QUATORZIEME COLLOQUE GRETSI - JUAN-LES-PINS - DU 13 AU 16 SEPTEMBRE 1993

Implantation d'algorithmes de traitement d'images sur une architecture multi-DSP avec l'environnement d'aide à l'implantation SynDEx

C. Milan* G. Richard* M. Paindavoine* Y. Sorel** Ch. Lavarenne**

* Université de Bourgogne, Laboratoire GERE, 6 bd Gabriel, 21000 Dijon
** INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex

Les algorithmes de traitement d'images impliquent des volumes de calcul suffisamment importants pour que leur implantation temps-réel nécessite des architectures multi-processeur. L'environnement d'aide à l'implantation SynDEx offre un modèle graphe flot de données conditionné pour spécifier à la fois les aspects régulier et irrégulier de ces algorithmes, une heuristique basée sur des transformations de graphes réalisant une adéquation entre l'algorithme et une architecture cible, et une génération automatique d'exécutif distribué sans interblocage. L'utilisateur est ainsi libéré des tâches lourdes de programmation bas-niveau et de la phase de mise au point temps-réel sur les multi-processeurs cibles, construits actuellement à partir de Transputers et de DSP TMS320C40.

Image processing algorithms involve computation amounts big enough to require multiprocessor architectures for their real-time implementation. For this purpose, the SynDEx environment offers a conditionned dataflow graph model to specify both the regular and irregular parts of image processing algorithms, a heuristic based on graphs transformations carrying out an adequation between the algorithm and the target architecture, and an automatic generator of dead-lock free distributed executives. Low level programming and real-time debugging of multiprocessor targets are thus drastically reduced. Present targets are based on Transputers and TMS320C40 DSPs.

1 Introduction

Les applications de traitement d'images temps-réel telles que le contrôle de qualité non destructif ou la vision robotique font appel à des algorithmes qui se décomposent habituellement selon trois niveaux :

- Le bas niveau concerne les prétraitements d'images. Les algorithmes utilisent en général des techniques de filtrage linéaire ou non linéaire, ces transformations s'appliquent à chaque pixel de l'image. Les données à traiter sont nombreuses et les algorithmes sont essentiellement réguliers (traitement identique pour chaque donnée).

- Le moyen niveau s'intéresse à l'extraction de primitives dans l'image traitée au bas niveau. Ce niveau consiste également à réduire le volume des données traitées : la structure des informations est de type image en entrée et de type liste en sortie.

- Le haut niveau concerne la fusion des primitives (listes) et effectue l'interprétation de la scène observée. Les algorithmes à ce niveau sont essentiellement irréguliers avec beaucoup de conditionnement.

Ces trois niveaux sont habituellement spécifiés, analysés et implantés au moyen de méthodes différentes. On présente d'abord un modèle de spécification commun aux algorithmes réguliers et irréguliers, le *graphe flot de données conditionné*, permettant l'analyse et l'implantation des trois niveaux simultanément. On présente ensuite une méthode d'implantation optimisée pour une exécution temps-réel de ces algorithmes sur des machines multi-processeurs. On illustre ensuite le modèle flot de données conditionné par un cas concret d'algorithme : la segmentation d'image au sens contours. Enfin on décrit les architectures multi-DSP utilisées.

2 Modèle flot de données

Les applications visées sont réactives, c'est-à-dire répondent aux stimuli de leur environnement en produisant des réactions et/ou en changeant leur état interne. De plus elles sont temps-réel, c'est-à-dire que la durée de chaque réaction est bornée. Dans le cas

du traitement d'image, le nombre de données traitées nécessite un important volume de calcul qui impose, pour son exécution en temps-réel, une architecture multi-processeur.

La réaction peut être modélisée par un graphe flot de données conditionné dont chaque sommet de type action représente un calcul ou un conditionnement ou une mémoire d'état ou une entrée-sortie, et dont chaque arc représente un flot de données, c'est-à-dire un transfert de données entre sommets, donc une précédence entre ces sommets. Ce modèle fait apparaître le parallélisme potentiel de l'algorithme (ordre partiel) qui permettra d'exploiter facilement le parallélisme disponible de l'architecture.

Une partie régulière d'un algorithme se représente simplement en répétant un sous-graphe appelé généralement "motif". Cette répétition de motif peut être factorisée (figure 1) afin d'une part de réduire le volume de la spécification, d'autre part de transformer la répétition en itération (transformation spatiale → temporelle). Dans le cas d'une implantation multi-processeur, la distribution des motifs sera faite de telle manière que chaque processeur exécute une partie de l'itération.

En plus des types de sommets utilisés pour les graphes irréguliers, on utilise pour les parties régulières les types de sommets suivants :

- *Fork* : flot vecteur → flot élément du vecteur
- *Join* : flot élément du vecteur → flot vecteur
- *Iterate* : dépendance inter-motifs
- *Index* : indice du motif

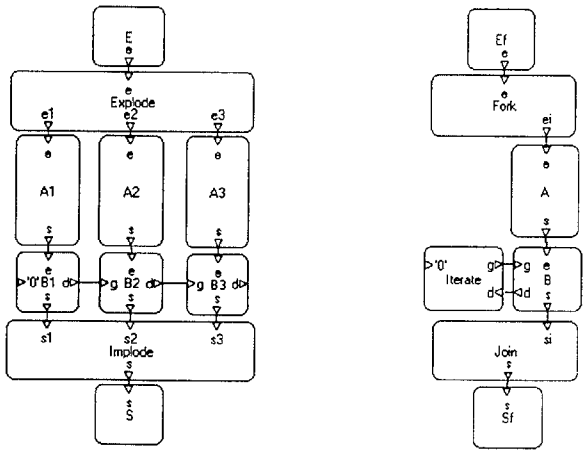
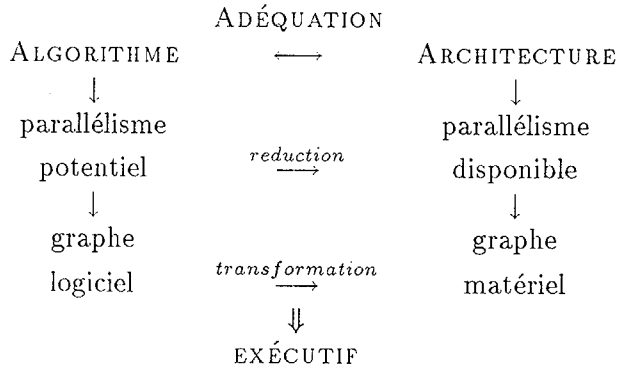


Figure 1: Factorisation d'un graphe régulier

3 Méthode d'implantation

Pour implanter un algorithme sur une architecture tout en respectant les contraintes temps-réel, on cherche à réaliser une adéquation entre l'algorithme et l'architecture. Cette dernière est représentée également par un modèle de graphe. L'adéquation consiste à réduire le parallélisme potentiel de l'algorithme au parallélisme matériel disponible. Cette réduction est obtenue en transformant progressivement le graphe flot de donnée jusqu'à ce qu'il corresponde au graphe matériel. Le produit de cette transformation est l'exécutif distribué temps-réel, permettant l'exécution de l'algorithme sur l'architecture.



Une adéquation efficace est obtenue en choisissant la transformation qui optimise les performances temps-réel. Nous avons développé une heuristique d'optimisation [1] basée sur des calculs dans l'algèbre $(\max, +)$ utilisant les durées d'exécution des sommets et des communications inter-processeur. Ces durées sont elles-mêmes calculées grâce au modèle matériel, paramétré par les caractéristiques des composants de l'architecture.

SynDEx [2] [3] est un environnement graphique interactif de développement pour applications de traitement du signal et des images. Il utilise les modèles et l'heuristique décrits précédemment. Il permet la saisie des graphes matériel et flot de donnée. Ce dernier peut également être spécifié au moyen des langages synchrones [4] et vérifié à l'aide de leurs compilateurs. Après optimisation, un diagramme prévisionnel d'exécution temps-réel de l'algorithme sur le multi-processeur est visualisé. Enfin, un exécutif sans interblocage est généré automatiquement en langage C pour des architectures à base de processeurs de type Transputer ou TMS320C40, libérant ainsi l'utilisateur des tâches lourdes de programmation bas niveau et évitant la phase de mise au point temps-réel sur le multi-processeur cible.

4 Segmentation d'image

Les trois niveaux d'un algorithme de segmentation d'image au sens contours correspondent à : un filtrage permettant de binariser l'image pour faire ressortir les pixels des contours, un suivi de ces contours [5] [6] pour générer des listes de codes de Freeman, et une fusion de ces listes pour déterminer les points caractéristiques d'objets en mouvement [7].

Ceci correspond au graphe factorisé de la figure 2. Le flot d'images acquis par le sommet `Image` est diffusé à chaque sommet `Filtre` (un par pixel, la factorisation du graphe permet de n'en représenter qu'un seul) qui réalise un filtrage sur les pixels voisins (à échelle de gris) pour générer un pixel binaire. Les flots de pixels sont réunis par le sommet `JoinP` pour former l'image binarisée diffusée aux sommets `Suivi` qui assemblent, de motif en motif, les listes de pixels contours. L'interdépendance entre motifs est représentée par le sommet `Iterate`. Il faut noter qu'au bas niveau, il n'y a pas de dépendance inter-motifs `Filtre`. Les flots de listes sont réunis par le sommet `JoinL` pour être interprétés par le sommet `Fusion`.

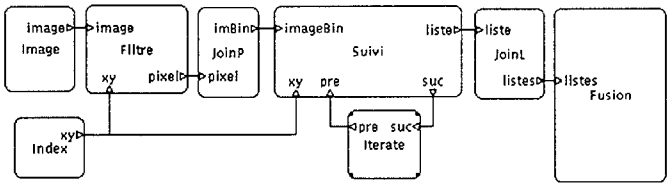


Figure 2: Algorithme de segmentation

On décrit maintenant un algorithme de suivi de contours et on donne figure 3 le graphe flot de données conditionné correspondant.

On définit un contour comme une ligne brisée (chaîne) constituée de segments successivement horizontaux et verticaux composés de pixels tous à 1 (les pixels de fond sont à 0). Le codage de Freeman d'une chaîne consiste à donner d'abord les coordonnées du pixel d'une des deux extrémités de la chaîne puis les coordonnées relatives des pixels suivants de la chaîne, codées dans l'ensemble {est, nord, ouest, sud}.

L'algorithme se décompose en quatre parties, chacune effectuant une réduction de données et un sous-échantillonage par rapport à la partie précédente :

1. Détection des extrémités des segments des lignes brisées : $P_{x,y}$ est une extrémité de segment horizontal si xh est vrai, vertical si xv est vrai :

$$xh = (P_{x,y} = 1) \wedge (P_{x-1,y} \neq P_{x+1,y})$$

$$xv = (P_{x,y} = 1) \wedge (P_{x,y-1} \neq P_{x,y+1})$$

   C'est une extrémité de la ligne brisée si une seule de ces deux conditions est vraie.

2. Constitution séparée des segments horizontaux et verticaux : deux extrémités qui partagent une même coordonnée x (resp. y) forment un segment horizontal (resp. vertical).

3. Constitution des lignes brisées : les segments sont assemblés par identité d'extrémités, l'assemblage d'une ligne brisée est terminé lorsqu'on a trouvé les deux extrémités de la ligne brisée.

4. Transformation d'une ligne brisée en liste de codes de Freeman : chaque segment de n pixels est expansé en n codes de Freeman identiques.

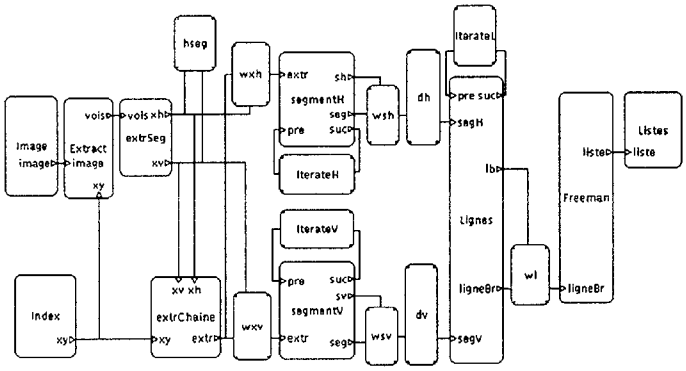


Figure 3: Moyen niveau : suivi de contours

Voici le rôle de chacun des sommets du graphe :
- `hseg`, `wxh`, `wxv`, `wsh`, `wsv`, `dh`, `dv` et `wl` sont des actions de conditionnement
- `IterateH`, `IterateV` et `IterateL` représentent des dépendances inter-motifs
- `Index` génère l'indice du motif c'est-à-dire les coordonnées du pixel correspondant
- `Extract` extrait de l'image binarisée les pixels voisins de celui correspondant au motif
- `Image` et `Listes` correspondent aux sommets `JoinP` et `JoinL` du graphe complet représentant l'algorithme de segmentation (figure 2)
- `extrSeg` détecte les extrémités des segments de lignes brisées
- `extrChaine` détecte les extrémités de lignes brisées et marque la coordonnée du pixel correspondant
- `segmentH` (resp. `segmentV`) constitue les segments horizontaux (resp. verticaux)
- `Lignes` assemble les segments horizontaux et verticaux pour produire les lignes brisées
- `Freeman` transforme les lignes brisées en liste de codes de Freeman.

5 Architecture multi-DSP

L'algorithme de segmentation au sens contours nécessite une capacité de calcul importante, ce qui nous a conduit pour son implantation temps-réel à choisir une architecture multi-processeur. Dans un premier temps, pour valider notre algorithme, nous avons utilisé une architecture basée sur quatre Transputers T800. Dans un second temps, pour respecter les contraintes temps-réel, nous avons envisagé une architecture multi-DSP basée sur le processeur TMS320C40 de Texas Instrument [8].

Fonctionnant à 40 Mhz, ce processeur de traitement du signal peut atteindre les 40 MFLOPS. Grâce à ses 6 liens de communication d'un débit de 20 Moctets/sec chacun, il permet de construire facilement des architectures multi-processeurs à mémoire distribuée. De plus, la réalisation de ces architectures est simplifiée par l'existence d'un standard "TIM40" de cartes modulaires enfichables sur une carte mère. Certains de ces modules, en plus du processeur et de sa mémoire, intègrent un "Frame Grabber" pour l'acquisition d'image en temps-réel.

6 Conclusion et perspectives

L'application de segmentation d'images au sens contours en temps-réel est suffisamment complexe pour nécessiter une implantation multi-processeur. Les premières expérimentations ont montré que la puissance de calcul des Transputers T800 est insuffisante pour respecter les contraintes temps-réel, ce qui nous conduit à utiliser le processeur de traitement du signal TMS320C40. La modularité des cartes à base de TMS320C40 et l'environnement d'aide à l'implantation SynDEx permettent de déterminer, de valider l'architecture et de produire automatiquement l'exécutif nécessaire pour réaliser cette implantation. Cette approche réduit considérablement le cycle de développement et conduit à des implantations optimisées et sans interblocage. La souplesse des architectures multi-processeurs basées sur le TMS320C40 et de l'environnement SynDEx permettent d'envisager la mise en œuvre d'architectures hétérogènes où les processeurs TMS320C40 coopèrent avec des opérateurs spécialisés non programmables (ASIC [9], carte dédiée...) réalisant des traitements bas niveau.

Bibliographie

[1] C. Lavarenne, Y. Sorel: *Performance Optimization of Multiprocessor Real-Time Applications by Graph Transformations.* Parallel Computing 93 (Septembre 1993).

[2] C. Lavarenne, O. Seghrouchni, Y. Sorel, M. Sorine: *The SynDEx software environment for real-time distributed systems design and implementation.* European Control Conference (Juillet 1991).

[3] C. Lavarenne, O. Seghrouchni, Y. Sorel, M. Sorine: *SynDEx, un environnement de programmation pour applications de traitement du signal distribuées.* 13ème Colloque GRETSI (Sept. 1991).

[4] A. Benveniste, G. Berry: *The Synchronous Approach to Reactive and Real-Time Systems.* Proc. of the IEEE vol79, n.9, pp.1270–1282 (1991)

[5] T. Pavlidis : *A thinning algorithm for discrete binary images.* Computer vision, graphics and image processing, Vol 2, chap 11 (1980).

[6] S. Ubeda : *Comparaison d'algorithmes d'amincissement d'images sur machine à base de transputer.* La Lettre du Transputer n° 13 (1992).

[7] E. Fauvet, M. Paindavoine, F. Cannard : *Human movement analysis with image processing in real time.* 19th International Congress on high speed photography & photonics – Cambridge (Sept. 1990).

[8] Texas Instrument : *TMS320C40 Manual reference.*

[9] M. Robert, M. Paindavoine, P. Gorria : *Architectures for integration of real time image processing systems.* IEEE Workshop on VLSI signal processing – Napa USA (1992).