MISE EN OEUVRE D'ALGORITHMES DE TRANSFORMATIONS GEOMETRIQUES SUR UNE ARCHITECTURE PARALLELE DE TYPE SIMD/SPMD

P.DUCLOS(1) F.BOERI(1) M.AUGUIN(1) G GIRAUDON(2)

(1)Laboratoire de Signaux et Systèmes, UA 814 du CNRS.
Université de Nice, 41 Bd Napoléon III 06041 Nice CEDEX
(2)Laboratoire PASTIS, INRIA centre de Sophia-Antipolis

OPSILA est une architecture originale permettant d'exploiter deux formes de parallélisme, le SIMD (Single Instruction Multiple Data) qui est une forme de parallélisme synchrone, et le SPMD (Single Progam Multiple Data) qui est une forme de parallélisme asynchrone.

Les algorithmes de transformations géométriques sont des traitements coûteux en temps de calcul et difficiles à mettre en oeuvre efficacement sur une architecture parallèle par la nature des accés aux données qu'ils supposent.

Nous proposons une mise en oeuvre efficace de l'ensemble des algorithmes de transformations géométriques, tant linéaires que non-linéaires, illustrée dans le cas de la rotation d'une image 256x256 avec interpolation bilinéaire.

Opsila is a new parallel architecture using both synchronous form of parallelism, the SIMD (Single Instruction Multiple Data), and asynchronous form, the SPMD (Single Program Multiple Data) .

Geometrical transformations are computationnaly costly and are quite difficult to implement on parallel architectures because of the required memory access. An efficient parallel approach for performing both linear and non-linear two-dimensionnal geometric transforms is proposed. The model problem is the rotation of a 256x256 image using bilinear interpolation technique.

I INTRODUCTION

Les architectures de super_ordinateurs réalisées à ce jour permettent de diminuer de façon significative les temps d'éxécution par rapport aux machines séquentielles classiques. Il est cependant nécessaire d'exploiter au mieux **le parallélisme potentiel des applications.**

Les techniques parallèles les plus exploitées dans les machines commercialisées sont :

- Le pipeline qui utilise le parallélisme du calcul itératif des expressions et des opérations (Cray 1, Cyber 205, VP 200 par exemple).
- Les systèmes multiprocesseurs qui permettent d'éxécuter de façon concurrente plusieurs tâches participant à la réalisation d'une application.

Les machines les plus récentes exploitent ces deux formes de parallélisme (Cray XMP, CEDAR par exemple).

Notre objectif principal est de montrer qu'une machine multi-processeurs appelée **OPSILA**, reconfigurable dynamiquement suivant les deux modes de fonctionnement **SIMD** (Single Instruction Multiple Data) et **SPMD** (Single Program Multiple Data), permet de traiter efficacement une grande variété d'applications et particulièrement les algorithmes du domaine du traitement numérique des images par utilisation judicieuse des formes de parallélisme qui leur sont associés.

Le démonstrateur OPSILA a été construit contractuellement avec la CIMSA/SINTRA et avec le support de la DRET.

II L'ARCHITECTURE OPSILA [Auguin 85]

II-1 Synoptique de OPSILA

Le synoptique général de OPSILA est illustré sur la figure 1. La machine possède une unité scalaire et de contrôle, et une unité vectorielle. La complexité du séquencement des opérations dans un processeur vectoriel conduit à considérer une unité de contrôle multi-processeurs. Dans OPSILA elle est composée de deux processeurs , le processeur scalaire (PS) qui gère l'éxécution de l'application et le processeur d'instructions (PI) qui gère l'éxécution des seules instructions vectorielles.

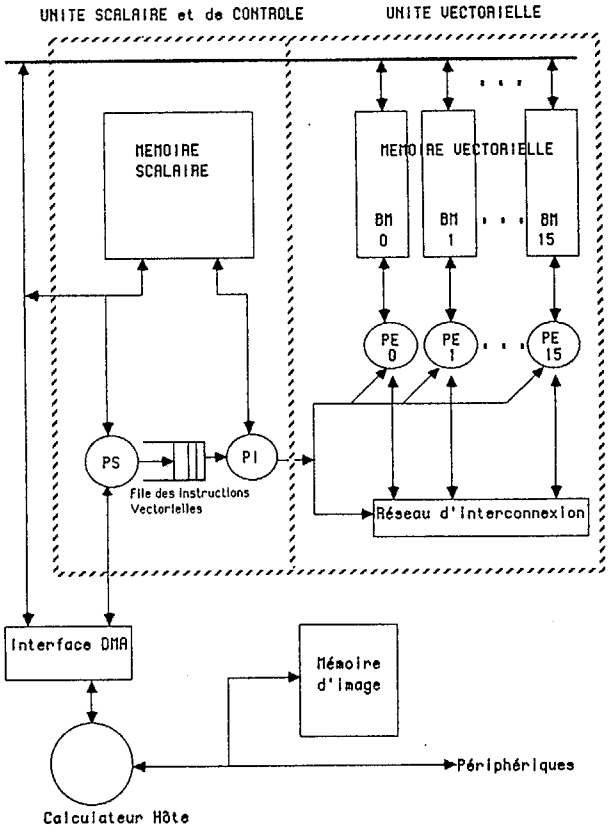


FIGURE 1 : Synoptique de l'architecture OPSILA

* **Le processeur scalaire** (PS) :
- extrait les instructions de la mémoire scalaire (MS).
- éxécute les parties de code scalaire.
- envoie à la partie vectorielle la description des instructions vectorielles et de leurs opérandes. Pour ce faire, une file d'attente (file des instructions vectorielles) permet un recouvrement des éxécutions des parties scalaires et vectorielles.

* **Le processeur d'instructions** (PI) :
- décompose les instructions vectorielles en séquences d'opérations éxécutables par la partie vectorielle. La partie vectorielle comportant $p=2^n=16$ processeurs, les vecteurs de taille quelconque sont automatiquement décomposés en vecteurs de taille p au plus. Un procédé de masquage des PE permet de traiter les tranches incomplètes.
- gère les conflits d'accés aux bancs de la mémoire vectorielle (cf II-2).
- initialise, sur commande du PS, le mode SPMD et gère les mécanismes de synchronisation.

* **La partie vectorielle** est composée de $p=2^n=16$ processeurs élémentaires (PE) et de p bancs de mémoire (BM) contenant les données vectorielles. La structure de la mémoire vectorielle est telle que deux adresses consécutives désignent deux bancs adjacents. Les données sont échangées de BM à BM ou de BM à PE par l'intermédiaire d'un réseau d'interconnexion de type Oméga/Bénes.

II-2 Les modes opératoires de OPSILA.

OPSILA peut fonctionner suivant deux modes : le SIMD et le SPMD.

- **Le mode SIMD** : parallélisme synchrone.
Il est adapté aux traitements vectoriels. Dans ce mode, OPSILA permet de réaliser une vectorisation analogue , avec les mêmes contraintes , à celles des machines pipeline du type Cray 1S par exemple. Ce mode est étroitement lié à l'utilisation de la mémoire vectorielle et du réseau d'interconnexion.
Un vecteur au sens de OPSILA (ou vecteur linéaire) est un ensemble ordonné de composantes dont les adresses sont en progression arithmétique dans la mémoire vectorielle. Il est défini dans l'unité scalaire et de contrôle par un couple (E,R) ou E est l'adresse de la première composante et R la raison d'accès aux éléments suivants.
Les sections de code SIMD sont rangées en mémoire scalaire. Lorsque le PS rencontre une instruction vectorielle, il la place ainsi que les caractéristiques (E,R) des vecteurs opérandes dans la file des instructions vectorielles (cf figure 1).
Le PI gère l'éxécution par la partie vectorielle de cette instruction :
- Il décompose les vecteurs opérandes de taille quelconque en tranches de 16 composantes. Chaque tranche transite par le réseau d'interconnexion qui réordonne les composantes accédées avant de les ranger dans les registres des processeurs élémentaires.
- Il gère automatiquement les conflits d'accés aux bancs de mémoire en réalisant le nombre d'accés successifs nécessaires pour reconstituer la tranche de 16 composantes. En effet, lorsque la raison d'accés R est impaire, l'ensemble des p composantes d'une même tranche est distribué dans les p bancs de mémoire. Il n'y a pas de conflit, les p composantes sont lues en un cycle. Si la raison R est paire, les p composantes sont situées dans un sous-ensemble de bancs. Plusieurs accés successifs à la mémoire vectorielle sont nécessaires il s'agit d'un accés conflictuel.

L'intérêt du PI, nous venons de le voir est de soulager l'utilisateur de la gestion des accés à la mémoire vectorielle. L'efficacité de la partie vectorielle est optimale lorsque l'on traite des vecteurs linéaires, et plus particulièrement ceux dont les composantes sont accessibles sans conflit. Toutes les applications ne peuvent s'exprimer dans de tels termes. Pour étendre le domaine d'utilisation du mode SIMD, il faut disposer d'accés vectoriels plus élaborés que les accés à des vecteurs linéaires. Deux primitives d'accés vectoriels indirects appelées GATHER et SCATTER ont été définies [Schubert 86].
Soit I et V' deux vecteurs de k composantes et V un vecteur de m composantes.
L'opération GATHER est définie par :
$V'(i) := V(I(i)), \quad i \in [1,k]$
L'opération SCATTER est définie par :
$V(I(i)) := V'(i) , \quad i \in [1,k]$

Ces accés sont moins performants que les accés à des vecteurs linéaires car leur gestion est plus complexe, tant au niveau du calcul des adresses des composantes (R est indéfini) qu'à celui des conflits d'accés aux bancs de mémoire. Les conflits dépendent du vecteur d'indices I et ne peuvent être détectés à priori par le PI comme dans le cas d'accés linéaires (cf Figure 4).

- **Le mode SPMD** : parallélisme asynchrone
L'éxécution simultanée et indépendante de plusieurs flots d'instructions permet de prendre en compte de façon plus efficace des applications ou des parties d'applications qui sont traitées pratiquement en séquentiel sur les machines purement vectorielles ou pipelines.
En mode SPMD chaque PE est complètement indépendant des autres, il ne peut accéder qu'au banc de mémoire de même numéro. OPSILA est alors composé d'un ensemble de 16 processeurs séquentiels qui éxécutent un même programme de façon asynchrone sur des données locales propres aux bancs qui leurs sont associés (cf figure 1). Le code des sections SPMD est duppliqué au chargement dans les bancs aux mêmes adresses locales. L'unité scalaire initialise le mode SPMD en fournissant aux PE, via le PI, l'adresse locale dans les bancs du début de la section de code à éxécuter. Quand tous les PE ont terminés ils sont resynchronisés et OPSILA est de nouveau en mode SIMD. En conséquence ce mode de fonctionnement n'est efficace que si la charge de travail des PE est équilibrée (cf figure 2).
Le changement de mode est dynamique. Les synchronisations nécessaires pour initialiser le mode SPMD et pour retourner en mode SIMD sont du type "FORK-JOIN" global sur l'ensemble des PE. Comme ces synchronisations sont simples, leur mise en oeuvre est efficace : le passage au mode SPMD s'effectue en un cycle de la machine et le retour est effectif un cycle après la fin d'éxécution du dernier PE.
Dans le mode SPMD les processeurs ne peuvent pas échanger d'informations par le réseau d'interconnexion. Les échanges n'ont lieu qu'en mode SIMD.

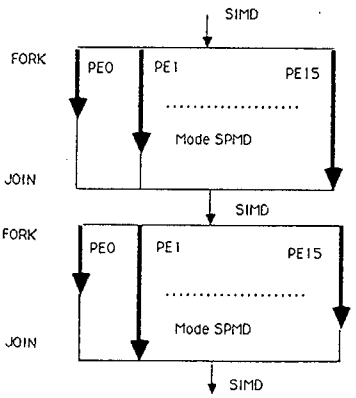


FIGURE 2 : Mécanismes de synchronisation entre les modes SIMD et SPMD

II-3 Outils de programmation

Le calculateur OPSILA est programmable à plusieurs niveaux. Les applications sont écrites en langage HELLENA [Jegou 86]. C'est un langage séquentiel de type PASCAL étendu par des instructions vectorielles (SIMD) et parallèles (SPMD). Un assembleur est également disponible.
Les instructions vectorielles de HELLENA sont issues d'une extension des opérateurs

scalaires aux tableaux et vecteurs comme cela se fera dans FORTRAN 8X.

La notion de patron d'accés étend ces facilités au traitement de sous-objets vectoriels comme par exemple une diagonale ou une colonne d'un tableau à deux dimensions.

La structure "spmd ... fin_spmd" définit une séquence à exécuter en mode SPMD.

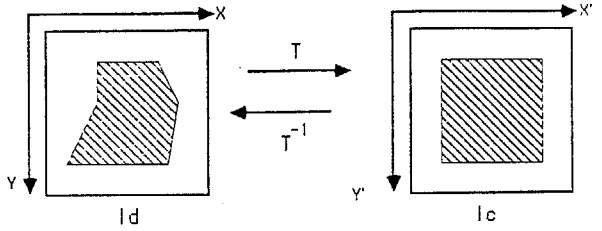
III MISE EN OEUVRE DES ALGORITHMES DE TRANSFORMATIONS GEOMETRIQUES SUR OPSILA

III-1 Les algorithmes de transformations géométriques

Les transformations géométriques interviennent dans de nombreuses applications du traitement des images (imagerie médicale, reconnaissance des formes, cartographie aérienne).

La complexité numérique de ces algorithmes est telle qu'ils justifient pour les plus utilisés d'entre eux (la rotation par exemple) la conception d'opérateurs spécialisés [Gasti 86] [Kajfaz 85].

Le problème consiste à déterminer une Image corrigée Ic à partir d'une Image déformée Id et d'un modèle T de la déformation : Ic = T(Id) (cf Figure 3).



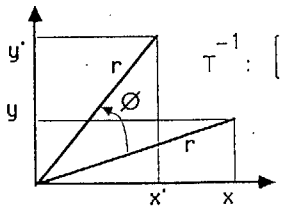
T : Modèle de Déformation Géométrique
Id : Image Déformée ; Ic : Image Corrigée
Corrections simples : translation, rotation, perspective
Corrections plus complexes : compensation de déformations topologiques

Figure 3 : transformations géométriques.

Pour des raisons de cadrage on préfère considérer la transformation inverse T^{-1} qui associe à chaque pixel de l'image corrigée une valeur d'intensité issue des pixels de l'image dégradée.

Les algorithmes de transformations géométriques comportent conceptuellement 3 phases :

a) Le calcul des coordonnées transformées par application du modèle de déformation. Nous présentons ci-dessous la géométrie et le modèle de déformation T^{-1} associés à la rotation.



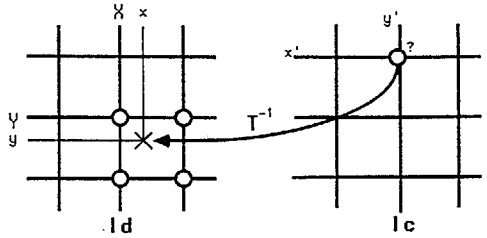
$$T^{-1} : \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \text{Cos}\varnothing & \text{Sin}\varnothing \\ -\text{Sin}\varnothing & \text{Cos}\varnothing \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

b) Le résultat de l'application du modèle de déformation est à valeur dans RxR, et ne coïncide donc pas avec la grille de discrétisation de l'image Id. Une phase d'interpolation entre un voisinage de pixels est nécessaire. Il faut **accéder à un ensemble local de pixels** appartenant à Id.

c) Le calcul de la valeur d'intensité associée à chaque pixel de Ic en fonction des valeurs d'intensité des pixels appartenant au voisinage précédemment décrit, et d'une fonction d'interpolation. Le choix de cette fonction dépend de l'application visée. La méthode du plus proche voisin qui est la plus simple à réaliser donne des résultats visuels trés médiocres. Nous préférons implémenter l'interpolation bilinéaire qui permet d'obtenir des résultats tout à fait correct pour un coût calcul raisonnable. Cette fonction d'interpolation nécéssite l'accés à un voisinage de 4 pixels. Elle est définie par :

$$\begin{aligned} Ic[x',y'] := {} & (1-\alpha)(1-\beta)\ Id[X,Y] \quad + \\ & (1-\alpha)\beta\ \ Id[X,Y+1] \quad + \\ & \alpha(1-\beta)\ \ Id[X+1,Y] \quad + \\ & \alpha\beta\ \ Id[X+1,Y+1]. \end{aligned}$$

avec : X = partie_entière(x),
Y = partie_entière(y),
α = x-X, β = y-Y.



III-2 Implémentation sur OPSILA

La principale difficulté de mise en oeuvre de cette classe d'algorithmes sur des machines parallèles est due à la nature irrégulière des accés à réaliser sur les pixels de l'image déformée. Un autre obstacle réside dans l'opération de clipping. Elle consiste à vérifier que chaque coordonnée transformée appartient bien à l'espace de l'image Id. Dans l'affirmative l'algorithme se poursuit par la phase d'interpolation et dans le cas contraire la valeur d'intensité associée au point traité est forcée à une valeur arbitraire. Il faut donc être capable de réaliser efficacement des tests en parallèle.

Nous proposons une méthode générale de mise en oeuvre de cette classe d'algorihtmes basée sur le mode SPMD (parallélisme asynchrone) et les accés vectoriels indirects GATHER.

Le mode SPMD est synonyme de partitionnement de l'espace des données. Dans notre cas, nous confions à chaque processeur élémentaire la charge d'élaborer un sous-ensemble des pixels de l'image corrigée Ic (cf Figure 4). Chaque PE connait les coordonnées des pixels qu'il possède dans son banc de mémoire puisqu'il sait à la fois son numéro d'ordre parmi p (cf I), la fonction de rangement des objets en mémoire vectorielle et le nombre de lignes et de colonnes de l'image à traiter Ic. Sachant que les matrices sont rangées linéairement colonne par colonne en mémoire, dans le cas d'une image 256x256 chaque PE doit traiter l'ensemble des lignes de numéro :

n0_PE + k*16 + 1, k ∈ [0,256/16-1]

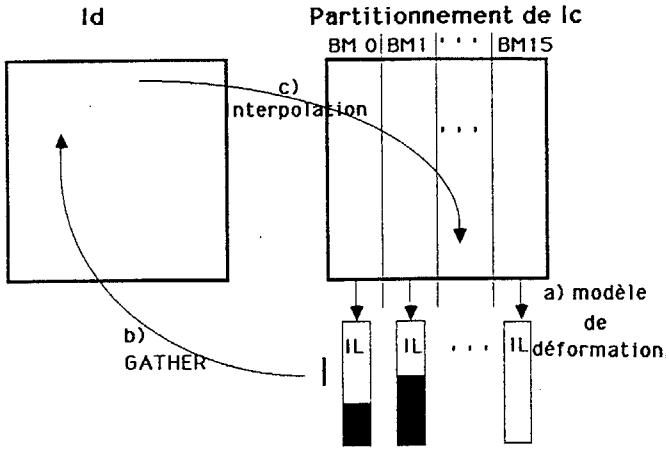


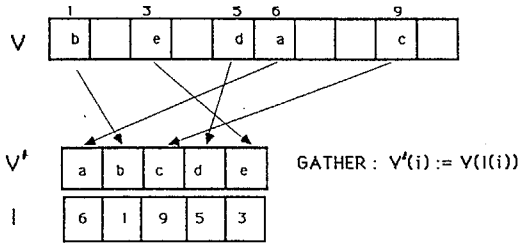
Figure 4 : Principes de parallélisation.

Le calcul des coordonnées transformées est réalisée en mode SPMD. Chaque PE applique localement le modèle de déformation aux pixels qu'il possède dans son banc.

Il en déduit pour chaque pixel, dans le cas ou le clipping ne s'applique pas, un ensemble de coordonnées à valeur dans Id définissant le voisinage utilisé par la fonction d'interpolation. Ces coordonnées

sont stockées dans un vecteur local d'indices IL. Le mode SPMD permet de réaliser très efficacement la phase de clipping qui est un traitement conditionnel.

L'accès aux pixels de Id est réalisé en mode SIMD grâce à la fonction d'accès vectoriel indirect GATHER dont le principe est illustré ci-dessous.



L'ensemble des vecteurs d'indice locaux IL constitue un vecteur global I (cf Figure 4). Les vecteurs d'indice locaux IL n'ont pas tous la même longueur car leur contenu dépend du volume des opérations de clipping éxécutées localement par les PE.

Afin de pouvoir les considérer globalement comme un vecteur unique d'indice, ils sont complétés par des valeurs d'indice non-significatives du GATHER (zone sombre sur la figure 4). La longueur du vecteur global I est donc égale à 16 fois celle du plus grand vecteur local IL. Les pixels lus par la fonction GATHER sont stockés dans une structure intermédiaire qui n'est pas représentée sur la Figure 4.

La phase d'interpolation est réalisée en mode SPMD. Chaque PE traite localement les valeurs d'intensité des pixels de Id obtenus par le GATHER, et modifie en fonction du schéma d'interpolation les pixels de la partition de Ic qu'il possède dans son banc.

III-3 Evaluation des performances

Le coût des transformations géométriques est dû principalement à la complexité du modèle d'interpolation qui peut consister en un système de 6,8 ou 12 équations et aussi à celle de la phase d'interpolation. L'interpolation bilinéaire nécéssite de l'ordre de 10 opérations arithmétiques par pixel, et l'interpolation bicubique de l'ordre de 100.

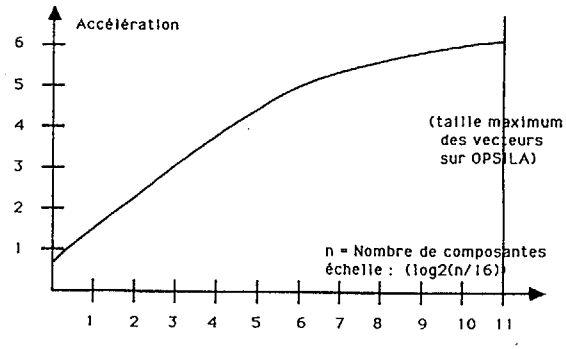
Nous définissons l'accélération comme étant le rapport des temps d'éxécution des algorithmes séquentiel et vectoriel.

Dans ces deux phases, essentiellement de calculs, l'accélération est proche du nombre de processeurs mis en jeu. Il s'agit d'un schéma de parallélisation très efficace. En effet, il n'y a pas de synchronisations ni d'échanges de données entre les processeurs élémentaires au cours de ces deux phases.

La figure ci-dessous illustre les performances des accés vectoriels indirects GATHER dans le cas de la rotation d'une image 256x256 avec interpolation bilinéaire.

L'accélération obtenue est de l'ordre de 6 pour des vecteurs de grande taille ce qui est notre cas.

Il faut remarquer que le coût total des accés de type GATHER est faible relativement au coût de calcul de cette classe d'algorithme.



Dans le cas de la rotation d'une image 256x256 avec un schéma d'interpolation bilinéaire les accés vectoriels indirects GATHER représentent 21% du coût total de l'algorithme.

L'accélération de l'algorithme parallèle de rotation par rapport à une mise en oeuvre séquentielle est de 14.

L'approche que nous proposons ici peut directement être étendue à une machine offrant un parallélisme plus large.

La fonction de répartition des pixels dans les bancs de la mémoire vectorielle (rangement linéaire par colonne) assure l'équilibrage de la charge de travail des processeurs. Elle permet également de traiter avec la même efficacité des fenêtres de taille quelconque.

IV CONCLUSION

Les études que nous avons réalisé à ce jour [Duclos 86] ont montré l' adéquation de l'architecture OPSILA à traiter les algorithmes du bas niveau du traitement numérique des images. Nous pouvons citer comme réalisations la convolution par un filtre de taille quelconque, la Transformée Rapide de Fourier, le filtrage médian, et le calcul d'histogramme.

Nous avons illustré dans le présent article les possibilités de cette architecture dans le cadre des transformations géométriques.

L'intérêt de la méthodologie est qu'elle autorise la résolution de tous les problèmes de corrections géométriques, et particulièrement les corrections de déformations topologiques (warpers) avec un taux de parallélisme maximum.

Nos recherches concernent maintenant les algorithmes traitant les structures de donnée de type liste et graphe (détection de contours et création de chaines, optimisation d'étiquetage par relaxation).

B I B L I O G R A P H I E

Auguin M. "Etude et réalisation de structures de calculs parallèles. Application aux méthodes numériques." Thèse d'état Univ. Nice 1985

Duclos P. Boeri F. "Etude du parallélisme en traitement des images orientée vers des réalisations sur l'architecture OPSILA." Rapport de recherche sur contrat DRET. LASSY Nice 1986

Gasti W. Armbruster P. "Processeurs detransformations géométriques." CESTA p433 Nice 1986

Jegou Y. "Définition du langage Hellena." Rapport technique INRIA 1986

Kajfaz P. Zavidovique B. "Implémentation matérielle de transformations géométriques bibimensionnelles d'une image." Grenoble 17ieme colloque AFCET 1985

Schubert K. "Vectorisation et mise en oeuvre sur OPSILA de la méthode des éléments finis."Thèse de Docteur-Ingénieur." Univ.Nice 1987

Siegel H.J. "Image Processing on a Partitionable SIMD/MIMD Machine." Langage&Architecture for Image Processing. Duff&Levialdi Academic Press 1981