Du traitement d'images dans ses rapports avec l'architecture des ordinateurs : deux études : la machine ROMUALD et le système KIDS

Bernard-Yves Bretagnolle

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR–INGENIEUR
«INFORMATIQUE»

par

Bernard-Yves BRETAGNOLLE

DU TRAITEMENT D'IMAGES DANS SES RAPPORTS AVEC L'ARCHITECTURE DES ORDINATEURS.

Deux études: la machine ROMUALD et le système KIDS.

Thèse soutenue le 20 janvier 1984 devant la commission d'examen.

F. ANCEAU	Président
C. BELLISSANT	
J.P. HATON	Examinateurs
S. LEVIALDI	
J.C. SIMON	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

Je tiens à remercier:

François Anceau, professeur à l'ENSIMAG, qui a dirigé mon travail de recherche. J'ai pu apprécier la confiance qu'il me faisait en m'acceptant dans son équipe et en me laissant une très grande autonomie dans le développement de mon travail. Je ne saurais dire combien il me fut agréable de travailler avec un patron qui savait se rendre disponible et n'était jamais avare de son temps, de ses conseils et de ses connaissances non plus que de ses critiques argumentées.

Camille Bellissant, professeur et directeur du Département Informatique de l'IUT de Grenoble. Pionnier grenoblois de la reconnaissance des formes, il a largement contribué à me faire découvrir ce domaine et à éveiller mon intérêt pour le traitement d'images. Son soutien et ses conseils m'ont été très profitables en particulier lors du développement de la machine ROMUALD.

Jean-Paul Haton, professeur à l'Université de Nancy 1 et animateur du groupe "Reconnaissance des Formes" de l'AFCET. C'est une joie, pour moi, qu'aprés avoir régulièrement manifesté son intérêt pour mon travail, il ait aujourd'hui accepté de le juger.

Stefano Levialdi, professeur à l'Université de Rome. J'ai eu la chance de rencontrer le Professeur Levialdi dés 1980, à Bonas, et j'ai pu, alors, apprécier ses qualités humaines et pédagogiques. Depuis, j'ai largement puisé dans ses travaux et ceux de l'équipe qu'il dirige. La renommée internationale que lui assure la qualité de ces travaux fait que c'est, pour moi, un honneur particulier qu'il ait accepté de participer au jury de cette thèse.

Jean-Claude Simon, professeur à l'Université Pierre et Marie Curie Paris VI et président de l'*International Association for Pattern Recognition*". J'ai tiré un grand profit de son action résolue pour le développement de la recherche française en reconnaissance des formes et plus particulièrement en traitement d'images. Mon travail doit beaucoup à ma participation aux écoles d'été de l'OTAN, à Bonas, dont il était

l'hôte et l'organisateur. Après avoir eu le plaisir de collaborer avec lui lors d'un projet de lecture automatique, je suis très honoré qu'il ait accepté de juger mon travail de recherche.

Je remercie, aussi, Françoise Renzetti et ses collègues de la bibliothèque de l'IMAG qui m'ont toujours accueilli de façon sympatique et compétente.

Enfin, je voudrais, de quelques mots génériques et complices, évoquer ce générateur de temps perdu aux multiples visages dont les appels téléphoniques, les visites ou les invitations m'ont bien souvent amené à tout plaquer là, pour quelques heures, quelques jours ou plus encore. Temps perdu? Voire!

CONTRE-CHANT

Vainement ton image arrive à ma rencontre
Et ne m'entre où je suis qui seulement la montre
Toi te tournant vers moi tu ne saurais trouver
Au mur de mon regard que ton ombre rêvée.

Je suis ce malheureux comparable aux miroirs
Qui peuvent réfléchir mais ne peuvent pas voir,
Comme eux mon œil est vide et comme eux habité
De l'absence de toi qui fait sa cécité.

Louis Aragon
"Le fou d'Elsa"

A mes parents,
A J. Lacan et A. Rimbaud,
Et à quelques autres
Dont ma vie porte la marque.

SOMMAIRE

INTRODUCTION

"Prenez un bouquet d'orchidées, jetez-le au feu. En quelques secondes, il n'y a plus qu'un peu de cendres et de fumée. Grâce aux lois de la thermodynamique, nous savons pourtant que rien ne s'est perdu: la plus grande masse du bouquet d'orchidées s'est transformée en vapeur et en chaleur; même si vous réussissiez à disloquer les atomes du bouquet en une gerbe nucléaire, la somme totale des énergies demeurerait constante. Rien ne se perd, rien ne se crée, telle est la loi. Rien ne se perd? Et mon bouquet d'orchidées, alors? Je veux dire sa forme? <<Ca, nous disent les savants, ça ne compte pas>>. Pardon?

Non, ça ne compte pas. Toute notre vie repose sur des trucs qui, pour la physique, n'existent pas. Des fantômes. C'est que les formes, voyez-vous, sont, en soi, sans énergie. Ce qui ne signifie pas sans force; mais elles restent hors des équations de la thermodynamique, donc elles échappent au monde de la physique et donc au <<réel>>."

La forme de tes hanches est une onde.
Patrice van Eersel. Actuel n° 45-46.
Juillet-Août 1983.

"<<Do you believe in ghosts?>>
<<No,>> I say.
<<Why not?>>
<<Because they are un-sci-en-ti-fic.*>>*

The way I say this makes John smile. <<They contain no matter,>> I continue, <<and have no energy and therefore, according to the laws of science, do not exist exept in people's minds.>>

The wiskey, the fatigue and the wind in the trees start mixing in my mind. <<Of course,>> I add, <<the laws of science contain no matter and have no energy either and therefore do not exist exept in people's minds. It's best to be completely scientific about the whole thing and refuse to believe in either ghosts or the laws of science. That way you're safe. That doesn't leave you very much to believe in, but that's scientific too.>>

Robert M. Pirsig,
Zen and the art of motorcycle maintenance,
The Bodley Head Ltd, 1974.

Ces prolégomènes peuvent sembler incongrus voire inutiles à l'intelligence de la thèse qui suit, et, s'il faut rester cartésien, ils le sont sûrement. Leur dessein est de proposer au lecteur une *toile de fond*, métaphore de la situation des deux domaines scientifiques essentiellement évoqués par la suite: le traitement d'images et l'architecture des ordinateurs.

Ni l'un ni l'autre n'ont, à ce jour, connu leur Carnot ou leur Einstein, et, en l'abscence de *la* formule, il faut admettre que dans ces domaines, la vérité n'est pas une. Les critères de choix et de jugement évoluent au gré de notions esthétiques de commodité ou de clarté. Cette subjectivité induit l'apparition de styles, d'écoles et de techniques diverses.

C'est dans cette situation, qui est, d'un certain point de vue, une providence pour la recherche, que le travail décrit dans cette thèse a été accompli. Il s'agit d'étudier le traitement d'images dans ses rapports avec l'architecture des ordinateurs, c'est à dire de proposer, subjectivement certes, mais de façon aussi cohérente que possible, une *vue* de ce que ces deux domaines, nécessairement appelés à se rencontrer, peuvent échanger. En bref, de répondre aux questions: <<Quelles contraintes le traitement d'images impose-t-il à l'architecture des ordinateurs?>> et <<Quelles solutions l'architecture des ordinateurs peut-elle proposer?>>.

Pour cela, cette thèse contient, dans son chapitre I, une présentation générale du traitement automatique des données image et décrit ses principaux domaines d'application et ses techniques essentielles.

Ensuite, le chapitre II est une large étude des matériels de traitement d'images. Dans ce chapitre, le paragraphe II.5 est le plus important. Il est consacré aux unités de traitement et, outre la description de quelques systèmes de traitement d'images, il présente des considérations et des propositions suggérant une *approche systémique* de la conception des systèmes informatiques pour le traitement d'images. Cette *approche systémique* vise à mener de front et de manière cohérente la conception du matériel et du logiciel. Il semble, en effet, que la structure interne des machines informatiques pour le traitement d'images ne doive être qu'un prolongement des algorithmes qui leur seront proposés. Ainsi, la conception doit être menée globalement, et la séparation entre le matériel et le logiciel -qui ne dépend que de contingences de réalisation ou de choix sur les performances à atteindre, souvent essentielles en ce domaine, et non d'un principe de base- doit permettre des interfaces aussi simples, aussi naturels et aussi directs que possible.

Dans le cadre de cette thèse, deux études ont été menées qui sont présentées dans une deuxième partie. Le chapitre III est consacré à ROMUALD, système multi-microprocesseur pour la saisie et le traitement d'images. Le chapitre IV propose une architecture plus ambitieuse: le système KIDS. Ce système allie les aspects logiciels et matériels et comprend un langage (dont la définition est donnée en annexe 1) et un ensemble matériel constitué d'une cascade de modules fonctionnellement spécialisés débouchant sur un processeur matriciel construit à l'aide d'un circuit VLSI, étudié en détail au paragraphe IV.6.

Ce travail, mené en position de chercheur associé -à temps partiel-, a fort heureusement bénéficié d'une activité principale d'ingénieur au Centre Interuniversitaire de Calcul de Grenoble où la machine ROMUALD a été réalisée. De même, les aspects *langage* du chapitre IV portent la trace de cinq années d'activité professionnelle dans l'équipe "Langages et Compilateurs" de l'IMAG.

Enfin, les Ecoles d'été du Forez, à Chalmazel, consacrées à la micro-informatique et à la micro-électronique, les Instituts d'Etudes Avancées de l'OTAN, à Bonas, consacrés au traitement d'images et les activités du groupe "Reconnaissance des Formes" de l'AFCET ont été des lieux très riches de formation et d'échanges dont la marque est présente tout au long des pages qui suivent.

CHAPITRE I

Le traitement automatique des données image.

Présentation générale.

I.1 Introduction

L'informatique, science du traitement automatique de l'information déplace, au fil des années, son attention des machines vers leur utilisation, de l'abstraction mathématique vers les hommes, leurs besoins collectifs et individuels. Les données visuelles, essentielles chez l'homme pour la communication et l'orientation, peuvent aussi, au même titre que les données numériques ou textuelles, être traitées de façon automatique.

I.1.1 Formation d'images

Toute scène visuelle est une image. L'image se forme sur la rétine grâce à la présence d'une lentille optique: le cristallin. L'œil peut ainsi être considéré à la fois comme un système de création d'images (le cristallin) et comme un système de saisie ou d'enregistrement d'images (la rétine). L'œil est un système de création/saisie d'images rendant compte de radiations électromagnétiques de longueurs d'onde

particulières situées dans le spectre visible.

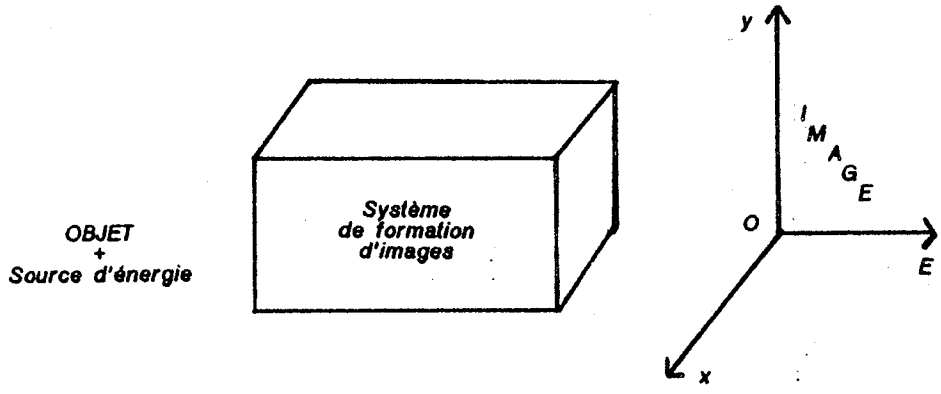


Figure 1.
Formation d'images: schéma de principe.

Des images peuvent être formées et enregistrées pour d'autres longueurs d'onde et par des systèmes autres que l'œil humain. La figure 1 étend un schéma de principe de formation d'images, donné par Andrews et Hunt dans [6]. Un objet est supposé se trouver dans un espace décrit par un système de coordonnées à n dimensions. Chacune de ces dimensions représente une caractéristique -en gros physique- de l'objet. Soit l'objet est irradié par une source d'énergie, soit il est lui-même une source d'énergie radiante. L'énergie radiante réfléchie, transmise ou émise par l'objet se propage à travers l'espace. Un système de formation d'images, idéalisé sur la figure 1 par une boîte, intercepte une partie de cette énergie radiante propagée et la transforme en une surface dans l'espace à trois dimensions (x,y,E). L'image ainsi obtenue est, en quelque sorte, une projection dans un plan (x,y) de l'objet observé, la dimension E rendant compte de l'énergie reçue dont les différents paramètres (quantité,type,...) peuvent être interprétés de diverses manières (distance de l'objet, nature physique, couleur,...).

De multiples variations à ce schéma de principe sont envisageables. En particulier, l'introduction du temps dans ce système permet de former soit une image représentant un phénomène (si le temps est une des coordonnées de l'espace où se trouve l'objet), soit une suite d'images. Enfin, l'image obtenue dans l'espace à trois dimensions (x,y,E) peut, dans le cas général, être transformée en une scène visuelle -c'est à dire que l'œil peut appréhender.

Bien que dans la suite de cette thèse, lorsqu'il est fait mention d'image -sans plus de précision-, ce soit de l'image assimilable à une scène visuelle dont il est question, il était important de mettre en valeur le rôle du système de formation d'images. En effet, la transformation qu'il effectue réduit l'information, la sélectionne et aussi l'altère, et dans de nombreux cas d'applications, traiter des images sans se référer explicitement à leur mode de formation conduirait à l'échec.

La présentation faite ci-dessus concernant la formation de ce qui sera désormais appelé image, montre à l'évidence que, sous réserve de disposer d'un système de formation d'images, à peu près n'importe quoi peut conduire à une image. Les seules limites de cette expansion boulimique potentielle sont de trois ordres:

- limites d'imagination: avoir l'idée d'exhiber d'un sujet d'étude des caractéristiques pouvant être transformées en images;
- limites de faisabilité: être capable de construire le système de formation d'images;
- limites de raison, enfin: s'assurer d'une part que l'image ainsi créée est porteuse d'informations pertinentes et utilisables, et d'autre part que les informations ainsi obtenues ne pourraient pas l'être plus facilement par une autre voie.

La formation d'images apparaît donc comme un domaine en soi. Les informations "pertinentes et utilisables" qui sont ainsi mises en image peuvent être d'importances diverses et donner lieu à deux types d'exploitation.

Un premier type d'exploitation, direct, laisse immédiatement la place à l'homme qui, par un examen -nécessitant ou non un apprentissage- de l'image va accroître ses connaissances ou fonder son jugement. Cette exploitation des images a été un des facteurs importants de transformation de notre société. Les exemples sont nombreux comme par exemple: la photographie, la radiographie et plus récemment, l'écographie dont le succès, en quelques années, est tel qu'on peut le mesurer à ses effets secondaires: dans certaines couches de notre société, toute femme enceinte de quelques mois qui ne connaît pas le sexe de l'enfant qu'elle porte court le risque d'être classée réactionnaire vieux-jeu ou délicieuse romantique!

Ce sont bien les mêmes paris de transformation de la société -avec,

là-aussi, des risques d'abus- qui sont faits dans le deuxième type d'exploitation d'images. L'image est, dans ce cas, soumise à un traitement dont le but est de fournir à l'homme un résultat utile ou utilisable.

I.1.2 Images digitales

Le processus de formation d'images présenté en figure 1 peut être réexaminé en termes quantitatifs. L'espace à trois dimensions (x,y,E) qui contient l'image est, potentiellement, continu et non borné, donc formé d'une infinité de points. Réaliser un traitement automatique suppose que les informations à prendre en compte soient dénombrables et situées dans un espace borné. Deux opérations vont donc être nécessaires:

- en utilisant des informations connues a priori sur le phénomène observé et sur la nature du processus de formation de l'image, une première opération va borner l'espace de travail. Cela signifie, trivialement, que, d'une part dans le plan (x,y) deux valeurs maximales x_m et y_m vont être choisies cernant dans ce plan le lieu des informations utiles; d'autre part, une valeur E_m doit être déterminée, bornant l'espace de mesure.
- définir, à l'intérieur de cet espace borné, et pour chaque dimension, une fonction de projection en un nombre fini de points. Ces fonctions de projection, dépendent à nouveau d'une part de la nature du phénomène observé, de la taille des détails significatifs relativement à celle de l'image par exemple, mais aussi d'autre part de la nature et de la fidélité des systèmes de formation et de lecture de l'image.

Ces deux opérations sont effectuées par un système approprié diversement nommé: capteur, digitaliseur, système de saisie, etc et le résultat est communément appelé image digitale. Une image digitale est un objet informatique assimilable à une matrice NxM de valeurs. Chacune de ces valeurs est comprise dans un intervalle discret et borné de cardinal connu V.

Les valeurs N et M (nombres de points de projection sur, respectivement, les axes Ox et Oy) définissent ce qu'il est convenu d'appeler la définition -spatiale- de l'image digitale. La valeur V (nombre de valeurs différentes pouvant caractériser un point du domaine spatial) est appelée précision de mesure ou de quantification.

I.1.3 Discussion

Le cheminement présenté ci-dessus, qui va du réel à une image digitale ne doit pas être considéré comme un schéma absolu et obligé. Il n'a pour but que de mettre en évidence un fait fondamental: une image digitale est le résultat de transformations, souvent peu fiables et toujours réductrices du réel.

I.2 Traitement d'images: les principaux domaines d'application

I.2.1 Le traitement d'images: raisons d'être

Les principales raisons pour lesquelles, des images étant formées, il est fait appel à un traitement automatique recouvrent assez bien les principales indications de l'informatique: la quantité et/ou la complexité des images, l'incapacité humaine à saisir l'information, et, enfin, la nécessité de conserver et de communiquer.

I.2.1.1 La quantité d'information

Bien que le processus amenant à la constitution d'une image digitale soit un processus réducteur, la quantité d'information subsistante est généralement encore très grande. A titre d'exemple, la figure 2 donne quelques valeurs, en supposant que l'image est digitalisée sous forme d'une matrice carrée de côté N dont chaque élément peut prendre 2^B valeurs différentes.

Taille du côté de l'image N	Nombre de valeurs de codage 2^B	bits par élément B	éléments par image N^2	bits par image BN^2
64	16	4	4.096	16.384
	64	6		24.576
128	64	6	16.384	98.304
	256	8		131.072
256	64	6	65.536	393.216
	256	8		524.288
512	64	6	262.144	1.572.864
	256	8		2.097.152

Figure 2.
Quantité d'information dans une image.

Ce tableau, éloquent en lui même, indique les valeurs les plus communes parmi celles rencontrées en traitement d'images. Il faut cependant noter que la tendance est à l'accroissement. Par ailleurs, les images digitales produites par les systèmes embarqués dans les satellites d'observation présentent une quantité d'information bien plus grande. Une image digitale LANDSAT II, par exemple, est une matrice de 3200 colonnes et 2400 lignes, chaque élément nécessitant 24 bits de représentation, soit un total de 207.360.000 bits [70].

Certes, la seule quantité de l'information ne justifie pas, en elle-même, l'utilisation de moyens de traitement automatique. En effet, un être humain est doté de systèmes biologiques et d'une capacité d'apprentissage et de jugement qui seraient le plus souvent suffisants pour *lire* une image, l'interpréter, extraire l'information recherchée. Ce qui impose le traitement automatique des images, c'est la volonté ou la nécessité de traiter -rapidement si possible, ce point sera examiné plus loin- de grandes collections ou de longues séquences d'images. Ceci peut être illustré par deux exemples:

- le satellite LANDSAT II, dont il est fait mention plus haut, émet en permanence des images de la Terre. Traiter, un jour, toutes ces images et en extraire toutes les informations utiles, tel est le pari des spécialistes de la télédétection. Pour ce faire, l'utilisation d'ordinateurs est plus que nécessaire puisqu'encore aujourd'hui insuffisante.
- la détection précoce des cancers de l'utérus, meilleure méthode pour vaincre cette maladie, suppose un examen détaillé et régulièrement renouvelé de frottis vaginaux [21]. L'extension de cette méthode de prévention à toute une population impose un traitement automatique [106].

D'autres exemples pourraient être fournis à profusion. Ce qui est à retenir, c'est que le traitement automatique d'images peut prendre une place prépondérante dans tous les cas où la grande quantité des images rendrait leur exploitation par l'homme soit impossible, soit non économiquement viable.

I.2.1.2 La complexité de l'information

Les images proposées pour un traitement automatique, et surtout

la nature des informations que l'on veut en extraire, dépassent parfois les capacités humaines normales. Cette complexité du traitement peut être due à différents facteurs, parmi lesquels:

- la quantité de calculs, d'opérations à réaliser;
- la quantité de détails, d'objets différents à prendre en compte;
- la trop faible lisibilité de l'image;
- etc.

Ainsi, la limitation des capacités humaines, que ce soit dans les domaines du raisonnement, de la mémoire, de la perception, ... ou plus simplement dans ceux de l'attention ou de la volonté, rend nécessaire un traitement automatique des images.

I.2.1.3 Incapacité humaine à saisir l'information

Ce paragraphe, bien que pouvant, de par son titre, être rapproché du précédent, envisage plus exactement le problème posé par les non ou mal-voyants. Ce qui est en jeu, c'est la possibilité de remplacer une partie au moins des informations qui ne sont plus transmises -ou le sont mal- par le canal visuel, par des informations transitant par d'autres canaux -généralement auditif ou tactile.

I.2.1.4 Nécessité de conserver ou de communiquer

L'apparition de la photographie avait déja apporté un moyen de conservation et de communication d'images. D'importantes améliorations ont été apportées par le bélinographe ou la télévision, par exemple. Mais ces systèmes de conservation et de transmission sont chargés de défauts. La conservation d'objets physiques limite la facilité d'utilisation ultérieure. Les temps de transmission sont longs, les réduire suppose une limitation de la définition spatiale et de la précision de quantification. Une bonne présentation des besoins modernes peut être faite en utilisant l'exemple des satellites LANDSAT qui forment des images de 207 millions de bits. Ces satellites émettent en permanence et ce depuis plusieurs années. Dans l'état actuel de la technique, ces images ne sont pas traitées en ligne mais stockées pour une utilisation ultérieure. Des techniques modernes de traitement d'images, techniques de codage d'images, techniques de compression d'images sont utilisées pour

transmettre ou stocker ces images avec pour but de minimiser l'emploi de la ressource critique (temps ou espace de stockage) sans perdre d'information ou en en perdant un minimum.

I.2.2 Les principaux domaines d'application

Il faudrait faire œuvre d'historien pour déterminer la date d'apparition de la première application en traitement d'images. Si l'on se limite aux traitements par des machines digitales, les premières publications apparaissent à la fin des années 50. Depuis, et principalement dans les dix dernières années, le traitement automatique d'images digitales a connu une expansion formidable tant du point de vue de la variété des applications concernées que du point de vue de l'amélioration des techniques. Aussi, seuls quelques grands centres d'intérêt sont examinés ici dans un but d'exemple et sans prétention exhaustive.

I.2.2.1 Les applications militaires

Ces applications doivent être citées essentiellement pour deux raisons:

- elles ont vraisemblablement été les premières réalisations opérantes,
- elles mobilisent des moyens matériels et humains souvent gigantesques. Par exemple, à Grenoble, on peut considérer qu'il y a, en 1983, environ 200 personnes qui travaillent sur des projets de traitement d'images. Sur ces 200 personnes, 100, au moins, travaillent sur des projets d'origine militaire.

Il est bien sûr difficile de donner une bonne information sur ces applications en raison du secret qui, naturellement, les protège. Quelques exemples sont connus:

- surveillance: détection du mouvement, de la chaleur, modification à long terme d'un paysage, recherche d'objets artificiels;
- défense, attaque: discrimination de la cible par rapport au fond, poursuite automatique, réplique, contrôle de tir; classification des cibles; identification des engins *amis*, détection et rejet des leurres, détection des camouflages;

- photointerprétation et gestion de banques de données de cartes, étude de la viabilité des terrains.

Il semble cependant que dans la plupart des cas, ce qui distingue les applications civiles n'est pas une différence dans les méthodes, mais plutôt une différence dans les moyens mis en œuvre. Ainsi, quand un robot industriel est capable de reconnaître et de saisir une pièce circulant sur un tapis roulant à 1 mètre/seconde, le système de guidage d'un missile sophistiqué se déplaçant à plusieurs fois la vitesse du son, doit être capable de détecter un autre objet, volant dans les mêmes conditions, de décider s'il est *ami* ou *ennemi*, de le poursuivre en prenant en compte le fait que sa trajectoire peut varier, qu'il peut y avoir des occlusions par des nuages, et de l'atteindre ... le tout dans un temps minimum. Un autre exemple: il est généralement admis que lorsque les systèmes LANDSAT ont été mis en service, en 1976, donnant une précision au sol d'une centaine de mètres, les satellites militaires donnaient une précision d'une dizaine de mètres.

Du point de vue de l'architecture des systèmes de traitement d'images, les applications militaires sont parfois très particulières soit en raison des images elles-mêmes (trés définies), soit en raison de la nature du système de formation des images (système infra-rouge ultra-sensible à balayage cavalier, par exemple). Cependant, les méthodes de traitement étant les mêmes, beaucoup d'opinions émises dans cette thèse doivent rester applicables.

I.2.2.2 La télédétection

Des images de la surface de la Terre sont prises d'avions ou de satellites, à l'aide de procédés photographiques ou d'autres systèmes, actifs ou passifs, le plus souvent multi-bandes, dans de nombreuses applications en sciences de la Terre. La liste ci-dessous, inspirée de Kazmierczack [50], présente ces principales applications:

- géologie: études tectoniques, pétrographie, détermination d'anomalies dans les limites de glaciation ou d'enneigement;
- agriculture, sylviculture: détermination des récoltes et de la végétation, détection des maladies ou des parasites, catastrophes naturelles: détection et estimation des dégâts (feux, orages, grêle,...);

- surveillance de l'environnement: étude analytique des zones utilisées (habitat, loisirs, bureaux, industries,...), analyse du trafic routier (bouchons, comportement des conducteurs), étude des climats locaux, de la pollution (poussières, température des eaux);
- météorologie: étude des formations nuageuses (horizontales et verticales), détermination de la direction du vent (circulation nuageuse), détermination de la température du sol, des océans, de la couverture nuageuse, des banquises, mesures spéciales (conditions atmosphériques, fréquence des pluies);
- hydrologie: surveillance des réserves en eau: volumes de neige, glaces, lacs naturels et artificiels;
- océanographie: observation des précipitations et des évaporations, recherche océanologique: sédiments, marées, mouvements, courants, glaces, températures, pollution, végétation;
- géodésie, topographie, cartographie: levé de terrains, établissement de cartes topographiques et thématiques, constitution de banques de données, analyse de terrains, interprétation.

Les traitements appliqués à ces images sont de deux types principaux. Le premier concerne l'acquisition des données, leur manipulation, leur compression, leur transmission, leur correction géométrique et radiométrique (voir à ce propos Goldberg [37]). Ce type de traitement, bien qu'on puisse en trouver l'équivalent dans la plupart des autres domaines d'application, est très particulier, important et complexe en télédétection en raison de l'originalité et de la complexité des capteurs et de la spécificité des conditions de prise de vue.
Le second type de traitements, bien que devant s'adapter aux besoins particuliers, est plus proche de ce qui est fait dans d'autres domaines d'application: extraction de caractéristiques, sélection et regroupement des données image, classification.

I.2.2.3 Le domaine biomédical

Ce domaine, bien que faisant le plus souvent appel à des techniques plus générales, a, lui aussi, des particularités certaines. Une première particularité -et ce n'est pas un détail sans importance- est que le service à rendre, l'aide à apporter s'adressent à des médecins ou à des biologistes dont les prérogatives et les habitudes, mais aussi la responsabilité, ne peuvent, encore moins qu'ailleurs, être largement

modifiées. Une deuxième particularité provient du fait qu'alors qu'en traitement d'images classique, le signal est raisonnablement bien défini, les images biomédicales sont, ordinairement, très pauvrement définies. Au lieu, par exemple, d'essayer de reconnaître ou de faciliter la détection, d'un engin blindé dans une photo aérienne, le problème médical typique consiste à détecter ou à faciliter la détection de tumeurs ou de métastases parmi des structures normales mais essentiellement variables. Alors qu'une forme prototype peut naturellement être donnée pour un engin blindé, c'est une tâche bien plus difficile pour une tumeur qui n'a, par essence, pas de caractéristiques conventionnelles bien définies. En traitement d'images biomédicales, le problème est bien souvent de découvrir des critères ou des mesures permettant une description de ce que l'on cherche qui puisse guider une reconnaissance automatique [35].

Les images biomédicales, qui peuvent être obtenues dans des conditions très diverses, se rattachent cependant à deux groupes: les images obtenues *in vivo*, directement sur le patient et des images prises *in vitro* [95]. Les médecins et les biologistes ont fait des efforts considérables pour permettre l'obtention de ces images.

Les principales techniques d'obtention d'images *in vivo* sont:

- les radio-isotopes;
- les ultrasons;
- la tomographie dite en français *scanner* (*X-ray computerised tomography*);
- la radiographie digitale;
- la résonnance magnétique nucléaire;
- la thermographie.

In vitro, au delà du microscope, la qualité d'une image est souvent celle de la coloration et de l'éclairage. Des techniques moins classiques, comme la micro-radiographie digitale, sont en voie d'être utilisées. Il serait long, et fastidieux, d'essayer de donner une liste des différentes applications existantes ou en projet. Il est cependant possible de les caractériser par grands thèmes.

- Images *in vivo*
  - détection de tumeurs ou autres anomalies de constitution,
  - étude du fonctionnement *mécanique* d'organes (cœur,

par exemple).
  - étude de la circulation de *quelque chose* dans l'organisme ou un organe;
- images obtenues *in vitro*
  - comptage, statistiques.
  - classification.
  - recherche d'anomalies.

Il faut noter enfin que certaines applications nécessitent de faire appel à des séries d'images successives représentant l'évolution d'un phénomène dans le temps.

I.2.2.4 La lecture automatique

Il s'agit là d'applications dont les esprits chagrins disent qu'elles ont fait couler plus d'encre qu'elles n'en ont lue. La lecture automatique a, en effet, été un des premiers centres d'intérêt pour des applications en traitement d'images et, dès le début des années 60, des chercheurs ont entrepris de résoudre ce type de problèmes.

Les principales raisons de cet intérêt doivent être cherchées d'une part dans la nécessité, pour la science du traitement de l'information encore à ses débuts, de tenter d'intégrer la masse colossale d'informations écrites pré-existantes, et d'autre part dans l'aspect de simplicité relative que présentaient, par exemple, des caractères imprimés.

Les études menées depuis ont montré que l'hypothèse de simplicité était très optimiste. Cette hypothèse se soutenait du fait que l'on a su, très tôt, formaliser la structure des caractères imprimés ou manuscrits [32]. Ainsi, disposer de grammaires génératives des formes à reconnaître, ouvrait la voie à la mise en œuvre de processus de reconnaissance à base d'automates, systèmes déjà largement explorés pour la reconnaissance des langages artificiels [74]. Cependant, la réalisation de tels systèmes s'est pendant longtemps heurtée à des difficultés dues soit à l'insuffisante précision des capteurs utilisés, soit à un manque de puissance des calculateurs disponibles, soit encore, et le plus souvent, au caractère non-économiquement réaliste des résultats obtenus.

Depuis, les applications développées sont diverses. Elles résolvent, en premier lieu, un problème plus simple mais de grande importance

économique: la lecture de *codes barres* ou de questionnaires à réponse multiple.

Pour ce qui est des caractères à proprement parler, certaines réalisations sont aujourd'hui commercialisées et présentent des résultats corrects [85]. Par ailleurs, le domaine d'application s'est diversifié vers les caractères arabes ou les idéogrammes chinois [73, 100], ou bien encore vers la reconnaissance de l'auteur d'un texte manuscrit ou d'une signature [11]. Des progrès théoriques ont été faits et bien qu'ils ne concernent pas exclusivement la lecture automatique, ils sont utilisés [53] et retrouvent parfois, et justifient, des méthodes jusque là heuristiques (par exemple, les squelettes de formes, [76]).

Enfin, la recherche est toujours d'actualité visant notamment à généraliser les conditions d'utilisation et à faciliter l'insertion des systèmes de lecture automatique dans la vie économique [77].

I.2.2.2.5 La robotique et le contrôle de processus

Ces deux domaines, bien que différents, sont rassemblés ici car ils sont proches du point de vue du traitement d'images. Que l'on cherche en effet à réaliser le système de vision d'un automate ou celui d'un robot, la nature des images, les contraintes de temps et les traitements généralement employés sont à peu près identiques.

Les images sont "simples" dans le sens où elles contiennent un nombre limité d'objets répondant d'un nombre limité de prototypes. Les conditions de prise de vue sont généralement suffisamment contrôlées pour donner des images bien définies et bien contrastées. Les contraintes de temps sont généralement primordiales: il faut réagir plus vite que le système qui est contrôlé. Enfin, les traitements effectués sont généralement limités, laissant la place, dans le cas des robots, à des systèmes sophistiqués de compréhension de scènes et de génération de plans d'actions [56].

I.2.2.6 Transmission et stockage d'images

Il est intéressant, et souvent nécessaire, d'essayer de réduire le volume d'information que représente une image: soit pour la transmettre

plus rapidement, soit pour occuper moins de place dans des organes de stockage. Cette réduction se fait par codage qui est soit un codage de données standard qui fonctionne en aveugle et ne prend pas en compte la nature des données image, soit un codage adapté (essentiellement à la relative régularité locale des images). De nombreuses techniques existent [19] et ont parfois, les contraintes de temps étant fortes (il faut coder aussi vite que l'interface de transmission ou de la mémoire de masse ne fonctionne), donné lieu à la réalisation de systèmes matériels spécialisés.

I.2.2.7 Les prothèses visuelles

Parmi les nombreuses aides aux handicapés de la vue, les projets de réalisation de prothèses visuelles sont les plus prometteurs pour le futur. Le sens visuel classique étant inopérant, il est inévitable d'utiliser d'autres voies sensorielles qui, cependant, même après apprentissage, ont un pouvoir de discrimination faible. Il est ainsi nécessaire, avant tout, de modifier ces images, pour les simplifier [80]. Après quoi, l'information visuelle est transmise au sujet via l'ouïe ou le toucher.

Une autre voie consiste à stimuler directement les centres de la vision du cortex cérébral par des impulsions électriques au moyen d'électrodes implantées, produisant ainsi des phosphènes (sensations ponctuelles de lumière sur la rétine). Il est actuellement possible de transmettre de la sorte des caractères Braille et des motifs simples. Il est clair que de multiples problèmes se posent: effet physiologique de l'implantation d'électrodes, types de stimuli à employer, choix judicieux des composantes de l'image devant être transmises, fonctionnement correct de la rétine et du nerf optique, etc. En outre, ce genre de manipulations corticales soulève des problèmes d'éthique médicale. Il est cependant réaliste de penser qu'à long terme une telle recherche sera menée à bien; elle offre un attrait certain du fait de sa simplicité conceptuelle et contourne les problèmes créés par l'utilisation des canaux tactiles ou auditifs.

I.2.3 Conclusion

Cette présentation rapide de quelques-uns des domaines d'application du traitement d'images montre l'extrême adaptabilité et

la puissance de celui-ci dans des situations très diverses. Cette diversité ne fait que s'accroître.

Cet accroissement est d'abord dû au développement propre du traitement d'images en tant que discipline scientifique qui a, dans les dix dernières années principalement, rassemblé ou construit les matériels, les techniques et les hommes.

L'influence grandissante de l'informatique sur la société, et surtout la baisse des coûts du matériel expliquent en deuxième lieu le développement du traitement d'images. Il est désormais possible d'offrir des services en traitement d'images à un niveau de prix inespéré il y a dix ans. Ainsi, les temps sont proches où le *boulanger du coin* utilisera un système de traitement d'images [103]!

I.3 Traitement d'images: les principales étapes techniques

Le fil conducteur de cette thèse étant l'architecture de systèmes de traitement d'images, il ne saurait être question de parler ici des techniques de traitement d'images pour elles-mêmes. Le paragraphe I.2 précédent, exposant les principaux domaines du traitement d'images, avait un double but: permettre au lecteur d'appréhender l'importance scientifique, économique et sociale du traitement d'images actuel et futur ainsi que des systèmes qui lui sont consacrés, de mettre par ailleurs en lumière, de façon tout à fait informelle, les contraintes de réalisation de systèmes de traitement d'images apportées par les exigences émises a priori concernant la valeur de l'un ou de plusieurs des paramètres: taille d'un objet élémentaire *image*, quantité d'objets *image* à traiter, temps de traitement. Décrire ici les principales étapes techniques du traitement d'images doit, de même, permettre de caractériser d'autres contraintes de réalisation. En effet, au stade où il est parvenu, le lecteur *naïf* -providence de ce genre de description descendante- peut dire d'un système de traitement d'images : ***<<Il faut une grande capacité de stockage; "ça" doit aller vite>>***... Mais rien n'est dit sur ce qu'un tel système doit faire. C'est donc en étudiant maintenant les principales étapes techniques utilisées qu'il va être possible de se faire une idée sur les algorithmes -et le genre d'opérations qu'ils utilisent- pour lesquels un système de traitement d'images peut être spécifié.

Les principales étapes techniques peuvent, classiquement, être regroupées en quatre ensembles:

- un premier ensemble est lié aux systèmes de formation d'images évoqués en I.1. Cet ensemble est cité ici pour mémoire, les systèmes de formation d'images étant, par nature, qualitativement différents des systèmes de traitement d'images, même si certaines fonctions peuvent être, en raison de choix ou de contraintes techniques, réalisées par les uns ou les autres.
- un deuxième ensemble, généralement appelé prétraitement, lui aussi de frontières variables, concerne ce qui, dans l'esprit de celui qui réalise l'application, *doit être fait, de toute manière, avant de pouvoir vraiment s'occuper du problème particulier que l'application cherche à traiter*.
- un troisième ensemble recouvre les techniques, dites de segmentation, qui divisent l'image en régions homogènes en fonction de certains critères.

- le quatrième ensemble est celui des techniques plus sophistiquées et c'est par elles que le résultat sera atteint. Ces techniques, bien que faisant le plus souvent appel à des méthodes générales, sont ordinairement très adaptées au problème traité et à ses particularités.

I.3.1 Le prétraitement: filtrages digitaux, convolutions [54]

Il s'agit là d'effectuer sur l'image des transformations visant à l'améliorer, au sens d'un ou de plusieurs critères. Ces critères peuvent être de nature esthétique ou subjective (restauration d'images) mais sont numériquement exprimables en terme de transformations de fréquences spatiales.

Bien que leurs réalisations effectives en soient aujourd'hui assez éloignées, ces transformations s'appuient sur une approche *fouriériste* de l'image considérée comme issue d'un signal analogique bidimensionnel. De façon analogue au cas unidimensionnel, un signal analogique bidimensionnel peut être caractérisé par une fonction Ia(x,y) qui possède une transformée de Fourier Ia(f,g):

$$Ia(f,g) = \int_{-\infty}^{+\infty}\int Ia(x,y)\ e^{-j2\pi(fx+gy)}\,dx\,dy$$

Les variables f et g sont appelées fréquences spatiales et expriment la tendance qu'a le signal Ia(x,y) à varier plus ou moins rapidement le long de ses coordonnées x et y [20].

I.3.1.1 Opérateurs bidimensionnels

Un opérateur bidimensionnel, S, transforme une image digitale Id(k,l) en une autre Id'(k,l):

$$Id'(k,l) = S[Id(k,l)]$$

Le domaine d'application d'un tel opérateur, fixé par k et l, peut être de taille plus ou moins grande. Lorsque ce domaine recouvre entièrement le domaine de définition de l'image, ou que leur rapport de surface est grand (par exemple supérieur ou égal à 1/16° ou 1/64°), on parle

d'opérateurs globaux ou régionaux. Dans ce cas, la transformation réalisée tient peu compte des détails, qu'ils soient partie intégrante du phénomène "mis en image" ou bien qu'ils ne soient qu'artefacts produits par le système de formation d'images (on parle alors, par analogie avec les signaux sonores, de bruit). Dans le cas inverse, où le domaine d'application de l'opérateur bidimensionnel S est petit par rapport à celui de l'image, ce domaine d'application sera appelé une fenêtre et l'opérateur sera qualifié d'opérateur local. La transformation préservera alors, ou mettra en valeur, les détails présents dans la fenêtre.

I.3.1.2 Opérateurs linéaires

Les opérateurs linéaires sont ceux qui satisfont la relation:

$$S[\alpha\ id_1(k,l) + \beta\ id_2(k,l)] = \alpha\ S[id_1(k,l)] + \beta\ S[id_2(k,l)]$$

Si, de plus, S est choisi invariant pour la translation (c'est à dire si sa réponse à $id(k-k_0, l-l_0)$ est $id'(k-k_0, l-l_0)$, et que sa réponse impulsionnelle est g(k,l), l'équation de convolution est vérifiée:

$$id'(k,l) = \Sigma_{k'}\Sigma_{l'}\ id(k',l')\ g(k-k',l-l')$$

$$= id(k,l) ** g(k,l)$$

et, dans le domaine spectral:

$$I'(f,g) = I(f,g)\ .\ G(f,g)$$

Les opérateurs bidimensionnels linéaires sont des filtres et quelques-uns sont bien connus et fréquemment utilisés soit comme opérateurs globaux soit comme opérateurs locaux. Par exemple, l'opérateur de moyennage arithmétique qui effectue un filtrage passe-bas atténuant les hautes fréquences spatiales et donc le bruit impulsionnel (d'acquisition, de quantification, etc). Un autre exemple d'opérateur local est un opérateur de SOBEL. Il fait partie de la large gamme de filtres passe-haut qui conservent les hautes fréquences spatiales et sont utilisés pour corriger les variations de luminance de l'image ou pour extraire les contours (ou bords) des objets présents dans l'image. Ces opérateurs approchent l'opération de dérivation, intrinsèquement contenue dans la notion de *contour discernable par la variation de luminance* (forte pente, variation

rapide), par le calcul d'un système de différences locales. Levialdi présente dans [63] une large discussion sur ce thème.

I.3.1.3 Opérateurs non-linéaires

Par rapport à leurs cousins linéaires, ces opérateurs présentent la caractéristique d'être plus fins, moins grossiers, en un mot, de donner souvent de meilleurs résultats, aussi constituent-ils la majorité de ceux utilisés en filtrage digital.

Non linéaires pour l'addition, on ne peut les caractériser par leur réponse impulsionnelle. Cependant, et c'est ce qui est à retenir pour le paragraphe suivant, leur application est de type convolutif. Un exemple d'opérateur non linéaire est l'opérateur de filtrage médian [60], de tendance passe-bas, qui diminue le bruit sans altérer trop fortement les contours.

I.3.1.4 Discussion

Les filtrages digitaux à l'aide d'opérateurs bidimensionnels de type convolutif ont, en traitement d'images, une importance majeure. La principale raison de cette importance tient dans l'universalité de leur emploi. Le spécialiste en traitement d'images fait du filtrage digital comme d'autres respirent, c'est là la base de son travail. Cette universalité d'emploi provient elle-même de la multiplicité et de la variété des opérateurs décrits dans la littérature et utilisés dans la réalité. Il est généralement possible de trouver des opérateurs convolutifs pour tout faire et dans le cas contraire, la variété infinie par nature des opérateurs non-linéaires permet d'en créer un nouveau qui donnera le résultat recherché. Du point de vue de l'architecture des systèmes, trois faits doivent être notés. Les deux premiers sont liés à la pratique du filtrage digital: grande quantité de données en jeu, multiplicité des opérateurs de filtrage utilisés au cours d'une même application. En effet, les opérations de filtrage digital sont effectuées en partant de l'image brute sur la totalité de ses points. Pour chaque point, le temps de calcul nécessaire pour l'opération de convolution du I.3.1.2 est soit directement fonction de la taille de la fenêtre considérée (dans le cas des opérateurs linéaires), soit de cette taille et, éventuellement, de la complexité de l'algorithme définissant la réponse du filtre dans le cas des opérateurs non-linéaires.

Par ailleurs, et c'est le deuxième fait, au cours d'une application, plusieurs filtres différents sont généralement utilisés de façon successive. C'est dire si la quantité d'opérations élémentaires à réaliser lors des opérations de filtrage est grande. Ainsi, un système de traitement d'images devra-t-il avoir une architecture conçue -d'abord conçue- pour *bien* réaliser les opérateurs convolutifs mis en jeu dans les filtrages digitaux. *Bien* les réaliser, c'est à dire: les réaliser vite, les réaliser tous. Ceci sera largement discuté par la suite.

Le troisième fait à noter concerne la nature même de l'opération de convolution et influe sur la *bonne* réalisation souhaitée ci-dessus. Mis à part le cas du traitement de scènes évoluant lors d'une séquence d'images (encore que, dans le cas de chacune de ces images la remarque soit valable), la notion de causalité n'intervient pas en filtrage digital. En effet, on dispose au même instant de l'intégralité de la scène, de l'intégralité du signal, de sorte qu'est rendue possible l'application parallèle sur tous les points de l'image de l'opérateur de filtrage. Cette remarque est au cœur des choix architecturaux des systèmes de traitement d'images.

I.3.2 Segmentation d'images

Le but des techniques de segmentation peut être, en première approche, défini comme étant de distinguer les objets présents dans l'image, entre eux et par rapport au fond. Les algorithmes de segmentation vont donc diviser l'image en un certain nombre de régions. La connaissance de ces régions sera ultérieurement utilisée par les étapes suivantes de classification et de reconnaisssance. Le nombre de ces régions est variable. D'une part en fonction de la nature du problème traité: si l'on a affaire à des caractères, deux régions, le fond et les caractères, suffisent généralement; si l'on a affaire à des objets plus complexes, par exemple dans les images cytologiques, le nombre de régions s'accroît: fond, hématies, cytoplasme et noyaux des leucocytes. Dans certains cas encore, le nombre de régions peut être très grand comme dans les images satellite utilisées pour l'étude des ressources terrestres.

Le nombre de régions issues de la segmentation varie aussi en fonction de la puissance plus ou moins grande des traitements de classification ou de reconnaissance utilisés par la suite.

En fait, segmentation et classification ou reconnaissance sont des étapes techniques très semblables et souvent confondues. En effet, dans tous les cas, un certain nombre de classes d'arrivées étant connues, il s'agit d'affecter chaque objet de départ à une de ces classes. Le problème essentiel est donc, pour chaque objet de départ, de faire un choix, celui de la classe à laquelle il appartient. Une bonne distinction, éclaircissant l'exposé et généralement admise et pratiquée, consiste à dire que la segmentation travaille sur des objets de départ qui sont des points de l'image et que les classes d'arrivée sont des régions, des domaines géographiques dans cette image. La classification ou la reconnaissance travaillent, comme ce sera exposé au paragraphe suivant, avec comme objets de départ des régions ou groupes de régions contigües et comme classes d'arrivée des caractéristiques sémantiques liées au problème posé.

Ainsi, le problème central de la segmentation est un choix, celui de décider, pour chaque point, à quel type de région il appartient. Dans le cas de la lecture automatique, par exemple, la segmentation consiste à construire deux régions ou ensembles de points: celle des caractères, celle du fond. Le critère de choix utilisé est le plus simple: les points clairs, dont la luminance est supérieure à un certain seuil, sont dits appartenir au fond, les points foncés, dont la luminance est inférieure au seuil sont dits appartenir aux caractères. Dans ce cas simple, le seul critère du choix est la valeur du seuil.

Dans le cas général, le choix n'est pas toujours aussi simple. Ainsi, ce qui caractérise les différentes techniques de segmentation, ce sont les critères de choix, leur nombre et leur façon d'intervenir dans la décision finale, et aussi la façon dont ces critères sont déterminés.

I.3.2.1 Détermination des critères

La détermination des critères de choix qui dirigent les opérations de segmentation est toujours, directement ou non, fondée sur une connaissance *a priori*. Ces critères, multiples ou non, sont toujours définis par des données préétablies ou fournies interactivement par l'utilisateur, et, éventuellement, par un algorithme d'adaptation paramétrique aux conditions du choix. C'est souvent un problème difficile que de les déterminer; en effet, ces critères de choix doivent être de nature quantitative, numérique, alors que dans la plupart des cas, la définition

typologique des régions à obtenir est plutôt d'ordre subjectif, voire esthétique. Le propos n'est pas ici d'approfondir ce problème mais de relever que ce passage du subjectif au quantitatif fait, le plus souvent, qu'il n'existe pas de mesure objective permettant de quantifier la qualité d'une opération de segmentation.

Ainsi, le jugement qualitatif est souvent encore, au moins dans les phases de mise au point, le résultat d'un examen visuel de l'image segmentée. C'est une des raisons qui amènent les systèmes de traitement d'images à être conçus pour permettre un travail interactif.

I.3.2.2 Les critères de choix

La segmentation est menée à partir d'une image digitale ayant généralement subi des opérations de filtrage. Ce dont la segmentation va disposer est donc une image digitale et l'assise objective du choix va donc être fondée d'une part sur les valeurs portées par chaque point de l'image digitale (généralement luminance associée ou non à teinte et saturation) et d'autre part sur des faits topologiques ou statistiques calculables à l'intérieur de cette image.

La segmentation consiste à diviser l'ensemble des valeurs prises en chaque point en un nombre choisi de sous-ensembles. Le résultat de la segmentation est donc une réduction du cardinal de l'ensemble des valeurs. Réaliser cette projection suppose la définition, dans l'espace de départ, d'intervalles.

La segmentation la plus simple aura des intervalles à bornes fixées a priori. Les techniques plus complexes de segmentation vont consister à faire varier ces bornes. Cette variation sera calculée par un algorithme choisi et éventuellement paramétrable qui prendra en compte des faits topologiques ou statistiques calculés globalement sur l'ensemble de l'image ou à l'intérieur d'une fenêtre locale. Ces faits peuvent être de nature extrêmement diverse, les principaux sont:

- les histogrammes, représentant la répartition statistique des points en fonction de leur luminance, teinte ou saturation,
- les calculs faits sur les histogrammes eux-mêmes (recherche de modes ou pics, approche entropique [79]),
- calcul des bornes en fonction d'une connaissance a priori des

régions recherchées (par exemple, rapport de surface),
- mesures locales: luminance moyenne, variance....
- influence du contexte pour créer des régions homogènes (surfaces *sans trous*, traits continus),
- prise en compte de la texture (structure "microscopique" de la répartition des valeurs)
- etc...

I.3.2.3 Discussion

L'approche de la segmentation présentée ici est trop rapide et il faudrait encore distinguer deux façons essentielles de procéder: la segmentation par les contours, qui cherche à identifier les régions par la discontinuité de valeurs sur leurs contours [63] et la segmentation par régions, duale de la précédente, qui consiste à rechercher la continuité de certaines caractéristiques [107].

Trois remarques peuvent cependant être faites à partir des paragraphes précédents.

La première concerne l'opération de segmentation en elle-même. Ce qui est important, au delà des techiques utilisées, c'est que la segmentation opère essentiellement une réduction de la quantité d'information et une structuration de cette information. Par conséquent, l'image résultant du traitement pourra désormais être stockée sous un volume réduit et surtout de façon structurée. Au delà de la segmentation, la classique matrice de valeurs peut céder la place à des structures d'information mieux appropriées (arbres quaternaires et pyramides [84], "processing cones" [45], etc) qui permettent aux traitements ultérieurs un accès plus efficace à l'information.

La deuxième remarque consiste à constater que dans la pratique, la segmentation est réalisée par -ou inclut- une ou plusieurs opérations de type convolution. Ceci ajoute du poids à la remarque faite au paragraphe I.3.1.4 qui notait que l'opération de convolution doit être au cœur de l'architecture des systèmes de traitement d'images.

Enfin, la troisième remarque concerne l'absence de critères objectifs pour un jugement de qualité de traitement, absence qui rend nécessaire un examen visuel des résultats et suppose une architecture logicielle

et matérielle fortement interactive.

I.3.3. Classification et reconnaissance des formes. Compréhension d'images

Classification et reconnaissance de formes sont des domaines très proches, au point que l'un ou l'autre terme sont parfois employés indifféremment. Une distinction, fragile, peut cependant être faite: alors que la classification relie les formes présentes dans une image à un certain nombre de classes ou types, la reconnaissance cherche, parmi les formes présentes, celles répondant à un ou plusieurs prototypes. Si l'on considère cependant, que les prototypes de la reconnaissance sont issus d'un apprentissage qui est souvent une classification, et que d'autre part la classification ne crée ordinairement pas des classes arbitraires mais des classes en nombre et de caractéristiques choisis, il apparaît que la discrimination est, pour le moins, floue.

Ce qui caractérise ces techniques, ce sont d'une part les objets sur lesquels elles travaillent (un nombre limité de régions, issues de la segmentation) et d'autre part les méthodes utilisées.

Ces méthodes sont des méthodes de type mathématique et peuvent être groupées en deux approches générales: l'analyse discriminante et l'analyse structurelle.

Dans l'analyse discriminante, un ensemble de mesures caractéristiques, ou propriétés, sont extraites des formes par calcul. Chaque forme est ainsi représentée par un vecteur de propriétés et la reconnaissance est généralement faite en partitionnant l'espace des propriétés.

Dans l'analyse structurelle, par contre, chaque forme est décomposée à l'aide d'un ensemble de sous-formes, ou formes primitives. Cette approche établit une analogie entre la structure des formes et la syntaxe d'un langage. La reconnaissance d'une forme est alors faite en analysant sa structure à l'aide d'un ensemble de règles de composition ou grammaire de formes [46, 88].

Pour ce qui concerne la compréhension d'images, ce terme est généralement employé lors de l'analyse de scènes tridimensionnelles. Le but est alors d'une part de reconnaître des objets tridimensionnels et d'autre part de déterminer leur arrangement relatif dans l'espace.

Plus généralement, il s'agit de constituer une représentation des connaissances sur la scène permettant à un système de la comprendre pour, par exemple, la modifier en dirigeant un robot [24].

Pour toutes ces techniques, il est possible de dire, contrairement aux précédentes, que l'aspect "image" s'estompe au profit de problèmes de structure de données et d'algorithmique: les données sont moins nombreuses, mieux structurées et les algorithmes beaucoup plus complexes. C'est sans doute pourquoi ce genre de problèmes est habituellement traité à l'aide de grands ordinateurs de structure classique. Le temps des architectures spécialisées viendra sûrement, peut-être par la réalisation de machines liées à la nature des outils d'expression algorithmique: machines LISP, machines PROLOG, etc... Mais l'heure est encore à la recherche tant pour l'extraction de propriétés, la définition de primitives ou la représentation des données que pour les algorithmes eux-mêmes et une définition plus précise, plus mathématique de ce qu'est une forme et de ce qu'est sa reconnaissance [75].

I.4 Conclusion

La présentation faite dans ce chapitre du traitement automatique des données images, bien que rapide et partielle, permet d'évaluer l'importance scientifique, économique et sociale de ce domaine de l'informatique.

Elle a posé, en outre, un certain nombre de faits qui doivent aider à définir l'architecture de systèmes de traitement d'images: la très grande quantité des données, le temps limité imposé à certaines applications, l'importance de l'opération de convolution, la nécessité de disposer d'un système interactif, l'importance que peut prendre la structuration des données. Ces faits seront repris, discutés, utilisés par la suite.

Cette présentation, enfin, met en évidence, vu la diversité et l'ampleur des applications, vu aussi la particularité et la relative stabilité des algorithmes de traitement utilisés, la nécessité actuelle de concevoir et de construire des systèmes spécialisés pour le traitement d'images.

CHAPITRE II

Matériels de traitement d'images

II.1 Introduction

Les systèmes matériels spécialement conçus pour le traitement d'images présentent aujourd'hui une réalité paradoxale: une très grande variété de réalisations ou de conceptions pour une production totale très faible. En dehors des domaines militaire et biomédical, pour lesquels des nécessités d'ordre socio-politique ont induit un développement rapide de la production, il est raisonnable d'estimer que, dans le monde entier, le nombre total de systèmes de traitement d'images ne dépasse pas mille ou quelques milliers de machines, parmi lesquelles bien peu ont été produites à un nombre notable d'exemplaires. Cette situation semble être liée à différents facteurs:

- le marché -économique- pour de tels systèmes n'est pas encore résolument ouvert. Cela implique d'une part que l'effet d'unification qu'engendre la fabrication en séries économiquement rentables ne joue pas. D'autre part, les perspectives d'avenir, le sentiment qu'un marché va s'ouvrir favorisent les activités de recherche débouchant sur la réalisation de nombreux prototypes différents.
- les techniques de traitement d'images connaissent depuis une

vingtaine d'années une évolution importante et rapide, et bien des pas effectués, bien des connaissances acquises ont pu donner lieu à des réalisations matérielles qui sont bien souvent le moyen de démontrer ou d'expérimenter l'efficacité des solutions proposées.

- les réalisations ont connu un essort quantitatif et qualitatif important grâce à l'intérêt précoce que certains spécialistes du traitement d'images ont porté, grâce à l'appel massif qu'ils ont fait aux développements des techniques et des réalisations de l'électronique digitale intégrée. L'apparition sur le marché des composants de microprocesseurs rapides et puissants ou de mémoires à grande capacité, mais aussi la possibilité plus récente de concevoir directement des circuits intégrés spécialisés ont été décisifs en rendant les coûts et les délais de conception et de réalisation de systèmes prototypes acceptables même par des équipes aux moyens limités.

La diversité des réalisations est telle qu'il semble aujourd'hui impossible, voire inutile, d'établir pour elles une classification fondée sur des critères d'architectures matérielles. Ce point est discuté au paragraphe II.5 où une présentation de quelques systèmes existants est proposée, organisée non pas en fonction de leurs architectures matérielles mais en liaison avec les choix de modèles algorithmiques qui ont, implicitement ou non, guidé leurs réalisations.

Cette présentation conduit au paragraphe II.6, discussion sur les liens logiciel/matériel dans les systèmes de traitement d'images.

Avant cela, les paragraphes II.2, 3 et 4 présentent rapidement quelques données, essentiellement descriptives, concernant respectivement les capteurs, les convertisseurs et les mémoires. Le chapitre est ainsi organisé pour parcourir, depuis l'image jusqu'au résultat, le schéma de principe illustré en figure 3.

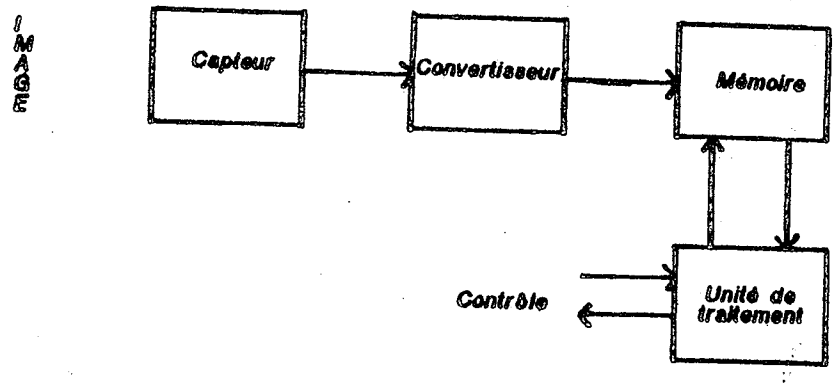


Figure 3.
Schéma de principe d'un système matériel de traitement d'images.

II.2 Les capteurs

Les capteurs d'images peuvent être classés en trois catégories: les capteurs photochimiques, les capteurs photoélectroniques et les capteurs de rayonnement.

II.2.1 Les capteurs photochimiques

Derrière l'appelation générique de capteurs photochimiques, c'est, plus prosaïquement, de la photographie et de ses dérivés dont il est question. La photographie, moyen de saisie d'images fiable, souple et précis, est en effet couramment utilisée, notamment parce qu'elle est aussi un moyen de stockage. Cependant, comme les traitements automatiques d'images sont effectués par des moyens électroniques, toute image photographique devra, pour leur être soumise, être présentée à un capteur photoélectronique. La photographie n'est ainsi qu'un premier pas, transitoire, vers le traitement automatique; aussi, l'attention sera concentrée par la suite sur les capteurs photoélectroniques et de rayonnement.

II.2.2 Les capteurs photoélectroniques

Les capteurs photoélectroniques sont basés sur un principe physique commun: la propriété de certains matériaux, dits photo-sensibles, de libérer, dans un champ de potentiel, des électrons lorsqu'ils reçoivent un flux lumineux. La quantité d'électrons libérés est directement proportionnelle au flux lumineux incident, tant que celui-ci ne détruit pas, thermiquement, la couche photo-sensible. Les différents capteurs réalisés suivant ce principe se différencient essentiellement d'une part par la nature du matériau photo-sensible utilisé et d'autre part par la façon de récupérer, pour mesure, les électrons libérés qui peut être réalisée soit par une photodiode soit par un photomultiplicateur.

L'organisation des systèmes capteurs photoélectroniques utilisés en traitement d'images est soit de type parallèle soit de type séquentiel. Dans les systèmes parallèles, l'image est obtenue par un ensemble de cellules photosensibles. Dans la pratique, ces cellules sont des photodiodes. Deux exemples de réalisation de tels capteurs sont présentés ci-après: les réseaux de photodiodes classiques et les réseaux de

photodiodes à transfert de charges. Dans les systèmes de type séquentiel, il n'y a qu'une cellule photosensible, constituée dans la pratique par un photomultiplicateur, complétée par un système de balayage permettant la mesure de l'intensité lumineuse en tout point de l'image.

Le paragraphe II.2.2.1 présente quelques méthodes de réalisation tandis que le paragraphe II.2.2.4 est consacré à une méthode particulière, largement utilisée, les caméras de télévision à tube classiques.

II.2.2.1 Les systèmes photomultiplicateurs

Dans son principe, un système photomultiplicateur travaille sur un flux lumineux provenant d'une région limitée de l'image à étudier. C'est la taille de cette région qui fixe la dimension d'un point élémentaire de l'image digitale utilisable. Le photomultiplicateur est constitué d'un tube électronique qui amplifie, à l'aide de grilles ou dynodes, le faisceau d'électrons arrachés au matériau photo-sensible par le flux lumineux. De tels dispositifs sont, le plus souvent, très fiables et permettent d'obtenir des mesures du flux lumineux stables et de grande précision (plusieurs centaines de niveaux distincts).

Différentes méthodes de balayage sont utilisées, assurant la transmission au photomultiplicateur du flux lumineux correspondant à un point de l'objet observé:

- systèmes par déplacement de l'objet observé: une mécanique assure le déplacement pas à pas de l'objet devant l'ensemble lentille optique-photomultiplicateur.
- balayage par disque de NIPKOW: un tel disque, percé de diaphragmes identiques disposés en spirales, interposé entre l'objet et le photomultiplicateur, assure le balayage du champ d'obervation fixe.
- balayage par miroirs vibrants: la transmission du flux lumineux, issu de l'objet observé, au photomultiplicateur est faite par deux miroirs mobiles selon des axes orthogonaux suivis d'un très petit diaphragme (taille d'un point). C'est en faisant varier l'angle d'incidence des deux miroirs que l'on obtient le balayage en x et en y.
- balayage par faisceau lumineux: un faisceau lumineux très étroit balaye l'objet et le photomultiplicateur, dont le champ recouvre

la totalité de l'objet, mesure le spot résultant.

De tels capteurs, très précis, présentent par contre des inconvénients: mise en œuvre et maintenance délicates, temps de saisie assez long.

II.2.2.2 Les réseaux de photodiodes classiques

De tels réseaux sont constitués d'éléments de base disposés en barres ou en matrices. Chaque élément de base est constitué d'une surface photo-sensible reliée à une capacité de stockage. Cette capacité est connectée à un transistor commandé par un système d'horlogerie et relié à une ligne de sortie au bout de laquelle est faite la conversion analogique-digitale. Chaque élément de base comporte, en outre, le plus souvent, une diode masquée, protégée par une couche de métal, empêchant ainsi l'exposition à la lumière. Ceci permet de fonctionner en différentiel et, de ce fait, de réduire les "bruits" d'échantillonage.

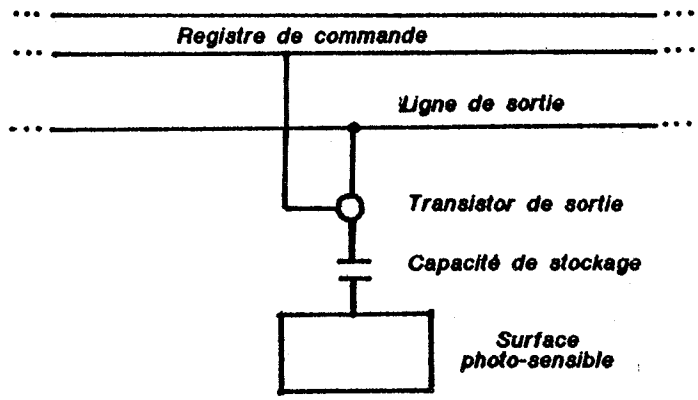


Figure 4.
Schéma de principe d'un élément de base
d'un réseau de photodiodes.

Des composants intégrés disponibles sur le marché, contiennent un grand nombre d'éléments de base et leurs registres de commande. Les fréquences d'échantillonage possibles atteignent aujourd'hui 10 Mégahertz.

De tels systèmes nécessitent cependant une horlogerie complexe et, dans le cas des barres, une mécanique de déplacement relativement à l'objet observé. Par ailleurs, des problèmes de précision se posent:

- problèmes géométriques: les surfaces photo-sensibles sont discontinues et ne couvrent pas tout le champ.
- problèmes électroniques: les quantités mesurées sont très faibles et présentent une non-uniformité élevée (de l'ordre de 10%).

De tels systèmes sont largement utilisés en raison, notamment, de leur faible coût.

II.2.2.3 Les réseaux de photodiodes à transfert de charges

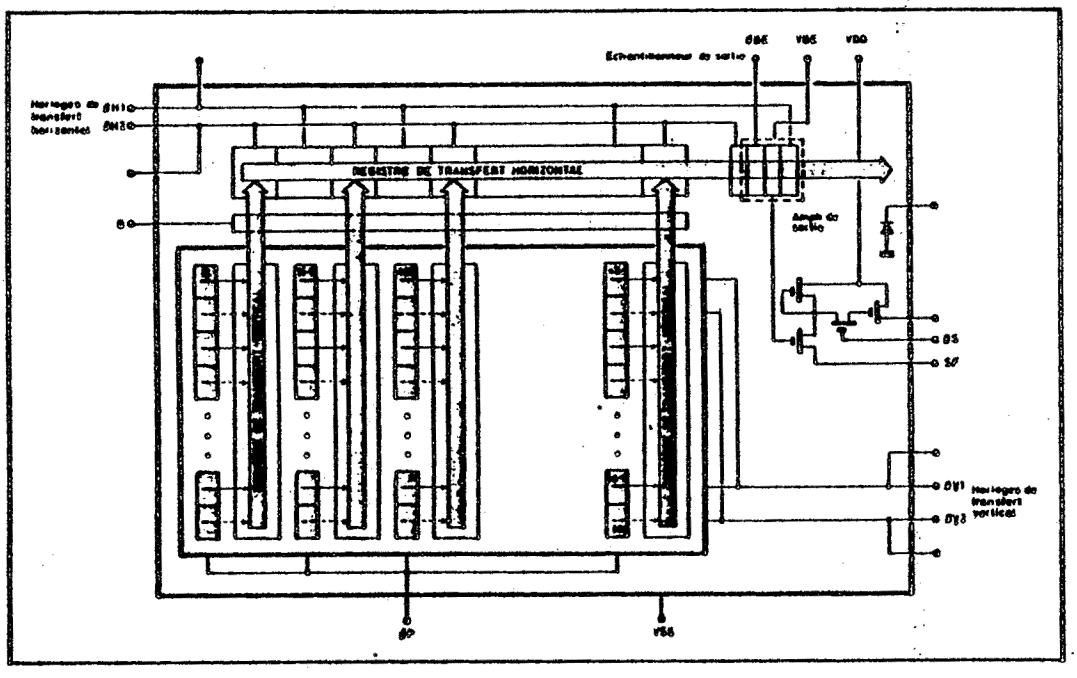


Figure 5
Schéma de principe d'une caméra à transfert de charges de résolution M * N points.
(d'après [94])

Du point de vue photoélectronique, ces systèmes sont identiques aux réseaux classiques. Ce qui les différencie, c'est la façon de récupérer les charges issues des éléments photo-sensibles pour les conduire à l'extérieur. Alors que dans les systèmes classiques, la conversion charge-tension est réalisée dans chaque élément de base afin d'appliquer la tension obtenue à une ligne de sortie, dans les systèmes à transfert de charges, ce sont les charges elles-mêmes qui sont conduites vers la sortie par l'intermédiaire de registres à transfert de charges (en anglais: *CCD, charge coupled devices*). De tels registres, maintenant

bien connus, présentent deux caractéristiques intéressantes: ce sont des mémoires de charge, ils peuvent fonctionner à haute fréquence (jusqu'à 1 Gigaherz, expérimentalement), tout en ayant dans le transfert un rendement très proche de 100 %.

Par rapport aux réseaux de photodiodes classiques, les réseaux à transfert de charges présentent deux avantages essentiels:

- un meilleur positionnement des cellules photo-réceptrices,
- le temps d'exposition est indépendant du nombre d'éléments.

Ces systèmes sont donc plus fiables, plus souples et plus simples à mettre en œuvre. C'est pourquoi il est prévisible que de tels réseaux seront de plus en plus utilisés en traitement d'images grâce à l'accroissement de leur définition spatiale (400x400 aujourd'hui) et à la diminution régulière de leur coût d'achat et de mise en œuvre.

II.2.2.4 Les caméras de télévision standard

Ce sont, de loin, les capteurs les plus couramment utilisés aujourd'hui.

Dans leur principe, les caméras de télévision sont des photomultiplicateurs. Le flux lumineux issu de l'objet observé frappe une surface photo-sensible et celle-ci libère des électrons. Dans une caméra de télévision, c'est en agissant sur le champ de potentiel permettant de récupérer les électrons qu'est réalisé le balayage de l'image.

Les systèmes télévision sont bien connus, largement employés et peu onéreux. Leur définition spatiale varie de 256x256 points à 512x512 expérimentalement 1024x1024). Le nombre de niveaux de gris obtenu est de quelques dizaines.

II.2.3 Les capteurs de rayonnement

Ce terme générique recouvre tous les capteurs réalisés, le plus souvent dans le domaine biomédical, permettant d'utiliser des images accessibles via des rayonnements non-lumineux: rayons X, γ, ultra-sons, résonnance magnétique nucléaire, rayonnement thermique, etc. Le modèle

le plus répandu de ces capteurs est le tomographe automatique à rayons X (*scanner*, en français).

La description de tels systèmes sort du domaine de cette thèse. Ce qu'il faut en retenir:

- ce sont des systèmes très complexes et très coûteux,
- ils mettent en œuvre des procédés de mesures physiques extrêmes: mesure de courants de 10 picoampères (*scanner* Thomson CE 10000), mesure de temps de l'ordre de 70 picosecondes (caméra positronique à temps de vol du LETI).

L'avantage essentiel de ces systèmes est qu'il permettent des coupes planes d'objets non-homogènes à trois dimensions.

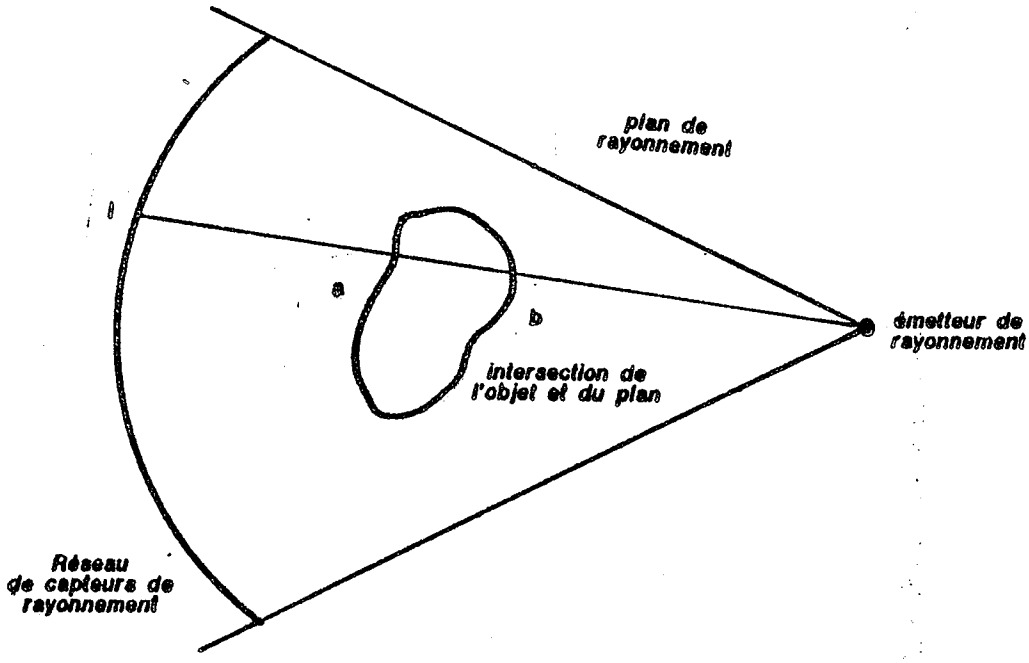


Figure 6.
Schéma de principe d'un *scanner*.

Le rayonnement mesuré au point I du capteur est une fonction modélisable dont les paramètres essentiels sont:

- le rayonnement émis,
- la capacité de transmission du rayonnement pour chaque point

de l'objet situé le long du chemin a,b,
- des phénomènes parasites issus de l'ensemble du plan et convergeant au point I (réflexion, diffraction).

Une analyse complète d'une coupe plane de l'objet ne peut être obtenue qu'en opérant une succession de mesures obtenues par rotation relative de l'objet et du système émetteur/capteur. Des calculs complexes utilisent cette série de mesures pour déterminer une image à deux dimensions correspondant à l'intersection de l'objet et du plan de rayonnement. Les valeurs obtenues pour chaque point élémentaire de cette image rendent compte de la capacité de transmission de ce point. Cette capacité rend compte de la nature physico-chimique de chaque point.

Ces systèmes à capteur de rayonnement sont donc en premier lieu des systèmes de création d'images. Cette création utilise des données qui ne sont pas des images visuelles et met en œuvre des ensembles matériels puissants et complexes.

II.2.4 Discussion

La présentation faite dans ce paragraphe sur ces capteurs n'étudie pas un certain nombre de problèmes importants, notamment ceux liés à leur réponse spectrale et leur rendement ou à leur définition spatiale et leur précision. A. Bijaoui ([10], chap 1 à 4) et C. Garbay ([35], chap II-A) présentent une étude plus détaillée de certains points.

II.3 Les convertisseurs

Les capteurs fournissent, en général, un signal analogique qui doit, pour être utilisé par des systèmes digitaux de traitement, être converti. La réalisation d'un convertisseur analogique-digital a été pendant longtemps un problème majeur lors de la construction de systèmes de traitement d'images. En effet, de nombreux facteurs concourent à la qualité et à la difficulté de conception d'un convertisseur analogique-digital: la précision (absolue et relative), la linéarité, la monotonie, la résolution, la vitesse de conversion, la stabilité et, bien sûr, le prix, mais encore le type de signal accepté en entrée, la nature des codes de sortie et l'encombrement physique.

Fort heureusement, la fin des années 70 a vu apparaître des convertisseurs analogiques-digitaux qui répondent de façon satisfaisante à la majorité des besoins en traitement d'images.

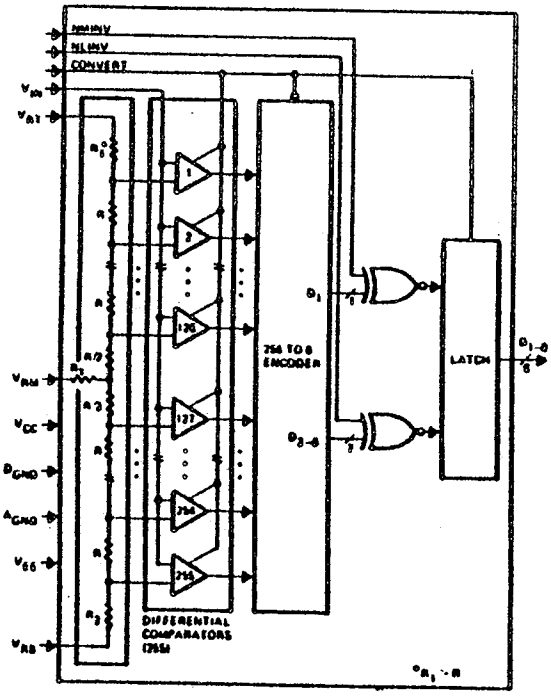


Figure 7.
Schéma fonctionnel du circuit
TRW TDC 1007J. (d'après [16])

La figure 7 présente le schéma fonctionnel du circuit TRW TDC 1007J, convertisseur intégré monolithique pouvant fonctionner à 30 Mégaherz.

qui permet de coder le signal d'entrée sur 8 bits.

De tels circuits s'interfacent directement avec des systèmes informatiques. Il existe, par ailleurs, des circuits intégrés de caractéristiques équivalentes réalisant des conversions digitales-analogiques. Ainsi, pour des systèmes capteurs du type télévision, le problème de conversion analogique/digitale peut être considéré comme résolu de façon satisfaisante par des composants standard.

II.4 Les mémoires

Dans ce domaine aussi, la situation a bien changé depuis quelques années. En effet, les contraintes de capacité, de temps d'accès et de coût inhérentes aux systèmes de traitement d'images sont aujourd'hui aisément satisfaites par des circuits intégrés disponibles sur le marché. Il s'agit là de mémoires digitales mais il convient de citer -pour mémoire- l'utilisation de mémoires analogiques. De telles mémoires, par exemple des enregistrements sur support magnétique du signal électrique issu du capteur, ont été utilisées mais le sont de moins en moins.

En ce qui concerne les mémoires digitales, leurs organisations et leurs modes de fonctionnement sont ordinairement très liés aux organisations et aux modes de fonctionnement de la ou des unités de traitement auxquelles elles sont reliées. Pour cette raison, des descriptions de réalisations concrètes de mémoires sont données seulement au paragraphe suivant, consacré aux unités de traitement.

Il est cependant possible de dresser une liste des caractéristiques qui permettent, généralement, de décrire les mémoires d'images.

II.4.1 Caractéristiques liées au nombre de niveaux de mémoire

Comme dans tout système informatique, les mémoires des systèmes de traitement d'images peuvent être organisées en niveaux en faisant appel à des modes de réalisation technologique différents:

- mémoires de masse sur disque magnétique,
- mémoires de travail classiques généralement réalisées à l'aide de circuits intégrés de grande capacité (16, 32 ou 64 K bits),
- mémoires caches, le plus souvent intercallées entre la mémoire de travail et une unité de traitement afin d'accélérer les accès ou, souvent, et de façon intimement liée à l'unité de traitement, afin de rendre facilement disponibles des données dispersées dans la mémoire de travail (Exemple: cache contenant les valeurs des points appartenant à une fenêtre donnée).

La tendance observée dans les réalisations récentes montre:

- un accroissement de la taille des mémoires de travail classiques

qui peuvent contenir au moins une image complète (par exemple 256 K octets), et souvent plusieurs, indépendamment de l'espace nécessaire au code des programmes et à leurs zones de travail,
- la diminution du rôle des mémoires magnétiques qui servent de mémoire à long terme et ne sont généralement plus utilisées au cours d'une étape donnée de traitement,
- l'importance croissante des mémoires caches, proches de l'unité de traitement, rapides et souvent complexes.

L'établissement des liens et la circulation des données entre ces différents niveaux peuvent être de nature très complexe et présentent parfois les caractères d'un véritable processeur mémoire, structuré le plus souvent en pipe-line, fonctionnant en parallèle avec l'unité de traitement et sous son contrôle.

Il va de soi que la réalisation de telles structures, pour être efficace, suppose que l'unité de traitement n'accède pas aux données de façon aléatoire mais suivant un algorithme régulier qui est réalisé par le processeur mémoire.

II.4.2 Caractéristiques liées aux accès

Classiquement, trois types d'organes permettent d'accéder aux mémoires des systèmes de traitement d'images:

- les systèmes de saisie qui, en sortie de leurs convertisseurs, accèdent à la mémoire pour stocker l'information digitale,
- les systèmes d'affichage qui, via un convertisseur digital-analogique, permettent de visualiser sur un écran ou tout autre support l'état de l'image en cours ou en fin de traitement,
- les unités de traitement qui utilisent les données en mémoire et les modifient.

Chacun de ces trois types d'organes a des contraintes spécifiques:

- système de saisie:
  - rapidité d'accès,
  - adressage le plus souvent séquentiel,
  - priorité d'accès liée au contrôle que l'on peut exercer sur le système capteur.

- système d'affichage:
    - . rapidité d'accès.
    - . adressage strictement séquentiel.
    - . priorité d'accès le plus souvent forte dans le cas des écrans.
    - . accès permanent;
- système de traitement:
    - . rapidité d'accès.
    - . adressage qui n'est jamais entièrement aléatoire mais parfois séquentiel, le plus souvent dispersé mais calculable, ou, enfin, aléatoire à l'intérieur d'un ensemble borné.
    - . priorité d'accès absolue, faute de quoi les performances totales du système se dégradent.

Les systèmes de contrôle associés à la mémoire et permettant de satisfaire au mieux l'ensemble de ces contraintes sont souvent fort complexes, en particulier lorsque le système de traitement d'images comprend plusieurs unités de traitement fonctionnant en parallèle.

II.4.3 Caractéristiques internes de réalisation

Une fois établi un cahier des charges de la mémoire, interviennent alors des choix concernant sa réalisation. Ces choix sont essentiellement conditionnés par:

- la nature et les performances des circuits intégrés utilisables,
- les temps d'accès,
- le mode d'accès.

Le choix d'un circuit est, là, primordial d'une part à cause de ses performances et de sa capacité, d'autre part à cause de son coût. En effet, si l'on écarte les coûts de conception, la part prise, dans le coût de fabrication d'un système de traitement d'images, par sa mémoire est bien souvent supérieure à 50 %. Le choix d'un circuit optimal doit donc prendre en compte:

- sa capacité, qui influence le nombre de boîtiers, de connexions, de plaques, etc.
- son coût unitaire,
- ses performances,

- la complexité et le coût des mécanismes de contrôle qui doivent lui être associés.

Des choix d'architectures peuvent enfin intervenir:

- choix de la taille d'un mot (1, 8, 16, 32 bits),
- choix d'un mode d'adressage (par exemple: utilisation de l'entrelacement des adresses de deux bancs mémoires qui permet, en cas d'adressage régulier, de diminuer l'intervalle de temps entre deux accès successifs),
- choix de la fonction de projection qui fait correspondre un point (x,y) de l'image à une plaque, un boîtier, une adresse... avec l'infinité de variantes que cela suppose.

II.4.4 Caractéristiques externes d'utilisation

Ces caractéristiques dépendent, en fait, de choix externes à la mémoire et sont liées aux besoins de l'un ou l'autre des organes qui lui sont connectés. Trois types essentiels peuvent être distingués.

II.4.4.1 Les mémoires circulantes

Basées sur une approche "producteur-mémoire-consommateur", les mémoires circulantes se trouvent dans des systèmes organisés, en tout ou en partie, en pipe-line. Mieux que de simples tampons d'attente, elles peuvent être organisées de façon à faciliter certains types d'accès aux données. La figure 8 présente le schéma d'une telle organisation. Sur ce schéma, la mémoire est constituée de trois registres à décalage R1, R2 et R3. La fonction de décalage est commandée par un signal Φ2. Chaque registre comprend autant de mots qu'il y a de points dans une ligne d'image et la taille de chaque mot permet de ranger la valeur significative (niveau de gris, par exemple) en chaque point. Le consommateur peut, dans ce cas, être par exemple un opérateur qui effectue une convolution sur une fenêtre 3x3 de l'image.

Le fonctionnement est le suivant:

- lors de la phase Φ1: producteur et consommateur lisent leurs données, calculent un résultat et le rangent dans les registres

de sortie RSP et RSG, respectivement;

- lors de la phase Φ2: les registres R1, R2 et R3 effectuent un décalage d'un mot vers la droite.

Dans de telles organisations, la notion de mémoire d'image devient très floue. En effet, il n'existe pas, dans le système, à un instant donné, une image complète à un stade de traitement donné. Par ailleurs, l'imbrication entre fonction de mémorisation et fonction de traitement est très forte et il devient artificiel de les séparer.

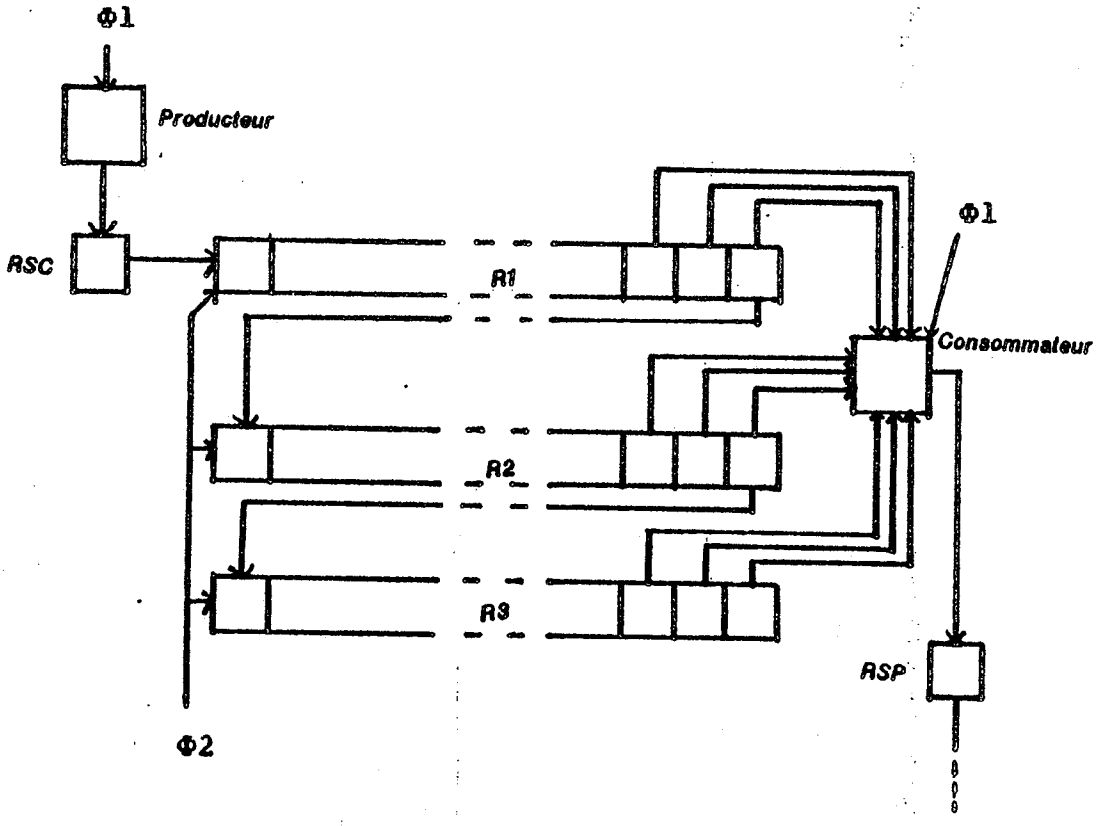


Figure 8.
Schéma de l'organisation d'une mémoire circulante permettant l'accès à une fenêtre 3x3 de l'image.

II.4.4.2 Les mémoires par plan de bits

Largement utilisées dans les systèmes graphiques, les mémoires par plans de bits le sont parfois dans des systèmes de traitement d'images. Fonctionnellement, de telles mémoires se présentent sous

la forme d'un ou plusieurs plans de mémoire. Chaque plan, quelle que puisse être sa réalisation matérielle, présente extérieurement une topologie correspondant à celle de l'image considérée: on accède à l'information en fournissant un numéro de ligne i et un numéro de colonne j. La valeur obtenue est un bit par plan pour chaque paire (i,j), ce qui correspond à une image binaire. En associant k plans, il est possible de travailler sur des images à 2^k niveaux de gris.

Ce type de mémoire est utilisé soit pour des raisons contingentes (disponibilité de plaques mémoires sur le marché, nécessité, *in fine*, de passer par une mémoire compatible avec un écran graphique, ...), soit en combinaison avec des processeurs matriciels (un processeur pour chaque point du plan). Dans ce dernier cas, c'est la structure du processeur qui est particulière, celle de la mémoire n'en est que la conséquence.

II.4.4.3 Les mémoires à mots

L'utilisation de mémoires à mots, mémoires classiques des systèmes informatiques d'usage général, résulte de choix divers:

- existence d'un savoir-faire largement répandu,
- volonté d'éviter des différences fonctionnelles entre mémoire d'image, mémoire de travail et mémoire de programme,
- nécessité, liée à une politique algorithmique, de pouvoir parcourir l'image selon différentes méthodes, en particulier de façon dispersée ou pseudo aléatoire,
- possibilité d'améliorer les performances de telles mémoires par utilisation de processeurs mémoire effectuant des algorithmes d'adressage sophistiqués (cf II.4.1).

Ce type de mémoires ne sera pas décrit ici en raison de son classicisme. Il faut cependant bien noter qu'elles sont le plus fréquemment utilisées dans les systèmes de traitement d'images, en particulier ceux à base de microprocesseurs conventionnels.

II.4.5 Discussion

L'exposé fait dans ce paragraphe II.4 consacré aux mémoires

des systèmes de traitement d'images montre que, nonobstant des caractéristiques générales communes (de capacité et de débit, par exemple), ces mémoires peuvent être extrêmement diverses. Cet exposé met aussi en évidence une notion: celle de non-séparabilité de la mémoire et des unités de traitement, tant du point de vue de la conception que de l'utilisation. Une explication à cela peut être avancée en comparant les systèmes de traitement d'images aux systèmes informatiques d'usage général.

En effet, si l'on veut bien accepter que puissent exister un programme type Pg pour un ordinateur d'usage général et un programme type Pi pour un système de traitement d'images, il est raisonnable de penser que, dans chaque cas, le temps total d'exécution de tels programmes peut être décomposé comme suit:

$$TT = TE + N*TA$$

où TE est le temps d'exécution par l'unité de traitement des instructions concourant directement au calcul du résultat, N est le nombre d'accès à la mémoire de données, TA est le temps moyen d'accès à cette mémoire incluant accès physique mais aussi calcul d'adresse, indexation, indirection, etc..

Ainsi,pour les deux programmes types cités ci-dessus:

$$TT_{Pg} = TE_{Pg} + N_{Pg}*TA_{Pg}$$

$$TT_{Pi} = TE_{Pi} + N_{Pi}*TA_{Pi}$$

Soient maintenant les hypothèses suivantes, liées à la nature des programmes Pg et Pi:

hypothèse 1: $TE_{Pg} \gg TA_{Pg}$

hypothèse 2: $N_{Pg} \ll N_{Pi}$

Sous ces hypothèses, les équations précédentes deviennent:

$$TT_{Pg} = TE_{Pg} + \epsilon$$

$$TT_{Pi} = TE_{Pi} + N_{Pi}*TA_{Pi}$$

avec:

TE_{PI} du même ordre que $N_{PI}*TA_{PI}$

voire:

$TE_{PI} < N_{PI}*TA_{PI}$

Ainsi, l'augmentation de performances dépend, dans le cas de l'ordinateur d'usage général, essentiellement de la rapidité de son unité de traitement. Par contre, dans le cas du système de traitement d'images, le nombre et le temps moyen d'accès à la mémoire de données jouent un rôle important, voire prépondérant.

Cette présentation est, certes, très intuitive et mériterait, pour avoir force de démonstration, de s'appuyer sur des mesures concrètes. Ces mesures semblent même délicates à réaliser, les notions de *programme type* et *d'instructions concourant directement au résultat* étant particulièrement floues. Il faut donc considérer cette présentation avec prudence, bien qu'elle ait un double intérêt:

- ce type de raisonnement semble guider bien des concepteurs de systèmes de traitement d'images.
- il met en évidence la part de l'accès aux données dans les performances d'un système de traitement d'images et donc la possibilité, la nécessité de concevoir des mémoires de données particulières.

Ces systèmes "mémoires de données" auront pour but de diminuer le terme "coût d'accès aux données", en jouant sur ses deux facteurs:

- N_{PI}:
  nombre d'accès à la mémoire de données: peut être réduit en supprimant, diminuant ou réduisant le coût (en temps) des lectures multiples d'une même donnée.
- TA_{PI}:
  temps moyen d'accès à une donnée: peut être réduit d'une part en améliorant les performances physiques de la mémoire, d'autre part en choisissant des modes d'adressage, d'indexation et d'indirection efficaces et en les faisant effectuer par des systèmes fonctionnant en parallèle avec l'unité de traitement.

Toutes ces améliorations ne peuvent, à l'évidence, être apportées qu'avec une bonne connaissance, *a priori*, des demandes d'accès mémoire émises par l'unité de traitement et dépendent donc d'un choix algorithmique qui dirige ainsi non seulement la conception de l'unité de traitement mais aussi celle de la mémoire.

Il faut enfin noter, en marge de ce paragraphe consacré aux mémoires des systèmes de traitement d'images, l'existence de systèmes contrôlés par les données (*data flow*). De tels systèmes, bien que radicalement différents de ceux évoqués dans ce paragraphe, réalisent en fait les mêmes fonctions d'accélération du traitement par traitement séparé des accès aux données et des calculs sur ces données.

II.5 Les unités de traitement

Le monde des systèmes de traitement d'images est vaste et divers. Différents par leurs conceptions, leurs architectures, leurs technologies ou leurs performances, les systèmes de traitement d'images n'ont même pas toujours en commun des caractéristiques liées à leurs utilisations tant est grand lui-même le champ d'application du traitement d'images. Face à cet état de diversité, le but du présent paragraphe aurait pu être d'établir une classification générale. Ce travail a été entrepris mais était bon pour la corbeille à papier. Le paragraphe II.5.1 donne quelques éléments relatifs à cet échec.

Une présentation exhaustive via une classification n'étant pas possible, le choix a été fait de présenter au paragraphe II.5.5 quelques systèmes de traitement d'images, connus ou non. Ce choix est donc le résultat d'un certain arbitraire mais est aussi conditionné par une "philosophie": celle de distinguer les "architectures spéciales" des "machines spécialisées" (comme expliqué en II.5.2), celle aussi, sous un autre point de vue, présenté en II.5.3, de considérer qu'un système informatique, avant d'être un assemblage de matériel, peut être défini comme exécutant un certain jeu d'instructions sur un certain ensemble de données. Cette philosophie se retrouve dans le paragraphe II.5.4 qui propose et illustre une typologie restreinte des systèmes de traitement d'images.

II.5.1 Echec d'une classification

Une classification des systèmes de traitement d'images existants peut trouver son origine dans trois sources complémentaires:

- les classifications existantes de systèmes informatiques généraux,
- les publications déja faites sur ce sujet,
- la définition et la prise en compte de critères de classification nouveaux.

Les classifications déja existantes des systèmes informatiques généraux, comme par exemple [33] ou [67] sont couramment utilisées dans la littérature pour définir l'architecture de systèmes de traitement d'images. Cependant, un examen détaillé des publications montre que l'intérêt, l'originalité ou la particularité des systèmes décrits n'ont rien à voir avec les dites classifications. En fait, comme un travail précédent le

remarquait [15], ces classifications s'arrêtent là où commence notre intérêt et ne peuvent pas être utilisées.

Parmi les publications existantes présentant les systèmes de traitement d'images sous un angle général et descriptif, aucune, semble-t-il, ne propose de classification d'un type différent de celles mentionnées ci-dessus. Il s'agit généralement de catalogues où des réalisations connues alternent avec celles de l'auteur ou d'équipes qui lui sont proches. Ces publications de synthèse n'en sont pas moins intéressantes cependant et ce à deux titres:

- elles sont une source d'informations condensées et parfois comparées sur différentes réalisations,
- elles démontrent, lorsqu'elles les utilisent, le manque d'intérêt des classifications générales pour les systèmes de traitement d'images.

Restait alors à essayer de définir une classification nouvelle. Cet essai n'a pas abouti pour quantités de raisons dont quelques unes suivent.

Les classifications essayées étaient trop complexes: un système était décrit de façon beaucoup plus concise par une description propre que par celle de sa position dans la classification.

Les critères de classification essayés n'étaient pas tous indiscutables, par exemple: <<*Existe-t-il un mécanisme non-trivial d'accès aux données ?*>>. Si un décodeur d'adresses doit être considéré comme trivial, une table de transposition d'adresses devrait l'être aussi. C'est pourtant là que se situe parfois la différence intéressante.

Il semblait nécessaire de faire intervenir des critères négatifs (et souvent flous): << *Impossibilité de traiter efficacement telle classe d'algorithmes*>>.

D'une façon générale, il semblait nécessaire pour arriver au but, mais dommageable pour la clarté de la classification, de faire intervenir des critères appartenant à deux mondes différents: ceux liés à la structure matérielle et ceux liés à son usage.

Enfin, les données disponibles sur les systèmes candidats à cette classification étaient issues de publications. Pas toujours très techniques, parfois polémiques, toujours partielles, ces données étaient insuffisantes.

Le paragraphe II.5.6 reviendra, en conclusion, sur le problème de classification en énonçant en particulier sous quelles conditions elle pourrait un jour être utilement entreprise.

II.5.2 Machines spécialisées et architectures spéciales

Le but de ce paragraphe est de proposer deux définitions qui semblent utiles à l'étude des systèmes de traitement d'images, voire au delà, de l'ensemble des systèmes informatiques se démarquant de la structure classique des ordinateurs.

Définition 1: Une **machine spécialisée** est le résultat de l'adjonction, permanente ou temporaire, à un système informatique de fonctions matérielles et/ou logicielles permettant ou facilitant son utilisation pour une classe particulière d'applications.

Exemples de machines spécialisées:

La connexion à un ordinateur classique d'un ensemble matériel de saisie et d'affichage d'images en fait une **machine spécialisée** pour le traitement d'images.

L'adjonction, au sein d'un ordinateur classique, d'une unité de traitement supplémentaire, permettant par exemple le calcul rapide de transformées de Fourier ou de produits matriciels, en fait une **machine spécialisée** pour certaines applications numériques utilisées en traitement du signal ou en analyse numérique.

De la même manière, plus simplement, la connexion à un ordinateur classique d'une unité d'impression de bonne qualité, jointe à l'installation d'un système logiciel particulier, en fait une **machine spécialisée** pour le traitement de texte.

Définition 2: Une **architecture** est dite **spéciale** lorsqu'elle s'écarte du schéma architectural traditionnel des systèmes informatiques.

Remarque 1: cette définition est, volontairement, assez floue. Elle suppose, en premier lieu, que soit défini ce qu'il faut entendre par *schéma architectural traditionnel des systèmes*

Informatiques. Le modèle général qui peut être donné est celui qui, basé sur les travaux de Turing [96], a été pour la première fois décrit par J. von Neumann [17] et est connu sous le nom de "machine de von Neumann" ou "machine à programme enregistré". Ce modèle associe trois éléments principaux: une mémoire, un chemin de manipulation et de transformation de données et une unité de contrôle. C'est l'introduction de l'unité de contrôle qui a permis de passer du stade "machine à calculer" au stade "ordinateur".

La validité de ce modèle peut cependant être l'objet de vives discussions. Il est, en effet, possible de considérer qu'aucun des ordinateurs commercialisés de nos jours ne répond, au sens strict, à la définition de von Neumann. Ainsi, par exemple, la partie contrôle s'est complexifiée et est devenue elle-même programmable. Le chemin de données s'est modifié lui aussi et est souvent organisé en pipe-line où plusieurs instructions sont en cours de réalisation à un instant donné. La mémoire, elle-même, s'est transformée, par un système de hiérarchie, par exemple.

Il n'en reste pas moins que ces modifications ne peuvent être considérées que comme des améliorations du modèle de von Neumann, destinées à accroître l'efficacité d'ensemble du système.

L'apparition des systèmes multiprocesseurs et plus particulièrement des systèmes distribués et plus précisemment des systèmes à contrôle distribué [67] a permis la réalisation de systèmes informatiques dont certains ont pu dire qu'ils ne relevaient pas du modèle de von Neumann. Il semble qu'il y ait là une confusion entre l'algorithme complexe du système d'exploitation de ces machines et les machines elles-mêmes qui relèvent, à n'en pas douter, du modèle de von Neumann.

Remarque 2: Un second critère doit être pris en compte lorsqu'il s'agit de savoir si on parlera, à propos d'un système informatique donné, d'architecture spéciale ou non. Il s'agit de la *vue* qu'a l'utilisateur, le programmeur d'application, de la machine qu'il utilise. Si il peut (ou doit), pour programmer son application, utiliser des ordres "non-von Neumanniens" comme par exemple affecter une tâche à un processeur, s'il doit avoir une

connaissance d'une autre organisation que la trilogie "programmes-données-unité de traitement", et si, particulièrement, ses algorithmes doivent être modifiés du fait de cette connaissance, alors le terme d'**architecture spéciale** rendra compte de cet écart par rapport aux systèmes informatiques traditionnels.

Quels que puissent être la généralité d'usage ou l'intérêt des définitions proposées ci-dessus, leur raison d'être dans cette thèse est de donner un éclairage intéressant sur les systèmes de traitement d'images. On assiste, en effet, dans ce domaine, à l'apparition d'une emphase certaine du vocabulaire qui fait que tout *engin* utilisé pour des applications en traitement d'images se voit doté de dénominations diverses qui véhiculent une information floue et souvent abusive. Sur la base de ces définitions, il est possible de proposer deux types de dénominations qui constituent, en fait, en embryon de classification.

Type A: **machines** d'architecture classique qui ont été **spécialisées** (au sens de la Définition 1).

Type B: machines d'**architecture spéciale** (au sens de la définition 2) qui ont été **spécialisées** (au sens de la définition 1).

Ces dénominations doivent être considérées comme provisoires et seront complétées au paragraphe II.5.4.

II.5.3. Architectures d'ordinateurs et langages d'instructions

Le point de départ de ce paragraphe consiste à rappeler le truisme trop souvent laissé pour compte selon lequel, avant d'être un assemblage de matériel, un ordinateur a pour fonction d'exécuter un ensemble d'instructions sur un ensemble de données. C'est la sémantique associée aux instructions et aux données qui détermine la valeur d'usage de l'ordinateur considéré.

Des raisons historiques expliquent que cette évidence mérite d'être rappelée. A travers le temps, le développement industriel et commercial de l'informatique s'est fait en se centrant sur un marché particulier: celui des ordinateurs d'usage général. Cela a conduit les études en architecture des ordinateurs à se focaliser pendant longtemps sur ce problème précis et c'est ce qui explique, qu'en deçà d'une diversité

réelle. les ordinateurs d'aujourd'hui présentent une grande uniformité si on les examine du point de vue du langage d'instructions qu'ils proposent.

Cette thèse n'est pas le lieu où puisse être entreprise une analyse détaillée de ce phénomène. aussi ce paragraphe se borne-t-il à présenter quelques généralités.

Un exemple archétypique de langage d'instructions est fourni par la compagnie IBM qui. depuis l'apparition de la série 360. il y a près de 20 ans. propose des ordinateurs haut de gamme obéissant à des langages d'instructions comportant un important noyau commun assurant une compatibilité sémantique et syntaxique à travers tous les modèles des séries 360. 370. 30xx et 44xx.

Il est possible de proposer deux raisons de cette étonnante continuité: raison économique. d'abord. elle permet aux clients d'IBM de changer facilement de machine sans avoir à supporter les coûts et les perturbations liés à une modification des logiciels; raison de fond. ensuite: avec ce langage. il est possible de *tout faire*. pourquoi donc en changer.

Des variantes existent bien sûr autour de cette notion de langage d'instructions *bon à tout*. Digital. par exemple. a proposé sur son PDP/8 la solution *record* d'un langage qui ne comportait que 8 instructions de base. certes largement paramétrées en ce qui concerne l'accès aux données.

Par ailleurs. des jeux d'instructions légèrement différents ont été proposés pour des ordinateurs d'usage un peu plus spécifique comme par exemple ceux destinés à la gestion de processus industriels qui faisaient une large place à la gestion des interruptions ou des priorités.

Il faut noter. enfin. la part. marginale cependant. prises par les machines langages comme celles de BURROUGHS sur la base d'Algol.

Cette uniformité est issue de l'échec des tentatives d'augmenter le niveau du langage d'instructions qui devient alors spécialisé pour une gamme d'applications dont on déborde assez rapidement. Une autre raison vient du fait que les mécanismes logiciels sont complexes. évolutifs et souvent mal définis avec un tas d'exceptions. Leur réalisation dans la machine amène à les simplifier et à les figer donc à les rendre

moins utilisables. Très souvent les programmeurs sont amenés à doubler en logiciel les mécanismes matériels jugés incomplets.

C'est peut-être pourquoi, pour la réalisation des ordinateurs d'usage général de très grande puissance, la tendance semble être de proposer des machines avec des jeux d'instructions ultra-simples mais qui permettent de réaliser du matériel ultra-rapide.

Ainsi, avec le temps, la différence sémantique entre les primitives des langages de programmation et celles des langages d'instructions s'est accrue; de ce fait, les compilateurs chargés d'assurer la correspondance entre les uns et les autres sont devenus de plus en plus complexes.

Il est possible de dire que la puissance d'expression des langages d'instructions n'a pratiquement pas évolué depuis 20 ans. Quelques progrès ont bien sûr été réalisés, parmi eux:

- l'apparition aux côtés de données à valeurs entières, logiques, réelles ou caractères, d'objets sémantiquement plus complexes:
  - les piles d'appels procéduraux (instructions CALL et RETURN),
  - les piles d'usage général (instructions PUSH et POP),
  - les tableaux (instructions d'indexation et de contrôle de dépassement de bornes);
- l'apparition d'instructions répétitives prenant en compte une partie du contrôle (par exemple: comparaison de chaînes de caractères),
- l'apparition dans le langage d'instructions de notions complexes comme celles de changement d'environnement (iAPX 432 d'Intel).

Il n'en reste pas moins que ces progrès laissent subsiter:

- de grandes différences sémantiques avec les langages de haut niveau du type Algol, Pascal ou PL/1,
- une inadéquation criante avec les besoins de langages comme Lisp ou Prolog,
- la référence encore largement répandue à des êtres para-technologiques comme registres, accumulateur, vecteur d'état, etc qui semblent être des vestiges d'un passé durant lequel les instructions de ces langages découlaient plus des possibilités de la machine qu'on était parvenu à construire que

de l'analyse des besoins de ses futurs utilisateurs.

Cet état de fait semble très dommageable pour l'avancement de l'informatique. En effet, les langages d'instructions permettent, en théorie, d'écrire les programmes les plus efficaces lors de l'exécution et les assembleurs sont bien plus faciles à écrire (et généralement plus rapides à exécuter) que les compilateurs. Ainsi, pourquoi ne pas tout écrire en langage machine? Le problème, bien connu, est que ces langages d'instructions de bas niveau, bien que très efficace à assembler et à exécuter, sont des outils inefficaces pour le développement de programmes, spécialement lorsque ceux-ci deviennent un peu longs. Il a été montré que la vitesse à laquelle les instructions (mises au point) d'un programme sont produites est indépendante du langage utilisé et qu'ainsi les langages de haut niveau présentent un avantage considérable sur les langages de bas niveau en raison de la correspondance de plusieurs instructions bas niveau pour une seule de haut niveau. Mais les langages de haut niveau produisent un code qui est souvent moins efficace à l'exécution que le programme équivalent en langage machine. Il y a là un cercle vicieux qui fait intervenir des choix sur le temps d'exécution et l'efficacité du développement de programmes. Bien sûr, une conception attentive des langages de haut niveau associée à des systèmes de compilateurs-optimisateurs peut résoudre certains problèmes. Mais une question fondamentale subsiste: celle de l'inadéquation des langages d'instructions proposés avec les langages de programmation de haut niveau. c'est peut-être pourquoi les langages de programmation effectivement utilisés sont de bas niveau: Fortran, Basic et même Lisp qui est, finalement, de très bas niveau.

Concernant les machines spécialisées, la situation est différente et il semble possible de concevoir des machines et des jeux d'instructions spécialisés et ce d'autant plus aisément que la spécialisation est forte. C'est le cas en traitement d'images, particulièrement pour le prétraitement.

De ce point de vue -celui de la conception de langages d'instructions nouveaux, utilisables sur des architectures de machines nouvelles pour le traitement d'images- un examen attentif des réalisations permet de relever des comportements divergents.

D'une part, il faut noter l'existence de réalisations allant tout à fait dans ce sens:

- au niveau des systèmes matériels: définition précise d'objets: images, masques, fenêtres; choix d'opérations primitives et de paramètres; conception d'architectures réalisant ces primitives;
- au niveau des langages de programmation, même lorsque cela ne conduit pas à des systèmes matériels particuliers, la recherche de définitions précises de types de données et de primitives d'opérations est largement entreprise.

D'autre part, la force de l'habitude induite par le modèle classique d'ordinateur et par son langage d'instructions est très grande. Par exemple, les langages spécialisés pour le traitement d'images sont presque tous conçus pour être exécutés sur des machines classiques et donc compilés vers un langage d'instructions traditionnel. Autre exemple, les propositions de réalisations matérielles relèvent bien plus souvent d'une *approche architecturale classique* qui cherche -non sans mérite, car il s'agit d'une opération délicate- à augmenter, dans certains cas d'utilisation, l'efficacité de traitement de programmes écrits en langages d'instructions traditionnels, plutôt que d'une approche visant à définir d'abord un langage d'instructions nouveau et à construire le système matériel ensuite.

Tout se passe comme si le langage d'instructions traditionnel constituait un point de passage obligatoire. Or l'état de l'art en architecture des ordinateurs est aujourd'hui tel qu'il n'en est rien et que, bien plus, ce domaine scientifique réclame et espère des propositions de langages nouveaux qui lui permettent de confronter ses outils et ses méthodes à des problèmes plus nouveaux et moins largement parcourus que celui de l'ordinateur d'usage général.

Reste, bien sûr, à construire des propositions qui aillent au delà des prémisses actuelles, puis à rechercher, parmi elles, un langage d'instructions de traitement d'images qui puisse rassembler un certain consensus des utilisateurs. Il n'est pas aujourd'hui possible de dire si cette voie, présentée ici comme une bonne voie, sera largement suivie dans les années à venir.

II.5.4 Proposition d'une typologie restreinte et illustration

Les différentes considérations présentées dans le paragraphe II.5.3 permettent de compléter l'amorce de typologie proposée au paragraphe

II.5.2.

Définition 3: une machine sera dite **spécifique** pour une application donnée (par exemple le traitement d'images) si son langage d'instructions comprend des instructions spécifiques à cette application.

		Architecture spéciale	
		NON	OUI
Machine	NON	Type 1	Type 2
spécifique	OUI	Type 3	Type 4

Figure 9
Constitution d'une typologie restreinte

Cette définition permet de proposer, pour les systèmes matériels utilisés en traitement d'images une typologie restreinte définie dans le tableau de la figure 9.

Remarque: cette typologie ne s'intéresse qu'aux machines répondant au moins au critère **spécialisées** au sens de la définition 1 du paragraphe II.5.2.

La figure 9 peut être explicitée comme suit:

Type 1:

Il s'agit là de machines à architecture et à langage d'instructions traditionnels. Elles ont simplement été **spécialisées** par adjonction, par exemple, d'unités d'entrées/sorties d'images.
Exemple: de nombreux micro-ordinateurs peuvent devenir des systèmes de traitement d'images du type 1 par adjonction de cartes d'entrées/sorties d'images, parfois disponibles sur le marché, comme c'est le cas pour l'APPLE II, le SIRIUS, le LSI 11 de DEC, etc.

Type 2:

Il s'agit là de machines à **architecture spéciale**, mais dont le langage d'instructions reste traditionnel. Beaucoup de

réalisations actuelles relèvent de ce type. Une partition de ce type peut être envisagée, bien que le critère distinctif ne soit pas entièrement satisfaisant:

Type 2.a:

machines spécialisées (cf définition 1) à **architecture spéciale** (cf définition 2), à langage d'instructions traditionnel, **caractérisées** par le fait que la nature spéciale de leur architecture **ne dépend pas** de considérations liées à l'usage qui va en être fait pour le **traitement d'images**. Relèvent de ce **type 2.a** des réalisations (généralement multiprocesseurs) qui, comme EMMA [65, 89], par exemple, sont essentiellement des machines d'architecture spéciale, conçues pour un usage général et spécialisées, ensuite, pour le traitement d'images.

Type 2.b:

machines spécialisées (cf définition 1) à **architecture spéciale** (cf définition 2), à langage d'instruction traditionnel, **caractérisées** par le fait que la conception de leur architecture spéciale **prend en compte** des considérations liées à l'**usage** qui va en être fait en **traitement d'images**. La machine ROMUALD, présentée au chapitre III, peut être considérée comme relevant de ce **type 2.b**.

La validité du sous-classement proposé en deux sous-types est très relative dans la pratique. Si l'on considère, par exemple, le système SYMPATI réalisé à Toulouse et dont on trouvera une description détaillée dans [9], il semble possible de l'affecter à l'un ou à l'autre des sous-types selon que l'on accorde plus ou moins d'importance à la description qui en est faite en terme d'utilisation pour le traitement d'images et qui influence notablement l'appréciation. Ce sous-classement peut cependant être retenu dans certains cas moins ambigüs.

Type 3:

Il s'agit là de **machines spécialisées** (cf définition 1) à architecture classique offrant un langage d'instructions **spécifique** pour le traitement d'images. Résultat vraisemblable de la polarisation sur les problèmes d'architectures matérielles plus que sur les langages d'instructions, il ne semble pas qu'existent, à l'heure actuelle, des systèmes répondant

exactement à ce type. Un certain nombre de systèmes, particulièrement parmi les systèmes commercialisés, peuvent être rattachés à ce type. Il s'agit de systèmes construits de façon classique sur la base, par exemple, d'un microprocesseur existant. A ce processeur sont adjoints un ou plusieurs co-processeurs spécialisés pour effectuer une opération particulière. Il est possible de considérer que la séparation physique entre l'unité de traitement et les co-processeurs n'est affaire que d'opportunité de réalisation. Du point de vue du contrôle de l'ordonnancement des programmes, aussi bien que du point de vue de la présentation du système, il s'agit bien d'offrir une unité de traitement capable d'exécuter un langage d'instructions spécifique (au moins en partie) pour des applications en traitement d'images. Les machines Péricolor, commercialisées par Numélec, relèvent, par exemple de ce type 3.

Type 4:

Il s'agit là de machines spécialisées (cf définition 1) à architecture spéciale (cf définition 2) offrant un langage d'instructions spécifique pour le traitement d'images. Des réalisations de ce type existent et sont très diverses et l'absence d'un moyen de présentation, de classification générale est, dans ce cas, particulièrement regrettable. Des systèmes comme GOP [39], CLIP4 [28], Analyseur d'images [51] ou KIDS (présenté au chapitre IV) relèvent de ce type 4.

Comme toute présentation du type "classification", celle proposée ci-dessus n'est pas pleinement significative si elle n'est pas accompagnée d'exemples qui illustrent les différences de modes de fonctionnement dans des cas classiques.

Un exemple va donc être présenté pour illustrer cette classification. Il convient cependant de garder à l'esprit qu'il ne s'agit que d'un exemple tout à fait partiel qui va se préoccuper uniquement de l'étude d'une opération très banale, le seuillage binaire d'une image, et étudier cette opération en fonction d'un seul critère: le temps d'exécution. D'autres critères pourraient être associés à l'étude comme la taille du code, le nombre de paramètres, le taux d'utilisation du matériel, la complexité et le coût du système, etc.

Soit donc à effectuer le seuillage binaire d'une image. Cette opération est définie:

1) par ses données:
   - une image caractérisée par un nom I et le nombre N de points qu'elle contient.
   - une valeur de seuil S.
   - deux valeurs de remplacement INF et SUP.

2) une action à effectuer:
   la valeur portée par chaque point de l'image I est modifiée comme suit:
   - elle devient égale à INF si la valeur de départ est inférieure à la valeur de seuil S.
   - elle devient égale à SUP si la valeur de départ est supérieure ou égale à la valeur de seuil S.

Cette définition semble naturelle, cependant, elle élude deux problèmes pourtant importants qui dépendent de la nature, de la "philosophie" de la machine qui va être utilisée:

- l'image est définie par un nom I et un nombre de points N; reste à dire comment on accède à la valeur d'un point particulier de cette image, en un mot, quelle est l'organisation en mémoire de l'image.
- les valeurs utilisées S, INF, SUP et celles portées par chaque point de l'image de départ doivent être définies de façon exacte de même que la relation d'ordre nécessaire à l'opération. Cette remarque n'est pas purement formelle, en effet, de telles définitions ne sont pas triviales à établir et à mettre en œuvre lorsque l'on a affaire, par exemple, à des images colorées ou à des processeurs bits-sériels.

Cas des machines de type 1:

Soit une telle machine constituée d'un microprocesseur classique et d'une mémoire de mots. L'algorithme de seuillage va s'écrire (aux variantes de jeux d'instructions près):

```
1        LDA    R1,I      !charge R1 avec l'adresse de I
1        LD     R2,N      !charge R2 avec le nombre de points
2 E1:    LD     R3,@R1    !R3 = valeur du point courant
2        CP     R3,S      !cette valeur est comparée au seuil
3        JP     lt,E2     !saut si valeur < S
2        LD     @R1,SUP   !remplacement valeur courante par SUP
3        JP     E3        !suite
2 E2:    LD     @R1,INF   !remplacement valeur courante par INF
3 E3:    INC    R1        !R1=R1+1, passage au point suivant
3        DEC    R2        !tant que tout n'est pas traité
3        JP     nz,E1     !on recommence la boucle
```

Ces instructions peuvent être divisées en trois catégories indiquées en marge:

Catégorie 1: initialisation.
Catégorie 2: action nécessaire au seuillage.
Catégorie 3: contrôle.

En supposant, pour simplifier, que toutes ces instructions s'effectuent en un même temps t, on obtient les temps de calcul suivants:

Catégorie 1: 2t,
Catégorie 2: (3t)*N,
Catégorie 3: (4t)*N.

Soit un temps total de $(7N + 2)\ t$.

Remarque: les instructions de catégorie 3, liées au contrôle du séquencement de l'algorithme, participent au temps total d'exécution pour plus de la moitié.

Cas des machines de type 2

Soit une telle machine constituée selon le modèle de ROMUALD, présenté au chapitre III, de 8 microprocesseurs (identiques à celui considéré ci-dessus). L'algorithme de seuillage, dans chacun des processeurs fonctionnant en parallèle, va s'écrire:

```
1        LDA     R1,I'     II' = 1/8° de I affecté à ce processeur
1        LD      R2,N'     IN' = N/8
2  E2:   LD      R3,@R1
2        CP      R3,S
3        JP      lt,E2
2        LD      @R1,SUP
3        JP      E3
2  E2:   LD      @R1,INF
3  E3:   INC     R1
3        DEC     R2
3        JP      nz,E1
```

Cet algorithme, tout à fait semblable au cas précédent, n'en diffère que par le nombre de points traités par chaque processeur. Avec les mêmes conventions que précédemment, on obtient les temps de calcul suivants:

Catégorie 1: 2t,
Catégorie 2: (3t)*N/8
Catégorie 3: (4t)*N/8

Soit un temps total par processeur de (7 (N/8) +2) t.

Si, comme ce sera justifié au chapitre III, les temps de mise en œuvre du parallélisme sont négligés, ce temps est le temps total nécessaire pour l'algorithme. La remarque faite précédemment concernant les instructions de catégorie 3 est toujours vérifiée.

Cas des machines de type 3:

Soit une telle machine, constituée d'une mémoire et d'un microprocesseur disposant d'un jeu d'instructions spécifique. Pour les besoins de l'exemple, mais sans entrer aucunement en conflit avec les possibilités technologiques et architecturales des microprocesseurs modernes, l'hypothèse que le jeu d'instructions de ce microprocesseur comprend l'instruction SIR : seuille, incrémente et répète qui admet 5 paramètres: SIR I,N,S,INF,SUP peut être faite. Cette instruction effectue N fois, à partir de l'adresse de I l'opération de seuillage.

A des fins de comparaison, il est raisonnable de penser que chaque

pas de cette instruction répétitive peut être effectué en un temps 2t, par exemple, ce qui donne un temps total de l'ordre de 2Nt.

Remarque: les instructions de catégorie 3, de contrôle du séquencement de l'algorithme ont disparu du programme et se trouvent maintenant prises en compte directement par l'unité de contrôle du microprocesseur. Tous les temps de lecture et de décodage de ces instructions sont ainsi supprimés.

Cas des machines de type 4:

Soit une telle machine constituée de N processeurs. A chaque processeur est associé un point de l'image. Dans ce cas, la commande de seuillage va être exprimée globalement par un ordre à 3 paramètres:

SEUIL S.INF.SUP

envoyé simultanément à chacun des N processeurs. Toujours à des fins de comparaison, il est raisonnable de penser que l'opération de seuillage va être effectuée en un temps total de l'ordre de t.

La figure 10 donne une représentation schématique des grandeurs obtenues en indiquant, pour chaque type de machine, la position relative des temps de seuillage en fonction de la taille de l'image.

L'ensemble de cet exemple doit être considéré avec prudence, comme une approche très grossière illustrant les possibilités de chaque type de machines. Une discussion plus sérieuse et plus complète ne saurait être entreprise qu'à condition de disposer de machines concrètes réalisées dans des technologies comparables et présentant, pour le cas des types 3 et 4, des jeux d'instructions spécifiques plus fournis que l'exemple d'une seule instruction utilisé ici.

Il n'est toutefois pas possible de passer sous silence le fait que cet exposé ainsi que le schéma de la figure 10 supposent que le matériel considéré a un temps de réponse indépendant de N, taille de l'image. Cela est, bien entendu, faux pour des variations importantes de N, singulièrement dans le cas des architectures spéciales. En particulier, pour des valeurs très grandes de N, les difficultés et les délais de routage à travers un ensemble matériel très important influencent grandement les temps de réponse. Il faut enfin noter que la possibilité

de réaliser des systèmes de très grande taille (par exemple un processeur matriciel de 1000x1000 processeurs élémentaires) est, pour l'heure au moins, tout à fait hypothétique.

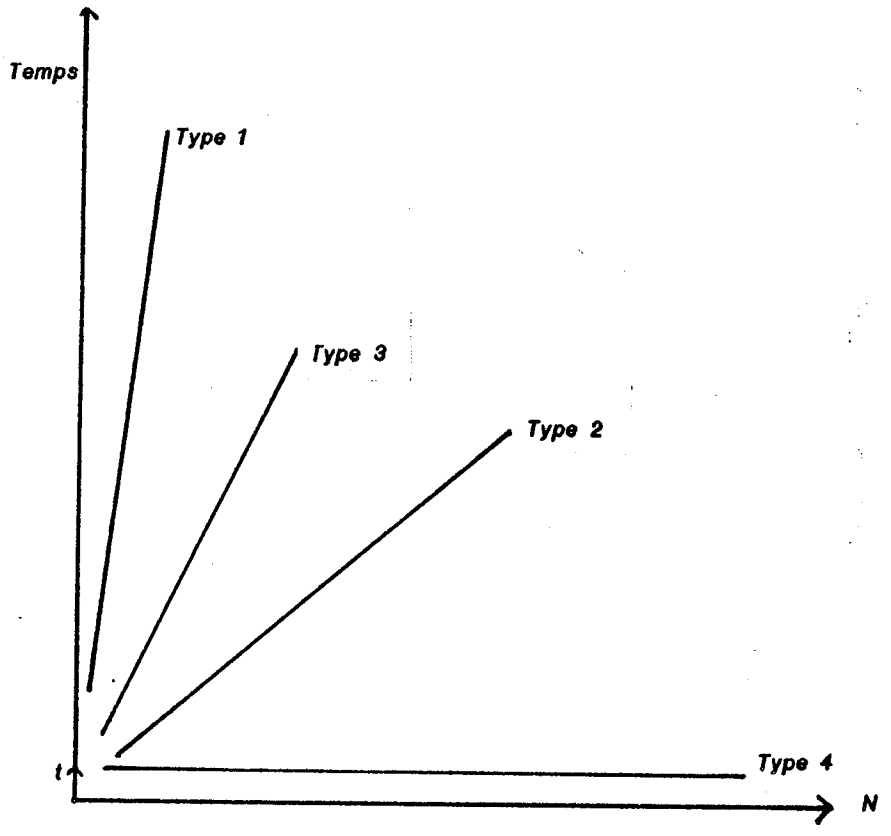


Figure 10.
Temps de seuillage en fonction
du type de machine et
de la taille de l'image

L'intérêt essentiel de cet exemple consiste à montrer l'important avantage qui peut être attendu grâce à l'étude d'un jeu d'instructions spécifique et à la réalisation des machines correspondantes.

En l'état actuel des systèmes de traitement d'images, deux problèmes essentiels restent largement ouverts.

Le premier concerne la définition de jeux d'instructions spécifiques pour le traitement d'images. En première analyse, ces jeux d'instructions doivent intégrer de façon harmonieuse au moins deux grands types d'instructions:

- des instructions répétitives (du type seuille, incrémente et répète) effectuant diverses opérations de base (comparaison, seuillage, addition, normalisation,...).
- des instructions travaillant sur des blocs de données. Ces blocs, par exemples des fenêtres, sont parfaitement définis par trois paramètres (centre, hauteur, largeur). De telles instructions doivent être largement paramétrables et effectuer toute sorte d'opérations du type convolution.

Le deuxième problème ouvert concerne la réalisation de systèmes matériels acceptant ces jeux d'instructions. C'est là que peuvent se poser les problèmes d'architecture les plus intéressants pour lesquels, par exemple, des solutions parallèles seraient choisies non plus simplement pour essayer *d'aller plus vite*, mais bien parce que les instructions à réaliser seraient effectivement parallèles.

II.5.5 Quelques exemples de systèmes de traitement d'images

Les exemples présentés dans ce paragraphe ont été, afin que cette thèse ne donne pas trop dans le penchant *somme bibliographique*, volontairement limités à quatre. De nombreuses autres réalisations sont citées en référence au cours de ce paragraphe et le lecteur intéressé pourra ainsi se reporter à une plus large information (voir aussi [25, 34, 53 et 105]).

Ces quatre exemples ont été choisis car ils sont une bonne illustration des préoccupations évoquées dans les paragraphes précédents. Trois d'entre eux (l'Analyseur d'Images, CLIP4 et GOP) sont bien connus et fréquemment cités en référence dans le domaine. Le quatrième, SCAPE, plus récent et en cours de réalisation, montre une originalité qu'il a semblé utile d'évoquer ici.

Ces quatre systèmes présentent, et c'est aussi ce qui les a fait retenir comme exemples, deux caractéristiques communes.

Premièrement, ces quatre réalisations trouvent leur origine dans une étude clairement explicitée de l'algorithmique des applications en traitement d'images. Parmi toutes les opérations possibles, le choix a été fait d'en retenir certaines comme fondamentales ou primitives. Ce choix est, en fait, de façon explicite ou non, celui d'une classe d'applications

mais aussi celui d'un langage et au-delà d'un jeu d'instructions. C'est la relative clarté de cette démarche qui a fait retenir ces quatre exemples.

Deuxièmement, l'architecture des systèmes matériels proposés est le résultat d'une analyse qui peut être présentée comme descendante, allant de la fonction à réaliser vers une solution qui en est, en quelque sorte, la traduction matérielle.

II.5.5.1 L'analyseur d'images

Une description de ce système est donnée dans [51] et [55] dont ce paragraphe s'inspire largement.

L'analyseur d'images a été conçu et construit au Centre de Morphologie Mathématique de l'Ecole des Mines de Paris sous la direction de J. Serra. Le cahier des charges choisi était de produire une machine susceptible d'aider les spécialistes en sciences naturelles (biologistes, géologues et métallurgistes particulièrement) à extraire des informations quantitatives de leurs spécimens.

Comme les images brutes ne permettent généralement pas des mesures, l'analyseur d'images effectue aussi des transformations d'images, pas nécessaire vers le but principal: la mesure. Les transformations effectuées ne sont pas quelconques et ont été largement étudiées afin de ne pas altérer les mesures. Connues sous le nom générique de "Morphologie Mathématique", ces transformations constituent un véritable corpus scientifique qui est largement présenté dans [86] et [66]. Ces études ont montré que, pour le cahier des charges fixé, l'analyseur d'images devait pouvoir réaliser les fonctions suivantes:

- saisir et seuiller des images,
- effectuer, point par point, des opérations logiques entre les images numérisées,
- mémoriser plusieurs images binaires,
- effectuer deux types de transformations:

  . la transformation "Premier Point", définie ainsi: l'image résultat contient, au plus, un point à 1 correspondant à la position dans l'image d'entrée du point à 1 le plus en haut à gauche. Cette transformation permet

de détecter des objets sur un fond.

. la transformation "de voisinage" qui admet comme paramètre une famille de masques binaires hexagonaux. Un point est mis à 1 dans l'image résultat si et seulement si le voisinage hexagonal qui l'entoure correspond à un des masques fournis en paramètre. Par exemple:

* le masque:

```
  1 1
 1 1 1
  1 1
```

est le paramètre pour effectuer une érosion.

* la famille de masques:

```
  . .
 1 1 1
  . .
```

où le point signifie "soit 1 soit 0" est le paramètre pour effectuer une érosion linéaire.

* si les masques sont de la forme:

```
  . .
 . 1 0
  . .
```

où le 0 peut circuler par rotation de $\pi/3$, ils pourront servir de paramètre pour une détection de contours.

- effectuer. Il s'agit là de mesures, des calculs numériques caractéristiques de propriétés géométriques (surface, périmètre,...) ou topologiques (nombre d'Euler,...).

Cette définition fonctionnelle de l'analyseur d'images se retrouve directement dans son architecture dont un diagramme est donné en figure 11.

Les objets à analyser sont captés à l'aide d'une caméra de télévision. Le signal analogique produit est seuillé puis échantillonné à une fréquence de 5 Mégaherz pour produire une image binaire organisée selon une grille hexagonale.

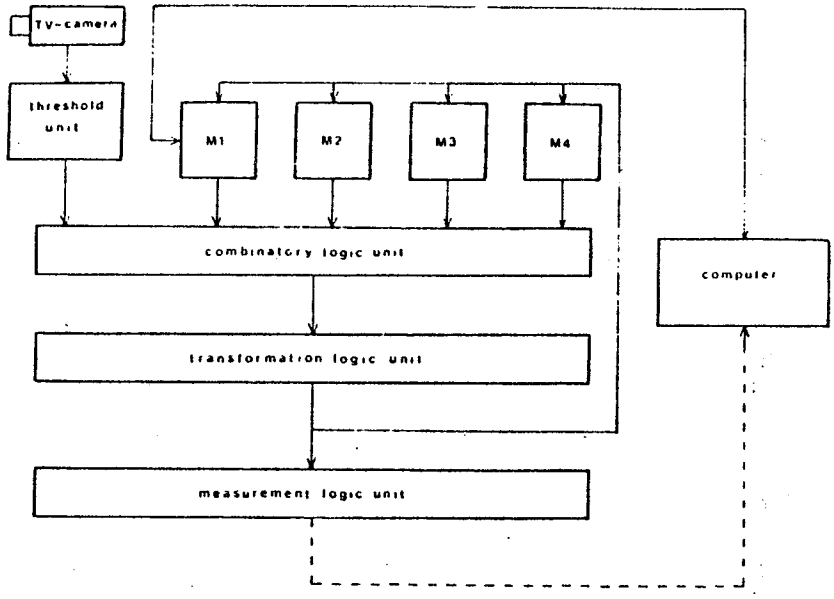


Figure 11.
Schéma de l'architecture de l'Analyseur d'images

Quatre plans mémoires (M1, M2, M3, M4) permettent de stocker des images de 256*256 bits. Les entrées de l'unité combinatoire sont les quatre plans mémoire et l'image seuillée. La sortie de cette unité est une image binaire combinaison booléenne, point par point, des images correspondant aux entrées sélectionnées.

L'image obtenue en sortie de l'unité combinatoire sert d'entrée à l'unité de transformation. Cette unité effectue l'une ou l'autre des transformations présentées ci-dessus. La durée d'une transformation est de 20 millisecondes. L'image résultat peut être rangée dans l'un ou plusieurs des 4 plans mémoire ou servir d'entrée à l'unité de mesure. La valeur numérique issue de cette unité est envoyée au calculateur gestionnaire.

Ce calculateur a quatre fonctions essentielles:

- calculs numériques sur les valeurs issues de l'unité de mesure,
- gestion d'une unité de disques, permettant des entrées/sorties avec le plan de mémoire M1,
- contrôle des différentes unités de l'analyseur,
- interprétation du langage d'application MORPHAL.

Ce langage, de haut niveau, a deux types d'instructions:

- des instructrions classiques arithmétiques, branchements, entrées/sorties (similaires à celles que l'on peut trouver en Fortran ou en Basic).
- des instructions spécifiques de traitement d'images qui permettent, à l'aide d'une syntaxe simple et claire, de programmer des applications en utilisant les fonctions présentées ci-dessus.

Ce langage est présenté dans [26] et, succintement dans [55].

L'analyseur d'images a connu plusieurs versions depuis 1970 et est commercialisé par LEITZ sous le nom d'ATS, depuis 1973.

II.5.5.2 CLIP4

CLIP4 (Cellular Logic Image Processor) a été réalisé à l'University College de Londres. Influencé par les travaux de Unger [101, 102], de Levialdi [61] et de Golay [36], ce système a fait l'objet de nombreuses publications. Ce paragraphe de présentation s'inspire largement de [29].

L'idée de départ est issue d'un domaine scientifique généralement appelé *logique cellulaire*, qui a guidé de nombreuses réalisations et dont une large présentation peut être trouvée dans [62, 78 et 83].

CLIP4 est un processeur matriciel pour les images. L'image est considérée comme une matrice carrée de points élémentaires. Une matrice de processeurs est assemblée dans laquelle il y a correspondance biunivoque entre processeurs et points élémentaires. La valeur (niveau de gris, par exemple) de chaque point est entrée dans le processeur correspondant sous forme d'un ou plusieurs bits. Des mémoires, permettant le stockage de ces valeurs et d'autres données, y compris les résultats, sont associées à chaque processeur de la matrice.

Comme le traitement d'images s'intéresse à leur structure, il est nettement insuffisant pour le fonctionnement de la matrice que le résultat issu d'un processeur soit une fonction de la seule valeur du point correspondant. Il doit exister un mécanisme permettant à chaque processeur d'accéder à d'autres valeurs dans la matrice. Un des résultats de la logique

cellulaire a été de montrer qu'en permettant la seule communication entre un processeur et ses huit voisins immédiats, il devient possible d'effectuer des fonctions sophistiquées permettant la réalisation d'applications en traitement d'images. Ces fonctions seront présentées dans la suite de ce paragraphe.

Le processeur matriciel se présente donc comme indiqué en figure 12.

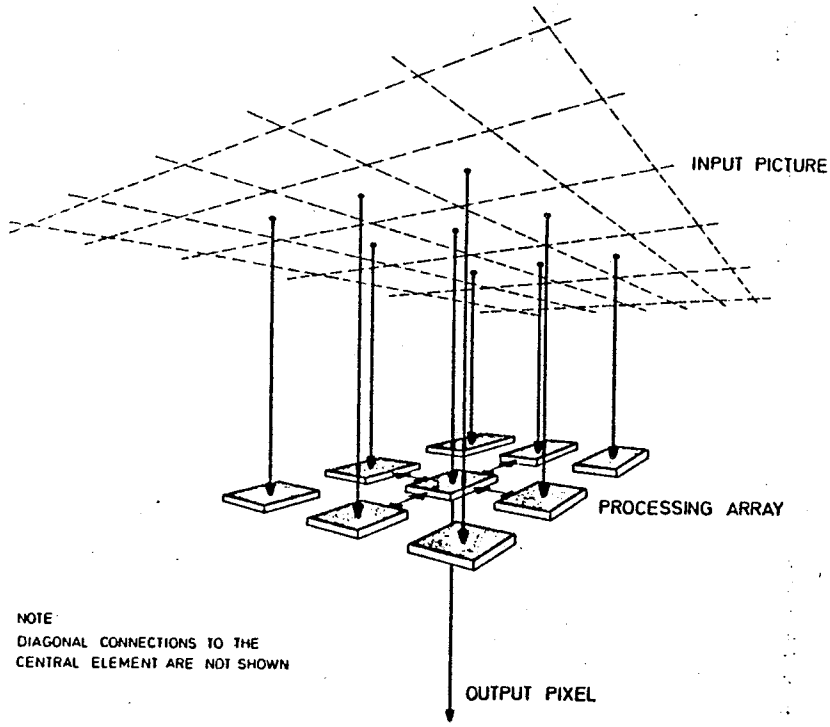


Figure 12.
Un processeur élémentaire et
ses connexions dans un processeur matriciel.

Cette organisation pose d'énormes problèmes de complexité dépendant en particulier:

- du nombre de bits choisis pour réaliser les connexions inter-processeurs.
- s'il est nécessaire d'avoir huit connexions différentes en entrée, est-il nécessaire que les huit connexions de sortie correspondent

à des résultats différents.
- la complexité de chaque processeur: largeur du mot (processeur booléen ou à 4, 8 bits), nature des opérations réalisées (booléennes, arithmétiques).

Par exemple, en limitant le nombre de bits de chaque connexion à 1 (images binaires), le schéma des connexions au processeur, booléen, est représenté en figure 13.

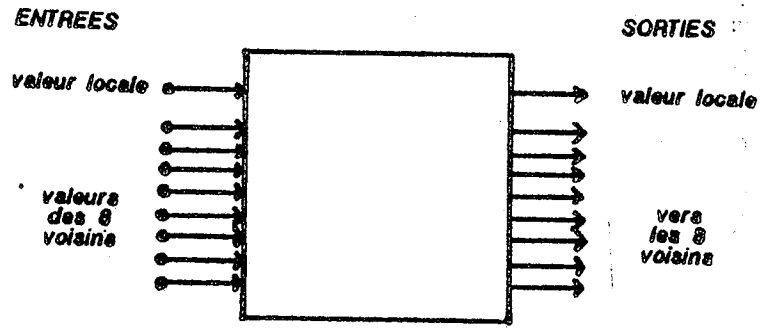


Figure 13.
Spécification d'un processeur booléen général dans une matrice.

Si chaque sortie est une fonction booléenne des 9 entrées, chaque processeur devrait être capable d'effectuer quelques 10^{1387} fonctions qui seraient sélectionnées par plus de 4600 lignes de contrôle! (voir à ce sujet [30]).

En fait, grâce à l'utilisation d'une mémoire interne au processeur, permettant de stocker des résultats intermédiaires, ces 10^{1387} fonctions peuvent être réalisées par des séquences d'opérations successives, tout en gardant un nombre raisonnable de lignes de contrôle.

Ce genre de problèmes étant posé, la conception de CLIP4 a été guidée par la définition des fonctions à réaliser. Pour chaque fonction, la structure du processeur booléen était étudiée, modifiée ou complétée. Dans l'article [29] déja cité, le lecteur trouvera une présentation très didactique de l'élaboration de l'architecture du processeur booléen de CLIP4, qu'il n'est pas possible de reprendre ici.

Les principales fonctions réalisées par la matrice de processeurs sont

les suivantes:

- fonctions booléennes de deux images (point par point: ET, OU, OUEX,...),
- décalage d'une image binaire d'une ou plusieurs positions dans l'une quelconque des huit directions de la matrice,
- opérations sur des voisinages locaux 3*3 (en référence à un masque 3*3, par exemple: érosion, dilatation, détection de contours,etc),
- opérations de propagation d'étiquettes (permettant d'isoler et de nommer un des objets de la scène),
- opérations arithmétiques: elles peuvent être menées soit en considérant que les bits des mots à additionner sont, pour chaque point, dans des plans successifs de la mémoire, soit que les bits à additionner sont dans des colonnes successives de la matrice.

Cet ensemble de fonctions est réalisé dans la matrice par des processeurs dont le schéma est donné en figure 14. Ce schéma nécessite les indications suivantes:

- les lignes terminées par un rond sont des lignes de contrôle,
- la partie "INPUT GATING" prend en entrée les valeurs issues des 8 voisins (via leur sortie N) et le registre C. Elle délivre en sortie la valeur du ET logique entre chaque entrée et la ligne de contrôle correspondante,
- A et B sont des registres tampons qui peuvent servir d'entrée au processeur booléen,
- la ligne de contrôle R sert lors des opérations arithmétiques à activer le système de retenue,
- les sorties D et N du processeur booléen sont des fonctions de ses entrées A et P sous le contrôle de 2 groupes de 4 lignes,
- D est la mémoire locale du processeur et est formée de 32 bits. Les lignes de contrôle de cette mémoire ne sont pas représentées.

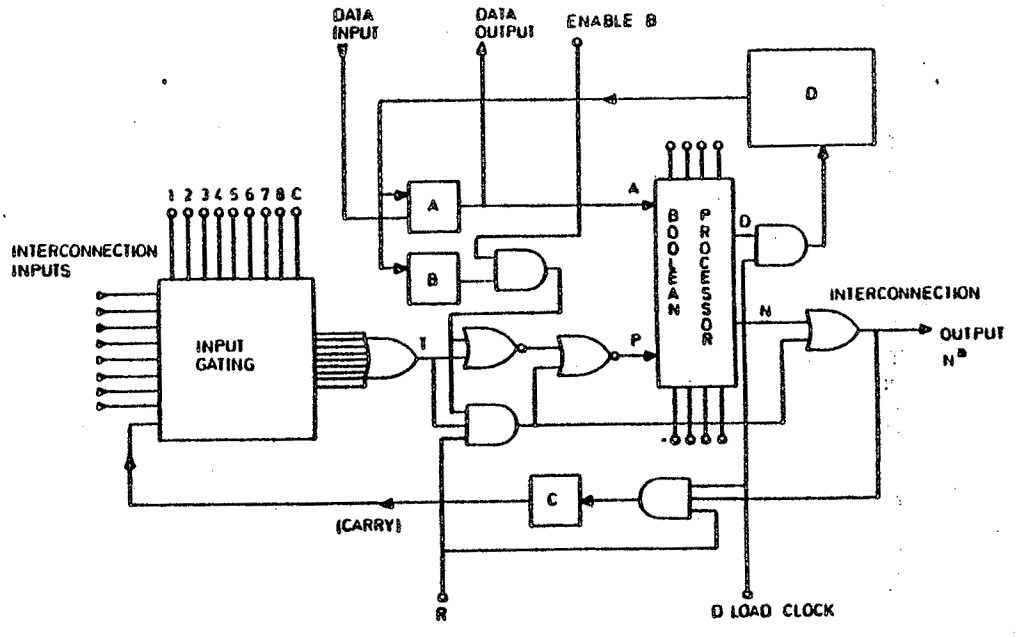


Figure 14.
Circuit logique d'un élément
de la matrice de CLIP4.

CLIP4 est formé d'une matrice de 96*96 processeurs booléens incluse dans un ensemble représenté en figure 15 qui permet de réaliser:

- les entrées/sorties d'images.
- le contrôle de la matrice.
- le comptage du nombre de bits à 1 dans une image (les registres A des éléments de la matrice sont en fait connectés par colonne pour former des registres à décalage; ces registres attaquent un arbre d'additionneurs).
- l'insertion de données dans les mémoires D de la matrice.
- des tests:
  - matrice vide: par un OU sur toutes les sorties D.
  - de propagation: en cas de propagation longue, le même mécanisme sur les sorties D, associé à un contrôle approprié du processeur booléen permet de tester la fin de propagation.

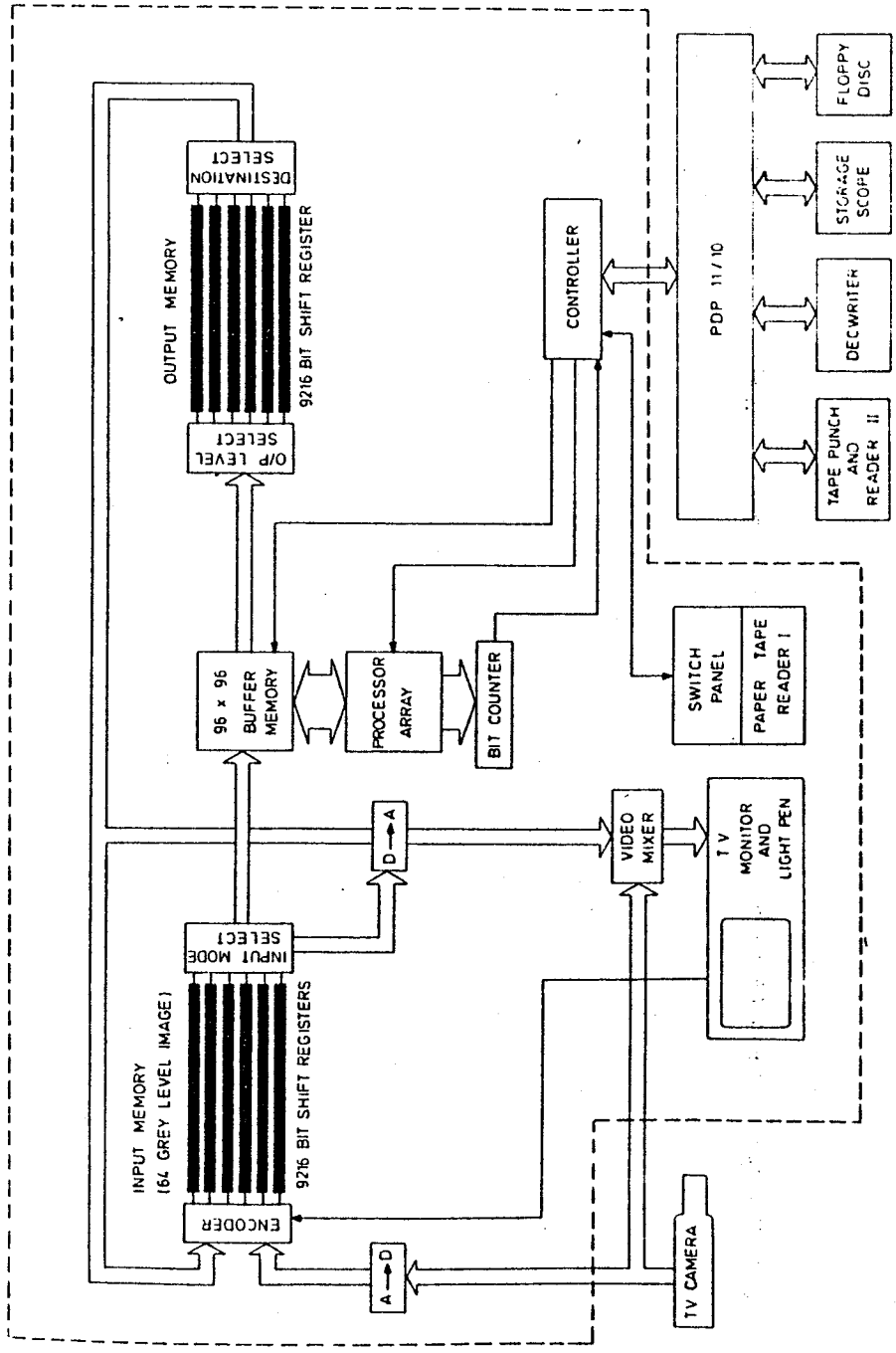


Figure 15.
Configuration du système CLIP4.

La matrice de processeurs de CLIP4 a été réalisée en circuits intégrés NMOS. Chaque circuit (40 pattes) contient 8 processeurs et leurs mémoires et fonctionne avec une horloge à 4 phases à la fréquence de 1 Mégaherz. CLIP4 est commercialisé depuis 1982.

II.5.5.3 GOP

GOP (General Operator Processor) a été développé à l'Université de Linköping en Suède, sous la direction de G.H. Grandlund. Le laboratoire de traitement d'images de cette université, très lié aux services de recherche de la Défense suédoise, est, pour le traitement d'images, un des plus importants et des plus actifs centres européens. Le laboratoire a, précédemment, réalisé un autre système de traitement d'images: PICAP [52].

GOP, de développement plus récent, est décrit dans [38, 40 et 41] qui ont inspiré ce paragraphe et s'inscrit dans un programme de recherche comprenant quatre grandes lignes:

- la représentation de l'information,
- opérations dépendantes du contexte,
- développement d'un processeur matériel,
- structures hiérarchiques de traitement.

La philosophie de conception de GOP consiste à faire franchir aux systèmes de traitement d'images un pas qualitatif important. En effet, les images, à niveau de gris ou colorées, pour peu qu'elles aient une résolution spatiale raisonnable, contiennent d'énormes quantités d'informations. L'analyse de telles images nécessite des traitements très importants. C'est pour cette raison que les systèmes de traitement d'images sont développés. Mais la voie classique consiste à simplifier le problème essentiellement de deux manières:

- orienter le processeur vers la réalisation d'opérations logiques sur des images binaires obtenues par seuillage sur des images originales,
- décider de mettre au second plan, voire de négliger, l'utilisation d'informations particulières (comme la texture) ou la réalisation de fonctions avancées (comme la segmentation dépendant du contexte ou la classification).

GOP a été conçu pour dépasser ces limitations et:

- travailler sur des images de n'importe quelle taille, à niveau de gris ou colorées,
- réaliser la segmentation d'images par des moyens plus précis que le seuillage, permettant la mise en évidence de détails plus fins dépendant de paramètres plus complexes (texture, couleur, orientation des contours,...),
- permettre la réalisation d'algorithmes dépendants du contexte par combinaison de traitements locaux et globaux,
- être adapté, aussi, à la réalisation des algorithmes de classification ou de relaxation, éventuellement contrôlés par des mécanismes de contre-réaction programmables,
- offrir des opérations arithmétiques diverses, en représentation fixe, flottante, complexe ou logarithmique avec une bonne précision,
- pouvoir être associé à tout ordinateur d'usage général,
- être très rapide (de l'ordre de 30 à 40 nanosecondes par point d'image pour des opérations non-triviales.

Tout ceci fait de GOP un système matériel dont la puissance et la complexité soutiennent largement la comparaison avec celles des récents ordinateurs ultra-rapides d'usage général. Du point de vue de l'utilisateur et en comparaison avec les autres systèmes de traitement d'images, GOP apparaît beaucoup plus puissant et largement mieux utilisable pour la réalisation de la quasi totalité des applications en traitement d'images.

D'une manière un peu abrupte, il est possible de dire que GOP est un processeur à une seule instruction. En effet, l'approche de l'équipe de Grandlund est de considérer que les opérations d'intérêt fondamental en traitement d'images sont du type suivant:

$$f = f(\alpha,\beta,\gamma, \ldots\ldots, \Sigma\Sigma_{ij} x_{ij} a_{ij}, \Sigma\Sigma_{ij} x_{ij} b_{ij}, \ldots)$$

Cette forme très générale inclut des opérations de type arithmétique aussi bien que logique. Le résultat est d'une part fonction d'un certain nombre de sommes de produits qui peuvent être des convolutions entre des masques (a_{ij}, b_{ij}) et des fenêtres dans l'image (x_{ij}). Le calcul de ces sommes de produits est généralement très long car il peut y avoir un très grand nombre de produits pour chaque somme. Cependant, le calcul est très régulier et ne nécessite que très peu de souplesse.

Le résultat est d'autre part fonction d'un certain nombre de paramètres α, β, γ, ... Ces paramètres permettent un très haut degré de non linéarité dans la procédure et sont, typiquement, des fonctions de plusieurs images ou des transformations de l'image ce qui signifie que leur valeur varie d'un point de l'image à un autre. Ces paramètres peuvent être combinés avec les sommes de produits pour réaliser une fonction particulière ou bien ils peuvent être des pointeurs sur différents programmes réalisant toute forme de contrôle ou de combinaison. Ces paramètres peuvent, par exemple, être utilisés pour contrôler un seuillage s'adaptant au contraste local ou un filtrage non isotropique selon une direction privilégiée.

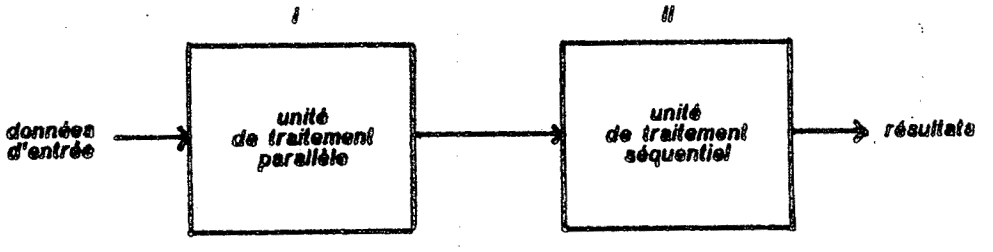


Figure 16.
GOP: architecture en deux parties.

Cette partie du calcul, dépendant de paramètres, nécessite une grande souplesse de la part du processeur. GOP a donc une architecture en deux parties, correspondant à ces deux types de calculs, qui est schématisée en figure 16.

Un schéma plus détaillé est présenté en figure 17. La première partie du processeur, à gauche sur le schéma, est un processeur à pipe-lines reconfigurables et parallèles. Les opérations *somme de produits* sont effectuées à partir de données issues de la mémoire de segments d'images et de poids issus de la mémoire de masques, qui sont combinés dans quatre pipe-lines parallèles.

La mémoire de segments d'images, d'une capacité de 16K mots de 16 bits a une configuration essentiellement paramétrable. Il est possible de faire varier le nombre d'images (jusqu'à 16) et la longueur de chaque segment d'image. L'accès se fait par lignes d'image, la mémoire se présentant, pour chaque segment, comme un buffer circulaire.

La mémoire de masques, d'une capacité de 16K mots de 20 bits, peut, elle aussi, être configurée d'un grand nombre de manières.

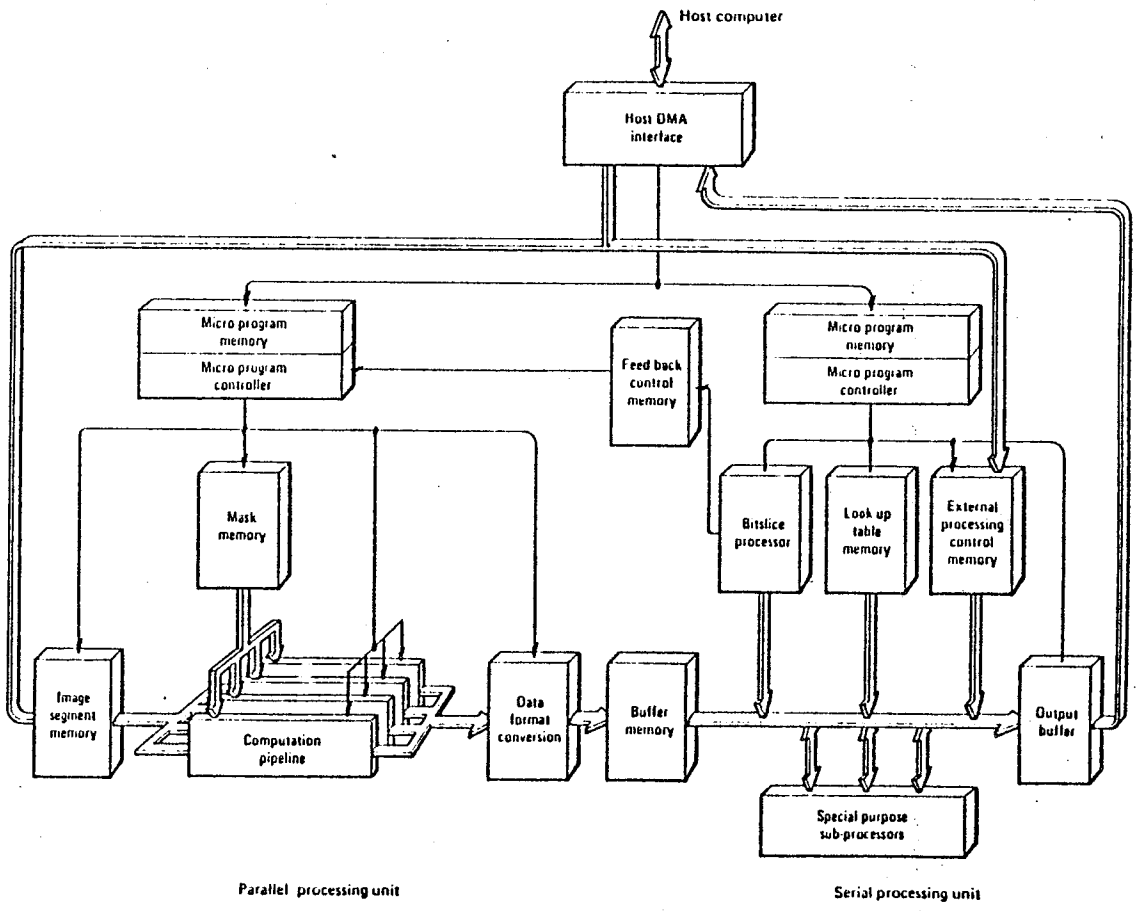


Figure 17.
Schéma détaillé du système GOP.

Les données issues de la mémoire d'images et de la mémoire de masques subissent, aux premiers étages des pipe-lines une conversion de modes. Chaque pipe-line, pour être efficace, travaille dans un seul mode: l'arithmétique entière, mais les données d'entrée peuvent être en différents modes:

- complexe: les 16 bits de données sont divisés en deux parties (exemple: magnitude et angle),
- deux images par mot,
- valeur sur 16 bits,
- images binaires.

Le plus grand soin est apporté dans les pipe-lines afin de ne pas perdre de précision dans les calculs. Ainsi, la sortie de chaque pipe-line présente des données sur 48 bits. Ces 48 bits sont à nouveau convertis sous différentes formes (entiers à 16 bits, valeurs logarithmiques sur 16 bits ou réels sur 32 bits) avant d'être transmis à la mémoire tampon qui assure la communication asynchrone entre les deux parties du processeur.

La deuxième partie du processeur, à droite sur le schéma, a une architecture totalement différente. Après traitement dans la première partie, la quantité d'informations est considérablement réduite. Maintenant, par contre, un haut degré de souplesse est nécessaire pour combiner, par des opérations généralement non linéaires, les résultats intermédiaires issus de la première partie.

En conséquence, la deuxième partie est un processeur séquentiel ad hoc. Le cœur de cette partie est constitué d'un processeur en tranches associé à un contrôleur et à une mémoire de microprogrammes (2K mots de 64 bits). Une mémoire de 16 K mots de 16 bits est disponible pour des calculs tabulés. Un certain nombre de processeurs supplémentaires sont connectés sur le bus permettant, par exemple, des multiplications rapides, des opérations sur des réels, etc.

La partie I fonctionne presqu'entièrement sous le contrôle de la partie II via la mémoire de contrôle par contre-réaction (*feed-back*).

Un système de cette importance ne saurait être décrit complétement dans les quelques pages précédentes qui tracent cependant les grandes lignes de GOP.

Pour les mêmes raisons de taille, de complexité et de souplesse, un utilisateur potentiel risque d'être effrayé par le travail nécessaire pour programmer tout ce qui est programmable, donner des valeurs à tout ce qui est paramétrable, concevoir la circulation des données et du contrôle. C'est pourquoi GOP doit être connecté à un ordinateur hôte sur lequel fonctionne un ensemble de logiciels permettant à l'utilisateur un accès simplifié au système.

Parmi ces logiciels, INTRAC, est un langage interactif de haut niveau essentiellement écrit en Fortran. Son rôle est de donner un moyen interactif et commode pour combiner des modules d'application pré-codés

en faisant varier les paramètres.

L'utilisateur *moyen* ne doit pas avoir besoin d'utiliser les assembleurs permettant de produire du micro-code pour les parties I ou II du processeur.

GOP est commercialisé depuis 1982.

II.5.5.4 SCAPE

SCAPE (Single Chip Array Processing Element) est un circuit VLSI, en cours de réalisation dans le groupe d'Architecture d'Ordinateurs dirigé par R.M. Lea à Brunel University. Ce groupe est engagé dans un projet visant à spécifier et concevoir des modules micro-électroniques permettant, à faible coût, de réaliser une large classe d'algorithmes de traitement d'images en temps réel.

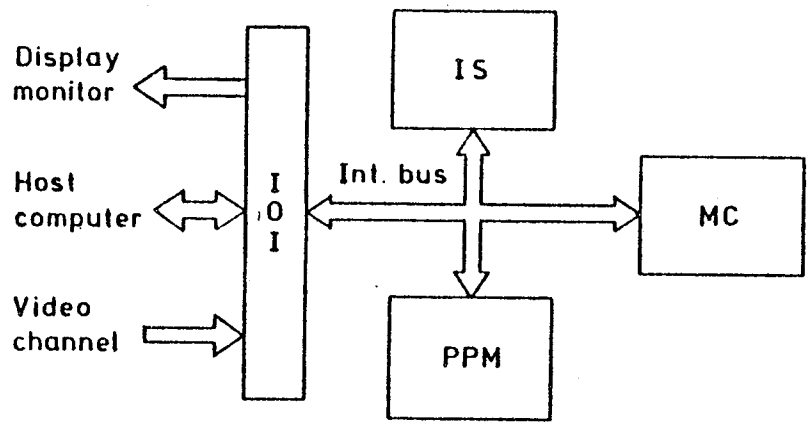


Figure 18.
Schéma d'un module typique.

La figure 18 présente le schéma d'un module typique qui comprend:

- une mémoire d'images de N*M points qui peut être partitionnée en pièces (*patches*) de n*m points (m= 16, 32, 64 ou 128 et N, M et n dépendent de l'application),
- un module de traitement de pièces (*Patch Processing Module: PPM*) qui va être présenté plus loin.

- un module d'entrées/sorties d'images et de communication avec un ordinateur hôte.
- un micro-calculateur contrôlant les transferts de données entre la mémoire d'images, le PPM et les entrées/sorties ainsi que l'exécution de programmes de traitement d'images.

Dans cette présentation, inspirée par [58, 59], l'accent va être mis sur la philosophie de conception, l'organisation structurelle et les principes de fonctionnement du PPM et du circuit VLSI SCAPE qui en est le composant majeur.

Contrairement aux précédentes, cette présentation sera peu détaillée. En effet, ce qui semble le plus intéressant à mettre en évidence, c'est d'étudier comment un groupe d'architecture d'ordinateurs plus particulièrement orienté vers la conception de circuits VLSI a marié les demandes fonctionnelles propres au traitement d'images avec le savoir-faire et les contraintes spécifiques de la conception de circuits intégrés, proposant une solution tout à fait originale.

Fonctionnellement, le PPM est identique aux processeurs matriciels comme CLIP4 présenté précédemment. Cependant, le choix a été fait de favoriser plus une diminution de coûts (de fabrication d'un système complet) que l'obtention d'une vitesse de traitement élevée.

En effet, si un processeur matriciel de m*n éléments exploite le parallélisme inhérent aux applications de traitement d'images en permettant une diminution du temps de calcul d'un rapport O(nm), son coût de réalisation effective subit un accroissement du même ordre, voire plus.

Le PPM se propose donc d'offrir la même réponse fonctionnelle que les processeurs matriciels mais à un plus faible coût, quitte à augmenter les temps de traitement. Ceci est la première clé du PPM: réaliser un système comprenant moins de circuits que les processeurs matriciels.

Ce problème de prix de revient intervient à nouveau dans les considérations suivantes. A première vue, les processeurs matriciels, organisés selon une topologie d'interconnexion entre plus proches voisins, semblent idéalement adaptés à la technologie VLSI. Or, dans un circuit intégré, si la surface occupée par la partie fonctionnelle est grande par rapport à celle occupée par les connexions (plots, amplificateurs, etc), le silicium disponible est *bien* utilisé et l'on a, en quelque sorte, un bon rapport

coût de fabrication/services rendus. Hélas, dans les circuits réalisés pour les processeurs matriciels, deux phénomènes concourent à une *mauvaise* utilisation du silicium: d'une part, la complexité des parties fonctionnelles proposées jusqu'ici est limitée, elles occupent donc une surface faible; par contre, d'autre part, l'organisation topologique de la matrice induit un grand nombre de connexions entre circuits et donc une grande surface de silicium qui *ne rend pas son maximum*. Par ailleurs, ce grand nombre de connexions conduit à une forte dissipation de puissance et à un coût d'assemblage élevé (supports, cablage, cartes, etc).

Améliorer la qualité de l'utilisation du silicium et diminuer le nombre de pattes des circuits est la seconde clé du PPM, présenté en figure 25.

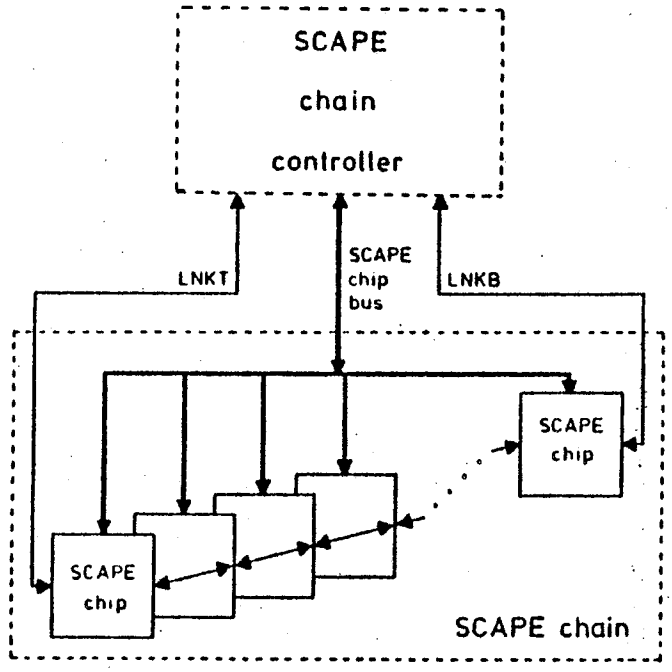


Figure 25.
Le module de traitement de pièces.

Bien que les circuits SCAPE du PPM puissent être *logiquement* configurés en matrice à deux dimensions avec interconnexion hexagonale ou orthogonale, ils sont *physiquement* disposés en chaîne linéaire reliée par un réseau linéaire de connexion. Ainsi, les cellules composant

les lignes de chaque pièce d'image sont liées bout à bout pour former une chaîne linéaire de cellules. chaque cellule est connectée directement au réseau d'interconnexion. Chaque cellule comprend de la mémoire de travail et une unité de traitement. La configuration logique de la chaîne en matrice est faite par le controleur de chaîne qui émet des ordres de traitement qui attaquent dans chaque cellule des mémoires associatives.

La présentation ci-dessus est très incomplète car d'une part, le projet est en cours de développement et les références citées donnent peu de précisions. d'autre part, la connaissance du fonctionnement des processeurs associatifs est peu répandue. ce qui nécessiterait d'en exposer le principe (le lecteur intéressé peut se reporter à [57]).

Ce qu'il a semblé intéressant de présenter ici est simplement l'originalité de la solution proposée. montrant qu'un dialogue fructueux est possible entre les domaines du traitement d'images et de la conception de circuits intégrés permettant parfois d'aboutir à des solutions qui sortent des sentiers battus.

II.6 Discussion

Ce chapitre ne présente pas les matériels de traitement d'images de façon exhaustive mais donne un aperçu général du domaine. L'accent est mis en particulier sur les unités de traitement. Bien qu'il soit clair, en effet, que toutes les applications en traitement d'images pourraient être menées à bien en utilisant des ordinateurs d'usage général classiques, il est clair aussi qu'elles le seraient dans des temps inacceptables tant du point de vue de l'opérateur humain que du point de vue des besoins.

Le traitement d'images peut être grossièrement caractérisé par la répétition d'un même calcul sur une structure de données régulière bien définie: une matrice de points d'images. Le nombre typique de répétitions peut varier de 10^4 à 10^7 (ce qui correspond au nombre de points dans l'image) alors que la quantité d'information pour une seule image varie de 10^6 à 10^9 bits et que la quantité de calcul peut être estimée à 10^3-10^4 opérations par point. Ainsi, un système capable d'exécuter de 1 à 100 milliards d'opérations par seconde est nécessaire pour obtenir des résultats *en temps réel* (vitesse vidéo, par exemple). Cela situe la demande de puissance en traitement d'images bien au-dessus -d'un facteur 100 à 1000- de celle disponible actuellement sur les super-ordinateurs d'usage général.

Pour répondre à cette demande de puissance, plusieurs solutions peuvent être envisagées et appliquées séparemment ou de façon coordonnée (sur tout ce domaine, voir [18, 49, 72 et 82]).

Une première classe de solutions, qui peuvent être appelées *matérielles* ou *architecturales au sens strict*, concerne l'organisation, la mise en relation, c'est à dire l'architecture au sens strict des objets matériels (essentiellement unités de traitement et mémoires) et des processus de contrôle associés. Ces solutions visent à modifier, pour aller plus vite, le modèle von Neumann de machines. Il a été montré [7] que ce qui limite la puissance des machines de ce modèle est essentiellement dû au goulot d'étranglement existant entre le processeur et la mémoire. Dans une machine classique, en effet, l'unité de traitement unique, connectée à une mémoire homogène, prend une instruction en mémoire (*instruction fetch*), la décode, l'exécute et répète ce cycle. Beaucoup d'instructions font des références supplémentaires à la mémoire pour en extraire des données ou y ranger des résultats. La performance

d'une telle machine dépend beaucoup de la vitesse à laquelle la mémoire peut être accédée, car, toujours dans le modèle classique, pendant ce temps, aucun traitement ne peut être effectué. Or le temps Ta d'accès à une mémoire n'est pas une constante technologique mais est lié à la taille de cette mémoire (en raison essentiellement de l'accroissement de longueur des connexions nécessaires et donc des temps de propagation le long de ces connexions). Mead et Conway ([68], paragraphe 8.2) proposent:

$$Ta = f(\sqrt{M})$$

où M est le nombre de bits de la mémoire.

Ce phénomène, peu sensible dans les petits systèmes, devient extrêmement préoccupant dans les cas où, comme en traitement d'images, la taille de mémoire est nécessairement très grande.

C'est pourquoi les solutions architecturales au sens strict visent à réduire l'effet de ce goulot d'étranglement en jouant sur différents paramètres:

- réduction ou élimination des accès mémoires tant pour les instructions que pour les données,
- modifier le mode d'accès *"un mot à la fois"* à la mémoire,
- accélérer les temps d'accès en diminuant la taille des mémoires.

Il est clair que si l'on peut jouer sur ces paramètres, le temps de calcul va diminuer; c'est ce que montre, de façon informelle, l'exemple illustrant le paragraphe II.5.4.

C'est dans ce but que des machines multiprocesseurs ont été conçues. Contrairement, en effet, à une idée reçue, les systèmes multiprocesseurs, dans le cas général et particulièrement en traitement d'images, permettent une amélioration des temps de calcul en agissant de façon positive sur le goulot d'étranglement processeur-mémoire, typique des architectures classiques, bien plus qu'en augmentant la capacité de traitement disponible. Ce raisonnement peut être soutenu par l'expérience. En effet, d'une part, les rares systèmes proposés présentant une mémoire unique connectée à plusieurs processeurs ont rapidement montré leurs limites (de 3 à 10 processeurs connectés à la même mémoire et l'accédant de façon plus ou moins harmonieuse par multiplexage temporel ou tout autre moyen dont le fonctionnement est dû à la nature des algorithmes exécutés

qui accèdent à la mémoire selon une loi de répartition qui, modélisée, permet de déterminer le nombre optimum de processeurs).
Ces systèmes ont rapidement évolué vers des systèmes à (au moins) deux niveaux de mémoire (une locale à chaque processeur et l'autre commune à tous) permettant l'exécution efficace d'algorithmes spécialement conçus pour limiter les accès à la mémoire commune; il s'agit alors essentiellement d'une action sur les paramètres cités plus haut.

D'autre part, parmi les grands systèmes multiprocesseurs de traitement d'images existants, il ne semble pas que l'augmentation de la capacité de traitement soit primordiale si l'on considère que dans la plupart des réalisations, chaque processeur élémentaire est un microprocesseur classique voire, le plus souvent, un processeur booléen très primitif fonctionnant avec des temps de cycle très éloignés des possibilités de la technologie (par exemple, CLIP4 fonctionne avec des processeurs booléens contrôlés par une horloge à 4 phases de fréquence 1 Mégaherz).

Ceci étant, reste à examiner comment les multiprocesseurs agissent sur le goulot d'étranglement processeur-mémoire.

Une première stratégie consiste à associer un processeur à chaque point de l'image. Ainsi, localement, le temps d'accès à la donnée est réduit au minimum. Ainsi, globalement, tous les processeurs accèdent à la donnée qui leur est associée dans un même intervalle de temps. Le temps de calcul dépend alors uniquement de la complexité du calcul et non plus de la taille de l'image. Cette stratégie est celle des processeurs matriciels.

Une deuxième stratégie consiste à mettre en place dans chaque processeur, et avant tout traitement, une ou plusieurs instructions. Cette stratégie, du type *pipe-line*, élimine ou réduit considérablement la phase d'*"instruction fetch"*. Les données nécessaires sont alors fournies à la chaîne de processeurs de façon séquentielle. Ainsi, le goulot d'étranglement est supprimé à la fois pour l'accès aux instructions et pour l'accès aux données. Dans ce cas, si le délai initial de propagation du premier point d'image à travers la chaîne de processeurs est négligé, le temps de calcul dépend linéairement du nombre de points dans l'image et non plus de la complexité du calcul.

Il est bien sûr possible de proposer des solutions qui combinent ces deux stratégies, le but ultime étant toujours de fournir au bon moment

la bonne donnée au processeur exécutant la bonne instruction.

Toutes ces solutions appellent quelques remarques. Premièrement, elles n'ont été rendues possibles que grâce à l'essor de la technologie VLSI autorisant la réalisation de configurations riches à haute densité et à coût acceptable, mais qui imposent des contraintes, en particulier sur le nombre de connexions entre circuits, influençant largement les réalisations. Deuxièmement, les machines construites sont peu ou pas souples et ne traitent correctement qu'une classe bien définie de problèmes. Troisièmement; enfin, la fiabilité de ces machines est faible compte-tenu d'une part de leur taille, d'autre part de leur tolérance le plus souvent nulle aux pannes et aussi de l'absence de mécanismes, voire de méthodes, de test.

Une deuxième classe de solutions, qui peuvent être appelées *systémiques* ou *architecturales au sens large*, concerne les liens entre les aspects logiciels et les aspects matériels [104]).

Que l'utilisateur dispose d'abord d'une machine ou bien qu'il veuille réaliser une application, le problème est d'arriver à une bonne adéquation entre un algorithme et une machine.

Cela signifie que n'importe quel algorithme n'est pas viable sur n'importe quelle machine. Mais cela ne suffit pas: ainsi, une bonne adéquation étant trouvée entre un algorithme et une machine (en termes de représentation des données et de fonctions primitives, par exemple), il faut encore qu'il puisse y avoir une bonne correspondance entre la façon dont l'algorithme est écrit et la façon dont la machine est commandée. Cela implique que le fossé logiciel-matériel existant entre un langage de programmation commode et un langage d'instructions efficace soit aussi restreint que possible.

Aux douceurs syntaxiques près, il n'y a, en effet, pas de meilleur langage pour utiliser une machine que son langage d'instructions (la situation des ordinateurs d'usage général n'est, de ce point de vue, qu'un blocage historico-économique qui semble, hélas, être appelé à durer, d'autant plus qu'aucune solution de rechange raisonnable ne semble être proposée).

Une approche systémique ou architecturale au sens large, du traitement d'images doit être de concevoir et de construire des machines qui réalisent le plus fidèlement possible une classe d'algorithmes. Un gain

de temps appréciable peut être attendu d'une telle démarche qui permet d'éviter à l'utilisateur la manipulation souvent inefficace d'outils plus ou moins généraux. Cette approche influence non seulement l'architecture matérielle de la machine et son jeu d'instructions mais aussi la définition d'un langage de programmation.

Certes, une approche systémique n'est pas totalement libre. Sa progression est en effet limitée par un grand nombre de contraintes parmi lesquelles celles imposées par l'architecture matérielle des machines efficaces réalisables (comme exposé précédemment) sont certainement les plus importantes.

En traitement d'images, des architectures matérielles et logicielles ont été définies. Une approche systémique, ou architecturale au sens large, suggère des architectures de systèmes. Le travail du concepteur de système est alors de spécifier les composants matériels et logiciels d'un système de base et, surtout, leur interface.

S'il n'y a pas correspondance entre les composants matériels et logiciels, de nombreuses difficultés apparaissent dans le système total. Ceci est particulièrement clair dans les cas où langages et machines, parallèles ou séquentiels sont mélangés. Par exemple, imposer un langage de programmation séquentiel à un processeur parallèle conduit généralement à l'échec, même si le langage permet, en plus de ses instructions séquentielles quelques opérations ou appels de procédures parallèles. Dans ce cas, le système est en fait un système séquentiel avec des compléments permettant l'exécution efficace de quelques tâches, et la structure particulière du processeur n'est pas complètement utilisée.

D'un autre côté, si un langage adapté à une exécution parallèle est implémenté sur une machine séquentielle, l'efficacité du traitement à l'exécution décroît considérablement en raison du temps nécessaire pour l'interprétation. Ceci peut être partiellement arrangé en remplaçant l'interpréteur par un compilateur. Mais alors, l'absence de correspondance logiciel-matériel sera sévèrement supportée par l'écrivain de compilateur dans sa tentative de surmonter le fossé architectural. Dans certains cas, le fossé sera tel qu'un compilateur fiable et efficace ne pourra pas être produit.

Bien sûr, il n'existe pas de machines ou de langages qui soient totalement séquentiels ou totalement parallèles: les ordinateurs séquentiels ont

des unités arithmétiques parallèles et les ordinateurs parallèles travaillent sur des séquences d'instructions. Mais la difficulté essentielle de l'approche systémique est d'essayer de définir des langages pour les nouvelles machines. Cette difficulté provient essentiellement de la distance existant entre les langages classiques bien connus et les architectures nouvelles.

La conclusion qui s'impose est que pour obtenir une efficacité optimale tant pour la compilation que pour l'exécution, un système complet doit assembler une machine et un langage qui soient aussi similaires, structurellement que le permettent les contraintes de souplesse.

La conception de tels systèmes complets est possible car langages et machines ne sont pas conceptuellement différents. En fait, un langage n'est qu'une machine logicielle et toute fonction matérielle peut être implémentée en logiciel et vice-versa. Ainsi, la répartition des fonctions entre logiciel et matériel est affaire de commodité et ne dépend pas d'un principe de base. L'essentiel est que cette répartition permette un interface le plus direct possible.

Les machines traditionnelles viennent conforter ce raisonnement. Bien qu'elles aient un niveau de matériel beaucoup trop bas pour correspondre aux niveaux de logiciel requis par les langages modernes, l'expérience montre que les langages les plus utilisés et les plus efficaces sur ces machines sont ceux dont la structure et le niveau reflètent le plus étroitement le modèle de machine de von Neumann, à savoir les assembleurs, Fortran, Basic, etc.

Les efforts de recherche en traitement d'images n'ont pas encore débouché sur des systèmes rassemblant un large consensus et la notion de *système général de traitement d'images* est encore bien mal définie et nécessite d'autres efforts. La chance du traitement d'images est qu'il s'agit d'un domaine relativement nouveau où les contraintes commerciales sont encore faibles vis à vis, de celles, par exemple, des fabricants de grands ordinateurs classiques. Dans les années à venir, une grande liberté sera encore possible pour expérimenter de nouvelles idées d'architecture de systèmes. Si cette liberté et cette chance ne sont pas exploitées rapidement, il ne sera plus possible de le faire lorsque le traitement d'images sera devenu une grosse affaire, avec ses contraintes économiques et l'inévitable résistance des utilisateurs au changement.

A l'heure actuelle, la communauté du traitement d'images prend

progressivement conscience de cette réalité et de nombreux systèmes matériels voient le jour. Le problème essentiel pour l'avenir est cependant celui de l'absence de bases en algorithmique et en langage, absence qui est la limitation majeure à l'utilisation productive des nouveaux matériels.

CHAPITRE III

Première réalisation: ROMUALD

III.1 Avant-propos

La machine ROMUALD, présentée dans ce chapitre, est le résultat d'un travail mené entre 1979 et 1981 [12, 13, 14, 15 et 48]. Premiers pas de l'auteur en traitement d'images, ROMUALD est un système très modeste par sa taille et ses ambitions - en terme de temps de calcul, notamment. Rapportée aux réflexions et aux exemples présentés au chapitre II, la conception de ROMUALD peut paraître élémentaire, voire naïve. D'un certain point de vue, elle l'est en effet, d'une part parce qu'elle répond à un cahier des charges aux exigences limitées, d'autre part parce que ROMUALD n'est pas un aboutissement mais un premier pas du travail présenté dans cette thèse qui, dans un second temps, s'est attaché à en étudier les conditions de dépassement.

III.2 Le cahier des charges

Le projet ROMUALD a été mené au sein du service d'étude et de développement de produits logiciels du CICG (Centre Interuniversitaire de Calcul de Grenoble). Il avait pour objectif général l'étude et le remplacement matériel et logiciel du système de saisie et de traitement d'images qui a fonctionné au centre de 1969 à 1976, date à laquelle il n'a plus été possible d'assurer la maintenance du matériel en raison d'une part de sa vétusté, d'autre part d'un accident électrique grave dont la réparation aurait été économiquement inacceptable pour le centre. Ce système, brièvement présenté en III.2.1 a été à la fois un exemple ou un modèle, mais aussi la source de réflexions critiques constructives résultant de l'expérience acquise lors de son utilisation passée. Ces réflexions, associées à des considérations portant sur les moyens disponibles et les besoins à satisfaire, ont conduit au cahier des charges de ROMUALD qui est présenté en III.2.2.

III.2.1 L'ancien système

L'ancien système de saisie et de traitement d'images a été installé en 1969 par T. Monnerot [69]. A cette époque, il s'agissait de la première connexion d'une caméra TV à un ordinateur réalisée en Europe. Par rapport à l'état de l'art et à la technologie utilisée, c'était une réalisation de première importance et il ne faudrait pas que les remarques critiques faites ci-dessous le fassent oublier. Si aujourd'hui il est possible de faire mieux, c'est d'une part, bien sûr, en raison de l'évolution de la technologie, mais aussi d'autre part parce que l'expérience acquise avec ce système et ceux de sa génération a permis de définir plus clairement les besoins.

La figure 20 présente un schéma des dispositifs d'acquisition, de traitement et d'analyse d'images qui existaient au CICG. Le capteur optique utilisé était une caméra de télévision à balayage en deux trames entrelacées. C'est un programme s'exécutant dans l'ordinateur PDP/8 qui commande la saisie de l'image. Pour cela, il existe un interface électronique ad hoc (réalisé par le LETI) qui, agissant en symbiose très étroite avec l'électronique du PDP/8, permet de saisir et de numériser une image sous forme d'une suite de colonnes. Chaque colonne est constituée pendant un balayage ligne par ligne en prélevant sur chaque ligne la valeur du point appartenant à la colonne en cours de constitution.

Il s'agit là, en quelque sorte, de la simulation d'une caméra lente dont le balayage s'effectuerait colonne par colonne. Il est ainsi possible d'acquérir une colonne en un balayage TV, soit 25 colonnes par seconde, et donc un temps de saisie de 14 secondes pour une image de 350 colonnes sur 624 lignes. Au fur et à mesure de l'acquisition, l'image était envoyée vers l'ordinateur IBM360/67 qui disposait des capacités de stockage et de traitement nécessaires.

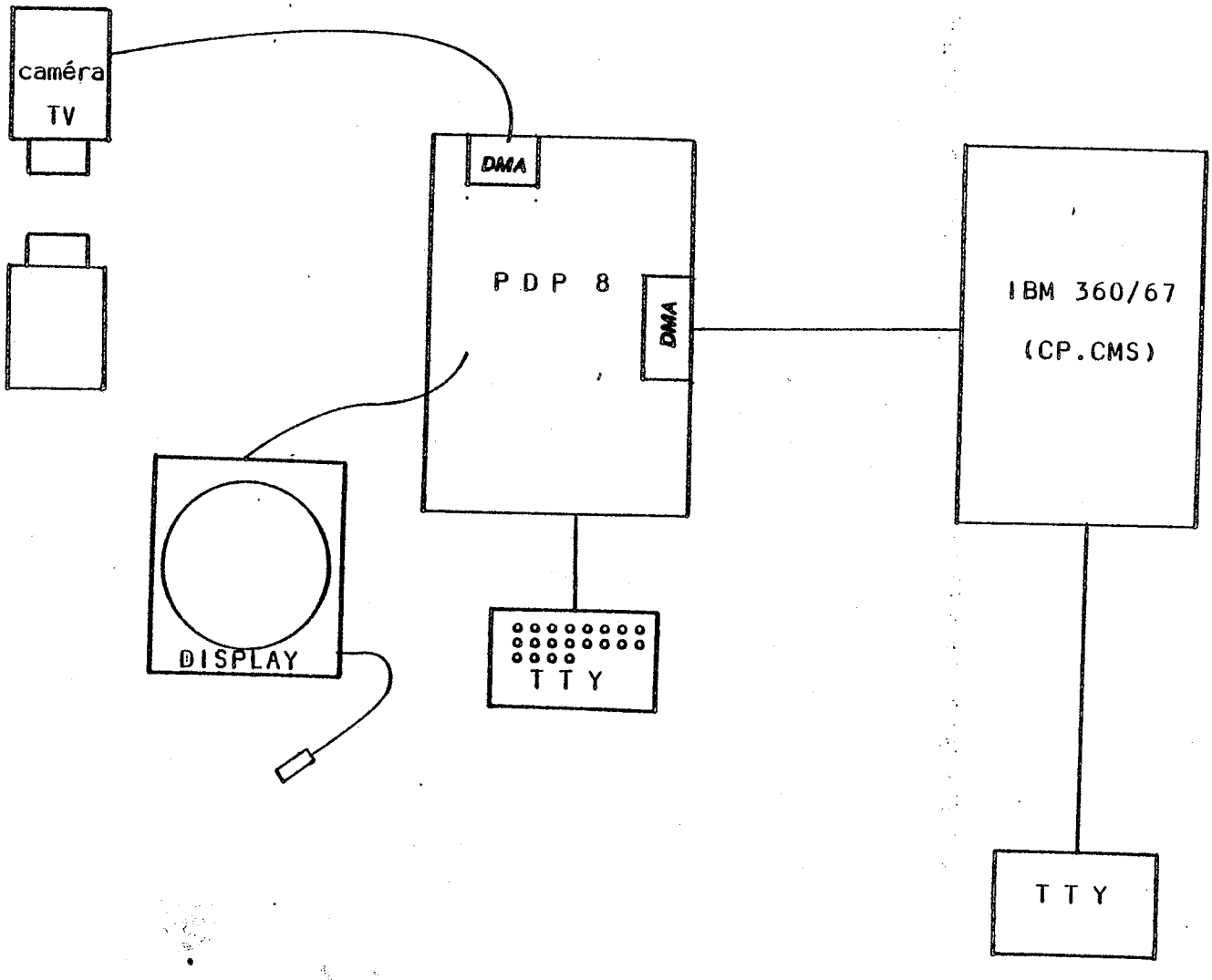


Figure 20.
Ancien système d'acquisition et de traitement d'images du CICG

Enfin, il était possible de faire l'opération inverse et d'afficher une image sur un tube cathodique. En raison de la lenteur des communications, l'affichage n'était utilisable qu'en faisant appel à fond à la rémanence de l'écran fluorescent ainsi qu'à celle que peut acquérir un œil bien entrainé.

Il était possible de choisir d'utiliser une seule trame donnant une image de 312*350 points ou deux trames (614*350 points). La caméra et l'électronique associée permettaient d'obtenir des valeurs stables d'intensité lumineuse réparties en huit niveaux de gris.

Concernant le traitement, l'ordinateur PDP/8 était pratiquement inutilisable car quasiment saturé par les tâches de saisie ou d'affichage et la gestion des communications. Le traitement était donc effectué sur l'IBM360/67 qui imposait au système ses contraintes propres (temps de réponse, coût d'utilisation) mais lui apportait ses avantages (interactivité, gestion de fichiers, langages, aide à la mise au point, etc).

L'utilisation de ce matériel a permis de dégager un certain nombre de critiques ou de souhaits concernant une nouvelle installation:

- la qualité de l'image est insuffisante: il faut arriver à une image de 512*512 points et en chaque point disposer d'un éventail plus large de niveaux de gris: 16, 32, 64 voire plus.
- la qualité de l'affichage est insuffisante: il faut pouvoir disposer d'une image affichée qui ait des qualités proches de celle saisie.
- l'ordinateur hôte est chargé d'effectuer presque tous les travaux et son utilisation devient coûteuse.
- le fonctionnement de l'ensemble est assez peu souple et lent.

Tout cet ensemble de souhaits à satisfaire, de conditions à remplir ou de buts à atteindre constituent un faisceau convergent de directions de travail et c'est sur ces bases que la construction de ROMUALD a été entreprise.

III.2.2. Le cahier des charges de ROMUALD

Le projet ROMUALD a été réalisé dans un important centre de calcul universitaire et l'analyse des besoins montrait que l'équipement à réaliser devait pouvoir être utilisé librement par une communauté

d'utilisateurs scientifiques pour la recherche ou l'expérimentation d'algorithmes de traitement. Les utilisateurs sont maintenant bien accoutumés aux systèmes conversationnels et demandent désormais un outil interactif aussi transparent que possible, excluant des procédures spécialisées ou sophistiquées. Le système de traitement d'images doit donc être aussi facile d'accès que n'importe quel autre service du centre de calcul.

L'utilisateur veut pouvoir décrire simplement (sans changer ses habitudes de programmation) un nouvel algorithme et observer ce qui arrive. Un temps de réponse de quelques secondes est donc adéquat. Il n'est pas nécessaire d'aller plus vite car la vocation d'un service de traitement d'images au CICG n'est pas, pour l'heure tout au moins, l'exploitation régulière de vastes ensembles de données images.

En somme, les contraintes étaient:

- être raisonnablement rapide, mais pas plus qu'un homme devant un terminal.
- éviter toute limitation sur les algorithmes pouvant être réalisés.
- exclure toute règle de programmation inhabituelle ou exotique.
- et, bien sûr, ne pas faire cher!

Dès lors, deux éléments de réponse s'imposaient:

- utiliser des microprocesseurs, universels, programmables de façon classique, peu coûteux.
- en utiliser plusieurs en parallèle de façon à permettre l'obtention d'un temps de réponse correct.

Cet énoncé appelle, en écho du chapitre II et de l'introduction de ce chapitre, deux remarques. Premièrement, le cahier des charges ci-dessus est tout à fait rudimentaire. Cela provient grandement d'une incertitude, d'une absence de définition, en terme de langage ou d'algorithmes, de ce que la machine allait avoir à réaliser. Deuxièmement, si naïveté il y a dans la définition de ROMUALD, cette naïveté provient sûrement du fait d'avoir accepté, sans autre examen que celui induit par une longue accoutumance aux machines et aux langages séquentiels classiques, que ROMUALD doive être utilisé sans changer les habitudes de programmation. Ces remarques étant faites, elles ne seront pas illustrées plus avant dans la description qui suit, le but de cette thèse

n'étant pas, aprés tout, de dénigrer un travail qui a montré lors de son utilisation qu'il rendait de façon satisfaisante les services qui en étaient attendus.

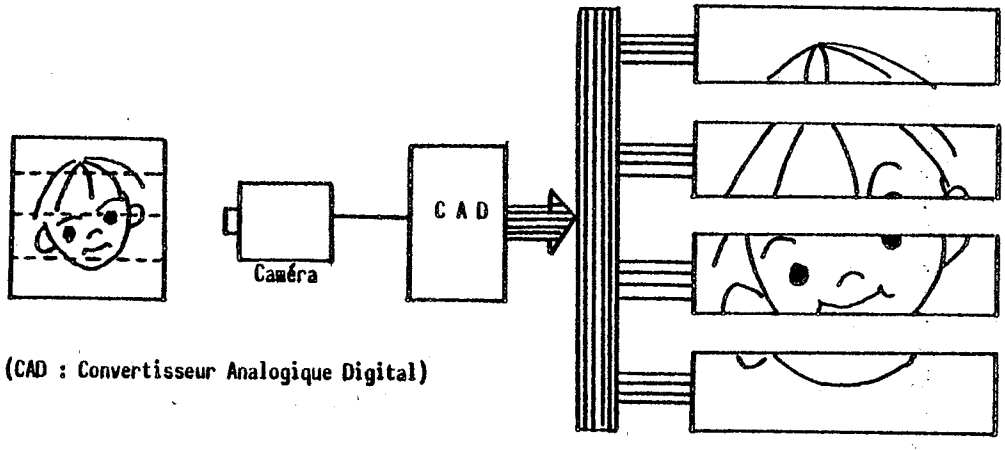
III.3. Architecture du système

Ce paragraphe se propose de décrire l'architecture au sens strict de ROMUALD, c'est à dire l'organisation de ses composants matériels et des processus de contrôle associés.

III.3.1. Principes généraux

Le choix étant fait d'utiliser des microprocesseurs et de permettre une diminution des temps de traitement en concevant un système multiprocesseurs, trois principes généraux restent à définir: l'organisation des liens entre mémoires et processeurs et particulièrement la façon de répartir les données image, le principe de contrôle des composants et la gestion des communications et enfin, l'organisation fonctionnelle de l'ensemble.

III.3.1.1. Organisation parallèle de la mémoire et des processeurs



(CAD : Convertisseur Analogique Digital)

Figure 21
Partition de l'image et projection en mémoire

La partition d'une image TV standard est aisée. Elle se divise, tout à fait naturellement, en tranches horizontales, chacune de ces tranches contenant un certain nombre de lignes adjacentes. L'image

entière peut être chargée dans une mémoire d'organisation matricielle qui apparaît être l'exacte projection de l'image et peut être partitionnée de la même manière. (voir figure 21)

A cette étape, une caractéristique très favorable du balayage TV apparaît: le balayage est un processus strictement sériel et le mieux qui puisse être espéré est de saisir une image entière dans le temps d'un balayage (40 millisecondes); ceci peut être réalisé assez aisément. Le problème de conception de l'architecture du système se résume à une seule question: << *Comment organiser les processeurs par rapport à la mémoire* ? >> La réponse devient, à ce stade de réflexion, assez simple et élimine naturellement une organisation multiprocesseurs classique (voir figure 22). En effet la possibilité de partitionner l'image conduit à relier un processeur à chaque tranche d'image en mémoire.

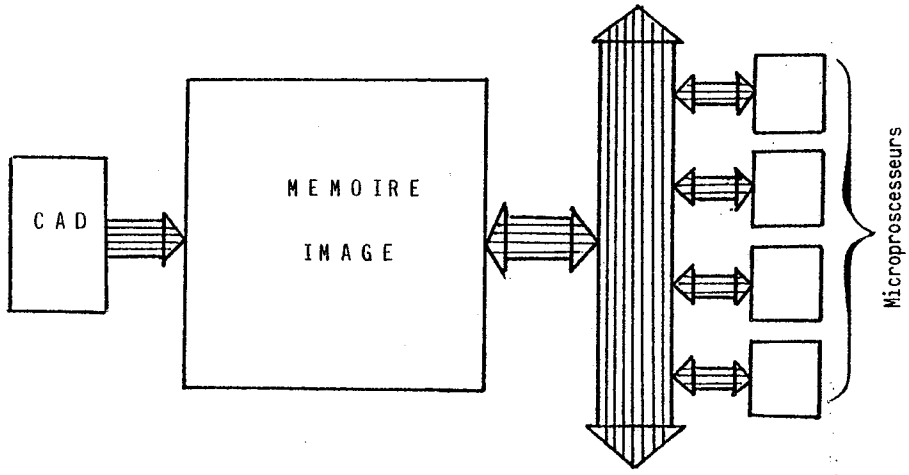


Figure 22
Organisation multiprocesseurs classique

Les problèmes d'accès concernant les programmes sont aisément résolus en les copiant dans des mémoires privées associées à chaque processeur (voir figure 23). Que se passe-t-il alors pour les problèmes d'accès à l'image?

Tout algorithme de voisinage va se trouver en difficulté dès lors qu'il opèrera près de la frontière d'une tranche où le voisinage déborde sur la tranche voisine.
La solution générale est de créer des voies de communications entre les tranches et de se trouver alors dans le cas, souvent décrit, de

liens, via une mémoire commune, par exemple, entre deux processus indépendants. Cette solution est apparue à la fois trop sophistiquée et trop restreinte vis °à vis des besoins et des intentions de départ et un processeur global, général, a été ajouté, capable d'adresser l'ensemble de l'image et permettant n'importe quel transfert de n'importe quel type entre des lieux quelconques. Il s'agit là, en effet, d'une mesure très générale et le mot *transfert* peut être compris d'une façon très extensive: exécuter du code situé n'importe où dans la mémoire, c'est aussi une espèce de transfert. De plus, n'importe comment, un tel processeur général est nécessaire.
Des solutions complémentaires peuvent être envisagées, par exemple, l'utilisation de tranches d'images partiellement redondantes, le recouvrement entre les tranches étant assuré par un chargement judicieux depuis le convertisseur analogique-numérique.

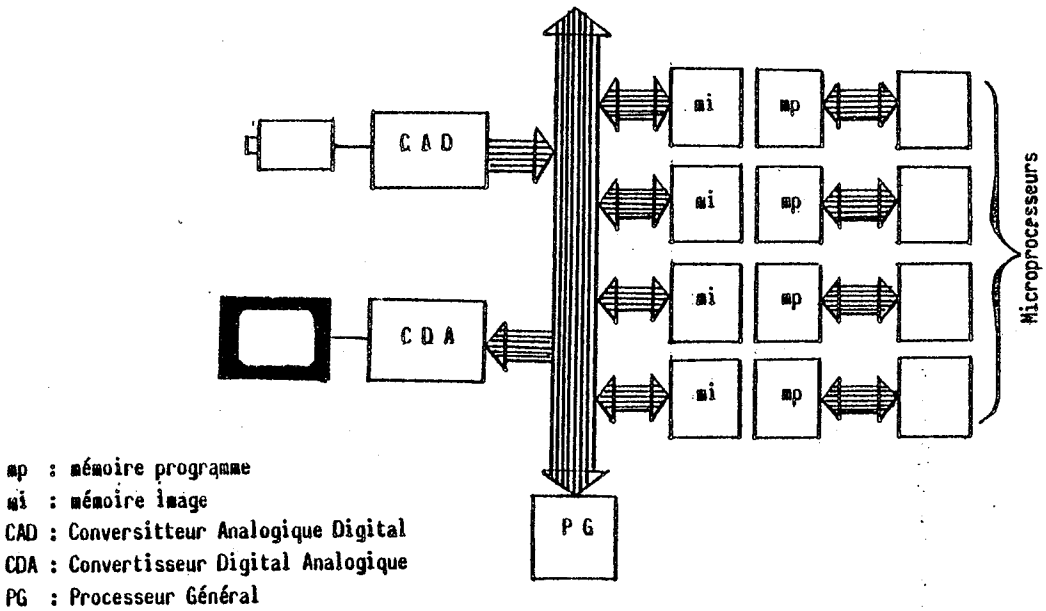


Figure 23
Principe de la solution proposée

III.3.1.2. Principe de contrôle et gestion des communications

Le schéma simple de la figure 23 ne peut fonctionner tel qu'il est présenté, avec des communications libres en tous points. Il faut fournir un système de construction de chemins de données et décider quels moyens employer pour cela. Comme le suggèrent de nombreux bons auteurs, une grande partie peut être faite - essentiellement en

logiciel - par des méthodes de *poignées de main, sémaphores, boîtes à lettres* ou toute autre variété d'*échanges polis*. Ces systèmes sophistiqués nécessitent, à chaque accès, des dizaines de microsecondes d'échanges logiciels. La durée de ces échanges est directement fonction de la plus ou moins grande dépendance mutuelle des processeurs partageant la mémoire.

L'option choisie vise à réduire au maximum ces échanges logiciels, en mettant en place un système *"tout ou rien"*. Quand le processeur local est actif, il est entièrement maître de sa mémoire, qu'il ne libère qu'en fin de traitement. L'organisation logicielle du système est telle que le processeur général n'utilise pas cette mémoire lorsque le processeur local est actif. Cela revient à affecter la mémoire à l'un des processeurs pour des séries d'accès fonctionnellement et temporellement bien définis. Les échanges logiciels n'ont lieu qu'entre ces séries d'accès et se limitent à une fonction de synchronisation du processeur général et de processeurs locaux basée sur une modification/test de leur état.

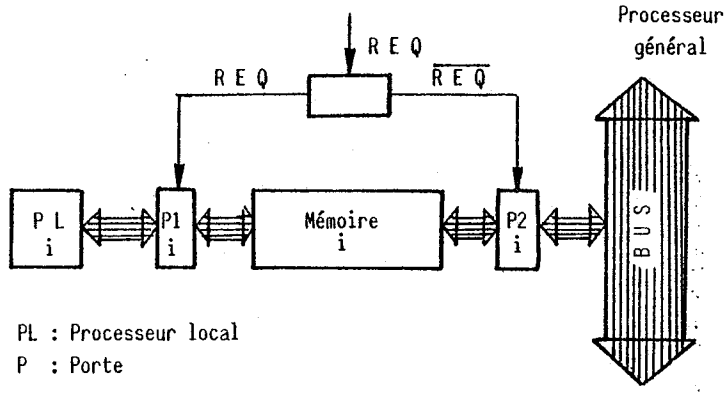


Figure 24
Le mécanisme de portes.

La figure 24 montre un exemple très simple d'une telle organisation. Les données contenues dans la mémoire i peuvent être utilisées et/ou modifiées par le processeur local i (porte P1 ouverte, porte P2 fermée) ou par le monde extérieur (porte P1 fermée, porte P2 ouverte). Les mécanismes matériel et logiciel nécessaires pour créer le signal REQuête sont très simples et seront décrits plus avant dans ce chapitre.

III.3.1.3. Architecture fonctionnelle

Fonctionnellement, quatre unités logiques constituent la machine ROMUALD:

- une unité de saisie, qui convertit le signal TV et range l'image numérisée en mémoire,
- une unité d'affichage, fort utile lors de la mise au point interactive des traitements, qui prend l'image numérisée et traitée pour la convertir en un signal TV transmis à l'écran d'affichage,
- une unité de traitement, qui effectue sur l'image les opérations demandées,
- une unité de commande, qui assure, d'une part le contrôle et l'ordonnancement des tâches confiées aux trois autres unités, et qui permet d'autre part, via un terminal, le dialogue interactif avec l'utilisateur.

Cette structure fonctionnelle peut être modifiée par adjonction ou suppression d'unités logiques: liaison rapide avec un ordinateur de grande puissance, suppression de l'unité d'affichage, etc...

ROMUALD se présente donc, en première approche, comme un système hiérarchique réparti fonctionnellement très classique. En fait, la véritable originalité de ROMUALD se trouve dans l'architecture de l'unité de traitement et dans l'organisation de ses liens avec l'unité de commande.

III.3.2. Architecture matérielle

Elle est présentée sous forme d'un schéma en figure 25. Ce paragraphe souligne quelques points importants ou remarquables. La structure et le fonctionnement particulier du processeur d'entrées-sorties d'images sont présentés en III.5.3.

III.3.2.1 Le mégabus

Cette architecture s'organise autour d'un *mégabus* par lequel transitent toutes les informations circulant dans le système. Toutes les unités opératives lui sont connectées. Hormis quatre octets de données en parallèle, le mégabus véhicule des adresses et des signaux de

contrôle.

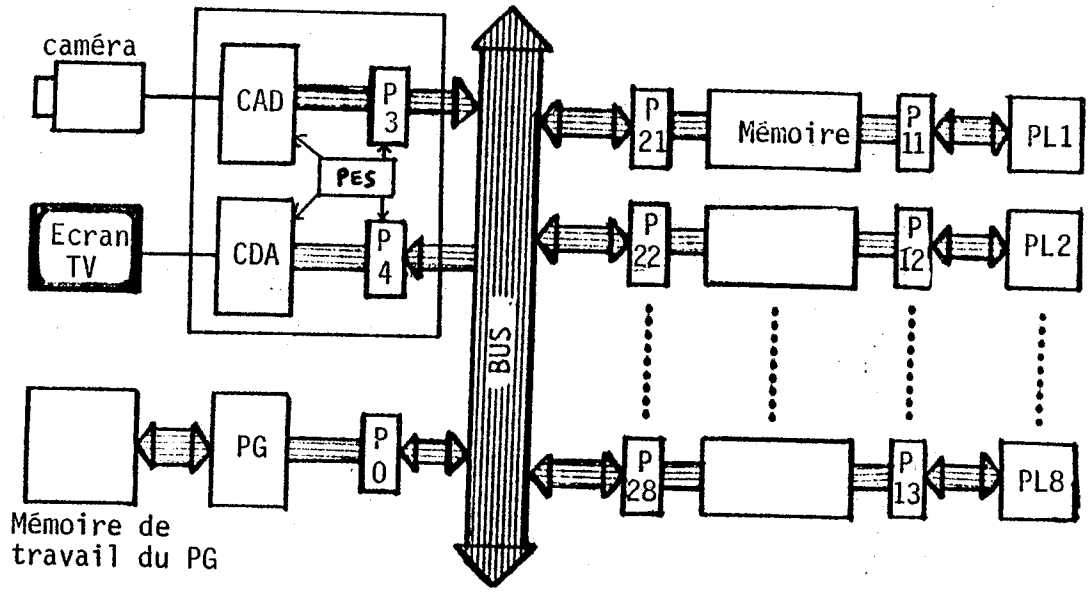


Figure 25.
ROMUALD: architecture matérielle

PES: processeur d'entrée-sortie d'images
CAD: convertisseur analogique-digital
CDA: convertisseur digital-analogique
P: porte
PG: processeur général
PL: processeur local

III.3.2.2 L'unité de traitement

Comme le montre la figure 25, elle est constituée de huit parties appelées par la suite: micro-unités de traitement. Elles sont physiquement séparées et chacune comprend:

- une mémoire, dite *mémoire image*, contenant une partie de l'image (par exemple, 64 lignes consécutives),
- un processeur de traitement,
- une mémoire, dite *mémoire de traitement*, contenant le code nécessaire aux traitements, les zones de travail, une zone de communication avec l'unité de commande.

- un système de portes, permettant d'isoler la micro-unité de traitement du reste de la machine et de reconnaître parmi les informations circulant sur le mégabus celles qui s'adressent à cette micro-unité de traitement.

III.3.2.3 Le système d'adressage

Le système d'adressage à l'intérieur de la machine est simple et évolué:

- A l'intérieur de chaque micro-unité de traitement, le processeur peut adresser la mémoire d'image et de traitement de la micro-unité de traitement, et seulement elle.
- L'unité de commande peut adresser d'une part sa mémoire locale, d'autre part l'ensemble des mémoires d'image et de traitement des micro-unités de traitement. Pour cela, les adresses circulant sur le mégabus sont organisées en deux parties: une partie indiquant le nom de la micro-unité de traitement à laquelle on s'adresse, une deuxième partie correspondant à l'adresse à l'intérieur de la micro-unité de traitement.

Ce sont les systèmes de portes qui réalisent le décodage des noms circulant sur le mégabus en les comparant au nom contenu dans chaque porte et attribué directement par l'unité de commande sans passer par le mégabus. Pour des raisons de clarté du schéma, ceci n'est pas représenté sur la figure 25. Ce mécanisme permet, comme la suite le montrera, d'accélérer considérablement certains échanges entre l'unité de commande et la micro-unité de traitement.

III.3.2.4 Les signaux de contrôle

Ils sont de quatre types:

- les signaux relatifs au fonctionnement des mémoires: bus prêt, lecture/écriture, etc... Ils sont compris dans le mégabus.
- un signal d'interruption qui, associé à la partie nom d'unité de l'adresse et circulant, comme elle, sur le mégabus, permet selon son état d'arrêter ou de démarrer l'unité ou la micro-unité de traitement considérée.

- une ligne de transmission de commandes, reliant l'unité de traitement au processeur d'entrées-sorties d'images et permettant de paramétrer son fonctionnement.
- un signal d'état, transmis directement par le processeur d'entrées-sorties d'images à l'unité de commande.

III.3.3. Système de contrôle et de communications

Les communications à l'intérieur de la machine sont toutes réalisées grâce à l'architecture matérielle et au système d'adressage présenté ci-dessus. Ce sont:

- les communications internes de chaque micro-unité de traitement,
- les communications entre la mémoire image de l'unité de traitement et le processeur d'entrées-sorties d'images qui peut, vis à vis du contrôle, être conçu comme composé de deux unités: une d'affichage, l'autre de saisie,
- les communications entre l'unité de commande et la mémoire de traitement de l'unité de traitement. Elles seront détaillées plus loin.

Le système de contrôle, à la mise sous tension de la machine la met dans l'état initial suivant:

- les unités de saisie et d'affichage sont à l'arrêt,
- chacune des micro-unités de traitement est à l'arrêt et les portes sur le mégabus sont ouvertes,
- l'unité de commande commence à exécuter un programme préchargé.

Le système de contrôle va être présenté en décrivant le fonctionnement de la machine pour la série suivante de commandes: saisie d'une image, traitement, affichage, ... , fin. Ces commandes sont suffisantes pour comprendre le mécanisme de contrôle. Leur ordre est indifférent vis à vis de ce mécanisme. Les moniteurs de contrôle des unités de commande, de traitement, de saisie et d'affichage, sont donnés ci-dessous, sous forme algorithmique.

** ALGORITHME A1 : unité de commande **

```
DEBUT
   execute = 'vrai' ;
   TANTQUE execute FAIRE;
      LIRE (commande);
      CAS commande DANS
saisir :
         DEBUT
            SI affichage ALORS affichage = 'faux';
                                 int ( unité_d_affichage )
            FINSI;
            charger ( unité_de_saisie );
            noms_différents;
            int ( unité_de_saisie );
            attente (fin_de_saisie)
         FIN;
afficher :
         SI affichage ALORS SKIP
                      SINON charger (unité_d_affichage);
                            noms_différents;
                            affichage = 'vrai';
                            int (unité_d_affichage)
         FINSI;
traitement :
         DEBUT
            SI affichage ALORS affichage = 'faux';
                                 int (unité_d_affichage)
            FINSI;
            nom_unique;
            copie_code; copie_parms;
            int (nom_unique);
            noms_différents;
            en_cours = 'vrai';
            TANTQUE en_cours FAIRE;
               en_cours = 'faux';
               POUR i = 1 JUSQUA 8 FAIRE;
                  SI actif(nom(i))
                  ALORS en_cours = 'vrai' FINSI
               FINFAIRE
            FINFAIRE;
            copier_résultats
         FIN;


fin :
         DEBUT
            SI affichage ALORS affichage = 'faux';
                                 int (unité_d_affichage)
            FINSI;
            execute = 'faux'
         FIN;
   FINCAS;
```

```
    FINFAIRE;
FIN;
```

Les précisions suivantes sont nécessaires pour comprendre cet algorithme:

- charger (nom_d_unité) : charge le mot de commande de l'unité désignée.
- int (nom_d_unité) : envoie une interruption à l'unité désignée.
- noms_unique : attribue à toutes les micro-unités de traitement le même nom.
- nom_différents : attribue à chacune des micro-unités de traitement un nom différent, respectivement U1,U2,...U8.
- attente (fin_de_saisie) : met la machine de commande en attente jusqu'à l'arrivée du signal de fin de saisie. Après quoi, l'exécution reprend en séquence.
- copie_code, copie_parms : chargent respectivement le code et les paramètres dans l'unité de traitement. Toutes les micro-unités de traitement sont chargées en parallèle grâce à l'utilisation du nom unique.
- nom : est un tableau de 8 éléments contenant les valeurs U1, U2, ... U8.
- actif (nom_d_unité) : est un mot dans la mémoire de traitement de l'unité spécifiée.
- copier_résultats : prend, si nécessaire le résultat du traitement dans chacune des micro-unités de traitement et compose le résultat final.

Les algorithmes suivants sont exécutés chaque fois que l'unité correspondante reçoit une interruption.

```
** ALGORITHME A2 Micro-unité de traitement **

    SI état ='arrêt' ALORS
       fermer (porte); état = 'actif';
       actif = 'vrai';
       execute_code;
       actif = 'faux'
    FINSI;
    état = 'arrêt';
    ouvrir (porte);
    halte;
```

```
** ALGORITHME A3 Unité de saisie **

        saisir_une_image;
        envoyer (fin_de_saisie);

** ALGORITHME A4 Unité d'affichage **

        SI état = 'arrêt' ALORS
           état = 'actif';
           TANTQUE 'vrai' FAIRE afficher FINFAIRE;
        SINON
           état = 'arrêt'
        FINSI;
```

Le mécanisme de contrôle utilise donc trois types d'outils:

- les interruptions.
- un mécanisme matériel installé dans les portes de chaque micro-unité de traitement.
- un ensemble de logiciels.

Cette présentation doit être précisée par trois remarques complémentaires. Il est clair que les algorithmes A2, A3, A4, se référant à l'état de leur unité et le modifiant, comportent des sections critiques. L'ensemble de la machine étant sous le contrôle de l'unité de commande, c'est l'algorithme A1 qui, en envoyant des interruptions vers les autres unités à des moments choisis et contrôlés, assure leur bon fonctionnement. La machine devant fonctionner de façon interactive sous le contrôle d'un opérateur humain, le problème de la sûreté de fonctionnement n'est pas crucial. Cependant, il est facile de compléter l'algorithme A1 en introduisant des mécanismes de *chiens de garde* lors des phases d'attente. Il est possible ainsi de reprendre le contrôle en cas de non-réponse de l'une des unités.
Afin de mettre en évidence le mécanisme de contrôle, la phase de lancement de l'unité de traitement a été simplifiée et sera détaillée au paragraphe suivant.

III.3.4. Parallélisme et répartition du traitement

L'adaptation des algorithmes de traitement d'images à la structure de ROMUALD pose des problèmes qui sont essentiellement de trois types:

III.3.4.1 Copies du code et des variables globales

La nécessité pour chaque micro-unité de traitement de disposer du code de l'algorithme à exécuter et d'un certain nombre de variables globales conduit à opérer des copies. Cela pourrait être très coûteux en temps. En fait, la quantité de code et de variables globales à copier est très faible. Par ailleurs, le mécanisme de chargement en parallèle des micro-unités de traitement permet de n'effectuer qu'une copie et non huit.

III.3.4.2 Algorithmes délivrant un résultat

Ce problème va être illustré par un exemple: le calcul de l'histogramme des niveaux de gris. Cet algorithme parcourt entièrement l'image et compte, pour chaque valeur de niveau de gris, le nombre de points ayant cette valeur. Un algorithme non parallèle peut s'écrire:

```
POUR i = 1 JUSQUA taille_de_l_image FAIRE
   hist(image(i)) = hist(image(i)) + 1
FINFAIRE;
```

Si l'image est répartie dans la machine ROMUALD, il est clair que chaque micro-unité de traitement va exécuter un tel algorithme sur sa partie d'image. En fin d'exécution, il y aura, dans chacune des huit mémoires de traitement, un tableau hist et il faudra alors réaliser l'addition vectorielle de ces huit tableaux pour obtenir un histogramme sur l'ensemble de l'image.

Dans la structure de la machine ROMUALD, c'est l'unité de commande qui a accès à tous ces tableaux et compose le résultat final.

III.3.4.3 Discontinuité de l'image du fait de la répartition

Certains algorithmes de traitement d'images effectuent en chaque point un traitement qui fait appel aux valeurs des points voisins. Ce voisinage est toujours limité et ce paragraphe examine, à l'aide d'un exemple, comment le problème se pose et peut être résolu.

Le calcul du gradient en chaque point d'une image est utilisé pour

mettre en évidence les points formant les contours des formes contenues dans l'image: la valeur du gradient en un point représente la variation de niveau de gris autour de ce point. Si l'image est une matrice carrée de 512 x 512, le calcul de la valeur du gradient au point j de la ligne i utilisera ses voisins. Cette valeur peut être approchée par un calcul simple utilisant ses huit voisins immédiats:

```
g (i,j) =    abs(image(i-1,j-1) - image(i+1,j-1))
           + 2 abs(image(i-1, j ) - image(i+1, j ))
           + abs(image(i-1,j+1) - image(i+1,j+1))
           + abs(image(i-1,j-1) - image(i-1,j+1))
           + 2 abs(image( i ,j-1) - image( i ,j+1))
           + abs(image(i+1,j-1) - image(i+1,j+1))
```

L'algorithme calculant la valeur de g en chaque point de l'image peut s'écrire:

```
s2 = 0;
   POUR i= 2 JUSQUA 511 FAIRE;
      POUR j = 1 JUSQUA 512 FAIRE;
         s1 (j) = g (i,j)
      FINFAIRE;
      image (i-2) = s2;
      s2 = s1
   FINFAIRE;
image (511) = s2; image (512) = 0;
```

où s1 et s2 sont des vecteurs de travail de 512 éléments et où le résultat vient remplacer l'image initiale.

L'image étant répartie dans la machine ROMUALD, pour calculer la valeur de g en chaque point, il faudra fournir à chaque micro-unité de traitement les valeurs des points contenus dans la ligne qui suit et dans la ligne qui précède la partie de l'image affectée à cette micro-unité de traitement.

Une bonne organisation des calculs permet de déterminer, avant le traitement, quelles sont les zones concernées. Après quoi, et toujours avant le traitement, le processeur général effectue des copies de ces zones d'un processeur local sur un autre. Le traitement peut alors démarrer, l'algorithme disposant de toutes les données qui lui sont nécessaires.

Une solution complémentaire est installée dans ROMUALD. Le processeur d'entrées-sorties d'images peut, lors de la saisie d'une image, écrire en parallèle dans les mémoires de deux micro-unités de traitement et ce à des adresses différentes. Les zones de recouvrement prévisibles sont ainsi préchargées lors de la saisie et disponibles pour le premier algorithme qui suit. Après expérience et utilisation, cette solution apparaît comme un investissement matériel de faible intérêt très peu utilisé dans la pratique.

III.3.4.4 Répartition du traitement

Les points III.3.4.2 et III.3.4.3 montrent que l'unité de traitement telle qu'elle a été considérée jusqu'à présent, ne peut pas toujours, à elle seule, réaliser l'ensemble des traitements demandés. Une intervention de l'unité de commande est parfois nécessaire, soit pour composer le résultat final, soit pour surmonter les problèmes que crée la discontinuité de l'image due à sa répartition. Les deux unités doivent donc collaborer. Pour cela, les algorithmes de traitement doivent être adaptés à la structure de la machine devenant ainsi des algorithmes répartis.
Le rôle pris dans le traitement par l'unité de commande doit être étudié avec soin. En effet, c'est un facteur de ralentissement de l'ensemble du traitement qui peut devenir important. Cette étude fait l'objet du paragraphe III.4.

III.4. Comportement du système

Ce paragraphe fournit des éléments permettant d'apprécier le comportement de ROMUALD considéré comme une configuration répartie spécialisée dans le traitement d'images. Pour cela les variations des composantes suivantes ont été étudiées:

- le nombre de micro-unités de traitement,
- les performances du système, en terme de temps nécessaire à l'exécution d'un algorithme donné,
- le coût total du système.

III.4.1 Etude des performances du système en fonction du nombre de micro-unités de traitement

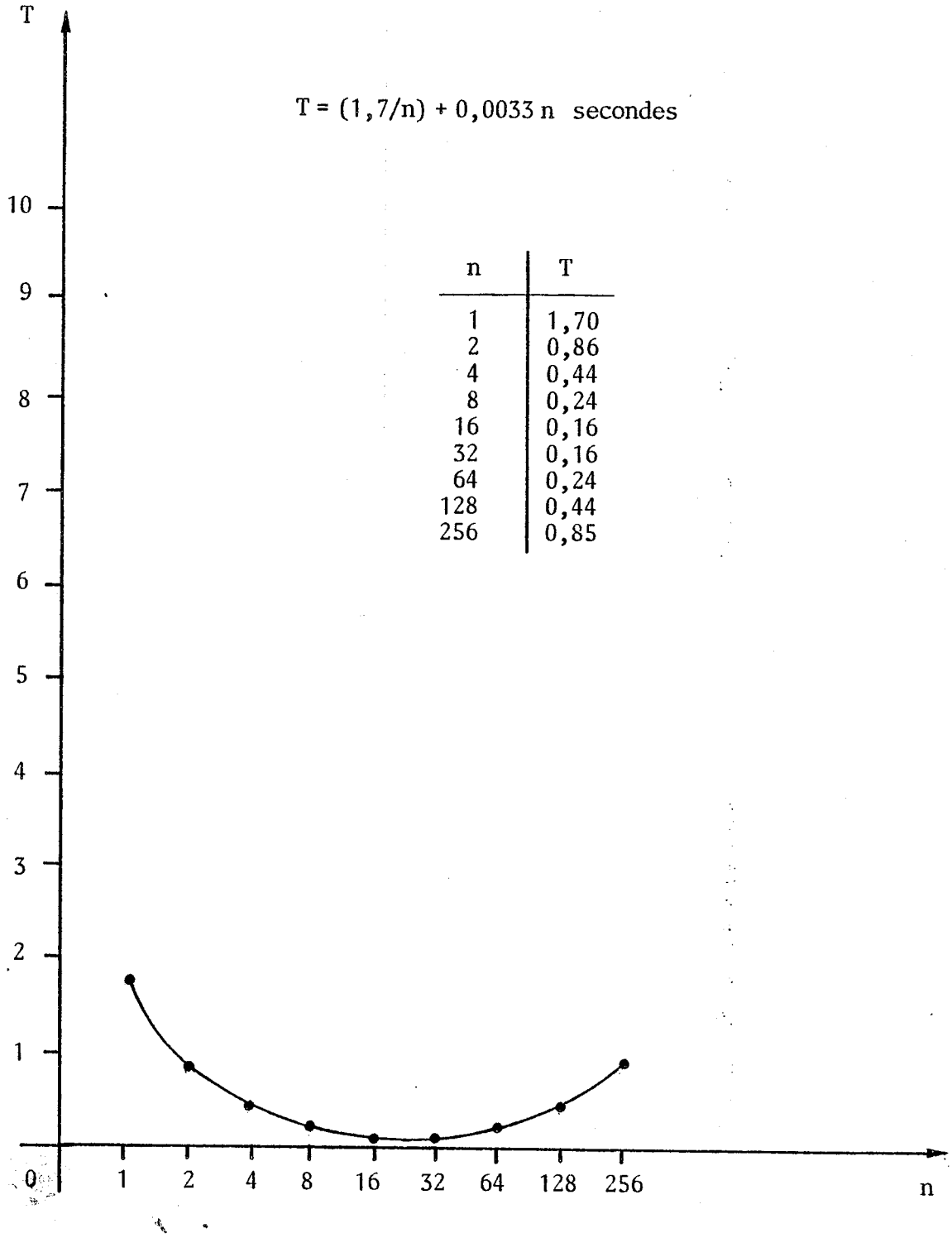
ROMUALD étant une machine spécialisée, il est normal, pour mener cette étude de se baser exclusivement sur le comportement du système lors de l'exécution d'algorithmes de traitement d'images. Ces algorithmes sont très ressemblants et peuvent être décrits ainsi:

- l'algorithme parcourt l'image un petit nombre de fois (généralement une fois),
- en chaque point, il effectue un petit nombre d'instructions (entre 5 et 50, par exemple),
- le résultat est soit une transformation de l'image, point par point, soit un petit nombre de valeurs.

Dans un système non réparti, le temps total nécessaire à l'exécution de l'un de ces algorithmes peut être évalué à l'aide de la fonction:

$$T = K\ N + Co$$

où N est le nombre de points élémentaires de l'image et K = Q1 + Q2 + ... + Qi + ... + Qn avec n = nombre de parcours de l'image et Qi = temps nécessaire pour effectuer les opérations du parcours i. Co est le temps nécessaire pour mettre en œuvre l'algorithme.



n	T
1	1,70
2	0,86
4	0,44
8	0,24
16	0,16
32	0,16
64	0,24
128	0,44
256	0,85

Figure 26

Temps de traitement (en secondes) en fonction du nombre de micro-unités de traitement, pour le calcul d'un histogramme. Processeurs: Z8000.

Dans l'architecture répartie de ROMUALD, le même algorithme va s'exécuter de façon tout à fait semblable dans chacune des micro-unités de traitement. Il faudra de plus tenir compte de la part prise par l'unité de commande dans le traitement (cf. III.3.4.4). Les exemples donnés en III.4.4.2 et III.4.4.3 montrent que cette part dépend à la fois de l'algorithme et du nombre de micro-unités de traitement.

Ainsi, dans le cas d'un traitement réparti, le temps total nécessaire à l'exécution d'un algorithme type de traitement d'images peut être évalué à l'aide de la fonction

$$T = K (N / n) + k n + Co$$

où K, N et Co ont les mêmes significations que précédemment, n désigne le nombre de micro-unités de traitement et k le temps mis par l'unité de commande pour assurer, parmi sa part du traitement celle afférente à une micro-unité.

L'étude détaillée des deux algorithmes cités a permis de donner des valeurs significatives aux constantes K, k et Co. Ainsi, pour le calcul de l'histogramme (cf III.3.4.2) effectué sur des microprocesseurs Zilog Z8000:

K = 39 cycles de 166,66 nanosecondes
k = 1975 cycles de 166,66 nanosecondes
et Co = 100 cycles de 166,66 nanosecondes

Pour une image de 512 x 512 points élémentaires, en arrondissant et en négligeant Co, la fonction générale devient la suivante:

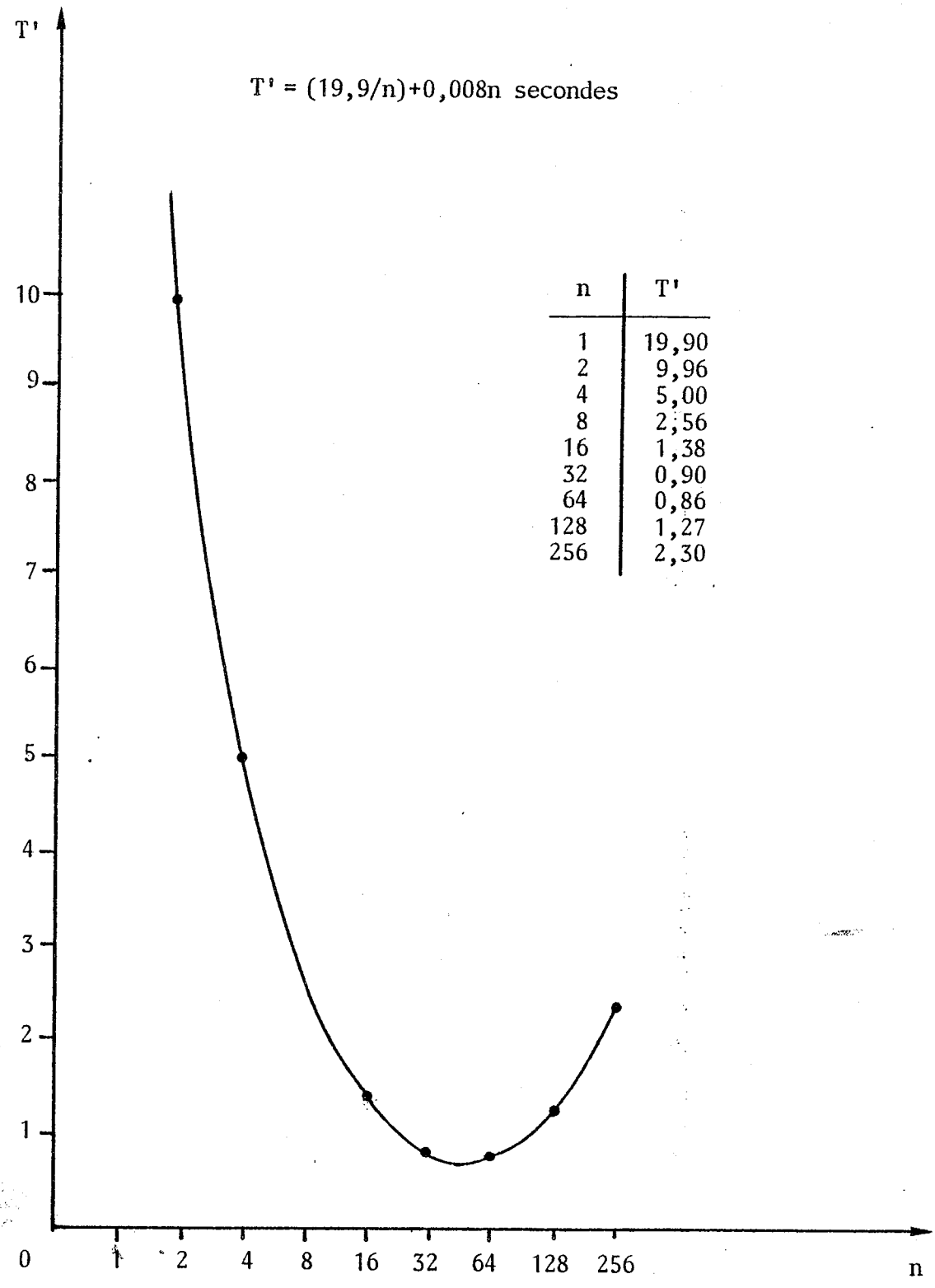
$$T = (1,7/n) + 0,0033\ n \quad \text{secondes}$$

qui est représentée en figure 26.

Bien sûr, les valeurs de K,k et Co varient d'un algorithme à l'autre. Ainsi pour l'algorithme du gradient, la fonction générale devient:

$$T' = (19,9 / n) + 0,0087\ n \text{ secondes}$$

dont la courbe, représentée en figure 27 est tout à fait comparable à la précédente.



n	T'
1	19,90
2	9,96
4	5,00
8	2,56
16	1,38
32	0,90
64	0,86
128	1,27
256	2,30

Figure 27

Temps de traitement (en secondes) en fonction du nombre de micro-unités de traitement, pour le calcul d'un gradient. Processeurs: Z8000.

Une remarque doit être faite ici: on voit sur les deux courbes citées ci-dessus qu'à partir de certaines valeurs de n (nombre de micro-unités de traitement), la courbe s'infléchit puis remonte. En effet, pour ces valeurs-là, le coût de la segmentation de l'image devient prépondérant par rapport au gain apporté par le parallélisme. Ou, plus intuitivement on met alors trop de temps à *recoller les morceaux*. Ce phénomène, ici mis en évidence dans le cas particulier de ROMUALD, correspond à la situation générale évoquée au chapitre II: le nombre d'instructions pour lesquelles les accès aux données sont faites en parallèle via les chemins processeurs locaux-mémoires locales devient insuffisant par rapport au nombre de celles où seul le chemin processeur général-mémoire est utilisé. Les conditions d'utilisation de la structure parallèle deviennent alors mauvaises. Ce phénomène peut être éclairé par l'étude de la formule générale T = (K*N/n) + kn + Co. Son point d'inflexion correspond à l'annulation de sa dérivée première soit quand n^2 = (K*N/k). N pouvant être considéré comme fixe pour une application donnée, plus les valeurs de K (liées au nombre d'opérations effectuées en parallèle) sont grandes par rapport à celles de k (liées au nombre d'opérations effectuées en monoprocesseur), plus la valeur de n pour laquelle la structure est intéressante croît. Par exemple, pour un algorithme très complexe, il est possible d'estimer K à 1 et k à 0,01. Alors, le point d'inflexion est obtenu pour n^2 = (K*N/k) = N * 10^2. Dans le cas où N = 26 * 10^4, l'architecture de ROMUALD serait payante jusqu'à n = 5.000 unités de traitement! Cet aspect n'apparaît pas sur les figures 26 et 27 qui se réfèrent à des traitements beaucoup moins complexes et qui n'envisagent que le cas d'un nombre limité -et raisonnable en terme de réalisation- de processeurs.

Ce qui est à retenir, et que ROMUALD permet de mettre clairement en valeur, c'est qu'il existe des valeurs de n pour lesquelles la fonction est pratiquement linéaire. Pour cet intervalle de valeurs, lorsque le nombre de micro-unités de traitement est multiplié par 2, le temps d'exécution est lui-même divisé par une valeur très proche de 2. L'architecture de ROMUALD est donc une bonne solution pour obtenir, dans certaines limites, une diminution du temps de traitement.

III.4.2 Coût du système

Ce paragraphe examine maintenant ROMUALD sous un angle économique. En effet, si tous les utilisateurs de systèmes de traitement

d'images demandent un accroissement des performances, ils ne sont pas pour autant prêts, à l'exception de quelques-uns, très exigeants - et fortunés -, à augmenter pour cela considérablement leurs dépenses.

Le coût de ROMUALD peut être essentiellement décomposé en deux parties:

- une première partie, ou *frais fixes* comprend essentiellement l'unité de commande, les unités de saisie et d'affichage, les périphériques d'entrée/sortie, la mémoire image, le cablage et la *mise en boîte* de l'ensemble,
- une deuxième partie ou *frais proportionnels* comprend essentiellement le coût des micro-unités de traitement - à l'exception de la mémoire image - et varie avec leur nombre.

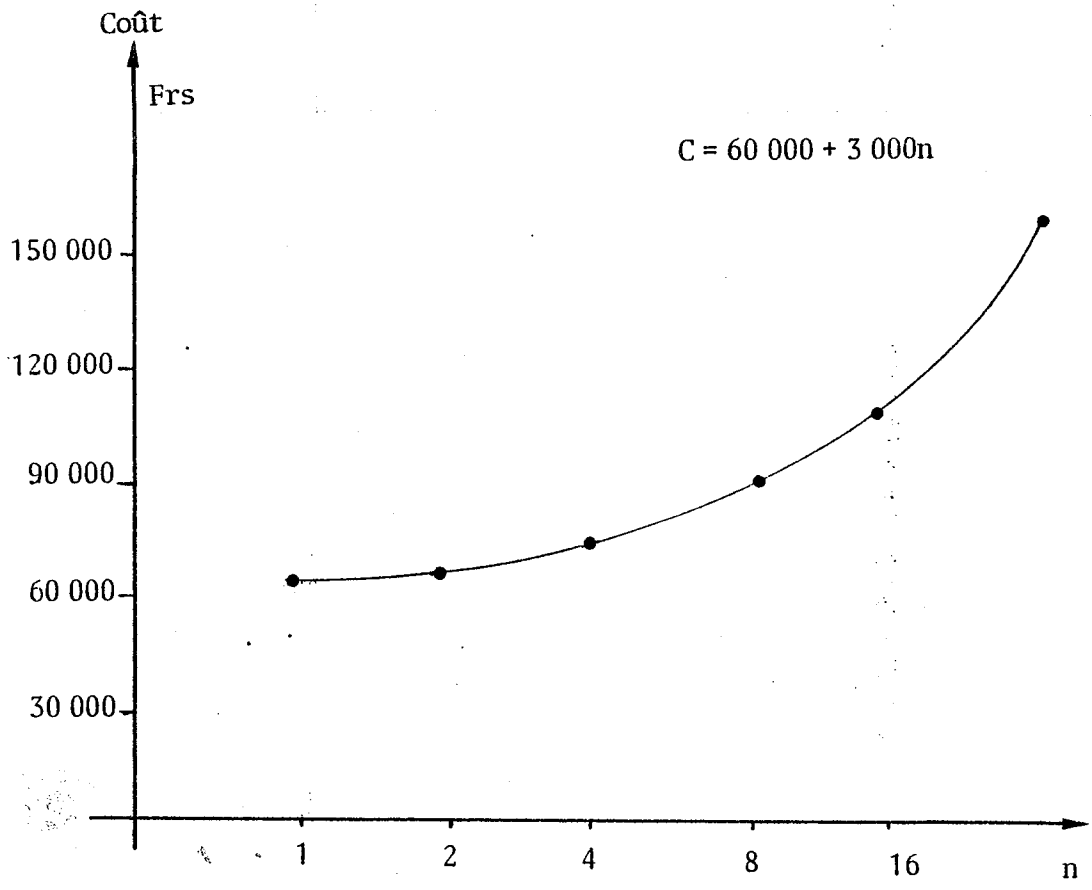


Figure 28

Variation du coût total (en francs) en fonction du nombre de micro-unités de traitement.

Soit un coût total exprimable – certes, grossièrement – par la fonction:

$$C = \text{frais fixes} + n \ (\ \text{frais proportionnels}\ )$$

Lors de la réalisation de ROMUALD, le rapport entre frais fixes et frais proportionnels s'est trouvé environ égal à 20. L'évolution du coût total du système en fonction du nombre de micro-unités de traitement est illustré en figure 28.

III.4.3. Conséquences.

Les études précédentes permettent de juger, d'une façon relativement objective et exacte, de l'intérêt et des limites d'une configuration répartie comme ROMUALD dans le domaine du traitement d'images. La figure 29 qui représente l'évolution approximative des performances en fonction des coûts, montre qu'en utilisant une architecture répartie de façon judicieuse et mesurée, il est possible d'accroître considérablement les performances sans augmenter exagérément les coûts.

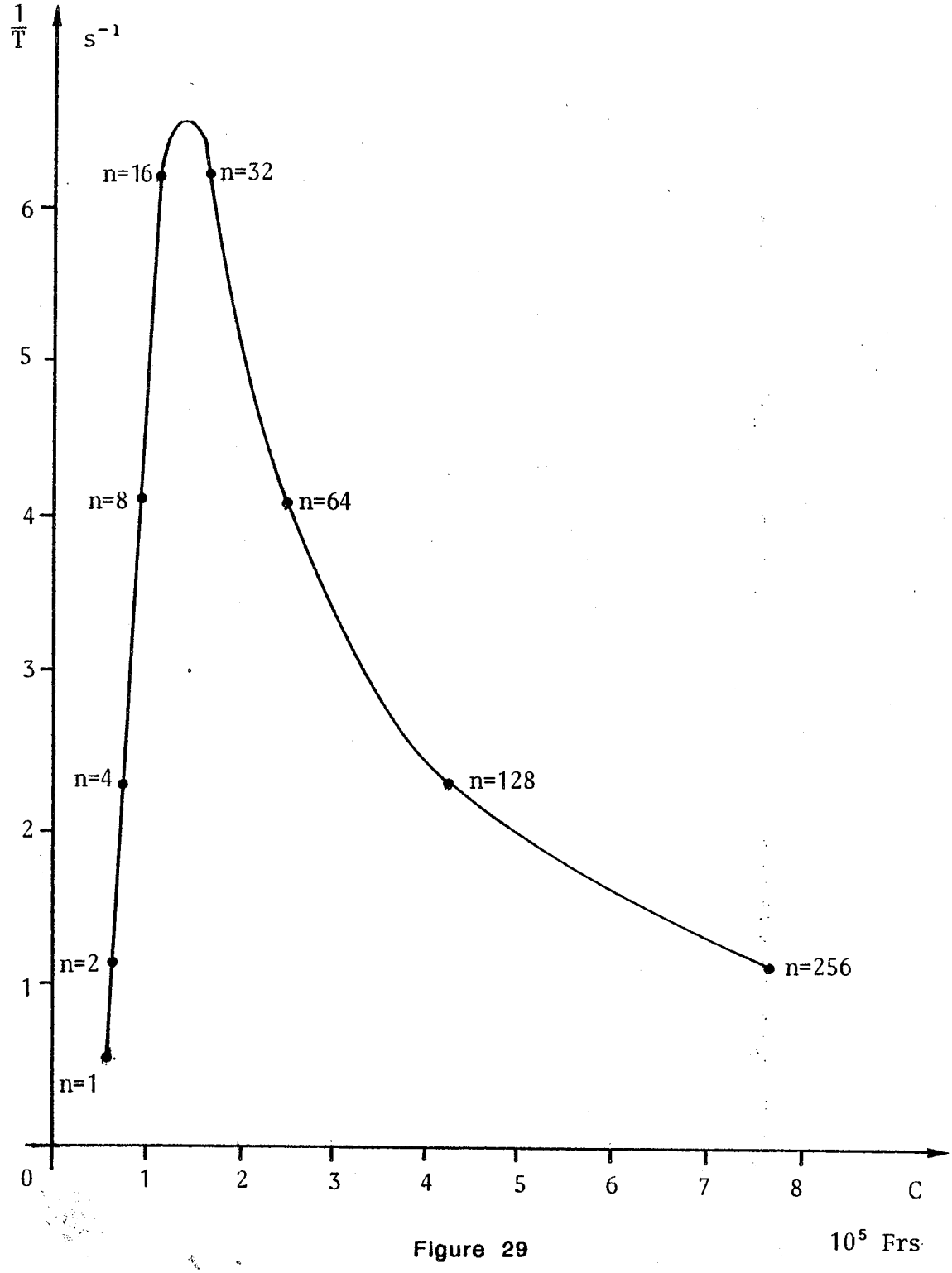


Figure 29

Variation de l'efficacité en fonction du coût.

Avec T = (1,7/n) + 0,0033 n secondes

C = 60 000 + 3 000 n Francs

n = nombre de micro-unités de traitement.

III.5 Aspects techniques de réalisation

III.5.1 Généralités

Démarrée au début de 1979, la première étape de la réalisation a permis de réfléchir sur l'objet du travail. Le mot *objet* étant pris dans deux sens: objet matériel (l'image et les traitements qu'on lui applique) et but à atteindre (quel service rendre).

Prenant en compte les idées mises en avant dans cette première étape, une architecture, largement présentée précédemment a été définie. Ce modèle d'architecture, matérielle et logicielle, est encore peu lié à des solutions pratiques.

Les choix d'un microprocesseur et du nombre de microprocesseurs en parallèle ont été effectués grâce à l'étude de comportement a priori présentée en III.4. Cette étude a été menée successivement pour trois microprocesseurs: Motorola 6800 (8 bits), Intel 8086 (16 bits) et Zilog Z8000 (16 bits). Le microprocesseur 16 bits Motorola 68000 n'existait pas encore au début de ce travail. Il aurait sûrement été un bon candidat. Le 6800 a rapidement été éliminé pour deux raisons :

- sa capacité d'adressage réduite posait de gros problèmes pour la réalisation du processeur général.
- ses performances étant nettement inférieures à celles des autres candidats, il aurait fallu, pour obtenir un résultat correct, utiliser un très grand nombre de processeurs parallèles, ce qui a été exclu pour des raisons technologiques : pour réaliser simplement le Mégabus, auquel les processeurs locaux sont connectés, il était souhaitable de rester dans les limites de la technologie TTL et pour cela, ne pas dépasser 8, éventuellement 16, processeurs locaux connectés. Or, pour être efficace avec des 6800, il aurait fallu 32, voire 64 processeurs.

Restaient alors deux microprocesseurs 16 bits: le 8086 et le Z8000. Le choix s'est porté sur le Z8000 essentiellement pour deux raisons:

- ses performances lors de l'étude de comportement étaient notablement meilleures que celles du 8086 (de l'ordre de 20 à 30 %).
- le mécanisme d'adressage segmenté du Z8001 s'adaptait très

bien à la réalisation du processeur général.

Le Z8001 fut donc retenu pour la réalisation du processeur général, et le Z8002 (à adressage classique) pour les processeurs locaux.

Vu les résultats de l'étude de comportement, et en tenant compte des préoccupations technologiques -citées plus haut à propos du 6800-, le nombre de processeurs locaux fut fixé à 8.

L'étape suivante, de construction d'une maquette avait trois buts:

- tester les capacités humaines pour réaliser le système défini,
- démontrer la cohérence et la pertinence des choix matériels et logiciels.
- fournir, si possible rapidement, une machine de test permettant de développer et de mettre au point du logiciel tant système que d'application.

La structure de la maquette était donc réduite par rapport au projet complet: un processeur général, deux processeurs locaux (correspondant chacun à un huitième d'image), une unité de saisie et une unité d'affichage.

Par ailleurs, afin de faciliter la réalisation et, semblait-il, d'en accroître la rapidité, des choix complémentaires avaient été faits:

- ne pas réaliser les vastes plans mémoire nécessaires et préférer la solution peu économique, mais fiable, consistant à acheter des plans mémoire montés et testés.
- faire réaliser une grosse partie du cablage et de *la mise en boîte* du système par un sous-traitant.
- confier la réalisation des convertisseurs à une société grenobloise, qui était bien accoutumée à traiter les problèmes des signaux analogiques et de leur conversion.

Les schémas définitifs de la maquette étaient terminés début décembre 1980, la fabrication début juin 1981, la mise au point et les tests, début septembre 1981.

L'expérience acquise lors de la réalisation de la maquette, sa mise au point puis son utilisation pendant la fin de l'année 1980 a été riche d'enseignements. Quelques points essentiels peuvent être notés:

- un gros défaut de la maquette résidait dans l'instabilité électrique -et donc logique- du Mégabus. Réalisé à l'aide de nappes de fils souples, et utilisant comme portes de transmission des composants classiques, ce bus n'a pu être stabilisé qu'après de longs tâtonnements, en faisant appel à des *bricolages électroniques* (capacités anti-parasites, rappel des lignes par des ponts de résistances, ...). Une nouvelle réalisation, utilisant des lignes en fond de panier pour éviter la diaphonie et faisant appel à des portes de transmission à trigger de Schmidt (beaucoup moins sensibles au bruit) a donc été étudiée.
- les difficultés rencontrées pour tester et démarrer les processeurs locaux ont conduit à les modifier afin de permettre de bonnes possibilités de mise au point.
- l'écriture de logiciels d'application sur la maquette a montré que le mécanisme utilisé pour commander le démarrage des processeurs locaux était malcommode et pouvait conduire à des situations peu sûres. Ce mécanisme a été revu et fiabilisé.
- enfin, quelques montages matériels utilisés dans la maquette devenaient très encombrants une fois étendus à une machine complète. Ces mécanismes ont été revus et densifiés en utilisant des procédés plus sophistiqués (par exemple: mémoires vives et mortes pour le décodage des adresses et des numéros de segments).

Pour réaliser un prototype complet, il fallait, par ailleurs, apporter des solutions aux difficultés esquivées dans la maquette: la mémoire image et les convertisseurs.

La mémoire a été choisie en tenant compte à la fois de considérations techniques mais aussi de la disponibilité des composants. Les bancs mémoires ont ainsi été conçus en utilisant des mémoires Intel 2118, mémoires dynamiques de 16 K x 1 bit, mono-tension.

Les convertisseurs, quant à eux, posaient deux problèmes essentiels:

- il fallait traiter des signaux analogiques et ce n'est pas là le domaine d'élection des informaticiens! Finalement, un examen attentif des composants disponibles montra que certaines pièces réalisaient déjà une grande partie du travail. Pour le reste, l'examen de schémas TV, la collaboration d'un électronicien, quelques transistors, résistances et autres capacités, liés à

la savante patience de P. Jutier sont venus à bout des difficultés.

- la partie logique des convertisseurs est, en fait , assez complexe. Un nouveau système assez souple a été réalisé, sur une idée originale de P. Jutier, par un ensemble de logique discrète contrôlé par un microprocesseur -et donc programmable.

Ainsi, les schémas définitifs du prototype ont été établis début 1981 et la machine complète testée et mise au point à la fin du mois d'août 1981.

Tout au long de la réalisation, de nombreuses difficultés ont été rencontrées, cette thèse n'est pas le lieu pour exposer tous ces petits malheurs -bien qu'ils aient été nombreux !- mais plutôt d'attirer l'attention sur certains points qui semblent critiques. Premièrement, le choix, dès 1979, du Zilog Z8000, a été un bon choix. L'influence défavorable de sa nouveauté sur le travail a cependant été mésestimée. A l'époque, en effet, -cela s'est un peu arrangé depuis- il n'existait qu'une documentation très chiche concernant ce processeur. Il a donc fallu travailler sur des photocopies de copies de rapports préliminaires à diffusion plus ou moins limitée ... et farcis d'erreurs.
Par ailleurs, la nouveauté du processeur faisait qu'il n'existait aucun circuit périphérique susceptible d'être inclus facilement dans une configuration à base de Z8000.
Enfin, l'absence, au début totale, d'outils de développement, de moniteurs de mise au point ou d'assembleurs a conduit à créer des solutions propres et cela a représenté une très lourde tâche.

En bref, cette nouveauté du Z8000 -et non sa valeur propre en tant que microprocesseur- a sérieusement compliqué le travail.

Deuxièmement, comme cela a été noté plus haut, les processeurs locaux de la maquette étaient pratiquement intestables. Les circuits prévus étaient certes efficaces et très compacts, mais toute erreur de conception ou de cablage pouvait très difficilement être repérée. Il s'agissait là d'une erreur de conception -due à l'inexpérience-. Dans le prototype final, ce qu'il fallait de facilités matérielles et logicielles pour permettre les tests a été ajouté.

Troisièmement, en dehors des outils de fabrication proprement dits, les outils de développement utilisés étaient, en tout et pour tout:

- un oscilloscope,
- un programmateur d'EPROM,
- un multimètre.

Pour compléter cet ensemble, un moniteur de mise au point a été codé. Enfin, à Pâques 1981, un assembleur croisé Z8000 a été réalisé à l'IRISA à Rennes.
Cette pauvreté a parfois compliqué la tâche. En particulier, les 5 premiers K octets de code (moniteur et programmes de tests) ont du être écrits et codés en hexadécimal à la main puis intégrés dans le système en utilisant le programmateur d'EPROM comme éditeur. *Ceci est assez lent...*

Quatrièmement enfin, il était prévu de simplifier le travail en incluant dans la maquette des éléments fabriqués ailleurs: la mémoire, les convertisseurs. Hélas, il a fallu d'une part pour la mémoire, réaliser un interface matériel qui compliquait artificiellement la structure des processeurs locaux , et d'autre part pour les convertisseurs, corriger une partie de leur logique afin de satisfaire à l'interface défini.

Choisies pour gagner du temps, ces solutions ont été d'un bénéfice douteux. Comme de plus, ces problèmes devaient être traités pour le prototype final, il y aurait eu intérêt à les aborder lors de la réalisation de la maquette ce qui se serait, sûrement, soldé par un avantage global.

Les deux paragraphes suivants décrivent de façon plus détaillée le principe de réalisation de deux fonctions importantes du système.

III.5.2 Adressage segmenté et reconfiguration dynamique

Une particularité du Z8001 (version à adressage segmenté du Z8000) permet de réaliser les fonctions de reconfiguration dynamique de façon particulièrement commode. La figure 30 présente le diagramme des temps (simplifié) du Z8001 pour des transactions mémoire.

Il apparaît que, pour une transaction mémoire classique de 3 cycles d'horloge, le numéro de segment est établi une période d'horloge avant le premier cycle de la transaction. Cette anticipation est prévue, dans les systèmes Z8000 standard pour permettre le fonctionnement d'une

unité Z8010, "Memory Management Unit", normalement utilisée pour réaliser un système d'adressage virtuel.

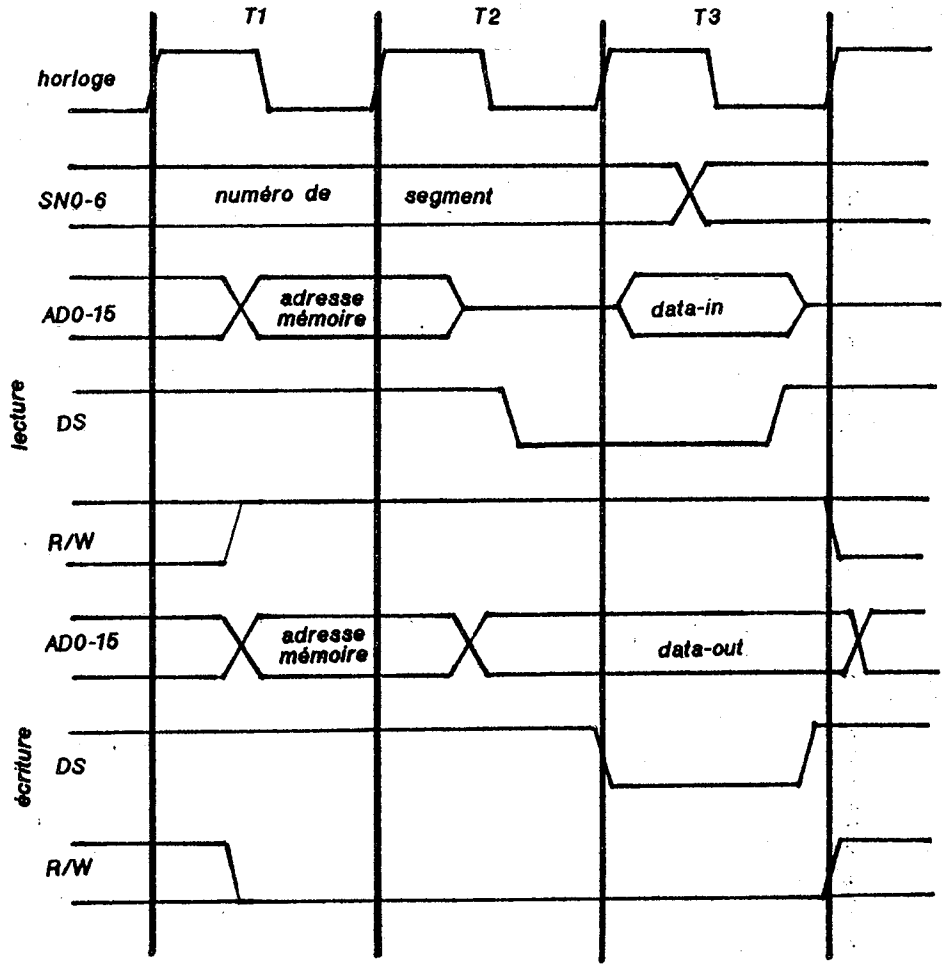


Figure 30
Diagramme des temps du Z8001 pour des transactions mémoire.

Dans ROMUALD, cette anticipation est utilisée pour réaliser les fonctions de reconfiguration dynamique qui interviennent à deux niveaux:

- un premier niveau concerne la réalisation d'un système de décodage d'adresses dynamiquement modifiable, qui a été présenté, sous une forme plus algorithmique que matérielle au paragraphe III.3.3. Il s'agit de réaliser un système permettant d'associer

une ou plusieurs micro-unités de traitement physiques à un nom logique apparaissant dans un programme sous forme d'une valeur numérique (valeur de segment, en l'occurence).
- un deuxième niveau concerne la restructuration et le contrôle du matériel permettant la réalisation des unités fonctionnelles telles que définies en III.3. Le fonctionnement du processeur d'entrées-sorties d'images, permettant la réalisation des unités de saisie et d'affichage, sera décrit au paragraphe III.5.3.

En ce qui concerne l'organisation des relations fonctionnelles unité de commande-unité de traitement, les fonctions à réaliser sont les suivantes:

- configurer la machine en unité de commande (les processeurs locaux sont à l'arrêt et les mémoires associées peuvent être adressées par le processeur général via le Mégabus).
- configurer la machine en unité de traitement (les processeurs locaux exécutent un programme de traitement et ont accès à leur mémoire associée. Le processeur général ne peut accéder ces mémoires).
- permettre au processeur général de connaître l'état de l'unité de traitement.
- détecter les erreurs logiques dans les commandes, particulièrement celles conduisant à des résultats aléatoires ou à des problèmes électriques.

La figure 31 présente le schéma de principe du matériel permettant la réalisation des fonctions de reconfiguration dynamique.

Sur la gauche du schéma apparaissent les signaux liés au processeur général Z8001. Sur la droite, ceux liés au Mégabus.

Les 7 bits du numéro de segment sont divisés en deux groupes. Le premier groupe (SN0-3) porte le nom logique du ou des processeurs auxquels la transaction en cours s'adresse. Le deuxième groupe (SN4-6) indique la fonction à réaliser.

Le premier groupe est décodé par une mémoire vive B. Les 4 bits SN0-3 sont utilisés comme adresse et les 8 bits de données lus en sortie indiquent, après décodage dans la mémoire, le ou les processeurs physiques concernés par l'opération en cours. La fonction de décodage

est entièrement programmable puisqu'elle dépend du contenu de la mémoire. Cette mémoire peut être écrite. Pour cela, la valeur à écrire apparaît sur les lignes data-out du processeur général, l'adresse à laquelle écrire est indiquée par SN0-3 et la fonction d'écriture est le résultat du décodage de fonction effectué dans la mémoire morte A.

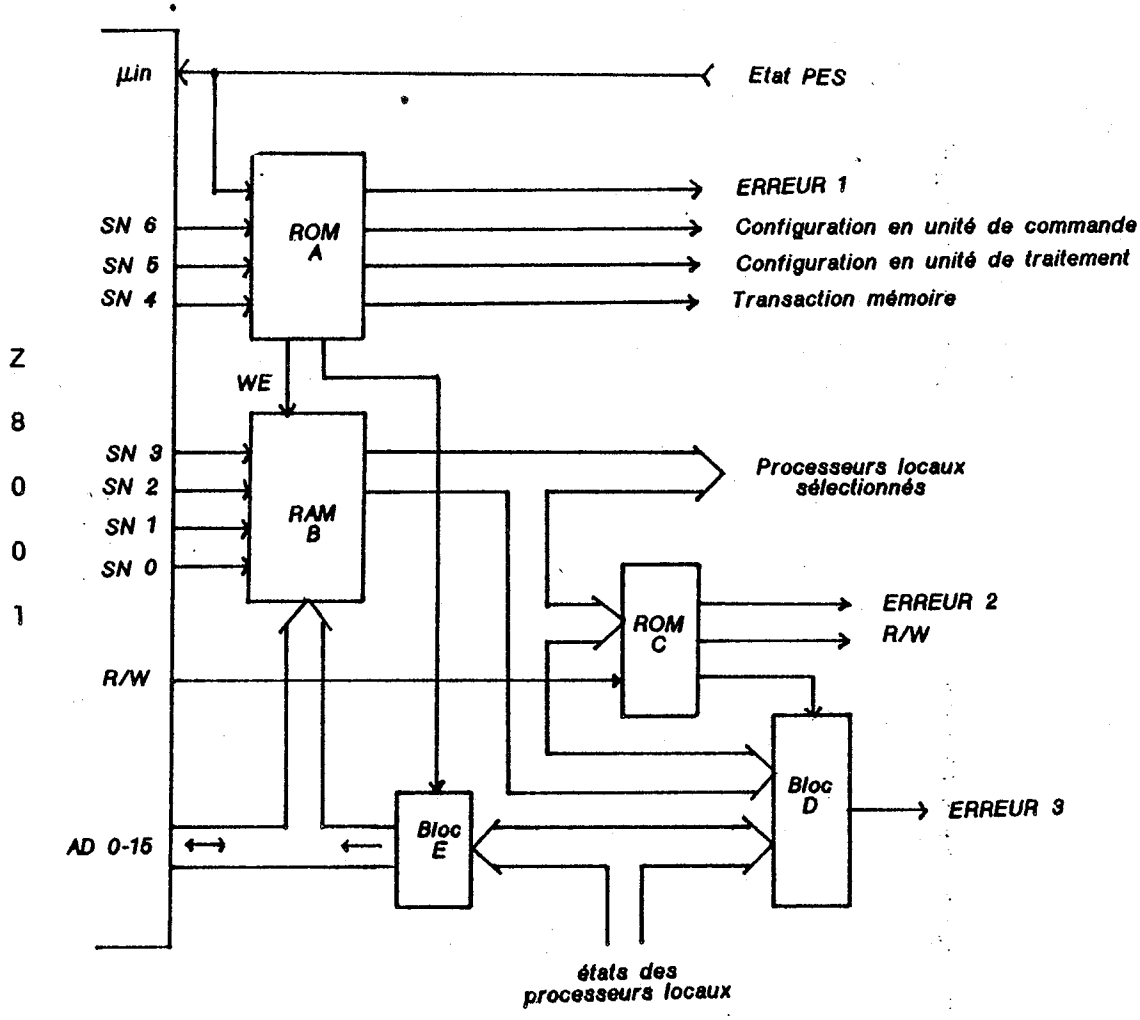


Figure 31
Reconfiguration dynamique:
schéma de principe du matériel.

Le deuxième groupe (SN4-6) est utilisé pour adresser la mémoire morte A en conjonction avec un signal provenant du processeur d'entrée-sortie d'images indiquant si celui-ci est actif ou non.

Les données lues en sortie de la mémoire morte A sont le décodage des fonctions à réaliser, à savoir:

- configurer la machine en unité de commande,
- configurer la machine en unité de traitement,
- écrire la mémoire vive B,
- lire, pour test, l'état de l'unité de traitement.

Une ligne de sortie, erreur_1, indique que la fonction demandée est inconnue (mauvais codage) ou non-réalisable (dans le cas où le processeur d'entrée-sortie d'images est actif et où on désire accéder au Mégabus).

La mémoire morte C est essentiellement destinée à détecter l'erreur_2 qui correspond, lors d'une transaction mémoire via le Mégabus, à une tentative de lecture simultanée de plusieurs mémoires (cas où la sortie de la mémoire vive B présente plus d'une ligne active, où la fonction issue de la mémoire morte A est transaction mémoire et où le sens de la transaction est lecture).

Le bloc D permet de détecter l'erreur_3. Ce bloc accepte en entrée d'une part les 8 lignes indiquant le ou les processeurs participant à la transaction, et d'autre part les 8 lignes, chacune issue d'une micro-unité de traitement, indiquant si cette unité est à l'arrêt ou non. L'erreur_3 signale une tentative de transaction mémoire sur une ou plusieurs unités qui n'est ou ne sont pas à l'arrêt.

Le bloc E permet la lecture par le Z8001 de l'état des micro-unités de traitement et lui permet de tester la fin de traitement.

Les erreurs 1, 2 et 3 sont combinées pour servir d'entrée à la patte "Segment Trap" du Z8001. Dans ce cas, le programme courant est interrompu et le programme de traitement de l'interruption peut analyser l'erreur en utilisant trois sources:

- l'état du processeur d'entrées-sorties d'images qui est communiqué au processeur général via son entrée μin,
- l'état des micro-unités de traitement, qui peut être lu comme indiqué ci-dessus,
- l'instruction fautive.

Enfin, dans le cas où une erreur est détectée, un mécanisme de sécurité,

non représenté, permet d'éviter que l'opération reconnue éronnée n'aille perturber le Mégabus, évitant ainsi des problèmes logiques et/ou électriques.

III.5.3. Processeur flexible d'entrées-sorties d'images

Ce processeur a été développé sur une idée originale de P. Jutier. Le signal vidéo standard de 625 lignes-50 Herz a servi de base de temps pour la conception du processeur. Pour la réalisation de ce processeur, trois classes de fonctions peuvent être dégagées:

- les fonctions liées à la génération d'une adresse mémoire, permettant de lire ou de ranger une valeur,
- les fonctions liées à la définition de l'image (choix entre une image sur deux trames, 512*512 points et une image sur une seule trame 256*256 points; mais aussi, pour un développement en cours, choix d'un système couleur ou noir et blanc).
- les fonctions liées à la génération de signaux de commandes, essentiellement pour les convertisseurs et les mémoires.

En terme de temps, il est possible de répartir les fonctions de ces 3 classes en 2 catégories, les fonctions lentes et les fonctions rapides. Les fonctions de définition de l'image sont lentes, celles de génération de signaux sont rapides.

La génération d'adresses peut être divisée en deux:

- une partie lente, effectuant la génération des bits de poids forts de l'adresse et correspondant aux valeurs de trame et de ligne.
- une partie rapide, effectuant la génération des bits de poids faibles de l'adresse et correspondant aux numéros de colonnes à l'intérieur d'une ligne.

Le principe de base consiste à remplacer une partie de l'électronique discrète utilisée classiquement dans ce genre de systèmes par un micro-ordinateur monolithique assurant les fonctions lentes. Le schéma de principe du processeur est représenté en figure 32.

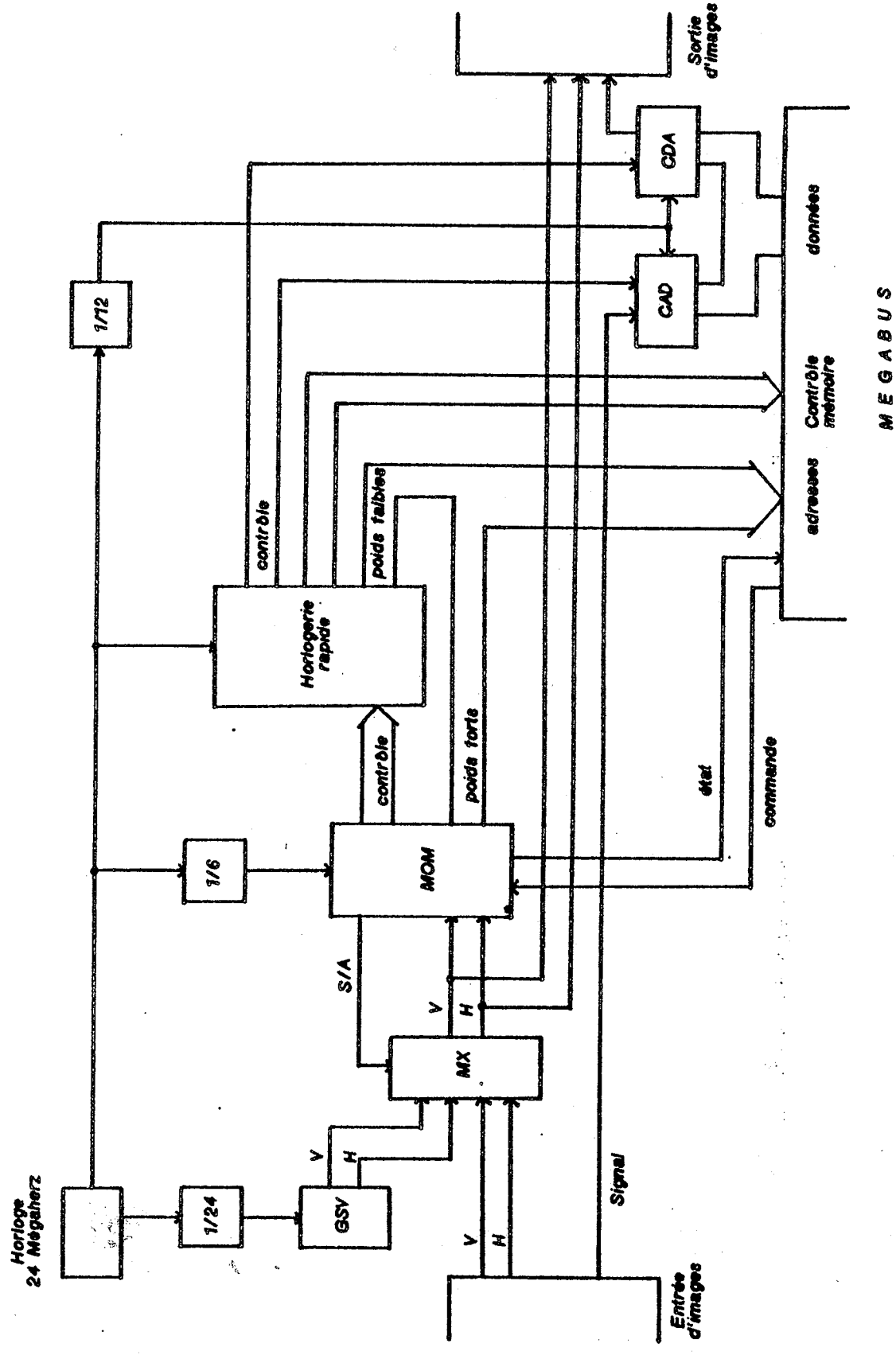


Figure 32.

Schéma de principe du processeur d'entrées-sorties d'images.

Les entrées d'images sont figurées à gauche du schéma sous une forme décodée à trois signaux: le signal H de synchronisation horizontale (changement de ligne), le signal V de synchronisation verticale (changement de trame) et le signal vidéo, porteur de l'information analogique correspondant à l'image.

Les sorties d'images sont figurées à droite du schéma sous la même forme.

En bas à droite, les connexions avec le Mégabus sont indiquées et comprennent:

- des adresses,
- des données,
- des signaux de contrôle pour la mémoire,
- un signal d'état, indiquant au processeur général, comme cela a été vu au paragraphe précédent, si le processeur d'entrées-sorties d'images est actif ou non.
- une ligne de commande, qui est une ligne de transmission série permettant au processeur général de transmettre des commandes et des paramètres au micro-ordinateur monolithique MOM.

Sur la partie haute du schéma circule un signal d'horloge de base à 24 Mégaherz qui est utilisé, en différents points du schéma, aprés division éventuelle et qui permet d'assurer un bon synchronisme.

Le fonctionnement de l'ensemble peut être décrit ainsi:

Lorsque le MOM reçoit une commande par la ligne provenant du processeur général, il y a décodage puis établissement d'un état initial correspondant.

La ligne S/A est activée selon qu'il s'agit d'une saisie ou d'un affichage. Le multiplexeur MX transmet alors au MOM des signaux H et V provenant soit de l'entrée d'image, pour une saisie, soit du générateur de synchro vidéo GSV pour un affichage.

Le logiciel du MOM est capable de traiter ces signaux et en particulier de distinguer, grâce à leur position relative, le début d'une trame paire du début d'une trame impaire.

En début de trame paire, le MOM compte quelques lignes, qui seront ignorées, puis affiche d'une part la partie poids forts de l'adresse et d'autre part une série de bits de commandes qui permettent à l'horlogerie rapide de fonctionner.

Cette horlogerie, aprés un délai permettant d'ignorer le début de la ligne, non significatif, génère d'une part les bits de poids faibles de l'adresse et d'autre part les signaux de commande du convertisseur approprié ainsi que ceux permettant l'accès à la mémoire, puis répète cette opération 256 ou 512 fois en incrémentant sa partie adresse.

Quand cette opération est finie, le MOM est en attente du prochain top H, et va répéter l'opération 256 fois en incrémentant sa partie adresse.

Si la définition est 512*512, la même opération est reprise pour la trame impaire qui suit.

Ce fonctionnement est très simple et a le mérite considérable d'être largement programmable et donc flexible.

Cela a permis en particulier, sur ce schéma de base conçu pour des signaux de type télévision, de connecter un microscope électronique à balayage en n'effectuant que des modifications logicielles.[8]

Les communications avec le processeur général sont simples et réalisées par deux lignes:

- une ligne d'état indiquant si le processeur d'entrées-sorties d'images est actif ou non (le processeur général peut ainsi tester la fin d'une saisie, le mécanisme de reconfiguration peut éviter des erreurs de programme).
- une ligne de commande, sur laquelle s'effectue une transmission série prise en compte par le récasynchrone universel contenu dans le MOM.

Le micro-ordinateur choisi est le circuit Z8 de Zilog qui, dans la version utilisée, présente les intéressantes caractéristiques suivantes:

- mémoire vive de 256 octets,

- mémoire de programme montée en *"piggy-back"* sur le dessus du chip: EPROM de 16K bits,
- 40 pattes seulement dont 32 utilisables comme ports d'entrées-sorties programmables et reconfigurables,
- mécanisme d'interruption (utilisé pour les signaux H et V et l'entrée de réception série),
- etc...

III.6. Logiciels et applications

Il a été vu plus haut, combien les débuts de ROMUALD furent difficiles du point de vue de sa programmation. La disponibilité d'un assembleur croisé fonctionnant sur le système Multics du CICG a contribué à dédramatiser la situation. Il n'en reste pas moins vrai que la programmation de ROMUALD présente actuellement l'important défaut de décourager parfois les utilisateurs éventuels pour peu qu'ils soient accoutumés aux facilités de l'informatique moderne.

Cela est essentiellement dû:

- à la programmation en assembleur, et conséquemment à un système de mise au point de programmes de très bas niveau,
- au fait que l'utilisateur doit connaître et avoir assimilé l'architecture du système pour l'utiliser correctement,
- à la difficulté rencontrée pour offrir à l'utilisateur un ensemble de primitives de traitement d'images réalisant un bon compromis entre souplesse et temps de réponse rapide.

A l'heure actuelle, un pas décisif pour le système ROMUALD serait de le doter, au moins, d'un langage de haut niveau et des outils de mise au point symbolique associés.

Mieux encore serait la définition et la réalisation d'un langage spécialisé pour le traitement d'images intégrant les caractéristiques matérielles de ROMUALD.

ROMUALD a néanmoins été utilisé dans des conditions par ailleurs satisfaisantes pour le développement d'applications concernant l'anthropométrie automatique [27], différents travaux sur les circuits intégrés [8] et la lecture automatique [77].

III.7. Perspectives

En complément des développements logiciels primordiaux évoqués ci-dessus, la machine ROMUALD peut être améliorée d'un point de vue matériel. P. Jutier assure depuis 1981 le développement de nouvelles versions de ROMUALD. Durant l'été 1982, un nouveau prototype a vu le jour. Par rapport à la description faite dans cette thèse, le changement essentiel porte sur le processeur d'entrées-sorties d'images qui accepte et produit désormais des images vidéo couleur. Au prix de modifications matérielles délicates mais limitées à la partie convertisseurs et horlogerie rapide du processeur d'entrées-sorties d'images, grâce à une modification du logiciel de ce processeur et en utilisant le reste de l'architecture telle quelle, il est possible de travailler sur des images couleur de 512*512 points de 8 bits (3+3+2) ou de 256*256 points de 16 bits (5+5+6).

Le nouveau prototype est par ailleurs construit en utilisant une technologie très dense fondée sur la réalisation de plaques imprimées en multicouches et sur un passage de la quasi totalité de la logique discrète en matrice logique programmable (*PLA*). La fréquence de base des processeurs est passée de 4 à 6 Mégaherz et pourra évoluer jusqu'à 10 Mégaherz.

Par ailleurs, il sera possible, dans une version ultérieure, d'adjoindre à chacun des processeurs un co-processeur arithmétique rapide qui devrait avoir une influence positive sur la rapidité de traitement ainsi que sur les classes d'algorithmes utilisables en permettant des calculs en arithmétique flottante.

Ce développement devrait pouvoir déboucher sur une production de type industriel.

CHAPITRE IV

Deuxième pas: définition du système KIDS

IV.1 Introduction

Le projet de réaliser un système de traitement d'images universel et efficace semble encore devoir être -pour longtemps, peut-être- situé dans l'inconscient collectif comme objet implicite du désir -scientifique, tout au moins- et servir là à justifier et à autoriser l'aventure de la recherche. Participant à cette quête, le système KIDS peut être considéré comme une résultante logique des opinions et des analyses présentées dans les chapitres précédents et, au delà, d'une certaine appréciation sur l'état de l'art du traitement d'images.

Il s'agit donc de définir un système qui ne sera ni universel ni instantané, mais devra permettre d'effectuer efficacement une large classe d'algorithmes, correspondant à des représentations de données et à des façons de procéder regroupant un large consensus et pouvant être utilisé pour de nombreuses applications. Ce champ d'action, qui dépend d'un ensemble de choix limitant l'ambition du système, sera défini en IV.2. Ensuite, comme son appellation de *système* le souligne, la définition de KIDS sera faite en suivant une *approche systémique* consistant à concevoir matériel et logiciel de façon étroitement liée.

Bien qu'il soit évident, pour quiconque a une expérience de ce genre d'activité, que la conception d'un système n'est jamais, dans la réalité, le fait d'une approche descendante ou ascendante pure, mais bien plutôt le résultat d'un mélange de méthodes et d'intuitions, d'essais, d'échecs et autres remises en question, le système va cependant être présenté d'une façon linéaire qui, partant de la définition d'un langage algorithmique, décrira les couches successives d'interprétation, qu'elles soient matérielles ou logicielles, qui permettent d'arriver à l'obtention du résultat recherché par un programme écrit dans ce langage. Ainsi, le paragraphe IV.4 contient la présentation d'un langage algorithmique satisfaisant les choix du paragraphe IV.2. Le paragraphe IV.5 décrit ensuite l'architecture matérielle et logicielle du système jusqu'au paragraphe IV.6 consacré à la définition d'un circuit intégré VLSI, élément constitutif d'un processeur matriciel cœur de l'ensemble.

IV.2. Champ d'action du système

La définition du champ d'action du système est le résultat d'un choix arbitraire dont les avantages et inconvénients semblent difficilement quantifiables. La présentation qui va en être faite est très peu organisée et consiste essentiellement à énumérer un ensemble de contraintes à satisfaire. L'organisation de l'ensemble, sa cohérence apparaîtront peu à peu dans la suite de ce chapitre.

Une première ligne directrice concerne la rapidité du système. Le but visé est de construire un système très rapide. Par quoi il faut entendre que, là où ROMUALD, par exemple, permet un gain de temps d'un facteur de l'ordre de 8, le système KIDS devra améliorer les temps de calcul dans un ordre de grandeur de 10^2 à 10^4, selon les cas particuliers.

Parmi les types d'architecture matérielle, évoqués au chapitre II et permettant d'obtenir des gains de cet ordre, le choix sera de construire le système sur un modèle de processeur matriciel.

Une deuxième ligne directrice concerne les images qui vont être traitées. Le choix est fait de pouvoir traiter des images 512x512 points mais aussi de permettre une certaine souplesse. Ainsi, il sera possible de traiter des images de définition spatiale moindre: 256x256, 128x128, etc. Les images pourront être soit des images binaires, soit des images dites *grises*. Chaque point d'une image grise portera une valeur sur 2 à 8 bits. Les images grises pourront être considérées comme un ensemble de plans, chaque plan étant assimilable à une image binaire. Des fonctions de conversion doivent exister, chaque fois qu'elles auront un sens, entre ces différentes représentations.

La troisième ligne directrice concerne le style des opérations qui pourront être effectuées sur les images. Ces opérations pourront dépendre de 3 catégories:

- les opérations *"image-image"*: par exemple addition, soustraction, comparaison de deux images grises, ou encore, pour des images binaires, des opérations logiques: OU, ET, etc.
- les opérations *de voisinage*, largement présentées dans les chapitres précédents, faisant intervenir, pour chaque point, la valeur de ses huit voisins immédiats sous le contrôle éventuel

d'un masque.
- les opérations de type pyramidal qui, parce que peu connues, sont présentées au chapitre IV.3.

Une quatrième ligne directrice, liée à la précédente, concerne la possibilité de réaliser des fonctions de comptage sur des images, essentiellement:

- tester si aucun point d'une image ne répond à une certaine condition,
- calculer combien de points d'une image correspondent à une certaine condition.

Cinquième ligne directrice enfin, celle qui consiste à permettre des reconnaissances de formes par référence à un modèle. Le principe de reconnaissance sera la recherche d'un recouvrement optimal d'un objet de l'image par un modèle (*template* ou *pattern matching*). Les modèles devront pouvoir varier en taille (variation d'échelle) et en orientation (rotation).

Le système KIDS, tel qu'il est présenté dans ces pages ne peut être considéré comme un ensemble définitif, clos et parfaitement défini. Ce serait d'ailleurs une erreur que de s'attacher, si tôt, à un tel objectif. En effet, un système de cette ampleur ne peut être réalisé sans la collaboration et le soutien soit d'un partenaire industriel cherchant à répondre à un besoin du marché, soit d'un important centre de recherche en traitement d'images. Ces collaborations restent à trouver pour le système KIDS. Aussi, le travail présenté dans ces pages doit-il être considéré comme une proposition, une base pour un travail ultérieur.

Dans cet esprit, certains points, bien qu'importants en terme d'utilisation, ont été jugés de seconde importance dans la description faite ici qui cherche surtout à être claire et concise et à ne pas accumuler les détails et les points particuliers. C'est pourquoi, d'une part, la description formelle complète du langage du système a été placée en annexe. C'est pourquoi, ensuite, la description qui suit ne traite ni des images grises ni des opérations de *pattern matching* sur les modèles. Ces deux omissions de la description sont reprises au paragraphe IV.7, en fin de chapitre, qui indique les problèmes spécifiques liés à ces deux questions et comment ils peuvent être résolus.

IV.3. Les structures de données non classiques du système KIDS: les pyramides et les arbres quaternaires

IV.3.1. Motivation

Les images sont classiquement structurées en listes, graphes ou plus généralement en matrices. De nombreuses méthodes de traitement ont été développées sur ces structures et particulièrement sur les matrices, aussi semble-t-il exclu, pour un système réaliste de traitement d'images, de ne pas pouvoir les utiliser. Dans de nombreux cas, cependant, ces structures classiques très riches en information détaillée, sont trop finement définies et conduisent à un travail local, microscopique, point par point.

Ainsi apparaissent l'intérêt et la nécessité de structures hiérarchiques de représentation des images. Dans leur principe, ces structures hiérarchiques sont constituées d'une collection de représentations d'images, le principe hiérarchique les organisant étant soit d'ordre physique (chaque image de la collection étant une représentation de la même scène à différents niveaux de résolution spatiale ou dynamique), soit d'ordre sémantique (chaque image de la collection étant une représentation de la même scène considérée sous l'angle de certaines propriétés, par exemple, une image grise, des images binaires correspondant à différents niveaux de seuil, image des contours, squelettes, etc), soit une combinaison, adaptée aux besoins de l'application, de ces deux organisations.

Ce type de structuration de données peut avoir un impact très important sur la façon de mettre en œuvre et même de concevoir des algorithmes. Depuis quelques années, de nombreuses recherches ont été menées dans ce domaine dont Rosenfeld [84] a donné une présentation synthétique. Parmi ces travaux, ceux de Hanson et Riseman [43, 44], de Uhr [97, 98 et 99] et essentiellement de Tanimoto [90, 91 et 92] ont largement influencé le travail présenté dans ce chapitre.

Si une hiérarchie de type sémantique dépend essentiellement d'opportunité dans le développement d'une application et est donc une affaire d'organisation logicielle, une hiérarchie de type physique peut être prise en compte par une organisation matérielle adaptée. C'est pourquoi le système KIDS fournit des dispositifs matériels permettant l'utilisation d'un de ces systèmes hiérarchiques: les pyramides.

IV.3.2. Pyramides: définition

Une pyramide peut être définie de diverses façons parmi lesquelles la suivante correspond le mieux à ce que le système KIDS permet de réaliser.

Une pyramide est:

1) un ensemble de p-nœuds:

$P = \{ (k,l,j) \}$ tels que:

$$\begin{cases} 0 \leq k \leq L \\ 0 \leq l \leq 2^{L-1} \\ 0 \leq j \leq 2^{L-1} \end{cases}$$

2) une relation binaire F sur P:

$$F(k,l,j) = (k-1, [l/2], [j/2])$$

3) une fonction de valeur V projetant P dans un intervalle R de valeurs (par exemple: R = {0, 1, ... sf5lè ou R = {0, 1}).

La notation [A] signifie *le plus grand entier inférieur ou égal à A*. Les p-nœuds de la pyramide correspondent à des points d'images.

Les ensembles {(k,l,j)} correspondent à des matrices de points et sont appelés niveaux de la pyramide.

La fonction F est la fonction *père (father)* définie sur tout P à l'exception de l'élément (0,0,0) qui est la racine de la pyramide.

Chaque p-nœud q, à l'exception de ceux du niveau L ({(L,*,*)}) a quatre fils qui sont les p-nœuds q' tels que F(q') = q.

Une pyramide est généralement construite en utilisant comme fonction V une fonction de moyenne:

$$V(k,l,j) = \begin{cases} 1/4 \sum_{x=0,1 \; y=0,1} V(k+1, 2l+x, 2j+x) & \text{pour } k<L \\ A(l,j) & \text{pour } k=L \end{cases}$$

où A(i,j) est une matrice contenant l'image originale.

Une autre fonction utilisée est la fonction de ré-échantillonnage pour laquelle le cas k<L devient:

$$V(k,i,j) = V(k+1,\ 2i,\ 2j)$$

Dans le cas des images binaires, d'autres fonctions sont utilisées comme par exemple la fonction de majorité:

$$V(k,i,j) = \begin{cases} \text{si } k<L & \begin{cases} 1 \text{ si } \sum_{x=0,1;\ y=0,1} V(k+1,\ 2i+x,\ 2j+x) > 2 \\ 0 \text{ sinon} \end{cases} \\ \text{si } k=L & A(i,j) \end{cases}$$

ou la fonction d'homogénéité:

$$V(k,i,j) = \begin{cases} \text{si } k<L & \begin{cases} 1 \text{ si } \sum_{x=0,1;\ y=0,1} V(k+1,\ 2i+x,\ 2j+x) = 0 \text{ ou } 4 \\ 0 \text{ sinon} \end{cases} \\ \text{si } k=L & A(i,j) \end{cases}$$

Ainsi définie, une pyramide est une représentation multi-résolution d'une image par une succession de matrices, chacune comprenant quatre fois moins d'éléments que la précédente. Il est intéressant de noter qu'une telle représentation ne nécessite pas beaucoup plus de mémoire que la représentation matricielle classique. En effet, si les tailles des niveaux successifs sont 2^n*2^n, $2^{n-1}*2^{n-1}$, ... alors, le nombre total d'éléments de la pyramide sera:

$$2^n*2^n\ (1 + 1/4 + 1/16 + ...) < 2^n*2^n\ 4/3$$

Soit une occupation supplémentaire de la mémoire limitée à 1/3 de la taille de la matrice originale.

D'un point de vue traitement du signal, la perte de résolution induite par la fonction V peut être analysée. Dans le cas, par exemple, de la fonction de moyenne, elle peut être considérée comme un processus à deux phases: filtrage suivi d'un ré-échantillonnage.

IV.3.3. Les arbres quaternaires: définition

Parce que leur apparence extérieure les rapproche, les arbres quaternaires sont souvent présentés conjointement aux pyramides, voire comme des cas particuliers. Cette ressemblance formelle ne doit cependant pas faire oublier que leur utilisation en traitement d'images répond à des besoins très différents.

Les arbres quaternaires sont en effet utilisés pour représenter et traiter efficacement des images segmentées en grandes régions compactes.

La construction d'un arbre quaternaire se fait sous l'hypothèse de l'existence d'un critère permettant de décider si une image digitale est uniforme ou homogène. Par exemple, un critère d'uniformité peut être que la déviation standard de ses niveaux de gris est inférieure à un seuil donné t.

Sur la base de ce critère, il est possible de subdiviser récursivement une image en parties homogènes, et ce de la façon suivante.

Si l'image de départ est homogène, l'arbre se réduit à sa racine. Sinon, l'image est subdivisée en quadrants et le test d'homogénéité est fait pour chacun d'eux. Chaque cadran non homogène est à son tour subdivisé, et ainsi de suite. Le résultat de cette subdivision peut être représenté par un arbre quaternaire. La racine de l'arbre représente l'image entière et les quatre fils de chaque nœud représentent ses quadrants. Ainsi, les feuilles de l'arbre représentent des blocs (sous....sous-quadrants) qui sont homogènes. Si à chaque nœud est associée la valeur moyenne des niveaux de gris du bloc qu'il représente, alors, l'arbre quaternaire résultant spécifie complètement une approximation de l'image en régions homogènes, dans laquelle chaque région est représentée par sa valeur moyenne.

Une telle représentation a l'avantage d'offrir une structure de données qui peut être parcourue très efficacement par certains algorithmes.

Le cas où t, seuil du critère d'homogénéité, est égal à zéro est particulièrement intéressant. Dans ce cas, un bloc n'est considéré comme homogène que si les valeurs de ses éléments sont égales, ainsi l'image originale peut être reconstruite de façon exacte à partir de sa représentation par un arbre quaternaire.

De façon générale, mais dans ce cas particulièrement, il est important de noter que la représentation d'une image par un arbre quaternaire peut être beaucoup moins compacte que sa représentation matricielle. Cet inconvénient peut, dans certains cas être contrebalancé par l'accélération des algorithmes de traitement.

Cependant, dans le cas où l'image est composée de vastes régions compactes, les arbres quaternaires deviennent très intéressants. La taille d'un arbre quaternaire dépend essentiellement de deux facteurs:

- la position et la taille des régions homogènes. Par exemple, une région constante de $2^k \times 2^k$ dont les coordonnées sont multiples de 2^k est représentée par un seul nœud dans l'arbre. Mais si sa position est décalée d'un pas en x et en y, il faut alors de l'ordre de 2^{k+2} nœuds pour la représenter [47, 64].
- la longueur totale des frontières entre régions homogènes [31].

Dans le système KIDS, les modèles utilisés dans les opérations de *"pattern matching"* relèvent typiquement d'une représentation par arbres quaternaires. Leur traitement cependant ne peut pas être fait de façon naturelle dans un processeur matriciel. Le paragraphe IV.7 indiquera comment ces données peuvent être traitées.

IV.3.4. Architecture matérielle pyramidale

Considérées sous l'angle de l'architecture matérielle, les structures pyramidales imposent trois contraintes:

a) une augmentation de l'ordre d'un tiers de la mémoire disponible.
b) la possibilité de considérer chaque niveau de la pyramide comme une image matricielle ordinaire.
c) la nécessité d'offrir un système matériel permettant la réalisation des fonctions V de construction de la pyramide et, plus généralement, de permettre des calculs faisant intervenir comme données ou comme résultats un père et ses quatre fils.

La contrainte a) n'est pas bien lourde et est simplement liée à l'utilisation de moyens ou de technologies appropriées. A l'allure où progressent les technologies VLSI, elle peut être considérée comme une contrainte

secondaire.

La contrainte b) exige, considérée dans un schéma de processeur matriciel, que celui-ci soit suffisamment souple pour permettre de travailler sur des images de taille variable. Ceci a pour effet:

- de sophistiquer le mécanisme d'adressage entre les processeurs de la matrice et la mémoire qui leur est associée.
- de nécessiter une fonction permettant de geler le fonctionnement de certains processeurs de la matrice.

La contrainte c) est peut-être la plus exigeante. Bien qu'en effet elle se prête naturellement à un traitement par un processeur matriciel, elle exige, au delà de la contrainte b), que dans une même opération, un opérande soit de taille $2^i \times 2^i$ et l'autre de taille $2^{i-1} \times 2^{i-1}$.

Le paragraphe IV.6 présente le circuit VLSI constituant le processeur matriciel et indique comment une judicieuse organisation des mémoires d'images, des parties opératives et de leurs mémoires d'opération permet de satisfaire ces contraintes.

IV.4. Le langage du système KIDS

L'annexe A1 présente une définition formelle de ce langage et peut être consultée en complément de ce paragraphe. Cette annexe appelle une remarque de fond: la méthode de définition, largement inspirée de celle d'Algol60, n'est pas, et de loin, ce que l'on sait faire de mieux en fait de définition de langage. En particulier, la sémantique du langage reste relativement floue puisque non formalisée. Cependant, cette méthode de définition a deux avantages qui ont été déterminants:

- la définition est facile à écrire et à corriger,
- la définition est facile à lire et à comprendre.

Dans ce paragraphe, la présentation du langage est faite de façon informelle comme pour répondre aux questions successives d'un utilisateur potentiel, essentiellement:

- de quels types de données dispose-t-on?
- quelles opérations sont définies sur ces données?
- quelles sont les structures du langage permettant de construire un programme?

IV.4.1. Types de données

Les données peuvent être de 3 types simples : ***ENTIER***, ***REEL*** et ***BOOLEEN*** ou de 4 types structurés: ***TABLEAU***, ***IMAGE***, ***MASQUE*** et ***PYRAMIDE***.

Les types ***ENTIER***, ***REEL*** et ***BOOLEEN*** sont identiques à ceux rencontrés classiquement dans les langages de type Algol. Exemples de déclarations:

```
ENTIER a, b, c;
REEL q;
BOOLEEN test;
```

Les tableaux de type ***TABLEAU*** sont des ensembles structurés de valeurs ***ENTIER***, ***REEL***, ***BOOLEEN*** ou ***MASQUE***. Leur déclaration doit spécifier complètement le nombre de dimensions et pour chaque dimension la valeur des bornes. Exemples de déclarations:

```
TABLEAU ENTIER ta(1:3), tb, tc(1:5, 1:5);
```

TABLEAU REEL rt(1:c);

TABLEAU BOOLEEN copymask(1:9);

TABLEAU MASQUE m(1:8);

Dans cet exemple, *tb* et *tc* ont la même définition. Le tableau *rt* contient *c* éléments, *c* étant déclaré et initialisé dans un bloc englobant. Trois fonctions sont disponibles: *dim(tableau)* qui délivre le nombre de dimensions du tableau indiqué en paramètre, *borneinf(t, i)* et *bornesup(t, i)* qui délivrent la valeur de la borne inférieure (respectivement: supérieure) de la dimension *i* du tableau *t*.

Les objets de type *MASQUE* sont utilisés pour réaliser des opérations de voisinage sur des images. Exemple de déclaration:

MASQUE vedge;

Exemple d'affectation:

vedge = (?,1,0,?,1,0,?,1,0);

Un masque est un ensemble de 9 valeurs logiques (a,b,c,d,e,f,g,h,i) correspondant à la topologie:

```
a b c
d e f
g h i
```

Les valeurs logiques du masque spécifient:
1 -> le point image correspondant doit être à 1
0 -> le point image correspondant doit être à 0
? -> le point image correspondant est indifférent.

Les objets de type *IMAGE* sont spécifiés par leur type et leur taille. Exemple de déclaration:

IMAGE scène TAILLE 512;

Cette déclaration définit une image binaire de 512 * 512 points. Une fonction : *taille(i)* est disponible et délivre la taille de l'image *i*.

Les objets de type *PYRAMIDE* sont spécifiés par leur type, la taille de leur plus grand élément et le nombre de niveau de la pyramide. Exemple

de déclaration:

```
PYRAMIDE p1 TAILLE 512 NIVEAU 4;
```

Cette déclaration définit une pyramide constituée de 4 niveaux.
p1(1) désigne une image binaire de 256x256 points.
p1(2) désigne une image binaire de 128x128 points.
p1(3) désigne une image binaire de 64x64 points.
p1(4) désigne une image binaire de 32x32 points.

L'objet *p1* est de type *PYRAMIDE*. L'objet *p1(1)* est de type *IMAGE*.

Il existe une fonction : *niveau(p)* qui délivre le nombre de niveaux de la pyramide *p*.

IV.4.2. Opérations

Les opérations sur des valeurs de type *ENTIER*, *REEL* ou *BOOLEEN* sont les opérations classiques.

Les opérations sur les masques sont soit des affectations:

```
m1 = m2;
m3 = (1,1,0,1,1,0,1,1,0);
```

soit une transformation par la fonction *rot*:

```
m1 = rot(m2, 2);
m3 = rot(m2, -1);
```

La fonction *rot* effectue autour du point central du masque une rotation. Le nombre de pas et son sens sont indiqués par le 2e paramètre. Si *m2* est le masque:

```
a b c
d e f
g h i
```

alors, l'effet des deux instructions ci-dessus sera:

```
        c f i
m1 =    b e h
        a d g

        d a b
m3 =    g e c
        h i f
```

Les opérations sur des objets de type *IMAGE* sont soit des opérations simples:

```
dif = i1 OUEX i2;
i3 = ↑( i1 ET ↑i2 OU i3);
```

soit des opérations masquées:

```
i1 = i1 OUEX i2 AVEC (m1,0);
i3 = i1 ET i2 AVEC (m1*m2, 1);
```

Pour une opération masquée, l'opération simple est d'abord évaluée. En chaque point de l'image I ainsi obtenue est appliqué le premier masque. Si les valeurs logiques spécifées par ce masque correspondent aux valeurs du voisinage de ce point dans l'image I, alors, le point correspondant dans l'image résultat R portera la valeur 1. Sinon, la même opération est éventuellement effectuée en utilisant le ou les masques suivants. Si aucun masque n'a correspondu aux valeurs présentes dans l'image I, alors le point correspondant dans l'image R portera la valeur du point de l'image I. Le deuxième paramètre de l'opération de masquage est une expression dont la valeur, 0 ou 1, indique la valeur des points extérieurs à l'image mais couverts par l'application des masques sur ses points frontières.

Des fonctions sont disponibles pour travailler sur des images:

i = convert(tb); tb = convert(i);

convertit le tableau booléen *tb* en une image binaire *i*, ou réciproquement.

a = marque(b);

crée dans *a* une image constituée de 0 et de un seul point à 1. La position de ce point dans *a* correspond à celui des points à 1 de *b* situé le plus en haut à gauche.

a = décale(b, nbpas, dir, frontière);

permet de décaler l'image *b* d'un nombre de pas *nbpas*, dans la direction *dir* (*dir=1*: haut, *=2*: droite, *=3*: bas, *=4*: gauche) en introduisant sur les bords de l'image les valeurs spécifiées par *frontière*. Si *frontière=0* on introduit des 0; si *frontière=1* on introduit des 1; si *frontière=2*, le décalage correspond à une rotation cylindrique, les valeurs chassées d'un côté étant réintroduites de l'autre.

b = vide(a);

b=VRAI si *a* est constitué exclusivement de 0, *b=FAUX* sinon.

surface = compte(a);

surface représente le nombre de points à 1 dans l'image *a*.

Les objets de type *PYRAMIDE* acceptent des opérations de construction. Soit:

```
ENTIER i,j;
PYRAMIDE p TAILLE 512 NIVEAU 3;
```

p(i) désigne un niveau de la pyramide et donc un objet de type *IMAGE*. La notation *p(i, j)* est utilisée pour les opérations sur ces objets de type *PYRAMIDE*. Dans ce cas, la valeur de *j* désigne un fils correspondant à la topologie:

0 1
2 3

Il est ainsi possible d'écrire:

p(2) = p(3,0);

qui construit le niveau 2 de la pyramide qui est un ré-échantillonnage de son niveau 3 en utilisant le fils 0. Il est encore possible d'écrire:

p(2) = hom(p(3));
ou
p2 = maj(p(3));

hom étant la fonction d'homogénéité et *maj* la fonction de majorité telles que définies plus haut.
Il est encore possible d'écrire:

p(3,*) = p(2);

Les quatre fils du niveau 3 reçoivent la valeur de leur père du niveau 2.

p(3,1) = p(2);

Le fils 1 du niveau 3 reçoit la valeur de son père.

Enfin partout où cela a un sens, un objet déclaré de type ***IMAGE*** peut être utilisé en lieu et place d'un niveau de pyramide:

i = hom(p(2)); i(*)= p(1);

IV.4.3. Structures du langage

Un programme est un bloc. Un bloc, délimité par ***DEBUT*** et ***FIN***, contient une liste de déclarations et une liste d'instructions. Les instructions sont:

- des blocs,
- des instructions composées (liste d'instructions délimitées par ***DEBUT*** et ***FIN***),
- des instructions d'affectation,
- des appels de procédure,
- des instructions conditionnelles, de la forme:
  SI expr-bool ALORS instruction SINON instruction FINSI;
- des instructions répétitives, de la forme:
  BOUCLE n FOIS TANTQUE (expr-bool);
  Liste d'instructions;
  FINBOUCLE;
  Les parties ***FOIS*** et ***TANTQUE*** sont optionnelles. On peut sortir d'une boucle ***BOUCLE*** par une instruction ***SORTIE*** ou passer au pas de boucle suivant par une instruction ***SUIVANT***.

IV.5. Architecture du système

IV.5.1. Organisation fonctionnelle

L'architecture du système KIDS est construite de façon à réaliser trois classes de fonctions:

- une fonction d'*interface* avec l'utilisateur,
- une fonction *langage*, de compilation ou d'interprétation du langage d'application présenté ci-dessus,
- une fonction *calcul* réalisant les opérations indiquées par les programmes.

Ces trois fonctions sont réalisées par une cascade d'unités fonctionnelles (cf figure 33).

Figure 33.
Schéma fonctionnel général du système KIDS.

L'idée générale de ce système en cascade est de mettre en œuvre une chaîne d'automates d'interprétation conduisant en fin de compte à l'utilisation directe d'une unité matérielle produisant le résultat demandé.

Dérivant de la cascade précédente, il est possible de proposer une structuration plus détaillée des modules fonctionnels. La figure 34 présente une telle organisation. Cette organisation fonctionnelle pourrait, certes, être encore détaillée. La figure 34 est proposée ainsi car elle correspond assez étroitement à la structure matérielle et logicielle exposée dans la suite de ce chapitre.

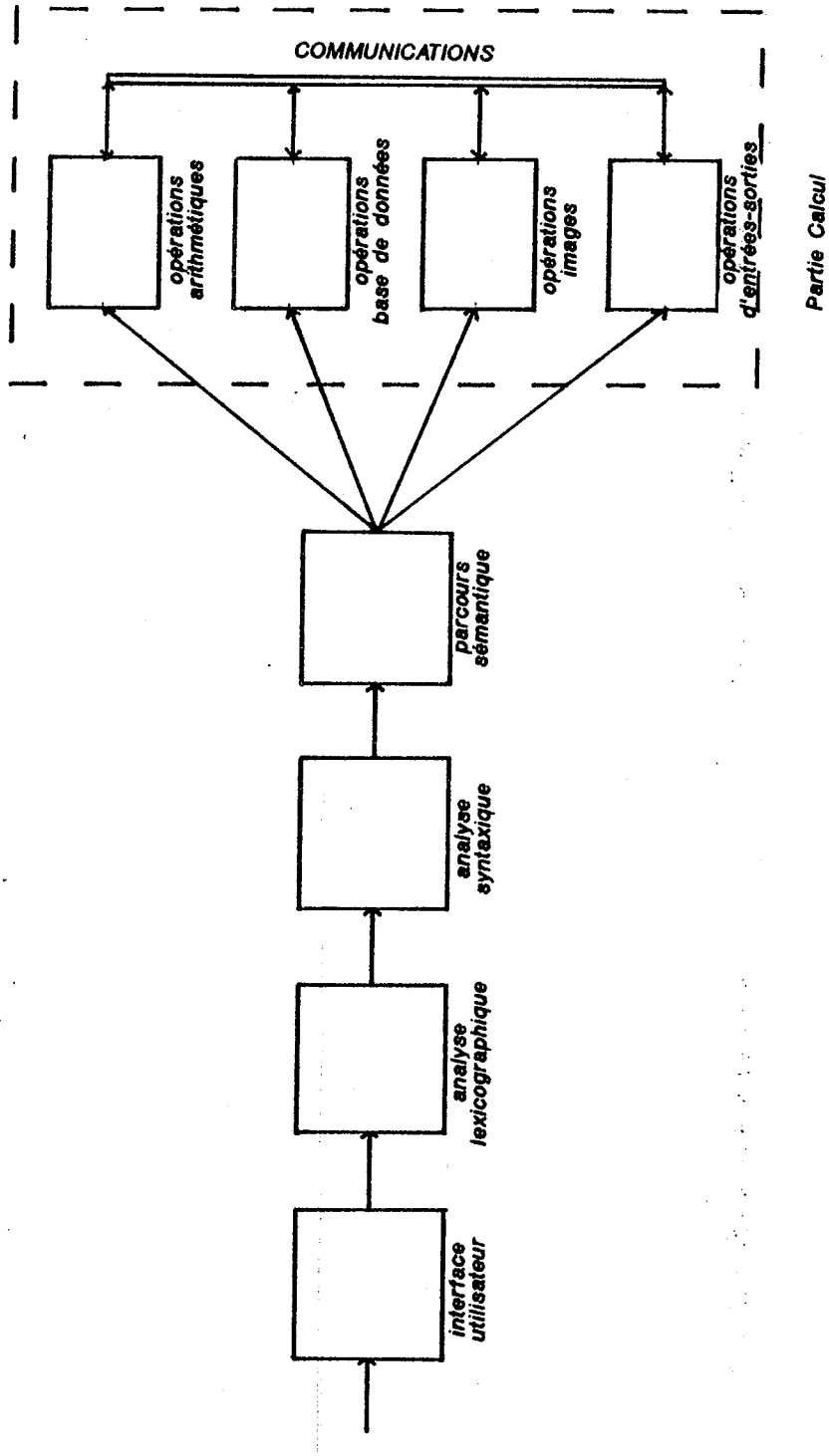


Figure 34.
Schéma fonctionnel détaillé du système KIDS

IV.5.1.1. Fonction interface.

L'unité fonctionnelle d'interface avec l'utilisateur peut, selon la qualité de la réalisation, prendre plus ou moins d'importance. Sa fonction minimum est de réaliser des fonctions système comme:

- dialogue avec l'utilisateur,
- système de gestion de fichiers,
- éditeur,
- commandes d'exécution d'un programme.

Reliée au terminal de l'utilisateur, cette unité accepte donc des commandes de type système. Son action est soit une transformation locale de ses fichiers, soit l'émission vers l'unité suivante d'une chaîne de caractères constituant le programme à exécuter.

IV.5.1.2. Fonction langage.

La fonction langage est réalisée par trois unités fonctionnelles successives:

- unité d'analyse lexicographique: accepte en entrée la chaîne de caractères constituant le programme et délivre en sortie une chaîne codée d'unités lexicales,
- unité d'analyse syntaxique: accepte en entrée une chaîne codée d'unités lexicales et délivre en sortie l'arbre abstrait du programme,
- unité de parcours sémantique: accepte en entrée l'arbre abstrait du programme et le parcourt en délivrant en sortie des demandes de calcul.

L'organisation algorithmique de ces trois dernières unités est une forme aujourd'hui classique du traitement en trois phases d'un langage algorithmique. Un exemple de cette organisation peut être trouvé dans [23].

Du fait de l'organisation en cascade, l'ensemble du système fonctionne au rythme de son unité la plus lente. Dans la plupart des cas concrets, il est prévisible que la partie *langage* de la cascade ainsi organisée en trois unités fonctionnera au rythme de l'unité lexicographique.

Cette limitation des performances peut être dépassée:

- soit en organisant à l'intérieur de la cascade des mécanismes permettant de stocker les différents textes intermédiaires échangés entre chaque unité. C'est une sorte de compilation partielle qui est ainsi réalisée et les textes intermédiaires stockés peuvent être réintroduits dans la cascade pour des exécutions ultérieures plus rapides.
- soit en remplaçant les trois unités langage de la cascade présentée ci-dessus par une structure série-parallèle plus complexe mais plus rapide. Des exemples de telles organisations existent, par exemple [5].

IV.5.1.3. Fonction calcul.

La fonction calcul est divisée sur la figure 34 en quatre sous-fonctions correspondant à des opérations de nature différente:

- sous-fonction de calculs arithmétiques: participe au fonctionnement de l'ensemble en gérant l'espace des noms et l'espace mémoire correspondant aux variables arithmétiques des programmes et en effectuant les opérations qui leur sont associées.
- sous-fonction de calculs image: participe au fonctionnement de l'ensemble en gérant l'espace des noms et l'espace mémoire correspondant aux variables images des programmes. Cette distinction fonctionnelle sous-entend une distinction matérielle: cette sous-fonction dispose de moyens spéciaux pour effectuer les opérations image; c'est là que prend sa place le processeur matriciel spécialisé pour le traitement d'images qui sera décrit plus loin.
- sous-fonction d'accès à une base de données: bien que les entrées-sorties ne soient pas décrites dans la définition préliminaire du langage donnée en annexe 1, cette fonction ne doit pas être oubliée en raison de sa particularité et des contraintes qu'elle implique pour une réalisation matérielle: connexion à un système de mémoire de masse, possibilité de transferts efficaces avec la partie de calcul sur les images.
- sous-fonction d'entrées-sorties d'images: il s'agit de réaliser là non seulement les fonctions de saisie et d'affichage, mais aussi celles d'adaptation réciproque des signaux analogiques

reçus ou produits aux définitions des objets image du programme (adaptation en terme de nombre de points dans l'image et nombre de bits par point).

Ces quatre sous-fonctions sont reliées entre elles à l'intérieur de la fonction calcul par un système de communication spécialement représenté car, dès ce stade de définition fonctionnelle, il apparaît que cette communication entre les quatre sous-fonctions doit être particulièrement efficace afin de ne pas ralentir le fonctionnement de l'ensemble lors des transferts de valeurs d'une sous-fonction à l'autre.

IV.5.2. Organisation matérielle.

L'organisation matérielle du système KIDS est conçue comme un reflet fidèle de l'organisation fonctionnelle. A chacune des fonctions ou sous-fonctions définies en IV.5.1 correspond une partie matérielle clairement définie sous forme d'un module physique réalisant la fonction souhaitée et présentant des interfaces de communication reflets de l'interface fonctionnel.

Comme il s'agit de réaliser un système matériellement important, il a semblé judicieux, afin de réduire les coûts de conception, de fabrication et de maintenance, d'utiliser, toutes les fois que c'était possible, des modules matériels identiques qui seront ensuite spécialisés par logiciel. Cette approche modulaire conduit à la définition, en première analyse, de trois modules principaux.

IV.5.2.1. Module élémentaire de cascade.

Ce module doit être utilisé pour réaliser, en tout ou en partie, chacune des fonctions ou sous-fonctions énumérées en IV.5.1. Son architecture matérielle est très rudimentaire et liée à la définition fonctionnelle suivante:

1) recevoir des données,
2) utiliser un ensemble de logiciels pour les transformer,
3) émettre des résultats.

Cela conduit à un schéma architectural illustré en figure 35.

Constitué autour d'un bus par un processeur et les mémoires de programmes et de travail, chaque élément de cascade établit une relation asynchrone avec l'élément précédent via une mémoire tampon du type file (*first-in, first-out*).

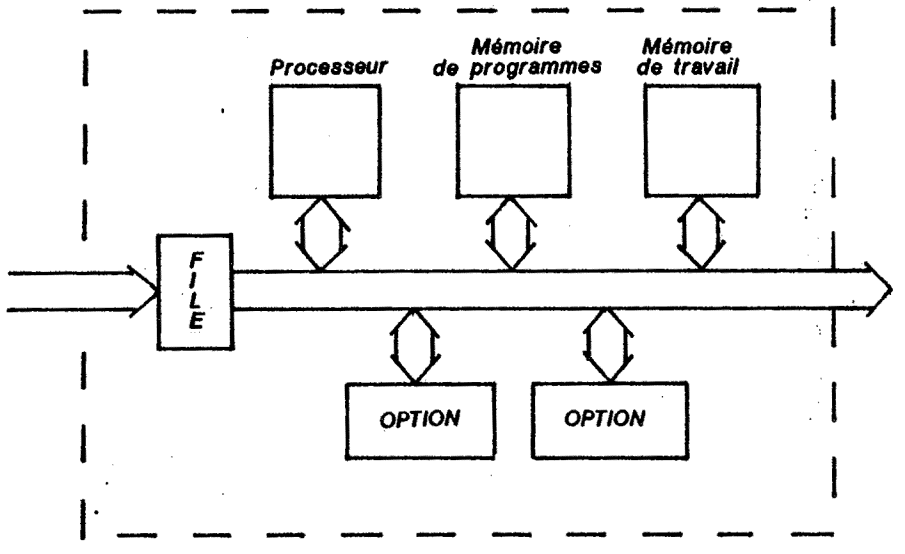


Figure 35.
KIDS: module élémentaire de cascade.

Les options permettent des extensions soit de l'espace mémoire soit de l'espace d'entrées-sorties.

Il est important de noter que:

- cette architecture est fondamentalement définie par son bus. Rien n'empêche de définir ce bus comme répondant à une norme existante.
- dans le même esprit, rien n'empêche de réaliser, matériellement, ce module élémentaire de cascade en utilisant des cartes standard disponibles sur le marché. Le choix de ces cartes est très libre, les fonctions demandées étant très largement répandues.
- si, pour des raisons diverses, il est choisi de réaliser entièrement ce module, les coûts de conception et de fabrication doivent rester faibles en raison de la simplicité du modèle architectural.
- quelle que soit la complexité logicielle de la fonction à réaliser sur cet élément de cascade, la dépendance du logiciel et du matériel reste très faible et, par ailleurs, la programmation

de tels éléments peut être faite de façon tout à fait classique en utilisant, éventuellement, des langages de haut niveau.

- la puissance nominale de chaque module élémentaire de la cascade est affaire de choix (dépendant d'une étude en simulation) et de contingences financières. Ainsi, la cascade peut varier depuis un ensemble de cartes à base de 6800 jusqu'à une série de VAX, par exemple.

IV.5.2.2. Module de communication.

Ce module est utilisé, lui aussi à plusieurs exemplaires, pour réaliser le système de communication entre les différentes sous-fonctions de la fonction calcul. La figure 36 présente le schéma de principe d'un tel module.

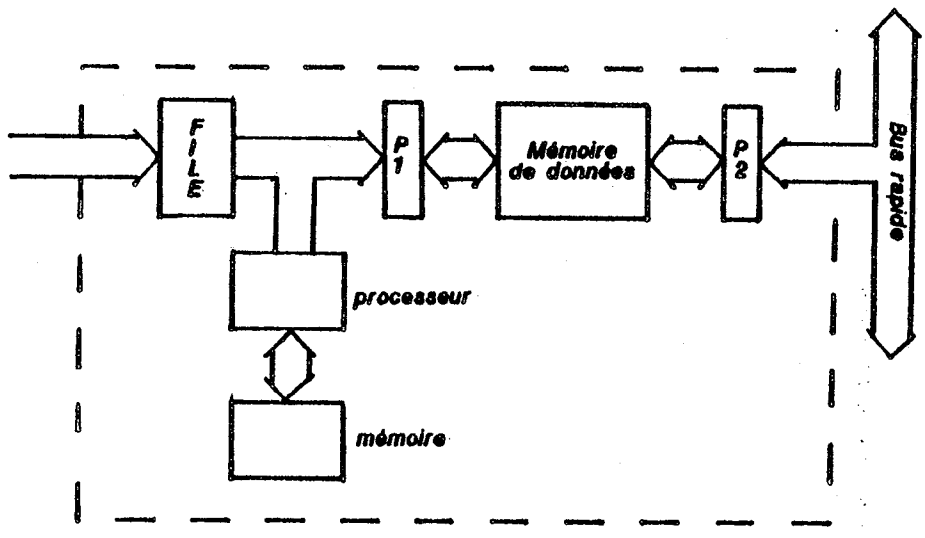


Figure 36.
KIDS: schéma de principe du module de communication.

Ce module comprend:

- un port d'entrée, conforme à la norme des modules élémentaires,
- un processeur et sa mémoire de programmes,
- une interface P1 d'accès à la mémoire de données.

- une mémoire de données: par exemple 512x512 bits,
- un interface P2 d'accès à un bus rapide.

Pour le bus rapide, permettant la communication entre les différents modules de communication, une seconde norme doit être utilisée. Vu les caractéristiques proposées aux paragraphes IV.5.2.3 et IV.6 concernant le processeur image, ce bus doit être capable de transférer 512 bits dans un temps de l'ordre de la microseconde.

Afin d'accomplir correctement ces performances, le processeur du module de communication peut être construit à l'aide de circuits *bit-slice*.

IV.5.2.3. Module de traitement d'images.

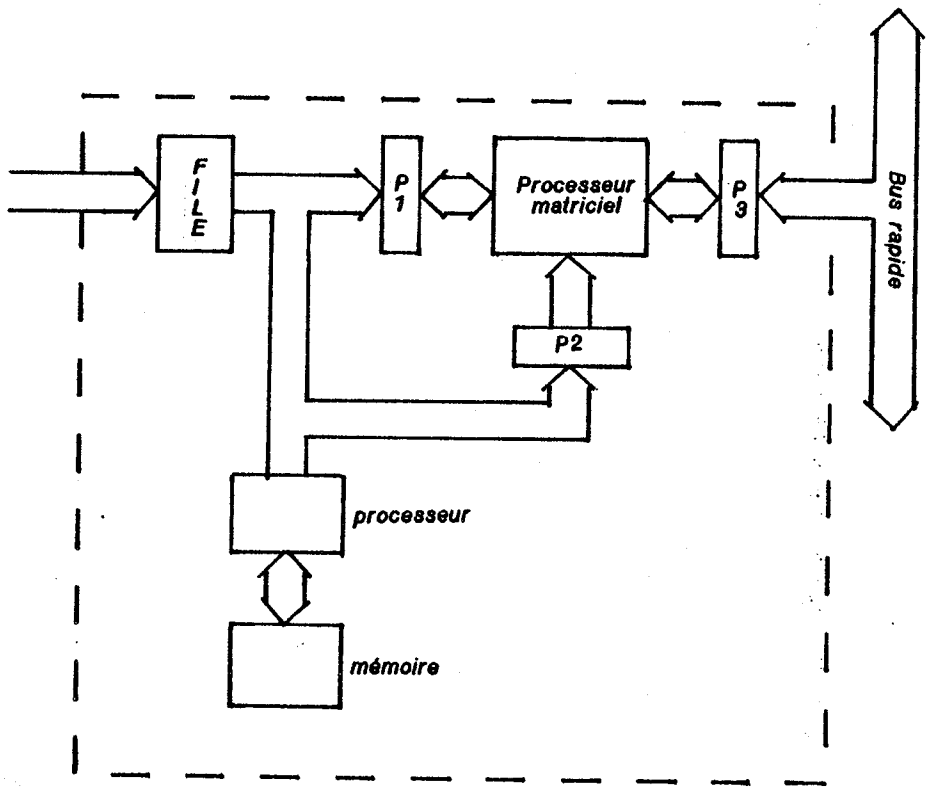


Figure 37.
KIDS: module de traitement d'images.

Cœur du système, ce module réalise les primitives de traitement d'images définies dans le langage. Son architecture, organisée autour d'un processeur matriciel de traitement d'images est présentée en figure 37.

Ce module comprend:

- un port d'entrée, conforme à la norme des modules élémentaires,
- un processeur et sa mémoire de programme,
- un interface P1 avec le processeur matriciel permettant de transmettre et de recevoir des signaux de contrôle,
- un interface P2 avec le processeur matriciel permettant de lui transmettre des instructions à exécuter,
- un interface P3, reliant le processeur matriciel au bus rapide.

La fonction essentielle de ce module consistera à:

- recevoir des ordres synthétiques de traitement d'images,
- transformer ces ordres et les paramètres associés en une suite d'instructions et de signaux de contrôle destinés au processeur matriciel,
- contrôler le fonctionnement du processeur matriciel.

Une bonne organisation des logiciels de ce module et des modules précédents doit permettre que la transformation *ordres synthétiques/suite d'instructions* soit très simple et consiste en une mise en correspondance de chaque ordre synthétique avec une suite d'instructions précodée en mémoire. Selon le cas et la valeur des paramètres, ces instructions sont ou non modifiées avant transmission au processeur matriciel et sont éventuellement répétées. Ce genre de fonctions peut être réalisé à l'aide d'un processeur *bit-slice*, éventuellement du même type que dans le module de communication.

Le processeur matriciel, quant à lui, est réalisé par une matrice 16x16 de circuits intégrés spécialisés dont l'architecture est décrite au paragraphe IV.6.

IV.5.3. Discussion.

Les paragraphes précédents ont présenté, très rapidement, le

schéma de l'architecture fonctionnelle et matérielle du système KIDS. Quelques points significatifs méritent d'être soulignés.

La réalisation sous forme de modules matériels permet une bonne correspondance avec l'organisation fonctionnelle, une grande facilité de conception, de réalisation ou de maintenance mais aussi de programmation des logiciels système.

Cette conception modulaire est relativement simple puisqu'elle ne fait appel qu'à:

- trois modules différents,
- deux normes d'interface (liaison type bus de module élémentaire et liaison type bus rapide),
- trois types de processeurs:
  - . processeur du module élémentaire,
  - . processeur des modules de communication et image,
  - . processeur matriciel,
- quatre langages de programmation:
  - . un langage d'utilisation: le langage d'application du système KIDS présenté ci-dessus,
  - . trois langages de programmation système:
    - * essentiellement un langage de haut niveau pour les modules élémentaires,
    - * une quantité limitée de programmes pour les processeurs *bit-slice* (du type assembleur),
    - * une faible quantité de codes ad hoc constituant des prototypes de suites d'instructions du processeur matriciel.

Pour peu que la norme d'interface de liaison du type *bus de module élémentaire* soit raisonnablement choisie en fonction des disponibilités du marché, l'ensemble du système peut être complété en faisant appel à des extensions standard: liaisons à un terminal, à une mémoire de masse, à un complément de mémoire, etc.

Par ailleurs, cette architecture est fondée, fonctionnellement et matériellement sur un schéma d'interprétations successives. Ce schéma permet d'une part de bonnes facilités de simulation et d'autre part la possibilité de réaliser le matériel pas à pas en remplaçant progressivement les modules logiciels de simulation par des modules

matériels. Ceci est d'une importance capitale. En effet, il semble exclu de mettre en œuvre la fabrication d'un tel système sans qu'aient été menées au préalable des études de simulation qui permettent de fixer quelques-uns des paramètres essentiels:

- puissance des processeurs,
- taille de mémoire nécessaire,
- largeur et débit des bus,
- etc.

Une bonne conception des outils de simulation doit enfin permettre de reprendre telle quelle ou d'adapter facilement la majorité des modules logiciels du simulateur pour en faire les logiciels système du matériel construit.

IV.6. Le processeur matriciel. Proposition pour un circuit VLSI.

Ce paragraphe se préoccupe des problèmes de réalisation du processeur matriciel de traitement d'images et propose, d'une façon aussi détaillée qu'il est réaliste de le faire avant d'avoir entrepris une réalisation, la description d'un circuit VLSI. Le paragraphe IV.6.1 réunit un ensemble de considérations préliminaires liées à des nécessités fonctionnelles, des contraintes de réalisation ou des choix méthodologiques. Le paragraphe IV.6.2 décrit l'architecture interne du circuit proposé.

IV.6.1. Préliminaires.

Comme l'essentiel des propositions faites dans ce chapitre, ce paragraphe et les suivants ne traitent que du problème des images binaires. Toutefois, et le paragraphe IV.7 reprendra ce point, les propositions de ce paragraphe IV.6 permettent de traiter des images grises considérées comme une succession d'images binaires organisées, reliées sémantiquement, c'est à dire par logiciel. Ce paragraphe de préliminaires rappelle d'abord les fonctionnalités essentielles et les opérations attendues du processeur matriciel. Les aspects topologiques liés à ces opérations sont ensuite examinés. Enfin, une troisième partie indique quelle politique a été prise vis à vis des contraintes liées à la conception et à la fabrication de circuits VLSI.

IV.6.1.1. Rappel des fonctionnalités.

Le processeur matriciel a pour but de réaliser de façon efficace, c'est à dire rapide, les opérations dites *expression image* et *expression pyramide* contenues dans la présentation du langage d'application du système KIDS. Il doit, par ailleurs, réaliser ou permettre la réalisation des fonctions de test (***vide***), de décalage (***décale***), de marquage (***marque***) et de construction de pyramide (***hom*** et ***maj***). Les opérations images sont de deux types:

- les opérations point par point, sur une ou plusieurs images: le processeur doit calculer une fonction booléenne construite avec les opérateurs **↑**, ***ET***, ***OU***, ***OUEX***
- les opérations supposant l'accès à un voisinage comme les

opérations masquées ou la fonction de décalage.

Les opérations pyramide peuvent être considérées comme étant soit des opérations image, soit des fonctions *hom* et *maj* réalisables par calcul booléen.

fonction *hom*:

(I0 ET I1 ET I2 ET I3)
OU
(Ī0 ET Ī1 ET Ī2 ET Ī3)

fonction *maj*:

(I0 ET I1 ET I2)
OU
(I1 ET I2 ET I3)
OU
(I2 ET I3 ET I0)

Les autres fonctions supposent des mécanismes matériels supplémentaires:

fonction *vide*:

consiste à réaliser un OU logique entre tous les points d'une image. Ceci peut être fait en deux phases: un *OU* par programme à l'intérieur de chaque circuit suivi d'un *OU* physique sur les résultats de chaque circuit.

fonction *compte*:

suppose l'existence dans la matrice de mécanismes additionneurs. Les résultats de ces additions doivent pouvoir circuler dans la matrice afin de réaliser des cumuls et des sorties de résultats.

fonction *marque*:

peut être réalisée en logiciel en utilisant la fonction *vide* et un mécanisme sélectif de test et d'activation des circuits.

Enfin, la quantité de mémoire nécessaire dans la matrice doit être suffisante pour contenir plusieurs pyramides binaires. Si la taille maximale d'une image est fixée à 512x512 points, 4096 K bits de mémoire permettent de contenir huit pyramides binaires (ou une pyramide grise à 8 bits) et plusieurs images binaires. C'est cette taille qui est retenue dans

la description qui suit. Il va de soi, cependant, que la quantité de mémoire contenue dans la matrice peut varier largement et dépend moins d'un principe de base que des possibilités d'intégration de la technologie choisie pour réaliser ces circuits.

IV.6.1.2. Aspects topologiques.

Comme les chapitres précédents l'ont montré, toute matrice de processeurs suppose une topologie liée aux opérations à effectuer et se traduisant en terme de connexions. Dans le cas du système KIDS, les aspects topologiques se présentent sous deux formes:

- topologie pyramidale: l'élément de base d'une topologie pyramidale est constitué, comme en figure 38, par quatre points mémoire fils, un point mémoire père et un processeur.

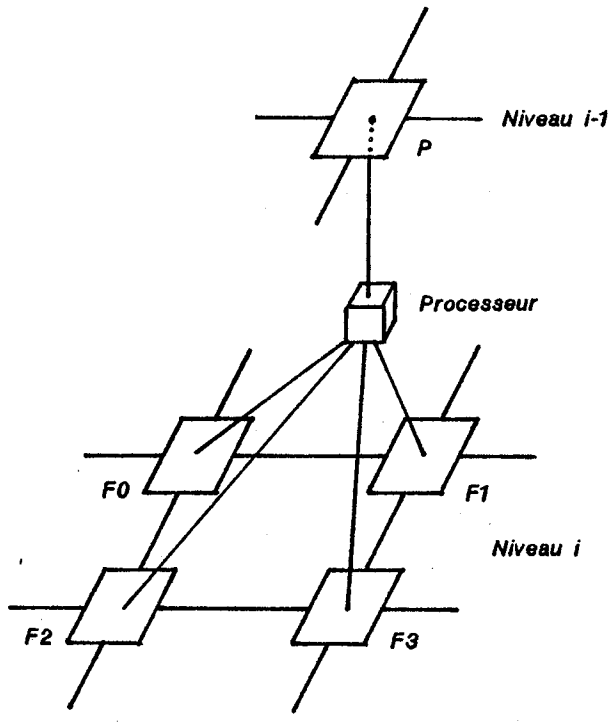


Figure 38.
Elément de base d'une topologie pyramidale.

Il est peut-être possible de construire des machines sur ce

modèle. Cependant, une telle organisation présente trois défauts majeurs:

- nécessité d'un très grand nombre de processeurs (pour une pyramide binaire complète sur une base de 512x512 points, il faut 256x256 + 128x128 + ... + 2x2 + 1 = 87.381 processeurs),
- très grande difficulté pour réaliser les connexions et la disposition des processeurs (voir à ce sujet [91, 92],)
- mauvaise utilisation du matériel réalisé: dans le cas général, un seul niveau de processeurs travaille à un instant donné soit, au mieux, 65.536 sur 87.381 (premier niveau) ou 16.384 sur 87.381 (deuxième niveau), etc.

Le choix a donc été fait de substituer à cette topologie pyramidale *physique* une topologie pyramidale *logique*. Pour cela, l'idée consiste à réaliser un seul niveau de processeurs associé à une quantité suffisante de mémoire pour contenir les données de plusieurs niveaux de la pyramide. La configuration pyramidale logique se fait en sélectionnant par logiciel les données des niveaux concernés.

Il est de plus expérimentalement vérifié que les niveaux supérieurs de la pyramide (par exemple à partir du niveau 16x16) sont fort peu utilisés. Une réalisation matérielle peut fort bien les omettre quitte, si nécessaire, à faire réaliser les algorithmes de traitement des niveaux supérieurs par le processeur de contrôle du processeur matriciel. Il s'agit certes là d'un mono-processeur, mais la perte de temps sera minime compte tenu de la très faible quantité de données mises en jeu.

Pour résumer, les problèmes d'architecture de la matrice liés à la topologie pyramidale peuvent être résolus:

1) en réalisant une organisation pyramidale logique et non physique,
2) en ne réalisant que les niveaux inférieurs de la pyramide.

Suivant ce principe, les problèmes de connexion sont largement simplifiés. Il suffit alors en effet:

. d'assurer un mécanisme d'adressage permettant à un processeur d'accéder à quatre points fils dans une image et à un point père dans une autre,
. de mettre en œuvre un mécanisme permettant d'inhiber le fonctionnement de certains processeurs. En effet, si le niveau I de la pyramide contient 512x512 points, un traitement pyramidal entre les niveaux I et I-1 nécessite 256x256 processeurs, mais un traitement entre les niveaux I-1 et I-2 ne nécessite plus que 128x128 processeurs, etc.
. d'assurer une organisation des processeurs et de la mémoire qui permette cette diminution progressive du nombre de processeurs actifs tout en conservant les possibilités d'accès *pyramidal 4 fils-1 père*.

Ces points seront explicités au paragraphe IV.6.2.

- topologie de voisinage: les voisinages concernés sont des fenêtres 3x3 comme définis dans le langage pour les opérations images masquées. Pour de plus grands voisinages, l'utilisateur fera appel à des décalages (par exemple: 9 décalages intercalés de traitements pour une fenêtre 5x5).

Considérées dans le cadre d'un processeur matriciel simple (1 point = 1 processeur = 1 circuit), les connexions nécessitées pour un tel voisinage s'élèvent, comme l'introduction du paragraphe sur CLIP4 l'a montré, au nombre de 9 et sont aisément réalisables. Les problèmes apparaissent lorsque l'on cherche à intégrer plusieurs processeurs dans un même boîtier. En effet, au delà de un, pour un boîtier contenant n^2 processeurs, il y a 4(n+1) connexions entrantes et 4(n-1) connexions sortantes, soit un accroissement éventuel du nombre de pattes du boîtier de 8n. Différentes techniques sont utilisables pour réduire ce nombre: lignes bidirectionnelles, multiplexage temporel... Le circuit proposé utilise ces techniques pour réaliser un circuit contenant, pour le niveau inférieur de pyramide, 32x32 points et ne nécessitant que 76, 44 ou 28 pattes (suivant les choix de multiplexage temporel) pour permettre aux algorithmes d'utiliser les 8n = 256 connexions nécessaires.

Il faut enfin noter, concernant ces connexions que les informations

transitant ainsi par l'extérieur des boîtiers, circulent beaucoup plus lentement qu'à l'intérieur (d'un facteur de 4 à 5). Ainsi, le plus grand soin doit être apporté dans l'organisation du matériel et du logiciel, visant à établir les informations valides sur ces lignes suffisamment avant leur utilisation, ce qui permet d'éviter des temporisations coûteuses en circuits d'horlogerie et néfastes en terme d'efficacité.

IV.6.1.3. Contraintes de réalisation et choix méthodologiques.

Ce paragraphe énumère un certain nombre de considérations qui ont guidé la conception du processeur matriciel et du circuit VLSI, sans distinguer ce qui relève de contraintes de réalisation ou de choix méthodologiques tant les unes et les autres sont liés. La présentation est simplement organisée pour aller du plus général (l'extérieur du processeur matriciel) vers le plus particulier (le dessin des circuits).

La première de ces considérations porte sur le choix du nombre de boîtiers pouvant constituer le processeur matriciel. A l'évidence, moins il y a de boîtiers, plus l'ensemble sera facile à assembler et peu onéreux. Le nombre de boîtiers est, très théoriquement, borné par le nombre de points d'image à traiter: 512x512 = 256 K. En supposant, ce qui est vraisemblablement optimiste, que l'on puisse intégrer dans un même boîtier 64 processeurs, leurs mémoires, leurs connexions, etc. le compte de boîtiers est ramené à 4096. Ce chiffre est encore beaucoup trop élevé et doit être réduit. A ce stade, deux directions sont possibles:

- ou bien revenir sur le choix fait de traiter des images de 512x512 points. Dans ce cas, avec l'hypothèse précédente, le nombre de boîtiers décroît rapidement et devient acceptable: une image 256x256 ne nécessite plus que 1024 boîtiers et une image 128x128, 256.
- ou bien, s'interroger sur la validité d'une politique de parallélisme *forcené* pour la réalisation du processeur matriciel.

C'est la deuxième direction qui a été suivie. Elle consiste à introduire, dans le contexte parallèle du processeur matriciel, une part de fonctionnement séquentiel à l'intérieur de chaque boîtier. Ainsi, pour un boîtier tel qu'il est présenté dans les lignes ci-dessus, contenant n^2 processeurs, il est possible de concevoir un boîtier équivalent qui,

passant d'une organisation totalement parallèle à une organisation partiellement séquentielle, contient seulement m processeurs qui répètent n^2/m fois une instruction ou une suite d'instructions. Un choix judicieux de m par rapport à n permet de ne pas trop compliquer l'algorithmique. Cette solution a deux conséquences:

- premièrement, si chaque boîtier contient toujours la même quantité de mémoire, d'une part il contient désormais beaucoup moins de processeurs -et est donc plus facile à intégrer-, d'autre part, le nombre de connexions nécessaires à un instant donné diminue puisqu'il n'y a plus que m processeurs actifs simultanément. Ainsi, par exemple, pour un boîtier totalement parallèle de n^2 processeurs passant à un boîtier série-parallèle de n processeurs, c'est à dire si l'on passe d'une matrice de processeurs à une ligne de processeurs, le nombre maximal de connexions nécessaires à un instant donné passe de 8n à 6+n (3 points à droite, 3 à gauche et, éventuellement, n points vers le haut ou le bas). L'établissement correct des valeurs sur ces connexions est affaire d'organisation matérielle et logicielle.

  En bref, ce passage à l'intérieur de chaque boîtier, à un mode de réalisation série-parallèle permet de diminuer:

  . le nombre de processeurs à intégrer,
  . le nombre de connexions inter-boîtiers.

  Il devient ainsi possible de réaliser des boîtiers couvrant une partie d'image beaucoup plus vaste. Le choix a été fait, dans KIDS, de réaliser des boîtiers couvrant une partie d'image de 32x32 points et contenant 16 processeurs. Ce qui porte, pour image 512x512, le nombre de boîtiers à 256, valeur tout à fait admissible.

- deuxièmement, l'introduction à l'intérieur de la matrice d'un fonctionnement séquentiel induit une augmentation du temps de calcul. Cette augmentation, d'un facteur de l'ordre de 64 dans la configuration retenue, pénalise l'ensemble du système: c'est, en quelque sorte, le prix payé pour une réalisation simplifiée.

La deuxième considération, en terme de contraintes de réalisation et

de choix méthodologiques, concerne la *bonne* utilisation du silicium, telle qu'elle a été évoquée au paragraphe II.5.5.4. Il fallait chercher à diminuer la surface de silicium *mal* occupée par les connexions (plots, lignes, amplificateurs, etc) au profit de la surface de silicium *bien* utilisée: mémoire, processeurs. La solution série-parallèle présentée ci-dessus va tout à fait dans ce sens.

La troisième considération concerne les techniques de conception et de dessin. En terme de conception, bien que le circuit proposé présente certains aspects exotiques ou inhabituels, il a semblé primordial de choisir, pour l'architecture interne du circuit, des solutions classiques, éprouvées. Par exemple, la mémoire doit être réalisée de façon classique, quitte à l'entourer de mécanismes particuliers de sélection ou d'adressage. Les processeurs eux-mêmes seront réalisés selon un modèle classique.

Ces choix permettent d'une part de bénéficier d'un savoir-faire certain [2] et d'autre part d'obtenir facilement une grande régularité dans le dessin. Par ailleurs, le choix de faire *classique* partout où cela est possible offre l'avantage considérable de pouvoir bénéficier, lors du dessin du circuit, de la puissance de construction d'outils comme LUCIE [42]. Ces outils permettent en outre d'intégrer dans le dessin des briques prédéfinies '[3] (par exemple: plots, points mémoire, étage d'additionneur, etc), d'automatiser certaines parties de conception (dessin et optimisation des PLA de la partie contrôle [22]) et encore de réaliser des optimisations globales [4]. Ces atouts doivent permettre de diminuer considérablement les temps de dessin et de mise au point du circuit.

IV.6.2. Architecture du circuit VLSI.

En première analyse, résultant des considérations du paragraphe précédent, le circuit VLSI proposé peut être décrit par le schéma par blocs présenté en figure 39a, complété, pour la partie centrale, par la figure 39b, qui peut être considérée comme un embryon de plan de masse du circuit.

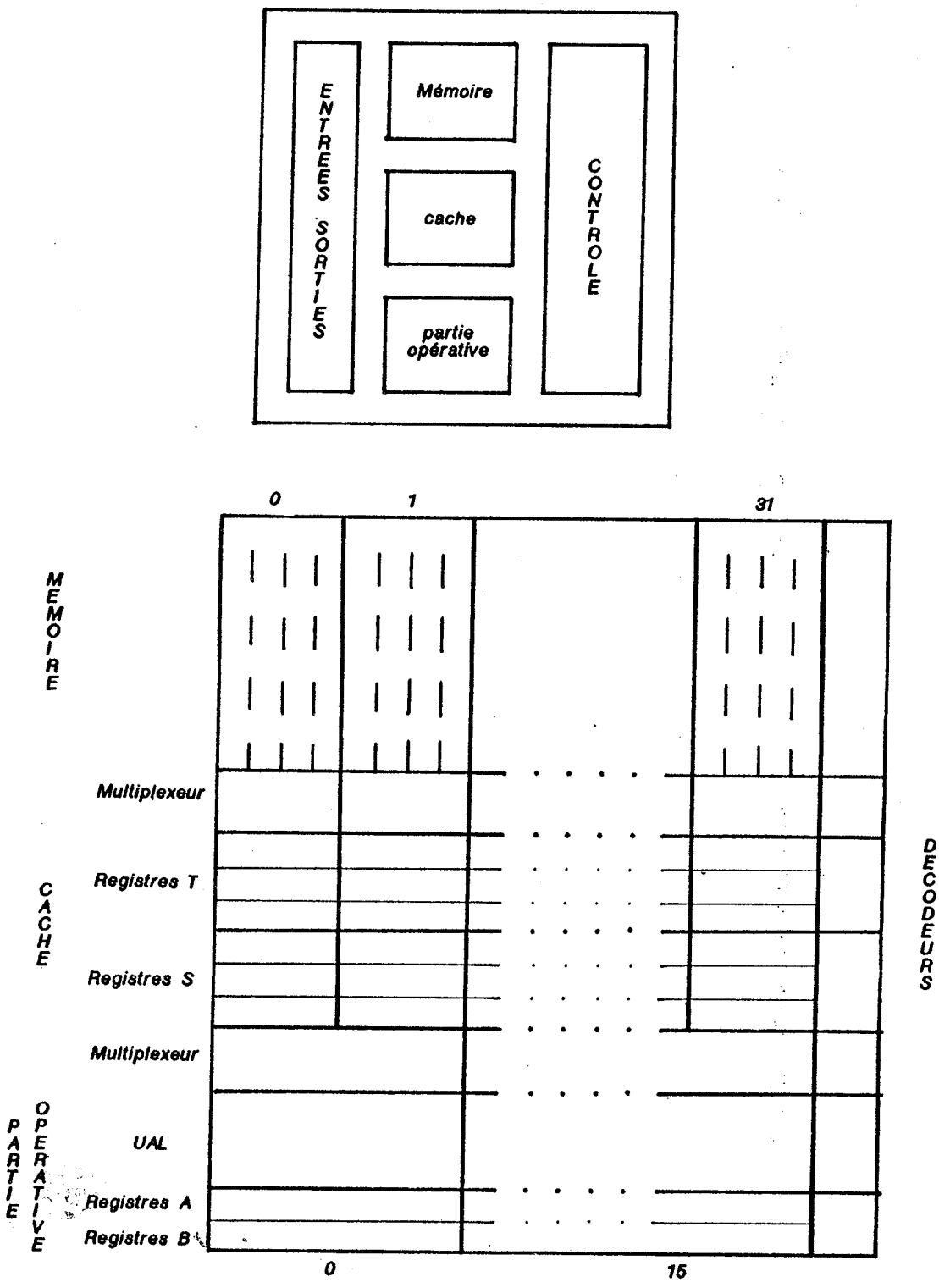


Figure 39.
KIDS: topologie du circuit VLSI.

Sur ce schéma, les données circulent verticalement dans la partie centrale. Les signaux de contrôle circulent horizontalement depuis la droite vers la gauche, et sont, progressivement remplacés par des lignes d'entrées-sorties de données.

Le bloc d'entrées-sorties, ne présentant pas de caractéristiques particulières, ne sera pas présenté en détail. Les quatre autres blocs: mémoire, cache, parties opérative et contrôle vont être présentés de façon plus détaillée dans les paragraphes suivants.

IV.6.2.1. La mémoire.

Chaque circuit comprend 16 K bits de mémoire statique organisée en mots de 32 bits. Ce bloc mémoire peut être construit librement de façon tout à fait classique. En particulier, sa taille et sa forme peuvent être largement modifiées pour satisfaire aux contraintes d'une optimisation globale. Son organisation générale peut avoir l'allure présentée en figure 40.

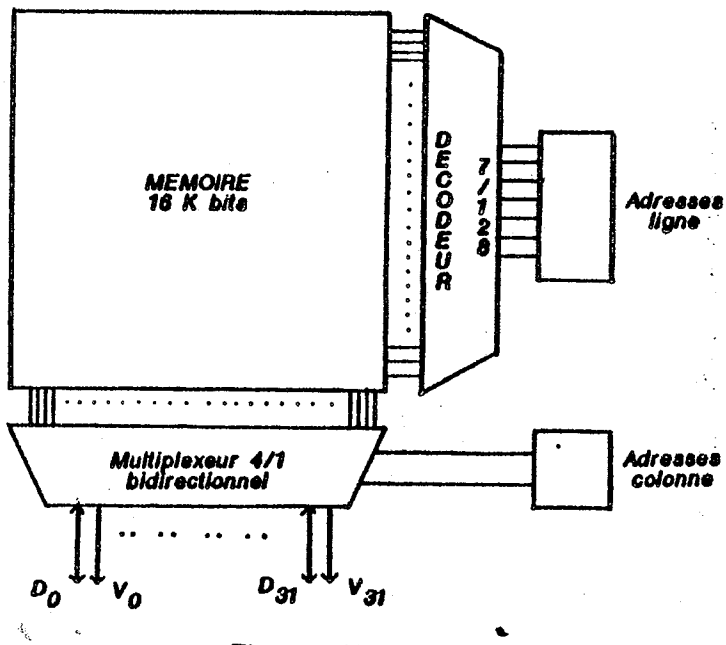


Figure 40.
KIDS: schéma de principe de la mémoire du circuit VLSI.

Les lignes de contrôle entrant par la droite sont au nombre de 11: 9 lignes d'adresses, une ligne de lecture/écriture, une ligne de validation.

Les lignes de données, bidirectionnelles, apparaissant en bas du schéma, sont au nombre de 32. Ces 32 lignes sont doublées de 32 lignes de validation dont l'utilisation sera présentée plus loin.

IV.6.2.2. Le cache.

Cette partie a été appelée *cache* faute de trouver une dénomination plus adaptée. Son organisation générale est présentée en figure 41.

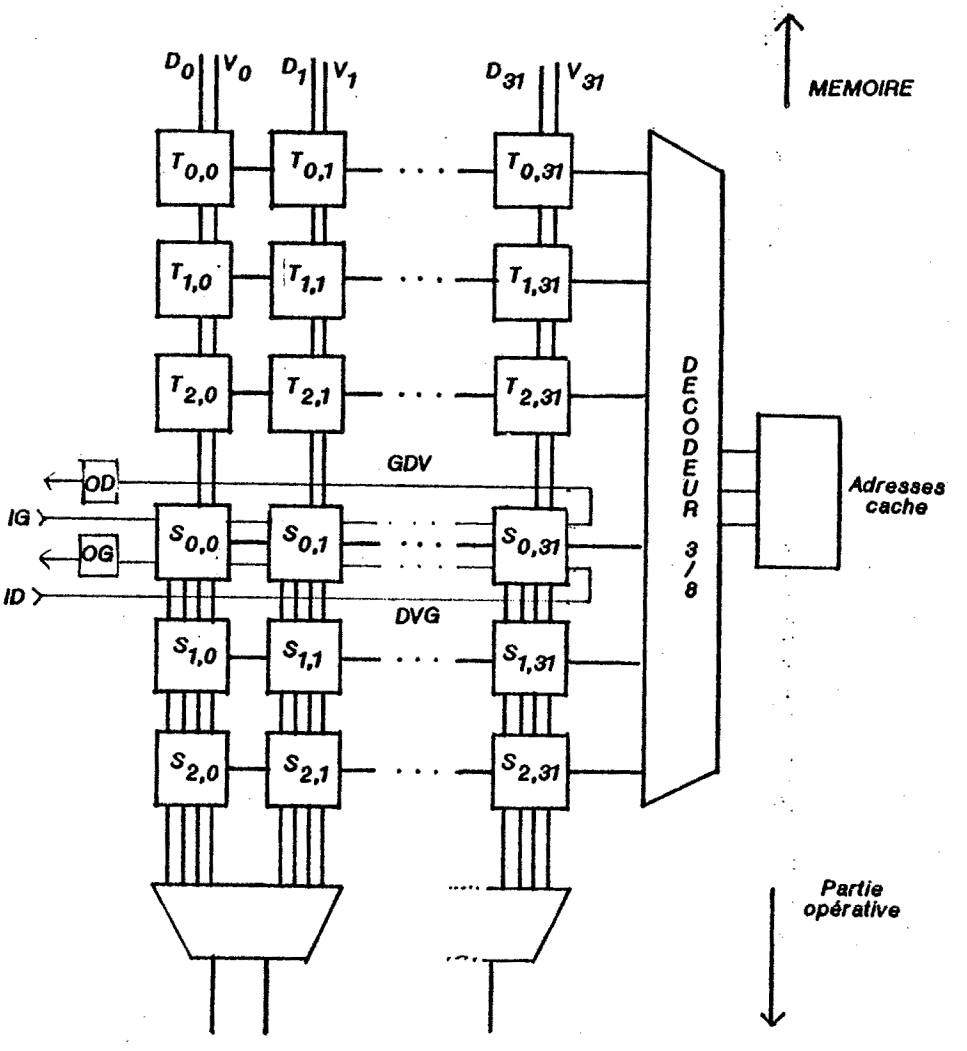


Figure 41.
KIDS: organisation générale du cache.

Son rôle est de réaliser un interface particulier entre la partie opérative et la mémoire, c'est à dire les fonctions suivantes:

- permettre des communications verticales dans la matrice. Pour cela, l'espace d'adressage du cache comprend deux registres d'entrées-sorties IO1 et IO2, de 32 bits. Ces registres sont physiquement installés dans la partie entrée-sortie et sont reliés aux lignes de données du cache (verticales sur le schéma) par des lignes horizontales.
- permettre les communications latérales dans la matrice lors des opérations masquées. Pour cela, le cache comporte 3 registres spéciaux S1, S2 et S3 de 32 blocs. Chaque bloc est constitué d'un point mémoire et de 3 multiplexeurs comme indiqué en figure 42.

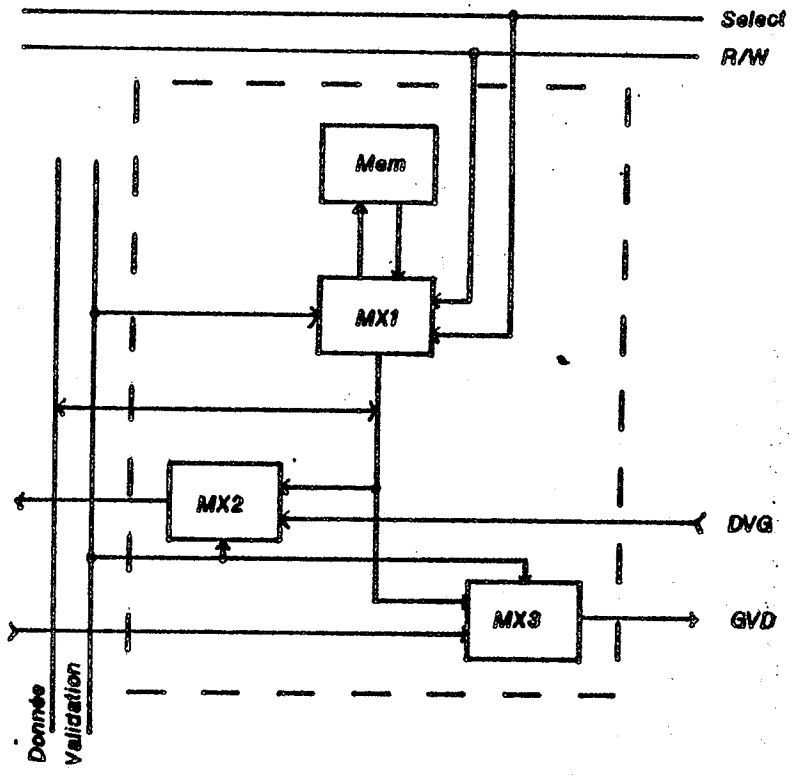


Figure 42.
KIDS: un bloc de registre spécial du cache.

Le fonctionnement de ce bloc est le suivant:

. la ligne "validation" n'est active que lorsque l'opération en cours porte sur la ligne de donnée associée.
. la ligne "select" n'est active que lorsque l'opération en cours porte sur ce registre.
. les lignes DVG (Droite Vers Gauche) et GVD (Gauche Vers Droite) transmettent dans chaque direction des valeurs qui, lorsque la ligne select est active sont celles du premier bloc à droite (respectivement à gauche) pour lequel la ligne "validation" est active. Si dans un bloc donné la ligne validation est active, les multiplexeurs MX2 et MX3 transmettent en sortie la valeur interne du bloc; sinon, c'est la valeur entrante de DVG (respectivement GVD) qui est transmise.
. Le multiplexeur MX1, bidirectionnel, est en haute impédance tant que "select" et "validation" ne sont pas actives. Sinon, il permet, selon l'état de R/W, la lecture ou l'écriture du point mémoire Mem.

Les 32 blocs de chacun des trois registres spéciaux sont organisés comme en figure 41.

Les lignes DVG et GVD sont connectées au bloc d'entrées-sorties, dans la partie gauche de la figure 41, pour permettre les connexions entre 2 circuits. Les blocs marqués OG et OD sont des bascules qui sont écrites à chaque opération d'écriture sur le registre.

- fournir trois registres de travail, T1, T2 et T3 de 32 bits. Chaque bit fonctionne sous le contrôle des lignes "validation", "select" et R/W vues précédemment.

- permettre à la partie opérative d'effectuer des lectures latérales lors des opérations masquées. Pour cela, 2 mécanismes complémentaires sont mis en place. D'une part, les blocs constituant les registres spéciaux sont complétés comme en figure 43. Les blocs A et B de la figure 43 permettent lorsque le registre spécial est sélecté, d'établir les valeurs correctes sur les lignes "donnée gauche" et "donnée droite". Ce mécanisme est complété par un multiplexeur, situé en fin de la partie cache à l'entrée de la partie opérative, qui permet de sélectionner, en lecture, une des 3 données.

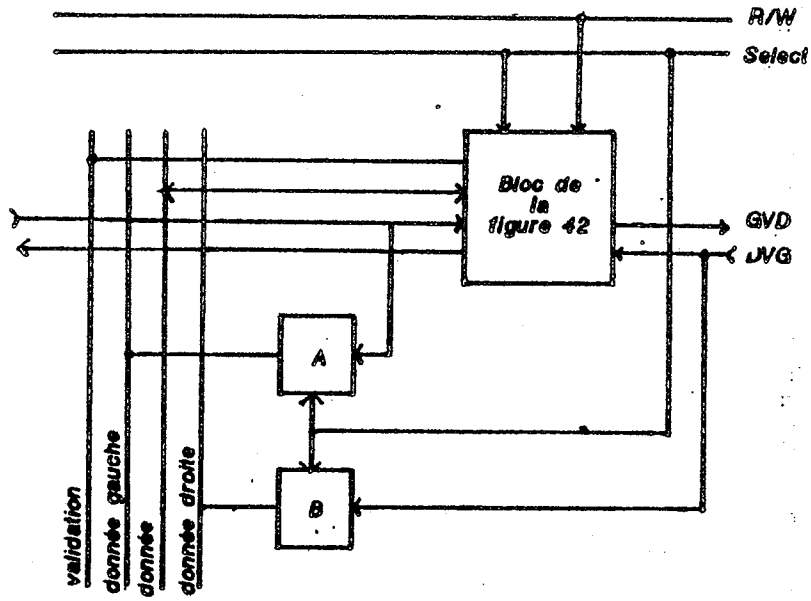


Figure 43.
KIDS: lecture latérale dans les registres spéciaux.

Bien que les blocs constituant les registres spéciaux soient peu orthodoxes, ils ne font appel qu'à des éléments classiques. Ainsi constitué, le cache peut être réalisé en suivant un modèle très régulier, formé de 32 colonnes identiques.

IV.6.2.3. La partie opérative.

La partie opérative est constituée de 16 blocs identiques. Chaque bloc est organisé comme en figure 44.

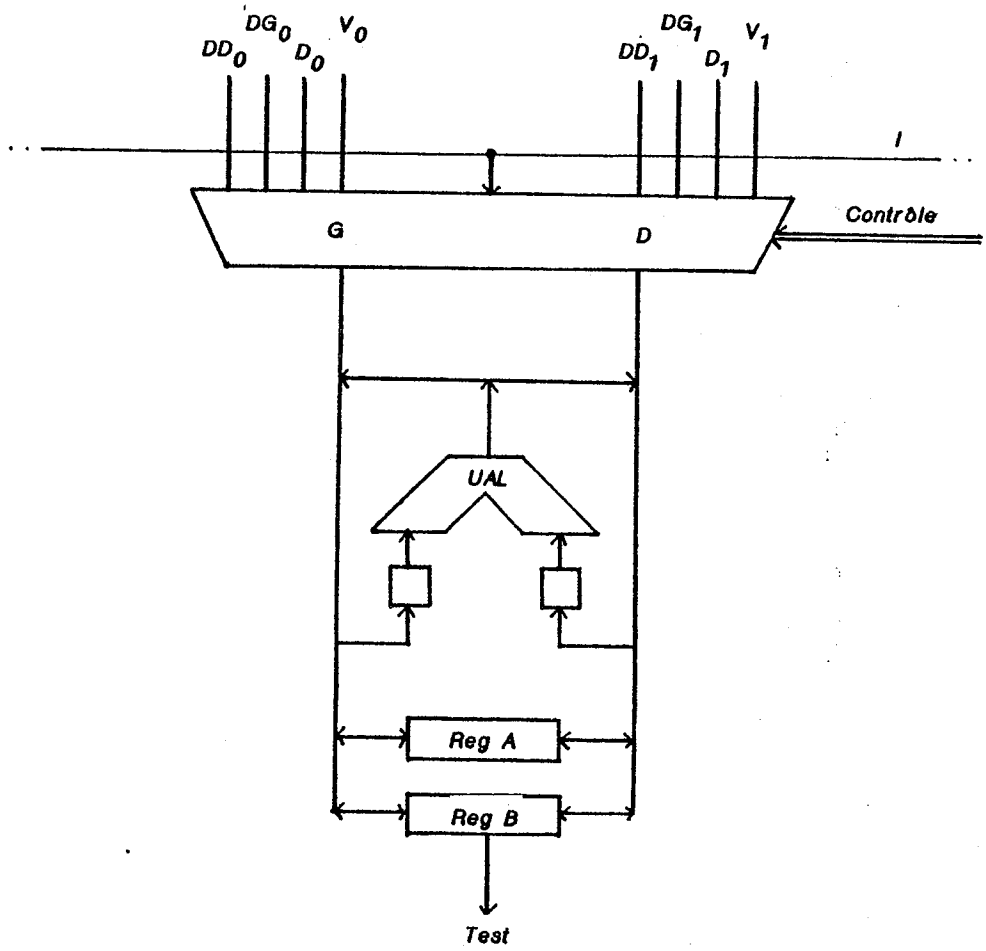


Figure 44.
KIDS: schéma d'un bloc de la partie opérative.

Ces blocs de partie opérative sont conçus pour exécuter des instructions rudimentaires de la forme:

code-op OP1, OP2, RES

où:

- code-op désigne soit:
  - une des 16 opérations logiques parmi les 8 suivantes et leurs compléments:

OP1	ET	OP2
OP1	OU	OP2
OP1	OUEX	OP2
↑OP1	ET	OP2

↑OP1	OU	OP2
↑OP1	OUEX	OP2
OP1	ET	↑OP2
OP1	OU	↑OP2

Ces instructions sont effectuées par le bloc marqué L.

. une opération d'addition, effectuée par le bloc marqué +.

- OP1 et OP2 spécifient les deux opérandes à l'aide de 3 champs d'adresse chacun:
  . champ1 (3bits): adresse du registre de cache concerné (T1, T2, T3, S1, S2, S3, IO1ou IO2).
  . champ2 (1 bit): indique si la donnée est à prendre sur les lignes gauches ou droites.
  . champ3 (2 bits): spécifie la fonction de sélection du multiplexeur concerné: choix entre les lignes D (donnée), DD (donnée droite), DG (donnée gauche) ou I. I est une ligne portant une valeur immédiate issue du bit de poids faible de champ1.

- RES spécifie le résultat et a la même forme que OP1 et OP2, mais, dans ce cas, champ3 spécifie toujours D.

Le schéma de partie opérative présenté en figure 44 n'est sûrement pas minimal pour cet ensemble de fonctions. Il a cependant été retenu pour deux raisons: d'une part, sa souplesse et sa généralité permettent une adaptation aisée à d'autres instructions qui s'avèreraient nécessaires lors de la poursuite du travail, d'autre part, il s'agit là d'un modèle classique qui, avantage considérable, peut être compilé par le système CAPRI [3].

Enfin, chacun de ces blocs est commandé, en ce qui concerne le séquencement et le choix des opérations par des lignes (horizontales et non représentées sur le schéma) issues de la partie contrôle, mais aussi par l'état d'une ligne associée à chaque bloc qui permet d'en geler le fonctionnement.

IV.6.2.4. La partie contrôle.

La partie contrôle a pour rôle d'assurer le séquencement et le décodage des instructions. Ces instructions sont de 4 types:

- transferts mémoire-cache,
- établissement du vecteur permettant de geler ou d'activer le fonctionnement de chaque bloc de la partie opérative,
- effectuer un OU entre les 16 lignes de tests issues des blocs de la partie opérative et transmettre le résultat à l'extérieur du circuit (ceci est utilisé pour réaliser la fonction ***vide*** définie dans le langage),
- instructions décrites au paragraphe précédent s'effectuant dans les blocs de la partie opérative.

Ainsi rapidement spécifiée, la partie contrôle doit être très simple à réaliser et occuper très peu d'espace. Il faut noter qu'il est possible de choisir de réaliser une partie contrôle plus sophistiquée comme cela sera évoqué au paragraphe suivant.

IV.6.3. Discussion.

Il serait long et fastidieux de reprendre maintenant les demandes fonctionnelles et de montrer comment, à l'aide de séquences éventuellement longues d'instructions, elles peuvent être réalisées par l'architecture matérielle proposée. Un travail ultérieur devra réaliser un simulateur du circuit et de la matrice et établira les programmes d'application nécessaires.

Pour l'heure, il est cependant possible de penser que la proposition faite permet d'atteindre les buts fixés. Les performances (en terme de temps) de cette solution mériteraient cependant d'être étudiées en détail avant toute réalisation. En effet d'une part le passage à un fonctionnement partiellement séquentiel, d'autre part la nature rudimentaire des parties opératives et contrôle, risquent de conduire à l'exécution de très longues séquences d'instructions dans la matrice et donc à des temps élevés. Ceci malgré les 4096 blocs opératifs travaillant en parallèle et le temps de cycle très faible que la simplicité d'un tel circuit permet d'espérer.

Il serait intéressant, en particulier, d'étudier la possibilité de sophistiquer la partie contrôle du circuit pour lui permettre d'effectuer des instructions répétitives ou mieux, des répétitions de suites d'instructions paramétrables. Une telle solution, dans l'esprit des propositions de *nouveaux langages d'instructions* développées aux paragraphes II.5.3 et II.5.4, diminuant le flot d'instructions et leur temps de chargement devrait permettre d'améliorer considérablement les performances de l'ensemble. Une telle partie contrôle, qui remplacerait le simple PLA actuellement nécessaire par un système utilisant une mémoire programmable de micro-instructions, est parfaitement réalisable.

Par ailleurs, le circuit proposé doit être examiné en terme de nombre de pattes. En première approche, les pattes nécessaires sont les suivantes:

. alimentation	IN	2
. horloge	IN	1
. reset	IN	1
. instruction prête	IN	1
. instruction	IN	21
. entrée horizontale droite	IN	3
. entrée horizontale gauche	IN	3
. connexion verticale haute	IN/OUT	32
. connexion verticale basse	IN/OUT	32
. sortie horizontale gauche	OUT	3
. sortie horizontale droite	OUT	3
. signal "vide"	OUT	1

Soit un total de 103 pattes. Ce chiffre est élevé mais n'est pas, aujourd'hui, trop élevé tant en terme de surface de silicium occupé par les connexions, qu'en terme de possibilité de mise en boîtier (*packaging*). De plus, il faut noter que ce nombre de pattes peut être diminué en utilisant une partie contrôle plus complexe qui permette de gérer un multiplexage temporel sur les lignes instruction et surtout sur les lignes d'échanges verticaux. Si par exemple les instructions sont lues en 2 fois et les échanges verticaux effectués en 2 voire 4 fois, le nombre de pattes nécessaires descend à 61 voire à 45, valeurs qui sont mieux que raisonnables.

IV.7. Miscellanées.

Ce paragraphe a pour but de donner quelques informations sur des points qui n'ont pas été évoqués ou l'ont été trop brièvement dans la description des paragraphes précédents. Certes, à la fin de ce paragraphe, tout n'aura pas été dit sur le système KIDS. Il convient de garder à l'esprit que ce chapitre n'est qu'une proposition pour un matériel à réaliser et n'ambitionne pas d'être un cahier des charges ou une notice technique, et que le travail restant à accomplir pour parvenir à un système matériel et logiciel opérationnel est encore immense.

IV.7.1. Les images grises.

Le traitement d'images grises dans le système KIDS peut être assez facilement obtenu en modifiant la définition du langage proposée en annexe 1 et en réalisant les modifications des logiciels système correspondants. La partie matérielle, quant à elle, ne devrait pas être modifiée car elle permet de traiter, par logiciel, des images grises considérées comme une succession de plans de bits, chacun assimilable à une image binaire.

IV.7.2. La reconnaissance de modèles.

La reconnaissance de formes par des techniques de *pattern matching*, c'est à dire en cherchant à obtenir le meilleur recouvrement possible d'un objet par un modèle doit pouvoir être un intéressant domaine d'application du système KIDS.

Pour cela, le langage du système doit être complété, en particulier par la définition d'un type ***MODELE*** et d'opérations primitives associées. Ces primitives doivent permettre:

1) de manipuler des modèles, soit en liaison avec une base de données de modèles, soit pour les transformer (variation d'échelle et d'orientation),
2) d'assurer des conversions entre une représentation sous forme d'arbres quaternaires (bien adaptées aux opérations du 1)), et une représentation matricielle (bien adaptée aux opérations du 3)), (voir à ce sujet [81]),

3) d'effectuer les opérations de recouvrement et d'en mesurer la qualité (par exemple en calculant la surface d'une image de différences -obtenue par un OUEX- entre un objet et un modèle).

Les opérations citées en 1) et 2) pourraient être réalisées dans un module fonctionnel et matériel particulier, sous-fonction supplémentaire de la partie calcul, assez semblable au module de calcul arithmétique augmenté d'une liaison à une mémoire de masse et éventuellement doté d'un complément de mémoire permettant la tabulation des opérations de rotation.

IV.7.3. Allocation mémoire dans le processeur matriciel.

Rien n'a été dit à ce sujet dans les pages précédentes bien que de nombreux problèmes puissent se poser. L'exemple suivant en révèle quelques-uns.

Soit une pyramide *p* déclarée ainsi:

```
PYRAMIDE p TAILLE 512 NIVEAU 3;
```

et soit à réaliser l'instruction:

```
p(2) := hom(p(1));
```

Un algorithme va réaliser l'opération *hom* de la façon suivante:

- charger dans T1 et T2 la première et la deuxième ligne de la portion d'image *p(1)* présente dans chaque circuit,
- calculer la fonction *hom* comme décrit en IV.6.1.1.
- ceci étant fait, chacun des 16 blocs de la partie opérative contient, dans son accumulateur, une valeur à ranger dans *p(2)*. C'est là que se pose un problème qui va être décrit plus loin,
- recommencer 15 fois pour les paires de lignes suivantes.

Un problème se pose, en effet, lorsqu'il s'agit de ranger un résultat puisque ce résultat ne comporte que 16 bits et que le cache, et au delà la mémoire, présentent 32 lignes. C'est là qu'apparaît la nécessité

d'une politique d'allocation mémoire dans la matrice. En effet, l'ensemble partie opérative-cache permet, tel qu'il a été présenté, en jouant sur les lignes de validation du cache, sur l'activation-désactivation sélective des blocs opératifs et sur les lectures latérales (qui permettent toutes formes de décalages), de ranger ces 16 bits à n'importe quel endroit, sans détruire les 16 autres bits non-concernés qui peuvent correspondre à une autre image de même taille. Ceci étant, il convient de minimiser le nombre d'opérations pour ranger le résultat. Ce nombre d'opérations est lié à la politique d'allocation mémoire choisie. Différents choix peuvent être faits, par exemple, dans ce cas, les 16x16 bits de $p(2)$ sont rangés sur 16 lignes successives de la mémoire soit:

- de façon contigüe (en utilisant les bits de poids forts ou les 16 bits de poids faibles du mot de 36 bits),
- de façon alternée (en utilisant les 16 bits pairs ou les 16 bits impairs du mot de 36 bits).

Il va de soit que le même genre de problème se pose lors du passage du niveau 2 au niveau 3 de la pyramide ou lors d'une opération s'effectuant sur des images de moins de 32x32 bits par circuit.

Une étude attentive du matériel proposé montre que bien que les deux solutions soient réalisables, la solution alternée est plus régulière et plus rapide car elle permet d'utiliser pleinement les mécanismes de communication latérale du cache et nécessite généralement moins d'instructions pour adapter les données.

Par ailleurs, une optimisation globale de l'allocation de mémoire peut être faite en amont du processeur de calcul image par le module de parcours sémantique, visant à faciliter la mise en corespondance des images et des niveaux de pyramides intervenant ensemble dans une expression ou un bloc de programme. Une bonne solution à ce problème, qui n'est ici qu'effleuré, est essentielle pour maintenir un niveau de performance correct.

IV.7.4. Les entrées-sorties d'images.

Il est essentiel que le processeur matriciel puisse être chargé et déchargé dans des temps très brefs. Ceci peut être réalisé par une séquence d'instructions appropriées utilisant les systèmes de

communication verticale de la matrice et par un environnement convenable réalisant l'interface avec le bus de communication rapide.

IV.7.5. L'environnement du processeur matriciel.

Le processeur matriciel doit être accompagné d'un ensemble matériel qui permette:

- la transmission des instructions,
- la transmission des signaux de contrôle,
- la lecture, sélective ou non, des lignes "vide",
- la possibilité d'activer chaque circuit sélectivement,
- la réalisation des fonctions de frontière pour les opérations de décalage et les opérations masquées,
- les entrées-sorties d'images.

Cet environnement, s'il ne pose pas de problème de principe, doit par contre être étudié attentivement en terme de temps de réaction. En effet, les lignes à contrôler sont longues et peuvent être connectées à un grand nombre d'entrées, c'est pourquoi la réalisation de cet environnement doit être soigneusement étudiée.

IV.7.6. Le parallélisme de la partie calcul.

Le système KIDS, tel qu'il est présenté dans ce chapitre, réalise, sous forme d'interprétations successives, l'évaluation d'un programme écrit dans le langage d'application. Une intéressante amélioration des performances pourrait être obtenue en faisant en sorte que le schéma d'interprétation prenne en compte le parallélisme physique existant dans la partie calcul entre ses différentes sous-fonctions. Pour cela, le langage devrait être augmenté de primitives d'expression de ce parallélisme d'évaluation, par exemple du type:

PAR e1 ; e2 ; e3 FINPAR

où *e1*, *e2* et *e3* sont des expressions pouvant être évaluées en parallèle par des sous-fonctions différentes de la partie calcul. L'évaluation n'est poursuivie après *FINPAR* que lorsque toutes les *ei* sont évaluées.

CONCLUSION

Le travail présenté dans les pages précédentes est consacré au traitement d'images dans ses rapports avec l'architecture des ordinateurs. Le traitement d'images est un domaine scientifique en plein développement qui est présenté, dans ses grandes lignes, au chapitre I, tant du point de vue des techniques de traitement que des domaines d'application. Cette présentation relève un certain nombre de contraintes (de temps et de quantités de données, essentiellement) et de spécificités algorithmiques, qui sont une interrogation directe pour le concepteur d'architectures d'ordinateurs.

Le chapitre II présente dans leurs principes et à l'aide d'exemples quelques unes des voies possibles pour la réalisation d'architectures spécialisées pour le traitement d'images. L'accent est mis sur la synergie entre matériel et logiciel et sur la nécessité de penser en termes de conception de systèmes, en alliant l'algorithmique et les langages de programmation aux architectures matérielles. Les différents paragraphes du chapitre II montrent les avantages qui peuvent être attendus d'une telle approche, mais aussi les contraintes auxquelles elle est soumise. Cette approche, faite ici dans le domaine du traitement d'images, serait sûrement aussi profitable dans d'autres domaines pour guider la conception

de systèmes informatiques spécialisés.

La machine ROMUALD, présentée au chapitre III, a été le premier pas de la réflexion qui a conduit à cette thèse. Ce multi-microprocesseur a pour ambition d'offrir des temps de réponse raisonnables et une grande souplesse d'utilisation pour des algorithmes simples. Son extention à la couleur en fait un outil de grand intérêt.

Le système KIDS, présenté au chapitre IV, est une proposition d'un système matériel et logiciel de plus grande importance. Ses points forts sont:

- un langage de programmation pour le traitement d'images directement évalué par le système,
- la possibilité d'effectuer les opérations -aujourd'hui classiques, mais rarement implémentées en matériel- de logique cellulaire,
- la possibilité -nouvelle- de traiter des pyramides d'images,
- sa puissance lors des opérations de traitement d'images grâce à son processeur matriciel.

Certes, le système KIDS reste à réaliser. Cependant, la définition proposée permet de se rendre compte qu'un tel système est réalisable tant pour le logiciel que pour le matériel en faisant appel à des moyens classiques et relativement simples.

La phase suivante de ce travail passe par la réalisation d'un simulateur qui permettra de tester la cohérence de l'ensemble, de juger sa facilité d'utilisation et de déterminer certains paramètres de réalisation.

La conception modulaire du système doit permettre de réaliser assez aisément ce simulateur. En particulier, il serait intéressant de réaliser le module de simulation du processeur matriciel de traitement d'images sur la machine ROMUALD qui offrirait, outre un temps et un coût de simulation raisonnables, des facilités d'entrées-sorties d'images et permettrait de faire fonctionner le simulateur sur des images et des problèmes concrets.

Bien sûr, la machine ROMUALD, et plus encore le système KIDS, sont -contrepartie de leur spécialisation- limités à la réalisation de certaines fonctions de traitement d'images.

Les fonctions réalisées, fonctions de prétraitement, de segmentation ou d'extraction de caractéristiques, sont certes l'effet d'un choix mais traduisent aussi un aspect de l'état de l'art du traitement d'images: il n'est possible de réaliser un matériel spécialisé que dans les domaines où les techniques algorithmiques et les primitives d'opérations sont précisément définies et universellement reconnues.

Les années qui viennent permettront vraisemblablement de savoir si l'utilisation d'architectures matérielles spécialisées en traitement d'images pourra s'étendre à d'autres domaines et à d'autres techniques, alors mieux définies.

Il n'est pas exclu qu'il n'en soit rien et que le rôle des ordinateurs d'usage général, à jeu d'instructions traditionnel, quelles que soient leurs puissances ou leurs architectures, reste important.

Alors, le rêveur impénitent, avide de nouveauté, aurait encore la ressource d'élaborer un processeur biologique qui, via les molécules d'ADN utilisées comme antennes, interprèterait directement les ondes des champs morphogénétiques, le tout selon la théorie avancée par R. Sheldrake [87]!

ANNEXE I

Définition formelle préliminaire du langage de programmation du système KIDS.

A1-1 Sommaire.

Cette annexe donne une description formelle de la version préliminaire du langage de programmation du système KIDS. Ce langage est susceptible d'exprimer une large classe d'algorithmes de traitement d'images sous une forme suffisamment concise. Il est fondamentalement conçu pour être évalué par le système KIDS dont les fonctionnalités répondent étroitement aux primitives du langage. Bien que s'en distinguant en de nombreux points, le langage et sa définition ont été largement inspirés par les langages de la famille Algol [1 et 71].

Le paragraphe A1-2 explique la notation formelle par laquelle la structure syntaxique est définie. Il donne aussi une vue d'ensemble des constituants de base et des caractéristiques du langage.

Le paragraphe A1-3 définit les identificateurs, les nombres et les chaînes. Quelques notions importantes telles que quantité et valeur sont également définies.

Le paragraphe A1-4 explique les règles de formation des expressions et donne la signification de ces expressions. Il existe 5 sortes d'expressions: les expressions arithmétiques, les expressions booléennes, les expressions image, les expressions pyramide et les expressions masque.

Le paragraphe A1-5 décrit les commandes opérationnelles du langage appelées instructions. Les instructions de base sont: instructions d'affectation (évaluation d'une formule), instructions procédure (appel pour exécution d'un processus fermé défini par une déclaration de procédure). La formation de structures plus complexes ayant un caractère d'instruction est expliquée. Celles-ci comprennent: les instructions conditionnelles, les instructions répétitives, les blocs et les instructions composées.

Dans le paragraphe A1-6 sont définies les unités appelées déclarations, servant à la description des propriétés permanentes des unités qui interviennent dans un processus décrit par le langage.

A1-2 Structure du langage

A1-2.1 Généralités.

Le but de ce langage est la description de processus de calcul et particulièrement dans le domaine du traitement d'images. Le concept de base utilisé pour la description des règles de calcul est l'expression arithmétique bien connue, comprenant des nombres, des variables, des fonctions et des opérateurs. A partir de telles expressions, on forme des instructions d'affectation.

Pour décrire l'enchaînement des calculs, il est nécessaire d'introduire certaines instructions qui ne sont pas des instructions de calcul. Elles permettent, par exemple, des choix ou des répétitions d'instructions de calcul.

Les instructions sont précédées par des déclarations qui ne sont pas elles-mêmes des instructions de calcul, mais qui définissent l'existence et certaines propriétés des éléments figurant dans les instructions, telles que la classe des nombres représentés par une variable, la dimension d'un tableau de nombres ou l'ensemble des instructions définissant une procédure. Une séquence de déclarations suivie d'une séquence d'instructions et délimitée par ***DEBUT*** et ***FIN*** constitue un bloc. Toute

déclaration apparaît dans un bloc de cette façon et est valide seulement dans ce bloc.

Un programme est un bloc.

A1-2.2. Formalisme de la description syntaxique.

La syntaxe est décrite à l'aide d'un métalangage dont les instructions sont de la forme:

‹méta-identificateur› ::= règle de production

Un méta-identificateur est une suite quelconque de caractères apparaissant entre ‹ et ›.

Le symbole ::= signifie "est défini comme".

La règle de production est une expression méta-linguistique formée de symboles non-terminaux: les méta-identificateurs, de symboles terminaux: construits avec les symboles de base du langage, le tout relié par des opérateurs méta-linguistiques. Les opérateurs méta-linguistiques sont:

|	qui signifie	ou bien
[x]	qui signifie	0 ou 1 fois x
{x}	qui signifie	0 ou plusieurs fois x

La succession de symboles terminaux ou non-terminaux dans une règle de production implique la concaténation du texte qu'ils représentent en fin de compte.

A1-2.3. Définition.

‹programme› ::= ‹bloc›
‹vide› ::= (c'est à dire la chaîne nulle de symboles)

A1-3 Identificateurs, nombres et chaînes, concepts de base.

A1-3.1. Lettres

<lettre> ::= **a|b|c|d|e|f|g|h|i**
|j|k|l|m|n|o|p|q|r
|s|t|u|v|w|x|y|z|é|è|à|ù

Les lettres n'ont pas de signification propre, on les utilise pour former les identificateurs et les chaînes.

A1-3.2. Chiffres.

<chiffre> ::= **0|1|2|3|4|5|6|7|8|9**

On utilise les chiffres pour former des nombres, des identificateurs et des chaînes.

A1-3.3. Valeurs logiques

<valeur logique> ::= ***VRAI*|*FAUX*|?**

Les valeurs logiques ont pour signification: vrai, faux et indifférent.

A1-3.4. Identificateurs.

<identificateur> ::= <lettre>{<lettre>|<chiffre>}

Les identificateurs n'ont pas de signification en eux-mêmes. Ils servent à l'identification des variables et des procédures.

A1-3.5. Nombres

<entier sans signe> ::= <chiffre>{<chiffre>}
<entier> ::= [+|-]<entier sans signe>
<partie décimale> ::= .<entier sans signe>
<facteur de cadrage> ::= *E*<entier>
<nombre décimal> ::= <entier sans signe>[<partie décimale>]

‹nombre sans signe› ::= ‹nombre décimal›[‹facteur de cadrage›]
‹nombre› ::= [+|-]‹nombre sans signe›

Le symbole de base *E* signifie: "fois 10 à la puissance".

Les entiers sont du type ***ENTIER***. Tous les autres nombres sont du type ***REEL***.

A1-3.6. Chaînes

‹chaîne propre› ::= ‹séquence de caractères ne contenant pas "›
| ‹vide›
‹chaîne ouverte› ::= ‹chaîne propre›{""‹chaîne propre›}
‹chaîne› ::= "‹chaîne ouverte›"

A1-3.7. Quantités, portées.

Parmi les quantités on distingue: les variables simples, les variables organisées et les procédures.

La portée d'une quantité est l'ensemble des instructions et expressions dans lesquelles la déclaration de l'identificateur associé à cette quantité est valide.

A1-3.8. Valeurs et types.

Une valeur est un ensemble organisé de nombres (cas particulier: un nombre isolé) ou un ensemble organisé de valeurs logiques (cas particulier: une valeur logique isolée).

Les divers types ***ENTIER***, ***REEL***, ***BOOLEEN***, ***IMAGE***, ***PYRAMIDE*** et ***MASQUE*** définissent des propriétés fondamentales de ces valeurs.

A1-4. Expressions

Les calculs à effectuer dans un programme sont décrits à l'aide d'expressions: les expressions arithmétiques, les expressions booléennes,

les expressions masque, les expressions image et les expressions pyramide. Ces expressions sont construites avec des valeurs logiques, des nombres, des variables, des indicateurs de fonction et des opérateurs.

<expression> ::= <expression arithmétique> |
<expression booléenne> |
<expression masque> |
<expression image> |
<expression pyramide>

A1-4.1. Variables.

<identificateur de variable indiçable> ::= <identificateur>
<identificateur de variable simple> ::= <identificateur>
<variable simple> ::= <identificateur de variable>
<liste d'indices> ::= <expression arithmétique> {,<expression arithmétique>}
<variable indicée> ::= <identificateur de variable indiçable>(<liste d'indices>)
<variable> ::= <variable simple> | <variable indicée>

Une variable est une désignation de valeur. On peut utiliser cette valeur dans des expressions pour former d'autres valeurs et on peut la changer au moyen des instructions d'affectation. Le type de la valeur d'une variable est défini par la déclaration concernant cette variable.

Les variables indiçables sont celles de type ***TABLEAU***, ***IMAGE*** ou ***PYRAMIDE***. La valeur à laquelle se réfère une variable indicée est déterminée par la valeur numérique des expressions arithmétiques de la liste d'indices. Ces expressions, appelées indices, doivent être de type ***ENTIER***. La valeur d'une variable indicée n'est définie que si les valeurs des indices sont compatibles avec la déclaration de la variable considérée.

A1-4.2. Indicateurs de fonction.

<identificateur de fonction> ::= <identificateur>
<liste de paramètres de fonction> ::= <expression arithmétique> [,<expression arithmétique>]
<indicateur de fonction> ::= <identificateur de fonction>(<liste de paramètres de fonction>)

Les indicateurs de fonction permettent le calcul de certaines fonctions. Ces fonctions sont définies par l'implémentation et le paragraphe A1-7 en donne une liste. Ces fonctions sont disponibles dans tout programme sans déclaration explicite.

A1-4.3. Expressions arithmétiques.

<opérateur additif> ::= + | -
<opérateur multiplicatif> ::= * | /
<facteur> ::= <nombre sans signe> | <variable> |
<indicateur de fonction> | (<expression arithmétique>)
<terme> ::= <facteur> { <opérateur multiplicatif> <facteur> }
<expression arithmétique simple> ::= [<opérateur additif>] <terme> { <opérateur additif> <terme> }
<expression arithmétique> ::= <expression arithmétique simple> |
SI <expression booléenne> *ALORS* <expression arithmétique> *SINON* <expression arithmétique> *FINSI*

Une expression arithmétique définit le calcul d'une valeur numérique. Lorsqu'il s'agit d'expressions arithmétiques simples, on obtient cette valeur en évaluant les opérations arithmétiques indiquées sur les valeurs numériques effectives des facteurs de l'expression.

La valeur des expressions plus générales de la forme *SI expb ALORS ea1 SINON ea2 FINSI* est celle de *ea1* si la valeur de *expb* est *VRAI*, celle de *ea2* sinon.

Mises à part les expressions booléennes, les constituants des expressions arithmétiques doivent être du type *REEL* ou *ENTIER*.

Les opérateurs +, -, * et / ont le sens habituel d'addition, soustraction, multiplication et division. Le résultat de +, - et * est de type *ENTIER* si les deux opérandes sont de type *ENTIER*, sinon le résultat est de type *REEL*. Le résultat de / est *REEL*. Les règles de priorité sont:

1) expressions entre parenthèses,
2) * et /,
3) + et -

A1-4.4. Expressions booléennes.

<opérateur de relation> ::= *INF* | *INFEG* | = | *SUP* | *SUPEG*
<relation> ::= <expression arithmétique simple> <opérateur de relation> <expression arithmétique simple>
<valeur logique déterminée> ::= *VRAI* | *FAUX*
<primaire booléen> ::= <valeur logique déterminée> | <variable> | <indicateur de fonction> | (<expression booléenne>)
<facteur booléen> ::= [↑] <primaire booléen>
<terme booléen> ::= <facteur booléen> [*ET* <facteur booléen>]
<opérateur additif booléen> ::= *OU* | *OUEX*
<expression booléenne> ::= <terme booléen> { <opérateur additif booléen> <terme booléen> | <relation>

Une expression booléenne définit le calcul d'une valeur logique. Les principes du calcul sont analogues à ceux donnés pour les expressions arithmétiques. Les variables et les indicateurs de fonction introduits en tant que primaires booléens doivent être de type *BOOLEEN*.

Les relations prennent la valeur *VRAI* quand la relation correspondante est satisfaite pour les expressions qu'elle contient; dans le cas contraire, elle prend la valeur *FAUX*. La signification des opérateurs logiques est donnée dans le tableau suivant:

b1	*FAUX*	*FAUX*	*VRAI*	*VRAI*
b2	*FAUX*	*VRAI*	*FAUX*	*VRAI*
↑b1	*VRAI*	*VRAI*	*FAUX*	*FAUX*
b1 ET b2	*FAUX*	*FAUX*	*FAUX*	*VRAI*
b1 OU b2	*FAUX*	*VRAI*	*VRAI*	*VRAI*
b1 OUEX b2	*VRAI*	*FAUX*	*FAUX*	*VRAI*

A1-4.5. Expressions masques.

<valeur de masque> ::= (<valeur logique>,
<valeur logique>,
<valeur logique>,
<valeur logique>,
<valeur logique>,
<valeur logique>,
<valeur logique>,

‹valeur logique›,
‹valeur logique›)

‹expression masque› ::= ‹indicateur de fonction› | ‹variable› | ‹valeur de masque›

Les masques sont des ensembles de valeurs logiques et sont utilisés dans les expressions image pour effectuer, en chaque point d'une image des opérations dépendant des valeurs de ses huit voisins immédiats.

Dans une expression masque, le seul indicateur de fonction utilisable est *rot*, qui désigne une fonction à deux paramètres:

rot (masque, nbpas)

Le premier paramètre est une variable de type *MASQUE* ou une valeur de masque.
Le second paramètre est une valeur entière.
La valeur définie par la fonction *rot* est une valeur de masque obtenue en effectuant une rotation du premier paramètre d'un nombre de pas indiqué par le second. Un second paramètre positif indique une rotation dans le sens trigonométrique, une valeur négative une rotation en sens inverse. Par exemple:

rot ((?,1,1,1,1,0,1,0,0),2)

délivre comme résultat:

(1,0,0,1,1,0,?,1,1)

soit une transformation du masque:

? 1 1		*1 0 0*
1 1 0	en	*1 1 0*
1 0 0		*? 1 1*

A1-4.6. Expressions image.

‹désignation d'image› ::= ‹identificateur› [(‹expression arithmétique›)]
‹opérateur image› ::= *ET* | *OU* | *OUEX*
‹expression logique image› ::= [!] ‹désignation d'image› [‹opérateur image› [!] ‹désignation d'image›]
‹expression élémentaire image› ::= !(‹expression logique image›) | ‹expression

logique image>

<partie masque> ::= *AVEC(*<expression masque>{*<expression masque>},<expression arithmétique>*)*

<expression de construction d'image> ::= <expression élémentaire image>[<partie masque>]

<expression image> ::= <expression de construction d'image> | <indicateur de fonction>

L'évaluation d'une expression de construction d'image se fait de la façon suivante:

1) évaluation de l'expression élémentaire image en effectuant d'abord l'évaluation de l'expression logique image puis, le cas échéant, une complémentation indiquée par l'opérateur ↑. Une expression logique image est évaluée de gauche à droite en effectuant d'abord les complémentations d'opérandes (opérateur ↑) puis, à priorité égale, les opérations *ET*, *OU* et *OUEX* qui sont identiques point par point aux opérations booléennes.
2) évaluation, en utilisant le résultat, obtenu ci-dessus, de la partie masque. La valeur image délivrée par une expression de construction d'image avec partie masque est la suivante:
   en chaque point de l'image calculée en partie gauche de l'opérateur *MASQUE*, on teste si ce point et ses huit voisins correspondent aux valeurs logiques de la première expression masque figurant en partie droite. Si tel est le cas, le point correspondant de l'image résultat portera la valeur 1 ; sinon, ce test est fait, le cas échéant, avec les valeurs logiques de la deuxième, de la troisième, etc expression masque. Si aucun de ces tests n'aboutit, la valeur du point de l'image résultat sera celle figurant dans l'image calculée en partie gauche.

   L'expression arithmétique spécifiée en deuxième membre de la partie droite, vaut 1 ou 0 et indique la valeur des points extérieurs à l'image mais couverts par l'application des masques sur les points frontières de l'image.

Une expression image constituée à l'aide d'un indicateur de fonction peut utiliser 3 fonctions:

1) *convert(tb)*: permet de créer une image à partir d'un tableau booléen *tb*.

2) *marque(image)*: permet de créer une image ne contenant qu'un seul point à 1. Ce point correspond au point de l'image fournie en paramètre portant la valeur 1 et situé le plus en haut à gauche.

3) *décale(image, nbpas, direction, frontière)*: permet d'effectuer un décalage de l'image du nombre de pas indiqué et dans la direction indiquée. Le paramètre *frontière* indique par sa valeur, 0, 1 ou 2 que les points entrant dans l'image du fait du décalage porteront les valeurs (respectivement) 0, 1 ou celles simultanément chassées par le décalage.

A1-4.7. Expressions pyramide.

‹désignation de fils› ::= ‹expression arithmétique› | *

‹objet image par quadrant› ::= ‹identificateur›([‹expression arithmétique›,]‹désignation de fils›)

‹expression pyramidale ascendante› ::= ‹objet image par quadrant› | ‹indicateur de fonction›

‹expression pyramidale descendante› ::= ‹identificateur image›

Les indicateurs de fonction dans une expression pyramidale peuvent être:

1) *hom(image)*: qui calcule la fonction d'homogénéité,
2) *maj(image)*: qui calcule la fonction de majorité.

A1-5. Instructions.

Les commandes élémentaires du langage sont appelées instructions. Les instructions sont normalement exécutées en séquence. Toutefois, cet ordre peut être changé par les instructions *SUIVANT* et *SORTIE* et par les instructions conditionnelles.

Les instructions peuvent être groupées en instructions composées ou en blocs.

A1-5.1. Instructions composées et blocs.

‹instruction de base› ::= ‹instruction d'affectation› | ‹instruction procédure›
‹instruction inconditionnelle› ::= ‹instruction de base› | ‹instruction composée› | ‹bloc›
‹instruction› ::= ‹instruction inconditionnelle› | ‹instruction conditionnelle› | ‹instruction de répétition›
‹bloc› ::= ***DEBUT*** ‹déclaration› {;‹déclaration›}; {‹instruction›;} ‹instruction› ***FIN***
‹instruction composée› ::= ***DEBUT*** {‹instruction›;} ‹instruction› ***FIN***

Chaque bloc introduit un nouveau niveau de définitions. C'est à dire que tout identificateur apparaissant à l'intérieur du bloc peut être déclaré dans ce bloc. Dans ce cas:

a) la quantité représentée par cet identificateur à l'intérieur du bloc n'a pas d'existence en dehors de celui-ci.
b) toute quantité représentée à l'extérieur du bloc par cet identificateur est inaccessible à l'intérieur de celui-ci.

Les identificateurs apparaissant dans un bloc et qui ne sont pas déclarés dans ce bloc représentent la même quantité à l'intérieur de ce bloc et dans le bloc qui l'englobe immédiatement, et récursivement puisqu'une instruction d'un bloc peut elle-même être un bloc.

A1-5.2. Instructions d'affectation.

‹partie gauche› ::= {‹variable›:=} ‹variable›:=
‹instruction d'affectation› ::= ‹partie gauche› ‹expression›

Les instructions d'affectation servent à donner à une ou plusieurs variables la valeur d'une expression. Toutes les variables d'une partie gauche et l'expression en partie droite doivent avoir le même type.

A1-5.3. Instructions conditionnelles.

‹instruction conditionnelle› ::= ***SI*** ‹expression booléenne›
ALORS ‹instruction›
[***SINON*** ‹instruction›]
FINSI

A1-5.4. Instructions de répétition.

‹instruction de répétition› ::= *BOUCLE* [‹expression arithmétique› *FOIS*]
[*TANTQUE* ‹expression booléenne›] ;
‹instruction dans BOUCLE›
{ ; ‹instruction dans BOUCLE› } *FINBOUCLE*

La syntaxe de ‹instruction dans BOUCLE› est donnée par la règle ‹instruction› du paragraphe A1-5.1. en remplaçant la règle ‹instruction inconditionnelle› par:

‹instruction inconditionnelle› ::= ‹instruction de base›
| ‹instruction composée› | ‹bloc› | *SUIVANT* | *SORTIE*

Ce remplacement affecte aussi l'utilisation de la règle ‹instruction› dans le paragraphe A1-5.3.

L'évaluation d'une instruction de répétition se fait de la façon suivante:

a) création d'une variable temporaire booléenne *fois-présent*, initialisée à *FAUX*.
b) si la partie fois est présente:
   1) *fois-présent := VRAI*
   2) évaluer l'expression arithmétique, créer une variable temporaire *compteur* initialisée à la valeur de l'expression (qui doit être de type *ENTIER*).
c) si *fois-présent = VRAI* alors *compteur := compteur - 1*
d) si *compteur < 0* alors continuer en h).
e) évaluer, si elle est présente, l'expression booléenne suivant *TANTQUE*. Si la valeur de cette expression est *FAUX*, continuer en h).
f) évaluer les instructions suivant la tête de boucle et précédant *FINBOUCLE*. Si l'instruction à évaluer est *SUIVANT*, continuer en c).
   Si l'instruction à évaluer est *SORTIE*, continuer en h).
g) reprendre en c).
h) évaluer l'instruction suivant *FINBOUCLE* ...

Des instructions de répétition peuvent être imbriquées. L'action des instructions *SORTIE* et *SUIVANT* ne porte que sur la boucle les contenant immédiatement.

A1-5.5. Instructions procédure.

<identificateur de procédure> ::= <identificateur>
<paramètre effectif> ::= <chaîne> | <variable>
<liste de paramètres effectifs> ::= (<paramètre effectif>{,<paramètre effectif>})
<instruction procédure> ::= *APPEL*<identificateur de procédure>[<liste de paramètres effectifs>]

Une instruction procédure provoque l'exécution du corps d'une procédure définie par sa déclaration.

La correspondance entre paramètres effectifs de l'instruction procédure et paramètres formels de la déclaration de procédure se fait comme suit:

a) la liste des paramètres effectifs doit avoir le même nombre d'éléments que la liste de paramètres formels,
b) les deux listes sont associées en prenant les éléments 2 à 2 dans le même ordre,
c) pour chaque couple paramètre effectif-paramètre formel:
   - paramètre effectif et paramètre formel doivent avoir le même type,
   - si le paramètre effectif est une chaîne, le paramètre formel correspondant ne pourra être utilisé que dans une liste de paramètres (procédures de sortie, par exemple),
   - dans tous les autres cas, l'utilisation du paramètre formel dans le corps de la procédure correspondra à l'utilisation ou à la modification des valeurs désignées par le paramètre effectif au moment de l'appel suivant le principe connu de passage de paramètres par référence.

A1-6. Déclarations.

Les déclarations servent à définir certaines propriétés des quantités utilisées dans le programme et à les associer à des identificateurs.

Dynamiquement, cela implique les règles suivantes: au moment de l'entrée dans un bloc (à travers le *DEBUT*), tous les identificateurs déclarés pour ce bloc prennent la signification impliquée par la nature des déclarations

données. Si ces identificateurs ont déja été définis par d'autres déclarations à l'extérieur, ils prennent à partir de ce moment là une nouvelle signification. D'autre part, les identificateurs qui ne sont pas déclarés pour ce bloc gardent leur ancienne signification. Au moment de la sortie d'un bloc (à travers *FIN* ou par une instruction *SUIVANT* ou *SORTIE*) tous les identificateurs qui ont été déclarés pour le bloc perdent à nouveau leur signification locale.

Mis à part les identificateurs de fonction, tous les identificateurs d'un programme doivent être déclarés. Aucun identificateur ne peut être déclaré plus d'une fois dans un même bloc.

‹déclaration› ::= ‹déclaration simple› | ‹déclaration de tableau› |
‹déclaration d'image› | ‹déclaration de pyramide› |
‹déclaration de procédure›

A1-6.1. Déclarations simples.

‹type simple› ::= *ENTIER* | *REEL* | *BOOLEEN* | *MASQUE*
‹déclaration simple› ::= ‹type simple›‹identificateur›{,‹identificateur›}

A1-6.2. Déclarations de tableaux.

‹borne inférieure› ::= ‹expression arithmétique›
‹borne supérieure› ::= ‹expression arithmétique›
‹paire de bornes› ::= ‹borne inférieure›:‹borne supérieure›
‹liste de paire de bornes› ::= ‹paire de bornes›{,‹paire de bornes›}
‹section de tableau› ::= {‹identificateur›,}‹identificateur›(‹liste de paire de bornes›)
‹liste de tableaux› ::= {‹section de tableau›,}‹section de tableau›
‹déclaration de tableau› ::= *TABLEAU*‹type simple›‹liste de tableaux›

Une déclaration de tableau sert à déclarer qu'un ou plusieurs identificateurs représentent des ensembles de valeurs et donne les dimensions des tableaux, les bornes des indices et le type des valeurs. Les expressions permettant de calculer les valeurs des bornes ne peuvent dépendre que de variables non-locales pour le bloc dans lequel la déclaration de tableau est valide. En conséquence, dans le bloc le plus extérieur d'un programme, on ne peut trouver que des déclarations de tableaux

avec des bornes constantes. Un tableau n'est défini que lorsque les valeurs de toutes les bornes supérieures ne sont pas plus petites que celles des bornes inférieures correspondantes.

A1-6.3. Déclarations d'images.

<section d'image> ::= { <identificateur>,} <identificateur>
TAILLE <expression arithmétique>
<liste d'images> ::= { <section d'image>,} <section d'image>
<déclaration d'images> ::= ***IMAGE*** <liste d'images>

Les images sont des matrices carrées de valeurs ***0*** ou ***1***. L'attribut ***TAILLE*** donne la taille du côté de la matrice, qui doit être 16, 32, 64, 128, 256 ou 512.

A1-6.4. Déclarations de pyramides.

<section de pyramide> ::= { <identificateur>,} <identificateur>
TAILLE <expression arithmétique>
NIVEAU <expression arithmétique>
liste de pyramides> ::= { <section de pyramide>,} <section de pyramide>
<déclaration de pyramides> ::= ***PYRAMIDE*** <liste de pyramides>

Les pyramides sont des ensembles d'images. Le nombre d'images dans une pyramide est défini par la valeur de l'expression suivant ***NIVEAU***. Soit I la valeur de cette expression. La pyramide contiendra I images. L'image de niveau I sera de taille t=valeur de l'expression suivant ***TAILLE***. L'image de niveau I-1 sera de taille t/2, celle de niveau I-2 de taille t/4, etc jusqu'au niveau 1.

A1-6.5. Déclarations de procédures.

<paramètre formel> ::= <identificateur>
<partie paramètre formel> ::= [(<paramètre formel>,} <paramètre formel>)]
<spécificateur> ::= ***STRING*** | [***TABLEAU***] <type simple> |
IMAGE | ***PYRAMIDE***
<partie spécification> ::= { <spécificateur> { <identificateur>,} <identificateur>;}
<tête de procédure> ::= <identificateur> <partie paramètre formel>;<partie

spécification>

‹corps de procédure› ::= ‹instruction›

‹déclaration de procédure› ::= *PROCEDURE*‹tête de procédure›‹corps de procédure›

Une déclaration de procédure sert à définir la procédure associée à un identificateur de procédure. Le constituant principal d'une déclaration de procédure est une instruction (qui peut être un bloc), qui, par l'emploi d'instructions procédure peut être appelée depuis d'autres parties du bloc dans la tête duquel apparaît la déclaration de procédure. Associée au corps, on trouve une tête qui précise tous les identificateurs apparaissant dans le corps en tant que paramètres formels. L'association paramètres effectifs/paramètres formels se fait suivant le principe du passage de paramètres par référence. Les identificateurs du corps de procédure qui ne sont pas formels sont soit locaux soit non-locaux pour le corps suivant qu'ils aient été déclarés dans ce corps ou non. Parmi ces identificateurs, ceux qui ne sont pas locaux pour le corps peuvent fort bien être locaux pour le bloc dans la tête duquel apparaît la déclaration de procédure. Le corps de procédure agit toujours comme un bloc, qu'il en ait la forme ou non. Si l'identificateur d'un paramètre formel est à nouveau déclaré à l'intérieur du corps de procédure, on lui donne ainsi une signification locale et les paramètres effectifs correspondants sont inaccessibles dans la portée de cette quantité locale intérieure.

A1-7. Fonctions prédéfinies.

Nom	Paramètres	Résultat
abs	*ENTIER*	*ENTIER*
	REEL	*REEL*
signe	*ENTIER*	*BOOLEEN*
	REEL	*BOOLEEN*
rac2	*ENTIER*	*REEL*
	REEL	*REEL*
sin	*ENTIER*	*REEL*
	REEL	*REEL*
cos	*ENTIER*	*REEL*
	REEL	*REEL*

arctan	*ENTIER*	*REEL*
	REEL	*REEL*
ln	*ENTIER*	*REEL*
	REEL	*REEL*
exp	*ENTIER*	*REEL*
	REEL	*REEL*
dim	*TABLEAU*	*ENTIER*
borneinf	*TABLEAU,ENTIER*	*ENTIER*
bornesup	*TABLEAU,ENTIER*	*ENTIER*
taille	*IMAGE*	*ENTIER*
niveau	*PYRAMIDE*	*ENTIER*
compte	*IMAGE*	*ENTIER*
vide	*IMAGE*	*BOOLEEN*
hom	*IMAGE*	*IMAGE*
maj	*IMAGE*	*IMAGE*
marque	*IMAGE*	*IMAGE*
décale	*IMAGE,ENTIER,ENTIER,ENTIER*	*IMAGE*
rot	*MASQUE,ENTIER*	*MASQUE*
réel	*ENTIER*	*REEL*
entier	*REEL*	*ENTIER*
convert	*TABLEAU BOOLEEN*	*IMAGE*
	TABLEAU BOOLEEN	*MASQUE*
	IMAGE	*TABLEAU BOOLEEN*
	MASQUE	*TABLEAU BOOLEEN*

Certaines de ces fonctions ont déja été définies, pour les autres, ce sont les fonctions usuelles de l'analyse:

abs(expr) pour le module (valeur absolue) de la valeur de l'expression *expr*.

signe(expr) pour le signe de la valeur de *expr* (+1 pour *expr* positif, 0 pour *expr* nulle et −1 pour *expr* négatif).

rac2(expr) pour la racine carrée de la valeur de *expr*.

sin(expr) pour le sinus de la valeur de *expr*.

cos(expr) pour le cosinus de la valeur de *expr*.

arctan(expr) pour la valeur principale de l'arc tangeante de la valeur de *expr*.

ln(expr) pour le logarithme népérien de la valeur de *expr*.

exp(expr) pour la fonction exponentielle de la valeur de *expr*.

réel(expr) pour obtenir une valeur de type *REEL* de la valeur de l'expression entière *expr*.

entier(expr) pour obtenir la valeur de type *ENTIER* la plus proche de la valeur de l'expression réelle *expr*.

A1-8 Exemple d'utilisation

Cet exemple est donné ici afin de permettre au lecteur de se faire une idée de la commodité du langage lors de la programmation d'une application. Les algorithmes utilisés ne cherchent pas à être optimaux mais exploitent cependant de façon simple les facilités essentielles du système KIDS. En particulier, l'utilisation de la structure pyramidale se montre commode pour éliminer, certes grossièrement, le bruit, et efficace lors de la phase de dilatation d'une tache.

Le problème proposé est le suivant: à partir d'une image originale, supposée contenir des taches à 1 sur un fond de 0, sélectionner les taches les plus importantes et compter ensuite leur nombre, puis, pour chacune d'entre elles, calculer sa surface et son périmètre extérieur.

Le principe de l'algorithme est le suivant:

1) on construit dans la pyramide *p1*, utilisée de façon ascendante, un ensemble d'images permettant de sélectionner dans l'image originale les taches qui contiennent au moins un carré de 8x8 points. Ceci est réalisé par la procédure *sélection*.
2) la pyramide *p2* est ensuite utilisée de façon descendante pour obtenir une copie d'une des taches sélectionnées. Pour cela:

    2a) une image est créée dans *p2(1)* par la fonction *marque*. Ceci correspond à la sélection, dans l'image *p1(1)* de la tache la plus haute et la plus à gauche.

    2b) à chaque niveau, la marque présente dans *p2* est étendue jusqu'à correspondre à la tache référence présente

au même niveau dans la pyramide ***p1***.

2c) on construit alors le niveau inférieur dans ***p2***, s'il existe et on reprend en 2b).

3) l'image ***p2(1)*** contient alors la copie d'une tache sélectionnée (et d'une seule). Sa surface est calculée par la fonction ***compte***. Son périmètre extérieur est fourni par un appel de la procédure ***mesper***.

4) la tache mesurée est éliminée de l'image ***p1(1)*** et le processus reprend en 2), tant qu'il reste des taches à mesurer.

La procédure ***dilatation*** appelée par les procédures ***étendre*** et ***mesper*** utilise une famille de huit masques obtenus par rotation et permettant de détecter les points du fond (à 0) et touchant une tache (à 1) et de les remplacer par des 1, c'est à dire de créer autour de la tache initiale une couche de 1 supplémentaire.

La construction de la famille de masques, présentée ici dans la procédure afin de faciliter la lecture, mériterait, par souci d'efficacité, d'être effectuée une seule fois dans le bloc englobant qui contient les déclarations de toutes les procédures présentées.

```
PROCEDURE extraction (imageoriginale, nombredetaches, surfaces, périmètres);
IMAGE imageoriginale;
ENTIER nombredetaches;
TABLEAU ENTIER surfaces, périmètres;
DEBUT
    PYRAMIDE p1, p2 TAILLE taille(imageoriginale) NIVEAU 4;
    nombredetaches := 0;
    p1(4) := imageoriginale;
    APPEL sélection(p1);
    BOUCLE TANTQUE ↑vide(p1(1));
        nombredetaches := nombredetaches + 1;
        p2(1) := marque(p1(1));
        i := 1;
        BOUCLE;
            étendre(p2(i), p1(i));
            SI ↑(i=4) ALORS
            DEBUT
                p2(i+1,*) := p2(i);
                i := i + 1
            FIN
            SINON
                SORTIE
            FINSI
        FINBOUCLE;
```

```
            surfaces(nombredetaches) := compte(p2(4));
            APPEL mesper(p2(4)), périmètres(nombredetaches));
            p1(1) := p1(1) OUEX p2(1)
        FINBOUCLE
FIN;


PROCEDURE sélection(p);
PYRAMIDE p;
DEBUT
        ENTIER ncour;
        ncour := niveau(p);
        BOUCLE TANTQUE ^(ncour=1);
            p(ncour-1) := p(ncour,0) ET p(ncour,1) ET p(ncour,2) ET p(ncour,3);
            ncour := ncour-1
        FINBOUCLE
FIN;


PROCEDURE étendre(tache, référence);
IMAGE tache, référence;
DEBUT
        IMAGE test TAILLE taille(tache);
        BOUCLE;
            test := tache;
            APPEL dilatation(tache, tache);
            tache := tache ET référence;
            test := test OUEX tache;
            SI vide(test) ALORS SORTIE FINSI
        FINBOUCLE
FIN;


PROCEDURE mesper(a, peri);
IMAGE a;
ENTIER peri;
DEBUT
        IMAGE b TAILLE taille(a);
        APPEL dilatation(a, b);
        b := b OUEX a;
        peri := compte(b)
FIN;


PROCEDURE dilatation(a,b);
IMAGE a, b;
DEBUT
        TABLEAU MASQUE m(1:8);
        ENTIER i;
        m(1) := (1,2,2,2,0,2,2,2);
        i := 1;
        BOUCLE 7 FOIS;
```

```
        m(i+1) := rot(m(i), 1);
        i := i + 1
    FINBOUCLE;
    b := a AVEC (m(1)*m(2)*m(3)*m(4)*m(5)*m(6)*m(7)*m(8),0)
FIN;
```

ANNEXE 2

Références bibliographiques.

[1] AFNOR
"Langage de programmation Algol60 normalisé"
Norme française NF-Z-65-010, Mars 1967, 74 pages.

[2] Anceau F.
"Architecture and design of Von Neumann microprocessors"
NATO-ASI on Design Methodologies for VLSI Circuits, Louvain, Belgium, July 1980.

[3] Anceau F.
"CAPRI: a design methodology and a silicon compiler for VLSI circuits specified by algorithms"
3° CALTECH Conf. on VLSI, Pasadena, USA, March 1983, pp. 15-31.

[4] Anceau F. and Reis R.
"Complex integrated circuit design strategy"
IEEE Journal of Solid State Circuits, Vol SC-17, n° 3, June 1982, pp. 459-464.

[5] **André** F., Banâtre J.P., Leroy H., Paget G., Ployette F. et Routeau J.P.
"KENSUR: An Architecture Oriented towards Programming Languages Translation"
7° Int. Symp. on Computing Architecture, La Baule, May 1980.

[6] **Andrews** H.C. and Hunt B.R.
"Digital Image Restoration"
Prentice Hall Inc.

[7] **Backus** J.
"Can programming be liberated from the Von Neumann style? A fonctionnal style and its algebra of programs"
CACM, vol 21, n° 8, 1978, pp. 613-641.

[8] **Baille** G. et Laurent J.
"Présentation d'un outil d'aide à la mise au point de circuits intégrés VLSI prototypes"
Rapport de recherche IMAG n° 298, Mars 1982, 24 pages.

[9] **Basille** J.L., Castan S. et Latil J.Y.
"Système Multiprocesseur Adapté au Traitement d'Images"
2° Congrès AFCET-IRIA Reconnaissance des Formes et Intelligence Artificielle, Toulouse, Septembre 1979, Tome 3, pp. 142-151.

[10] **Bijaoui** A.
"IMAGE ET INFORMATION. Introduction au traitement numérique des images."
Masson éditeur, 1981, 242 pages.

[11] **Bonnefoy** J.P., Jounet P., Lorette G. et Gaudaire M.
"Reconnaissance automatique en temps réel de signatures manuscrites: définition et mise en œuvre d'une méthodologie générale"
3° Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, Nancy 1981, pp. 267-275.

[12] **Bretagnolle** B-Y. et Rubat du Mérac C.
"ROMUALD: un multiprocesseur interactif pour la saisie

et le traitement d'images"
2° Congrès AFCET-IRIA Reconnaissance des Formes et Intelligence Artificielle, Toulouse, Septembre 1979, Tome 3, pp. 101-109.

[13] Bretagnolle B-Y., Rubat du Mérac C. et Seguin J.
"Architecture of a multi-microprocessor for pictures capture and processing: ROMUALD"
Micro and Mini Computer Conference, Houston, Texas, November 1979, pp. 286-291.

[14] Bretagnolle B-Y., Jutier P. and Rubat du Mérac C.
"Looking for parallelism in sequential algorithms: an assistance in hardware design"
NATO-ASI on Digital image processing and analysis, Bonas 1980, Simon J.C. and Haralick R.M. editors, REIDEL 1981, pp. 95-103.

[15] Bretagnolle B-Y.
"Conception et réalisation d'un multimicroprocesseur pour la saisie et le traitement d'images"
Mémoire d'Ingénieur CNAM, Grenoble, 29 Octobre 1981, 72 pages.

[16] Bucklen W.K.
"Monolithic bipolar circuits for video speed data conversion"
TRW-LSI publications TP2-10/79, 7 pages.

[17] Burks A.W., Goldstine H.H. and Von Neumann J.
"Preliminary discussion of the logical design of an electronic computing instrument"
Institute for Advanced Study report to the US Army Ordonance Dept. 1946. Reprinted in:
Von Neumann J.
"Collected Works"
Vol 5, Pergamon Press, 1963, pp. 34-79.

[18] Cantoni V. and Levialdi S.
"Matching the task to an image processing architecture"
6° Int. Conf. on Pattern Recognition, Munich, Oct. 1982, pp. 254-257.

[19] **Castan** S. et Massip L.
"Un système de codage adaptatif en transmission d'images par MICD"
2° Congrès AFCET-IRIA Reconnaissance des Formes et Intelligence Artificielle, Toulouse, 1979, Tome 2, pp. 22-30.

[20] **Chassery** J-M.
"Introduction à l'analyse d'images"
Cours de 3° année ENSIMAG et DEA d'Informatique, Grenoble, 1980, 73 pages.

[21] **Christopherson** W.M.
"The control of cervix cancer"
Acta Cytol. 10:6, 1966.

[22] **Chuquillanqui** S. and Perez-Ségovia T.
"PAOLA: a tool for topological optimization of large PLAs"
19° Design Automation Conf., Las Vegas, USA, June 1982, pp. 300-306.

[23] **Cunin** P-Y., Simonet M. et Voiron J.
"Méthodologie d'écriture de compilateurs. Une expérience du langage Algol68"
Thèse de Docteur-Ingénieur, Grenoble, Avril 1976.

[24] **Dalle** P., Debord P. et Castan S.
"Système d'analyse et de compréhension de scènes par ordinateur: SACSO"
3° Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, Nancy 1981, pp. 297-308.

[25] **Danielsson** P-E. and Levialdi S.
"Computer Architectures for Pictorial Information Systems"
IEEE Computer, Vol 14, n° 11, 1981, pp. 53-67.

[26] **Digabel** H.
"Manuel d'utilisation d'AT4"
Rapport interne du Centre de Morphologie Mathématique, Ecole des Mines de Paris, Fontainebleau, 1976.

[27] **Drouot** Y. et Lagrôst J-M.

"Recherche d'un moyen de saisie de morphologies humaines" Projet de 3° Année ENSIMAG, Grenoble, Juin 1982, 96 pages.

[28] Duff M.J.B.
"Review of the CLIP image processing system"
Proc. National Comp. Conf., AFIPS Conf. Proc. 47, 1978, pp. 1055-1060.

[29] Duff M.J.B.
"CLIP 4"
In Special Computer Architecture for Pattern Analysis, Fu K.J. and Ichikawa T. eds, CRC Press, 1982, pp. 65-86.

[30] Duff M.J.B.
"Cellular logic and its significance in pattern recognition"
AGARD Conf. Proc. n° 94 on Artificial Intelligence, 25.1, 1971.

[31] Dyer C.R.
"Space efficiency of region representation by quadtrees"
Technical Report, KSL 46, University of Illinois, Chicago, IL, USA, 1980.

[32] Eden M.
"On the formalization of hand-writing"
Appl. Math. Symp. Vol 12, 1961.

[33] Flynn M.J.
"Some computer organizations and their effectivness"
IEEE Transactions on Computers, C21-9, September 1972.

[34] Fouse S.D., Nudd G.R. and Nygaard P.A.
"Implementation of image preprocessing functions using CCD LSI circuits"
Proc. of the Society for Photo-optical and Instrumentation Engineering, SPIE 225, 1980, pp. 118-130.

[35] Garbay C.
"Modélisation de la couleur dans le cadre de l'analyse d'images et de son application à la cytologie automatique"

Thèse de Docteur-Ingénieur, INPG, Grenoble, Décembre 1979, 230 pages.

[36] **Golay** M.J.E.
"Hexagonal parallel pattern transformations"
IEEE Trans on Comput. C19, August 1969, pp. 733-740.

[37] **Goldberg** M.
"Digital image processing of remotly sensed imagery"
NATO-ASI on Digital Image Processing, Bonas 1980, Simon J.C. and Haralick R.M. eds, REIDEL 1981, pp. 383-437.

[38] **GOP** (anonymous)
"GOP image processor. Technical specification"
Picture Processing Lab., Linköping University, Sweden, 12 pages.

[39] **Grandlund** G.H.
"An architecture of a picture processor using a parallel general operator"
Proc. 4° IJCPR, Kyoto, Japan, 1978, pp. 1076-1081.

[40] **Grandlund** G.H., Antonsson D., Arvidsson J., Hedlund M., Henden P., Knutsson H., Lunfgren K., Nilsson B., Von Post B. and Wilson R.
"The GOP image processor"
IEEE Workshop on computer architecture for pattern analysis and image database management, Hot-Springs, 1981, pp. 195-199.

[41] **Grandlund** G.H.
"GOP, a fast and flexible processor for image analysis"
Proc. of the 5° International Conference on Pattern Recognition, Miami Beach, 1980.

[42] **Guyot** A., Jerraya A. et Raymond J.
"LUCIE, Langage Universitaire de Conception de circuits Intégrés pour l'Enseignement"
IMAG, Septembre 1980.

[43] **Hanson** A.R. and Riseman E.M.

"Preprocessing cones: a computational structure for scene analysis"
COINS Tech. Report n° 74C-7, University of Massachusetts, Amherst, MA, USA, Sept. 1974.

[44] Hanson A.R. and Riseman E.M.
"Computer Vision Systems"
Academic Press, New York, 1978.

[45] Hanson A.R. and Riseman E.M.
"Processing cones: a computational structure for image analysis"
In Structured Computer Vision, Tanimoto S.L. and Klinger A. eds, Academic Press, 1980, pp. 101-132.

[46] Henderson T.C. and Davis L.
"Compilateurs de grammaires de formes"
3° Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, Nancy, 1981, pp. 415-423.

[47] Jones L. and Iyengar S.S.
"Representation of a region as a forest of quadtrees"
Proc. IEEE Conf. on Pattern Recognition and Image Analysis, 1981, pp. 57-59.

[48] Jutier P., Rubat du Mérac C. and Bretagnolle B-Y.
"Parallel image processing: uni or bidimentional? Example: ROMUALD"
3° International Meeting on Image processing, Salva di Fasano (Brindisi), Italy, December 1982.

[49] Kartashev S.I. and Kartashev S.P.
"Problems of designing supersystems with dynamic architecture"
IEEE Trans. on Computers, Vol C29, n° 12, december 1980, pp. 1114-1132.

[50] Kazmierczak H.
"Processing of image data for remote sensing application"
NATO-ASI on Digital Image Processing and Analysis, Bonas 1978, Simon J.C. and Rosenfeld A. eds, Nordhoff 1979.

pp. 317-327.

[51] **Klein** J.C.
"Conception et réalisation d'une unité logique pour l'analyse quantitative d'images"
Thèse, Nancy, 1976.

[52] **Kruse** B.
"Design and implementation of a picture processor"
Linköping Studies in Science and Technology Dissertations, n° 13, Linköping University, Sweden, April 1977, 122 pages.

[53] **Kung** H.T.
"Special-purpose Devices for Signal and Image Processing: An opportunity in VLSI"
Proc. of the SPIE, Vol 241, July 1980, pp. 76-84.

[54] **Kunt** M.
"Traitement numérique des signaux"
Traité d'Electricité, Vol. 20, Georgi ed. Lausanne, 1980.

[55] **Lantuejoul** C.
"An image analyser"
In Languages and Architecture for Image Processing, Duff M.J.B. and Levialdi S. eds, Academic Press, London, 1981, pp. 163-177.

[56] **Latombe** J.C.
"Survey of advanced general-purpose software for robots manipulators"
Rapport de recherche IMAG n° 330, Grenoble, Novembre 1982, 49 pages.

[57] **Lea** R.M.
"I2L micro-associative-processors"
ESSCIRC 1979, pp. 104-106.

[58] **Lea** R.M.
"A VLSI array processor for image processing"
Proc. of Workshop on Algorithmically-specialized computer organisations, Purdue University, 1982.

[59] Lea R.M.
"SCAPE: a Single Chip Array Processing Element for image analysis"
VLSI'83, Proc. of the IFIP TC10/WG10.5 Int. Conf. on VLSI, Trondheim, Norway, August 1983, Anceau F. and Aas E.J. eds., North-Holland Pub., pp. 285-294.

[60] Lester J.M., Brenner J.P. and Selles W.D.
"Local transforms for biomedical image analysis"
Computer Graphics and Image Processing, 13, 1980, pp. 17-30.

[61] Levialdi S.
"COPLAN: a closedness pattern analyser"
Proc. Inst. Elec. Eng. 115n, 1968, pp.879-880.

[62] Levialdi S., Duff M.J.B., Norgren P., Preston K. and Toriwaki J.I.
"Theoretical and practical considerations in the application of neighborhood logic to image processing"
Proc. of the 4° IJCPR, Kyoto, Japan 1978, pp. 139-145.

[63] Levialdi S.
"Finding the edge"
NATO-ASI on Digital Image Processing, Bonas 1980, Simon J.C. and Haralick R.M. eds, REIDEL 1981, pp.105-148.

[64] Li M., Grosky W.I. and Jain R.
"Normalized quadtrees with respect to translations"
Proc. IEEE Conf. on Pattern Recognition and Image Analysis, 1981, pp. 60-62.

[65] Manara R. and Stringa L.
"The EMMA system: an Industrial Experience on a multi-processor"
in Languages and Architectures for Image Processing, Duff M.J.B. and Levialdi S. eds., Academic Press, London, 1981, pp. 215-227.

[66] Matheron G.
"Random sets and integral geometry"

Wiley, 1975.

[67] **Mazaré** G.
"Structures multi-processeurs. Problèmes de parallélisme. Définition et évaluation d'un système particulier"
Thèse d'Etat, Grenoble, 1978, 327 pages.

[68] **Mead** C. and Conway L.
"Introduction to VLSI systems"
Addison-Wesley, 1980, 396 pages.

[69] **Monnerot** T.
"Picture input with a TV-camera"
Proc. of the 5° DECUS European Seminar, Stockholm, Sweden, 1969.

[70] **NASA** (Anonymous)
"Landsat Data Users Handbook"
Document n° 765D4258, Goddart Space Flight Center, Greenbelt, MD, USA, 1976.

[71] **Naur** P. and al.
"Revised Report on the Algorithmic Language Algol 60"
CACM, 6, 1963, pp. 1-17.

[72] **Nudd** G.R.
"Image understanding architectures"
AFIPS, Vol. 49, National Comp. Conf. 1980, pp. 377-390.

[73] **Oka** R.
"Handwritten chinese-japanese characters recognition by using cellular features"
6° Int. Conf. on Pattern Recognition, Munich, Oct. 1982, pp. 783-785.

[74] **Pavel** M.
"Fondements mathématiques de la reconnaissance des structures"
Hermann ed., 1969.

[75] **Pavel** M.

"Des formalismes en classification et reconnaissance des formes"
3° Congrès AFCET Reconnaissance des Formes et Intelligence Artificielle, Nancy 1981, pp. 311-322.

[76] Pavel M.
"Aspects topologiques et catégoriques de la reconnaissance des formes"
Cours de troisième cycle, fascicule 3, Lab. de Probabilités, Université de Paris VI.

[77] Porracchia A. et Tapponnier J.J.
"Reconnaissance automatique multipolices de caractères imprimés"
Projet de 3° année ENSIMAG, Grenoble, Juin 1983, 71 pages.

[78] Preston K., Duff M.J.B., Leviaidi S., Norgren P.E. and Toriwaki J.I.
"Basics of cellular logic with some applications in medical image processing"
Proc. IEEE, Vol 67, n° 5, May 1979, pp. 826-856.

[79] Pun T.
"Entropic thresholding, a new approach"
Computer Graphics and Image Processing, 16, 1981, pp.210-239.

[80] Pun T.
"Simplification automatique de scènes par traitement d'images en vue d'une restitution tactile pour handicapés de la vue"
Thèse n° 425, Ecole Polytechnique Fédérale de Lausanne, 1982, 157 pages.

[81] Ranade S., Rosenfeld A. and Samet H.
"Shape approximation using quadtrees"
Pattern Recognition, Vol 15, n° 1, 1982, pp. 31-40.

[82] Reddy D.R. and Hon R.W.
"Computer architectures for vision"

Computer Vision and Sensor-based Robots, Warren, MI, USA, Sept. 1978, pp. 169-186.

[83] **Rosenfeld** A.
"Cellular architectures: from automata to hardware"
University of Maryland, Computer Vision Lab., Tech. Rep. n° 1048, May 1981, 14 pages.

[84] **Rosenfeld** A.
"Quadtrees and pyramids: hierarchical representation of images"
NATO-ASI on Pictorial Data Analysis, Bonas 1982.

[85] **Schürmann** J.
"Reading machines"
6° Int. Conf. on Pattern Recognition, Munich, Oct. 1982, pp. 1031-1044.

[86] **Serra** J.
"Image analysis by mathematical morphology"
Academic Press, London, 1982.

[87] **Sheldrake** R.
"A new science of life"
Tarcher ed., Los Angeles.

[88] **Simon** J.C., Backer E. and Sallantin J.
"A structural approach of pattern recognition"
Signal Processing 2, 1980, pp. 5-22.

[89] **Stringa** L.
"EMMA: an unbounded modular multi-mini processor structure"
ACM Comp. Sci. Conf. Detroit, 1978.

[90] **Tanimoto** S.L. and Klinger A.
"STRUCTURED COMPUTER VISION. Machine perception trough hierarchical computation structures"
Academic Press, New-York, 1980, 234 pages.

[91] **Tanimoto** S.L.

"Towards hierachical cellular logic: design considerations for pyramid machines"
Dept. of Computer Science, University of Washington, Seattle. Technical Report 81-02-01, Feb. 1981, 20 pages.

[92] Tanimoto S.L.
"Programming techniques for hierarchical parallel image processors"
In Multicomputers and Image Processing, Algorithms and Programs, Academic Press, 1982, pp. 421-429.

[93] Thomé S.
"Reconnaissance du chiffre manuscrit"
Thèse de Docteur-ingénieur, Ecole Nationale Supérieure des Télécommunications, E-79005, Octobre 1979, 106 pages.

[94] Thoraval Y.
"CCD: des registres à transfert de charges"
Micro-systèmes Sept-Oct 1981, pp. 189-198.

[95] Todd-Prokropek A.
"Medical image analysis"
NATO-ASI on Pictorial Data Analysis, Bonas 1982.

[96] Turing A.M.
"On computable numbers, with an application to the Entscheidungsproblem"
Proceedings of the London Mathematical Society, Series 2, Vol 42, 1936-1937, pp. 230-265.

[97] Uhr L.
"Layered recognition cones networks that preprocess, classify and describe"
IEEE Trans. on Computers, Vol 21, 1972, pp.758-768.

[98] Uhr L.
"Psychological motivation and underlying concepts"
In Structured Computer Vision Tanimoto S.L. and Klinger A. eds, Academic Press, New York, 1980, pp. 1-30.

[99] Uhr L.

"Converging pyramids of arrays"
6° Int. Conf. on Pattern Recognition, Munich, Oct. 1982, pp. 31-34.

[100] Umeda M.
"Recognition of multi-font printed chinese characters"
6° Int. Conf. on Pattern Recognition, Munich, Oct. 1982, pp. 793-796.

[101] Unger S.H.
"Pattern detection and recognition"
Proc. IRE, Vol 47, 1959, pp. 1737-1752.

[102] Unger S.H.
"A computer oriented toward spatial problems"
Proc. IRE, October 1958, pp. 1744-1750.

[103] VOIR Sarl
"Mesures de volumes de pains par traitement d'images à l'aide d'un micro-ordinateur"
Document interne, Voir Sarl, 45, rue du Vercors, 38000 Grenoble.

[104] Wood A.
"The interaction between hardware, software and algorithms" in Languages and Architectures for image processing, Duff M.J.B. and Levialdi S. eds, Academic Press, 1981, pp. 1-11.

[105] Young T.Y. and Liu P.S.
"Impact of VLSI on Pattern Recognition and Image Processing"
VLSI Electronics: Microstructure Science, Vol 4, Chap. 10, 1982, Academic Press, pp. 319-360.

[106] Zahniser D.J.
"Automation of pap smear analysis: a review and status report"
NATO-ASI on Pictorial Data Analysis, Bonas 1982.

[107] Zucker S.W.
"Survey: region growing, childhood and adolescence"

Computer Graphics and Image Processing, 5, 1976, pp. 382-399.

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,
VU le rapport de présentation de Messieurs F. ANCEAU
J.P. HATON
S. LEVIALDI

Monsieur B.Y. BRETAGNOLLE

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité Informatique.

Fait à Grenoble, le 12 janvier 1984

Le Président de l'I.N.P.G.

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,

Thèse de Docteur-Ingénieur

Nom de l'auteur: Bernard-Yves **Bretagnolle**

Etablissement: Institut National Polytechnique de Grenoble

Résumé:

Cette thèse étudie le traitement d'images dans ses rapports avec l'architecture des ordinateurs. L'étude des principaux domaines d'application et des techniques essentielles du traitement d'images permet de dégager des interrogations directes pour le concepteur d'architectures d'ordinateurs. Quelques unes des voies possibles pour leurs solutions matérielles sont ensuite présentées dans leurs principes et à l'aide d'exemples.

Une deuxième partie présente deux études: la machine ROMUALD, multi-microprocesseur pour la saisie et le traitement d'images et le système KIDS, d'architecture plus ambitieuse, qui allie les aspects logiciels et matériels et comprend un langage de programmation et un ensemble matériel constitué d'une cascade de modules fonctionnellement spécialisés débouchant sur un processeur matriciel construit à l'aide d'un circuit VLSI spécialement défini.

Mots-clés:

Traitement d'images - Architecture d'ordinateurs
Parallélisme - Multiprocesseurs - Processeurs matriciels
Langage de programmation - VLSI