PREMIER COLLOQUE IMAGE
Traitement, Synthèse, Technologie et Applications
BIARRITZ — Mai 1984 —



# ARCHITECTURES DE MACHINES D'IMAGES : UNE APPROCHE METHODIQUE - QUELQUES EXEMPLES

# ARCHITECTURES OF IMAGE COMPUTERS : A METHODIC APPROACH - EXAMPLES

Michel MERIAUX

LA 369 CNRS - Université de Lille I - 59655 Villeneuve d'Ascq Cedex

## RESUME

La synthèse d'images par ordinateur fait appel à trois types d'informations : les objets à visualiser ou *scène*, les processus mis en jeu ou *actions*, les *pixels* formant l'image synthétisée. Suivant le centre d'intérêt que l'on a, scène, action, ou pixels, l'architecture adaptée aura des particularités que nous décrivons, et des possibilités particulières en ce qui concerne, par exemple, le parallélisme, l'interactivité, le temps réel, les possibilités d'intégration... Trois classes d'architectures sont donc définies, chacune étant explicitée sur un exemple.

Cette tentative de classification et de formalisation doit permettre d'une part d'orienter le choix d'une architecture en fonction de divers critères (temps réel, modularité, ...), d'autre part d'éviter le piège des architectures hétérogènes, voire hétéroclites qu'offrent de nombreuses réalisations commerciales. Elle met également en lumière, et nous le faisons sur quelques exemples, la nécessité d'isoler les algorithmes fondamentaux mis en jeu et de les étudier en fonction de l'architecture choisie (parallèle, pipeline, SIMD, MIMD etc ...).

## SUMMARY

Computer generation of images involves three types of informations : the objects to be displayed or *scene*, the processes or *actions*, the *pixels* building the resulting image. Every point of view has its corresponding peculiar architecture that we describe, especially in terms of parallelism, interaction, real-time, very large scale integration ...

Three classes are defined and explained with an example. These examples help us to give some rules to make a choice between architectures and to get a good implementation of this choice. They show too, if it were necessary, that all the fundamental "well-known" algorithms of image synthesis and processing must be re-studied to match such architectures

Michel MERIAUX

Architectures de machines d'images : une approche méthodique - quelques exemples

Architectures of image computers : a methodic approach - examples

## Introduction

Une machine d'images est un système informatique limité d'un côté aux moyens d'actions (entrées) que l'utilisateur a à disposition, de l'autre à l'écran (visuel) que celui-ci peut regarder. Cette définition très large inclut toutes les machines produisant, traitant ou manipulant des images ou dessins. Nous réduisons ce cadre en précisant que ce que l'on peut voir en sortie est une image bidimensionnelle, c'est-à-dire une surface assimilable à un tableau T(i, j) de valeurs visuelles (*pixels*).

Nous appelerons "scène", composée d'*objets* une représentation logique de l'image, bien souvent nécessaire pour une application donnée. Cette application met en jeu des processus, appelés *actions*.

## 1 - Principe

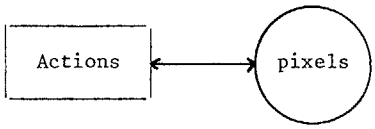
Ayant ainsi défini trois types d'informations, pixels actions, et objets, il en résulte de manière immédiate trois classes d'architectures suivant la préférence ou le centre d'intérêt que l'on a : une machine très proche des objets sera très différente d'une architecture implémentant des actions, etc ...

Une machine-objet est une machine organisée autour des objets de la scène, les actions à réaliser et les pixels de l'image finale ne devant pas cacher ces objets, mais contribuer à les faire apparaître ; une machine action est une architecture cherchant à implémenter au mieux tous les algorithmes mis en jeu, par exemple en pipe-line ou sous forme d'un multiprocesseur MIMD. Une machine pixel, quant à elle, est entièrement orientée image finale ; l'exemple type est le tableau de processeurs, chaque processeur étant affecté à un ensemble de pixels.

## 2 - Domaines d'application

### 2.1 - *Manipulation d'images*

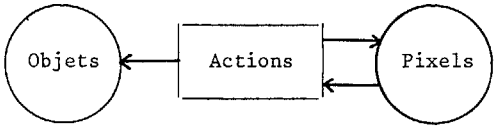
Il s'agit là seulement de réorganiser une image, voire de réaliser des traitements simples entre images (inclusion, incrustation). Il n'y a donc pas d'objets à proprement parler et le schéma se réduit à :




Comme on réalise généralement des actions simples et séquentielles, seule l'approche machine-pixel sera retenue. Nous trouverons deux limites : - d'une part une machine où l'on associe une logique simple à chaque pixel, d'autre part une machine où cette logique est associée à toute l'image.

### 2.2 - *Traitement / Analyse d'images*

L'utilisateur souhaite réaliser sur des images un ensemble d'actions, pour obtenir soit une nouvelle image, soit un ensemble d'informations plus condensées, analogues à des objets. D'où le schéma suivant :




On voit clairement que deux des classes de machines sont possibles, d'une part les machines-pixels où les actions seront réparties sur chaque processeur-pixel, d'autre part les machines-actions, où chaque processeur fonctionnel effectuera l'ensemble du calcul associé sur tous les pixels.

### 2.3 - *Synthèse d'images*

Elle met en jeu objets, actions, pixels, suivant le schéma suivant :


On trouvera donc ici les 3 classes de machines définies plus haut.

## 3 - Parallélisme

Chacune des classes définies permet un certain degré de parallélisme, que nous allons préciser.

### 3.1 - *Machine-objet*

Une machine objet associe un processeur par objet de la scène, sur lequel il effectue tous les traitements. Si les objets sont indépendants, i.e. s'il l'on ne tient pas compte de relations entre objets, il est clair que le parallélisme potentiel est égal au nombre d'objets.

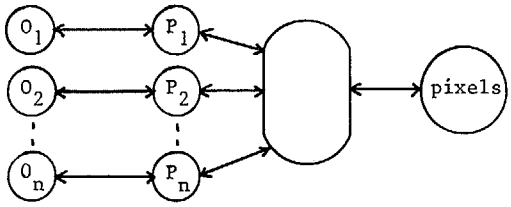



*Figure 1 : machine-objet*

Néanmoins, il reste à régler le problème d'accès à l'image (aux pixels). La machine proposée en exemple montre une approche possible de traitement de ces conflits, à l'aide d'un bus intelligent.

Si les $P_i$ réalisent les mêmes fonctions, on obtient donc une machine SIMD.

### 3.2 - *Machine-Action*

On associe un processeur à chaque action à réaliser, les objets étant traités séquentiellement. Le parallélisme est donc égal au nombre d'actions, si l'on est capable d'effectuer toutes les actions simultanément. Cela n'est généralement pas le cas, car l'ordre des actions n'est pas indifférent.

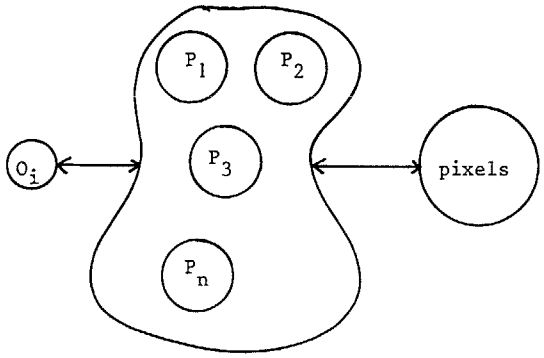
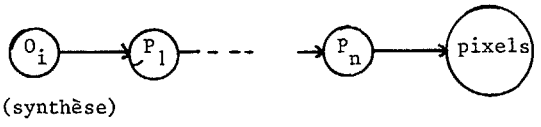


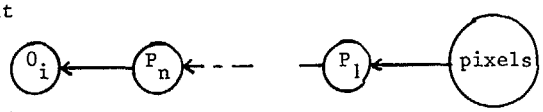

*Figure 2 : machine-action*

Un cas particulier très important est celui des machines pipe-line, soit




(synthèse)

soit




(traitement et analyse)

### 3.3 - *Machine-Pixels*

On associe un processeur à une zone géométrique de l'image responsable du traitement de toutes les actions pour tous les objets dans cette zone.

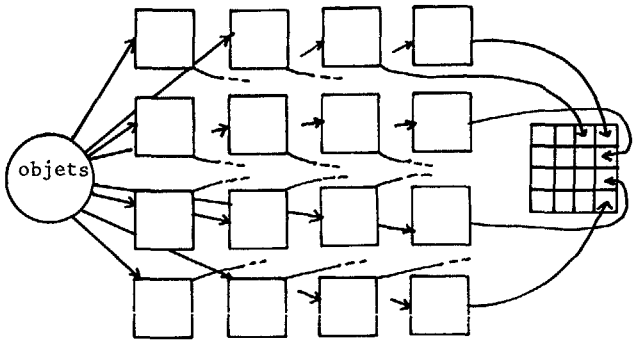



*Figure 3 : Machine-pixel*

Un cas particulier très important est celui des machines où l'on associe un processeur par pixel ; un exemple en sera donné dans la suite.

Le parallélisme est ici géométrique et est égal au nombre de processeurs (de zones) mis en jeu. En réalité les processeurs ne sont pas indépendants et certains ne sont pas concernés par certains objets.

### 3.4 - *Remarques*

De l'étude ci-dessus résulte le fait que le parallélisme est fonction de l'éventuelle dépendance entre éléments (processus). Nous supposons dans la suite qu'il existe *un* critère permettant d'affirmer si deux éléments sont dépendants ou non, i.e. si deux processus sont réalisables simultanément (ou en pipeline). Ainsi :

. Dans un algorithme de lignes cachées, les objets peuvent généralement être partitionnés en sous-groupe, avec comme critère : deux objets de deux sous-groupes distincts peuvent être traités indépendamment. Nous pouvons donc créer autant de processus identiques que de sous-groupes.

. Dans une machine cellulaire, deux pavés disjoints peuvent effectuer des processus différents.

. Sur un même objet, certaines actions peuvent être parallèles, d'autres doivent être séquentielles (ou pipe-line).

D'où le théorème suivant :

"Le parallélisme est maximum si l'on sait partitionner l'ensemble des processus (ou éléments) en un minimum de sous-groupes formés de processus (ou éléments) indépendants deux à deux".

La théorie des graphes fournit alors des méthodes permettant de trouver des solutions à cette question.

## 4 - Exemples

Nous donnons ici les grandes lignes de 3 architectures, chacune correspondant à une classe de machines.

### 4.1 - *Machine-Action*

C'est l'approche la plus classique : on cherche à décrire et à implémenter la suite des traitements à effectuer. Citons comme exemple le "Geometry Engine" de Clark ([CLA 82]), qui permet de réaliser les transformations géométriques 3D, le découpage, la mise en perspective, etc ...

Plutôt que de "câbler" un algorithme complet, nous proposons de chercher dans l'ensemble des algorithmes des noyaux communs, simples, mais fréquemment utilisés. La description suivante est un exemple du résultat de cette recherche.

#### 4.1.1 - *Le noyau ([MER 84])*

Nous nous sommes intéressés à plusieurs algorithmes, relevant de divers domaines, mais faisant tous intervenir la description d'images à l'aide d'arbres quaternaires. Il en est ainsi par exemple de :

- l'algorithme de Warnock
- une méthode de remplissage
- une méthode de reconnaissance de formes
- une méthode de compression d'images

Le noyau de cette description par arbres est un découpeur bidimensionnel. Notre étude s'est limitée à un découpeur de polygones par 4 fenêtres carrées (fig 4).

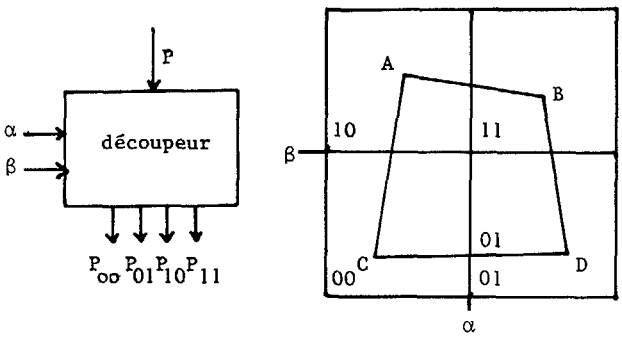



*Figure 4 : principe du découpeur*

Un tel noyau peut être utilisé récursivement ou exister en plusieurs exemplaires pour un problème donné.

#### 4.1.2 - *Implémentation*

Indiquons simplement ici quelques résultats de l'étude du noyau ci-dessus et de son utilisation, généralisable à tout autre algorithme :

1) Une étude *très* précise est indispensable pour produire une implémentation matérielle efficace en temps et en espace. Il est nécessaire de comparer diverses réalisations pour estimer quelle est la meilleure (ex : calcul d'intersection) ; de même, il convient d'étudier des solutions fortement parallèles.

2) Cette étude étant menée, le résultat en est bien souvent un circuit de faible complexité, aisément intégrable ; un composant VLSI pourrait en contenir en grand nombre. L'organisation de ces circuits élémentaires et de ces composants doit également être étudiée précisément.

3) Les performances obtenues sont très intéressantes par rapport à une réalisation logicielle usuelle.

### 4.2 - Machine-objet

L'exemple le plus connu d'une telle architecture est la machine proposée par Weinberg [WEI 81]. Cette machine comporte un pipe-line de n processeurs-objets, le pipe-line assurant le calcul de l'information finale à afficher.

#### 4.2.1 - Machine proposée

Nous avons étudié ([DUR 82] et [DUR 83]) une solution parallèle à ce problème, où le travail du pipe-line est effectué par un décideur unique suivant le schéma:

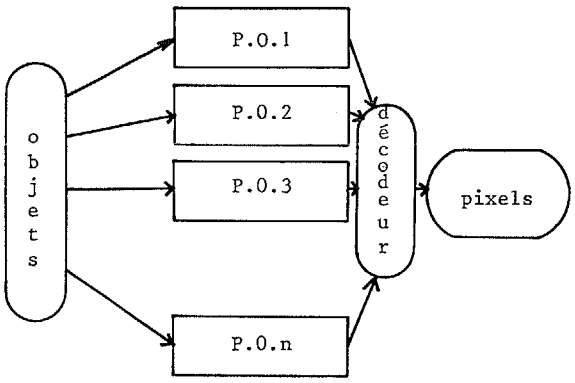



*Figure 5 : machine à découpage par objet*

Les avantages de cette approche sont les suivants :

. fiabilité plus grande
. identification des objets immédiate
. modification d'un objet prise en compte immédiatement.

Cette architecture nécessite l'étude de processeurs-objets capables d'effectuer beaucoup de traitements sur un objet donné.
Nous avons choisi les options suivantes :

. un P.O. est associé à un *type* d'objet (polygone, cercle, sphère, ...)
. il assure la conversion de ce type et l'évaluation d'attributs visuels (couleur, texture, transparence) en 3D
. il n'assure *pas* les transformations géométriques.

Le processeur décideur est en quelque sorte un "z-buffer", qui détermine le(les) objet(s) le(s) plus proche(s) de l'observateur en un (x, y) donné.

#### 4.2.2 - Résultats de l'étude

L'étude a montré que

1) Le décideur pouvait être très simple et extensible (bus)
2) On aboutit à quelques types de P.O. construits autour de briques élémentaires (interpolateurs pour l'essentiel), pouvant chacun faire l'objet d'un composant VLSI
3) Il est possible d'obtenir le temps réel, avec une technologie d'intégration courante.
4) La machine est modulaire et peut se configurer en fonction de la complexité d'une application (i.e. le nombre d'objets)

### 4.3 - Machine-pixel

Un exemple bien connu est la machine de traitement d'images MPP, qui associe un µ-processeur à une zone de l'image à traiter ; elle comporte 128×128 processeurs.

#### 4.3.1 - Machine cellulaire ([MER 83])

Nous proposons de pousser le schéma à l'extrême et d'associer une cellule de calcul à chaque pixel de l'image.

L'architecture est donc très simple et répétitive ; en réalité les problèmes sont de deux ordres :

1) *Tous* les algorithmes séquentiels usuels ne sont plus valides : ils doivent être réécrits pour être compatibles avec l'architecture et fonctionner en parallèle ou en pipe-line (i.e. propagation dans le réseau)
2) Les communications dans le réseau sont essentielles; l'étude a porté sur un réseau carré à 4 voisins et montre les limitations de celui-ci.

#### 4.3.2 - Implémentation

La cellule obtenue s'avère très simple ; elle satisfait aussi bien aux algorithmes de traitement qu'à ceux de synthèse. Sa complexité montre que l'on saurait intégrer un sous-réseau (4×4 ou 8×8) dans un seul composant, et donc réaliser par exemple une machine 512×512 à l'aide de 4096 boîtiers *tous identiques*.

L'évaluation des performances montre de plus qu'il est possible d'atteindre un très fort taux de parallélisme ($n^2$ au mieux) et donc des performances très bonnes.

## 5 - Résultats généraux

### 5.1 - Classification

L'étude des 3 exemples nous a montré que chacune des classes était plus ou moins adaptée à tel ou tel objectif initial, ce que résume le tableau suivant :

| objectif | action | objet | pixel |
|---|---|---|---|
| synthèse et traitement | 0 | 0 | 1 |
| traitement temps réel | 1 | 0 | 1 |
| synthèse temps réel | ? | 1 | ? |
| interaction | ? | 1 | ? |
| modularité taille image | 0 | 0 | 1 |
| modularité fonctionnelle | 1 | 0 | 0 |
| modularité complexité image | 0 | 1 | 0 |
| régularité (généricité des composants) | 0 | 1/2 | 1 |

### 5.2 - Observations

Chaque approche nécessite sa propre méthode d'investigation et ne peut se satisfaire de la description abstraite et théorique d'algorithmes souvent séquentiels. En outre, les architectures déduites ne sont implémentables qu'à l'aide de composants VLSI ; elles satisfont d'ailleurs en général aux règles de simplicité, régularité, modularité nécessaires à une intégration éventuelle.

## Conclusion

Si une telle classification peut paraître arbitraire, elle permet néanmoins de fixer un peu les idées dans la jungle des machines diverses et variées que l'on trouve en synthèse ou en traitement d'images. Elle nous a également servi de support pour l'étude de trois architectures complétement différentes les unes des autres, qui pourtant ont de nombreux points communs sur le fond. L'introduction de composants spécialisés, qui s'est déjà faite de manière un peu anarchique, doit passer par une définition précise des objectifs et de l'architecture concernée. C'est au prix de cette "formalisation" que l'on obtiendra des machines d'images performantes, évolutives et peut-être compatibles entre elles.

Michel MERIAUX

Architectures de machines d'images : une approche méthodique - quelques exemples.

Architectures of image computers : a methodic approach - examples.

Bibliographie

[CLA 82] : CLARK J.H., "The Geometry Engine : a VLSI geometry system for graphics"
Computer Graphics Vol 16 n° 3 July 1982.

[DUR 82] : DURIF P. & MERIAUX M., "Une architecture pour la synthèse d'images sans mémoire de trame"
Actes du Congrès AFCET, 1982.

[DUR 83] : DURIF P., "Etude d'une machine parallèle de synthèse d'images à découpage par objets"
Thèse de 3e cycle - Décembre 1983 - Lille.

[MER 83] : MERIAUX M., "A Cellular Architecture for image synthesis"
Microprocessing & Microprogramming - Euromicro Journal (à paraître).

[MER 84] : MERIAUX M., "A Two-dimensional clipping divider"
Publication interne - LA 369 - Université de Lille I.

[WEI 81] : WEINBERG R., "Parallel processing image synthesis and anti-aliasing"
Computer Graphics Vol 15 n° 3 August 1981.