EVALUATION DE L'OUTIL SYNDEX POUR L'IMPLANTATION D'UN ALGORITHME D'ETIQUETAGE EN COMPOSANTES CONNEXES SUR LA MACHINE TRANSVISION

Dominique GINHAC, Jocelyn SEROT, Jean Pierre DERUTIN

LASMEA - URA 1793 CNRS, Campus des Cézeaux, 63177 Aubiere
e-mail: Dominique.Ginhac@lasmea.univ-bpclermont.fr

RÉSUMÉ - L'objet de ce papier est de présenter les résultats de diverses implantations d'un algorithme d'Etiquetage en Composantes Connexes sur une architecture MIMD à mémoire distribuée dédiée à la vision (Transvision). Cette implantation a été effectuée en utilisant la chaîne de développement SIGNAL-SynDEx et avait pour but de tester l'efficacité d'un tel outil dans l'optique de prototypage rapide d'algorithmes de vision artificielle à fortes contraintes temporelles.

1 INTRODUCTION

La complexité sans cesse grandissante des applications temps réel de traitement d'images rend inévitable le recours à des machines parallèles dédiées. Toutefois, malgré les travaux intensifs suscités par ce sujet, la programmation de ces machines demeure un exercice délicat. Pour programmer une architecture MIMD à mémoire distribuée comme la machine Transvision [DBG91], l'utilisateur est notamment conduit à décomposer explicitement son application en tâches indépendantes et communiquantes puis à placer et ordonnancer manuellement ces tâches sur le réseau physique de processeurs.
L'étude réalisée consiste à tester et valider une méthodologie d'adéquation automatique algorithme-architecture. Cette méthode s'appuie sur le langage synchrone SIGNAL [GBGM91] pour

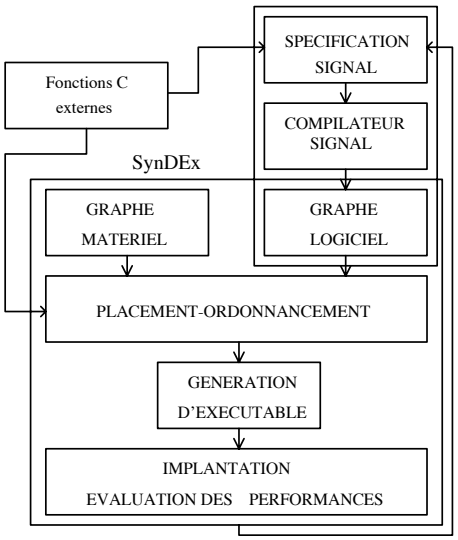


FIG. 1 – *La chaîne de développement SIGNAL-SynDEx*

la validation algorithmique et sur l'environnement de programmation SynDEx [LSSS91] pour la génération de l'exécutif distribué associé (cf. figure 1). A partir d'une spécification purement flot de données de l'application et d'une description de la machine cible, SynDEx utilise une heuristique de minimisation du temps de réponse pour placer et ordonnancer de manière automatique l'algorithme sur l'architecture cible. L'utilisateur est ainsi libéré des tâches lourdes de programmation bas niveau et de la phase de mise au point de l'algorithme sur la machine MIMD. L'utilisation de cette chaîne de développement vise à réduire de manière drastique le temps de cycle *conception-implantation*, objectif primordial dans une optique de prototypage rapide d'algorithmes.
Ce papier présente les résultats des diverses implantations d'un algorithme d'Etiquetage en Composantes Connexes réalisées en utilisant la chaîne SIGNAL-SynDEx. Il s'inscrit dans le cadre plus général d'une collaboration entre l'INRIA (projet SOSSO) et le LASMEA pour l'évaluation de l'outil SynDEx sur la machine Transvision.

2 L'ALGORITHME D'ETIQUETAGE EN COMPOSANTES CONNEXES

L'algorithme à implanter est un Etiquetage en Composantes Connexes (ECC), algorithme de traitement d'images moyen niveau largement utilisé dans diverses applications de vision artificielle. Par ailleurs, des implantations de l'ECC ont déjà été réalisées sur diverses machines cibles tant SIMD [CSS90] que MIMD [NJR90]. Notre méthodologie d'implantation peut ainsi

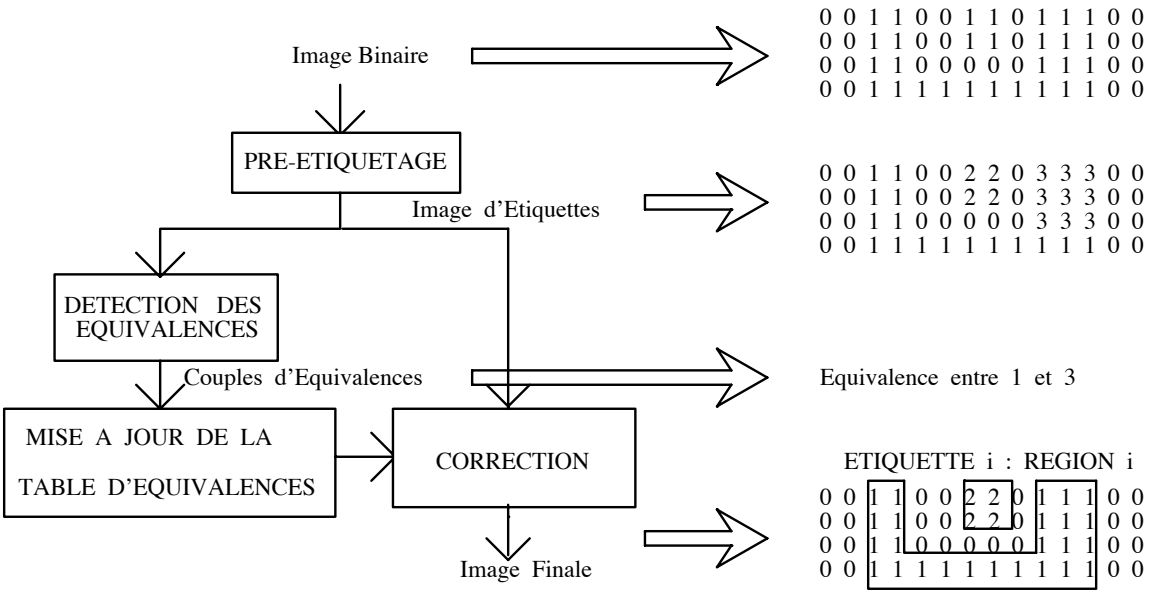


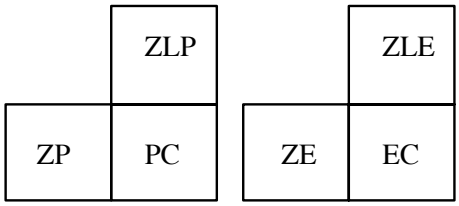
FIG. 2 – *Schéma fonctionnel de l'ECC*

être comparée en terme d'expressivité et de performances par rapport à ces implantations.
L'ECC consiste à traiter une image dans le but de discerner les objets la composant. Pratiquement, tous les pixels appartenant à une même composante connexe se voient attribuer une et une seule étiquette.
L'algorithme nécessite classiquement trois phases successives (cf. figure 2) :

1. une phase de pré-étiquetage calculant une image provisoire des étiquettes en fonction du voisinage immédiat des pixels,

2. une phase de détection des conflits d'étiquettes, conduisant à la construction d'une *table d'équivalence*,

3. une phase de correction réalisant la fusion des étiquettes equivalentes.



REGLES D'ATTRIBUTION D'UNE ETIQUETTE			
pc	zlp	zp	ec
0	X	X	0
1	1	0	zle
1	0	1	ze
1	0	0	cpt+1 création d'une nouvelle étiquette
1	1	1	ec = min(ze,zle) et détection d'une équivalence

FIG. 3 – *Masque en L inversé*

Une étude bibliographique nous a conduit à envisager 3 algorithmes correspondant à ce schéma:

- L' algorithme classique, décrit par exemple dans [ECC92] et basé sur un pré-étiquetage local par un masque en L inversé (cf. figure 3).

- Une première variante de cet algorithme [PRA93], pour laquelle une seule étiquette est affectée à chaque segment horizontal, la propagation de cette étiquette étant assurée par détection des connexités verticales entre segments.

- Une seconde variante, décrite par Selkow [SEL72], pour laquelle la phase de pré-étiquetage se déroule concurrement avec celle de détection des équivalences et peut donc effectuer certaines pré-corrections [1].

3 IMPLANTATIONS ET RESULTATS

Les implantations ont été réalisées en suivant un schéma de parallélisation de type SPMD [2] ce qui signifie que les mêmes opérateurs de calculs seront appliqués sur des données différentes. Pratiquement, l'image acquise est divisée en données de taille fixe, lesquelles sont réparties sur l'ensemble des processeurs.
Ce partage de données, effectué statiquement, introduit la notion de *granularité* de données, qui correspond en première approximation à la taille des communications inter-processeurs. Compte-tenu du format des images traitées, trois tailles de grain ont été envisagées: pixel, ligne et bande [GIN95].

3.1 Parallélisation avec un grain pixel

La programmation de l'algorithme d'ECC avec une granularité pixel peut s'effectuer intégralement en langage SIGNAL, qui sert alors de point d'entrée à l'outil d'aide à la parallélisation

1. Ces deux variantes ont pour but de diminuer le nombre d'erreurs introduites par la phase de pré-étiquetage et donc de réduire la taille et le temps de construction de la table d'équivalence.
2. Single Program Multiple Data

SynDEx. L'ensemble des opérateurs requis par la phase de pré-étiquetage (pour les trois versions de l'algorithme décrit au paragraphe 2) ont ainsi été écrits puis validés en SIGNAL [3].
Nous nous sommes toutefois heurtés à des difficultés pratiques lors de la génération effective du code pour la machine cible, difficultés liées à l'incomplétude de la passerelle SIGNAL-SynDEx à l'heure actuelle. L'instruction SIGNAL *array*, notamment, utilisée pour la synthèse de motifs répétitifs, suppose l'existence d' instructions SynDEx *fork*, *join* et *iterate* pour exprimer le repliement spatio-temporel au sein des graphes flot de données. Ces instructions ne sont pas supportées par la version actuelle de l'heuristique de placement-ordonnancement. Or, en leur absence, la formulation flot de données d'algorithmes opérant sur des images avec un grain pixel conduit à générer un nombre considérable de processus élémentaires (ou actions) interconnectés (65536 processus de pré-étiquetage pour une image 256*256 pixels dans le cas de l'ECC par exemple), ce qui amène très rapidement à des temps de placement-ordonnancement totalement prohibitifs.

3.2 Parallélisation avec un grain ligne

Au vu des limites énoncées au paragraphe précédent, une nouvelle approche de programmation a été mise en œuvre. Les fonctions associées aux actions du graphe SynDEx peuvent être des instructions SIGNAL élémentaires mais aussi des fonctions externes écrites en C [4].
Ces fonctions permettent d'augmenter la granularité des données du graphe logiciel dans la mesure où elles peuvent regrouper l'équivalent de plusieurs instructions SIGNAL. Une bibliothèque d'opérateurs écrits en C et implantant les différentes phases de l'algorithme d'ECC (seuillage, pré-étiquetage, détection d'équivalences, correction) et opérant sur des données de type *ligne de pixels* a donc été programmée.
L'implantation de ce schéma sur un réseau de deux Transputers a toutefois révélé certains limites du modèle théorique exploité par l'heuristique de placement-ordonnancement. Par exemple, le temps de réponse effectivement observé pour une implantation sur une configuration à 2 transputers est dix fois supérieur aux prévisions théoriques calculées par SynDEx !..
L'explication de cet écart nous a conduit à analyser finement les mécanismes mis en œuvre par SynDEx pour la communication inter-processeurs. En bref, ces communications sont gérées dynamiquement par un exécutif distribué bati à partir d'un ensemble d'*objets génériques* [ELS92]. Ces objets — implantés sous la forme de processus alloués de manière transparente à l'utilisateur par le générateur de code — assurent le formatage, le routage et le transfert des messages entre les processus de calcul définis par l'utilisateur. physiques. Lors d'une communication entre deux Transputers, les messages passent ainsi par (cf. figure 4):

- une porte d'émission P.E. attachée au processus émetteur qui récupère la donnée à transmettre,
- le bus logiciel B.L. qui effectue le routage des messages,
- une porte d'accès P.P.L. au lien physique,
- le lien physique,
- la porte d'accès P.P.L. du processeur récepteur,
- le bus logiciel du processeur récepteur,

3. Par génération de code C séquentiel
4. Fonctions au sens strict du terme, c'est à dire sans effet de bord, sans mémoire et à temps d'exécution constant

– une porte de réception P.R. qui va fournir la donnée transmise au processus demandeur.

Chacun des processus associés aux objets PE, PR, BL et PPL est exécuté en haute priorité à la différence du processus de calcul, traité lui en basse priorité. Chaque activation d'un de ces objets implique donc une interruption du processus de calcul avec sauvegarde de son contexte, interruption d'autant plus longue que le transfert de message entre ces objets se fait par recopie de tampons en mémoire. Par ailleurs, les séquences PE-BL-PPL d'une part et PPL-BL-PR

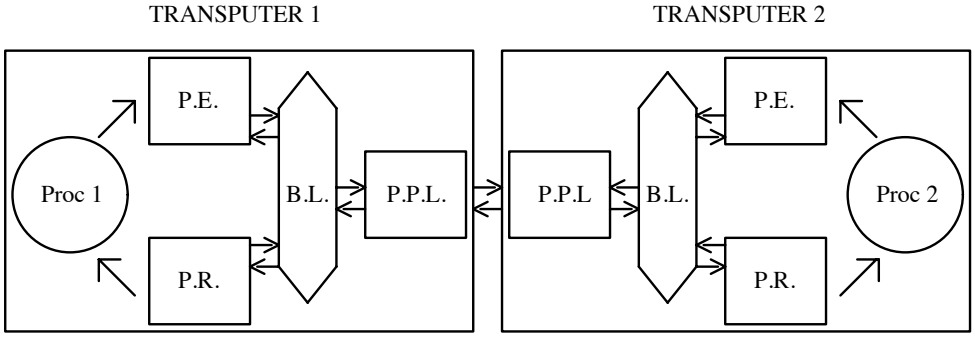


FIG. 4 – *Communication entre deux Transputers*

d'autre part sont supposées être exécutées de manière indivisible. Or, dès que plusieurs liens du Transputer sont sollicités pour effectuer des transferts de messages en parallèle, on assiste à un entrelacement des séquences de formatage et de routage sur le processeur émetteur (par exemple, exécution de tous les processus PE puis BL et enfin PPL) [5].
Cet entrelacement, associé au coût des changement de contexte et des recopies mémoire entraîne globalement une augmentation non négligeable du temps d'initialisation (start-up) et des réductions importantes de débit effectif des communications.

3.3 Parallélisation avec un grain bande

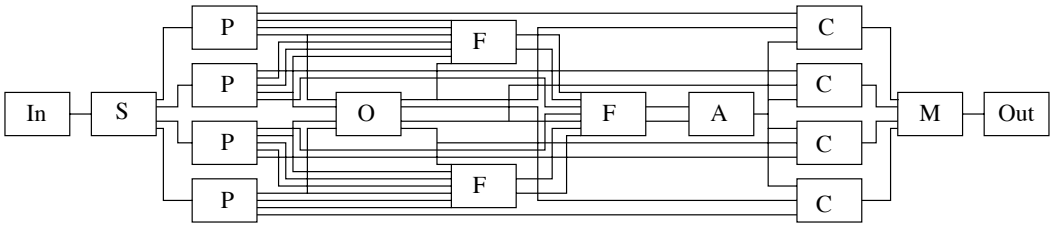
Les phénomènes mis en évidence au paragraphe précédent sont d'autant plus sensibles que la densité de communication est importante. Une solution pour limiter leur effet consiste donc à réduire le nombre de séquences de communication requises par l'application, en augmentant par exemple la granularité des données traitées.
Une manière de procéder consiste à diviser l'image source non plus en lignes mais en *bandes* consécutives. Le pré-étiquetage s'effectue alors de manière totalement indépendante sur chaque bande (cf. figure 5), une phase de fusion étant chargée de la gestion des équivalences inter-bandes [6]. A partir des tables d'équivalences produites séparément par les processus de pré-étiquetag de chaque bande, cette opération de fusion génère une table d'équivalence globale en répertoriant les connexités au niveau des frontières inter-bandes [7]. Cette table finale est ensuite envoyée aux opérateurs de correction [8]. L'expérimentation de cette implantation a dès lors mis en évidence deux aspects essentiels que l'heuristique de placement-ordonnancement SynDEx ne prenait pas en compte. Ces points sont illustrés sur la figure 6, qui montre les

5. Ce phénomène se répercute sur le processeur récepteur mais de manière plus complexe car les messages arrivent en général déjà décalés

6. On remarquera incidemment que le parallélisme potentiel de l'algorithme est alors plus important que dans les schémas précédents pour lesquels la phase de pré-étiquetage restait intrinsèquement séquentielle

7. Dans le cas d'une division en 2^n bandes, $2^n - 1$ fusions sont nécessaires. En fait, une derniere phase de correction — notée A sur la figure 5 — est nécessaire pour gérer les équivalences multiples

8. Chaque processus de correction des étiquettes n'effectuant son traitement que sur une bande, il est indispensable que cet opérateur puisse retrouver dans la table finale la partie correspondant à sa bande à traiter. Cette fonction est assurée par l'opérateur *Offset* noté O sur la figure 5



In : Acquisition de l'image
SP : Division de l'image en bandes
P : Pré-Etiquetage d'une bande
O : Calcul de la position des étiquettes de chaque bande dans la table d'équivalence finale
F : Fusion des tables d'équivalences par test des frontières des bandes deux à deux
A : Mise à jour de la table finale afin de détecter les équivalences multiples
C : Correction d'une bande
M : Regroupement des bandes traitées pour former l'image finale
O : Affichage de l'image résultat

FIG. 5 – *graphe flot de données de l'ECC pour n = 2*

graphes théorique (calculé par SynDEx) et réel (déduit du chronométrage de l'application sur l'architecture cible) résultant du placement-ordonnancement de l'application d'ECC sur 4 transputers. Le placement est représenté sur une échelle horizontale sur laquelle on affecte une colonne par processeur. L'ordonnancement des tâches est décrit verticalement par une échelle temporelle.
La différence la plus importante entre les deux vues se situe après l'étape S de division de l'image en bandes, étape réalisée par le processeur NV4.
D'une part, les mesures réelles montrent clairement que la tâche P de pré-étiquetage de la bande 4 ne démarre qu'après la fin des transferts des trois autres bandes vers les autres processeurs NVi alors qu'en théorie le processeur NV4 effectue simultanément le transfert de données et ce pré-étiquetage [9]. En pratique, les processus de communication et plus particulièrement les séquences internes de découpage et de routage des messages s'exécutent en haute priorité en interrompant le processus de calcul, comme on l'a vu au paragraphe 3.2. Le processus P de NV4 ne commence de ce fait son exécution qu'à la fin des séquences internes de formatage des messages destinés aux processeurs NV3, NV2 et NV1. Ce phénomème est totalement ignoré par le modèle utilisé dans l'heuristique de placement-ordonnancement qui suppose que les séquences de calcul et de communication sont exécutées en vrai parallélisme, comme le montre le graphe théorique de la figure 5 sur lequel l'opérateur P sur NV4 débute son exécution dès la fin de l'étape de division en bandes. Il en résulte un retard d'ordonnancement non pris en compte de t = 81 - 11 = 70 ms, retard qui se propage tout au long du graphe, entraînant un allongement significatif de la latence globale de l'application.
Ce recouvrement seulement partiel des calculs par les communications se traduit par ailleurs par un second phénomène lui aussi mis en évidence par la figure 6: Les opérateurs P de pré-étiquetage exécutés sur les processeurs NV1 à NV3 démarrent leur exécution avec un certain retard par rapport aux prévisions théoriques (retard mesuré de 13 ms pour NV1 à 63 ms pour NV3). Ces différences d'instant de démarrage sont dues aux communications entre le Transputer NV4 et les autres processeurs du réseau. Si les communications des données débutent bien

9. C'est une caractéristique essentielle du transputer d'autoriser le recouvrement calcul-communications grâce aux DMAs associés à chaque lien

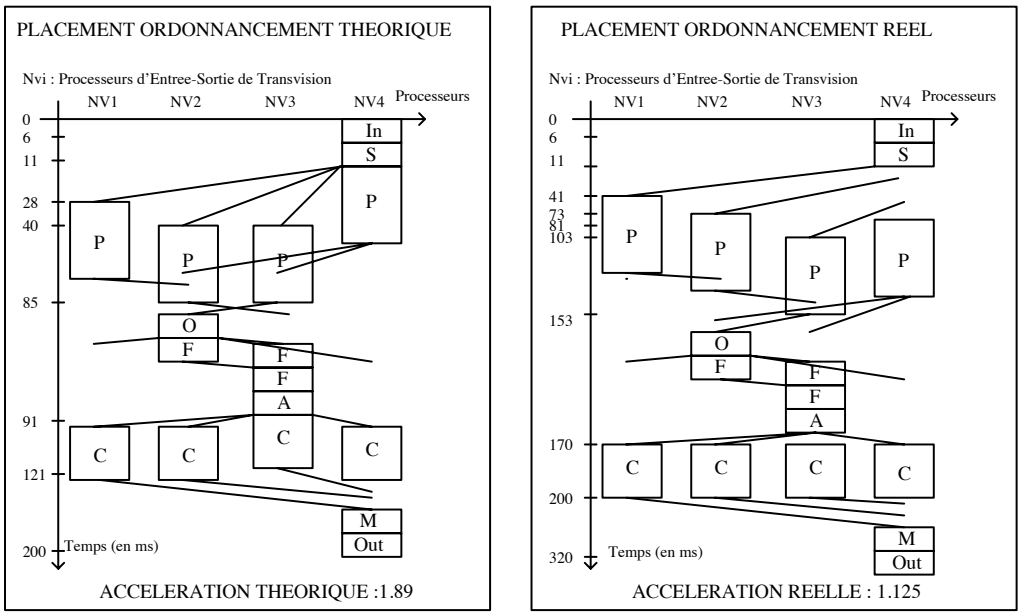


FIG. 6 – *Vues théorique et réelle du placement-ordonnancement de l'algorithme*

à t = 11 ms, conformément aux estimations de SynDEx, l'implantation réelle sur la machine cible révèle que les trois transferts sont en partie séquentiels. Le transfert d'un bloc de l'image comporte une première partie de découpage et de routage qui s'effectue par des recopies de mémoire et donc sollicite le CPU et une deuxième partie de transfert effectif sur le lien physique gérée de manière totalement indépendante par le DMA. Le modèle utilisé par SynDEx suppose que l'ensemble des processus associés aux communications (y compris les séquences de formatage et de routage) sont exécutés en parallèle. En théorie donc, les trois communications de NV4 vers les autres NVi sont purement parallèles. En réalité, seuls les transferts effectués par les DMA peuvent être parallèles puisqu'ils ne sollicitent pas le CPU. Cette différence entraîne aussi un allongement systématique du temps de réponse de l'application. Par suite, le facteur d'accélération (speedup) observé reste plus faible que la valeur théorique calculée par SynDEx.

4 CONCLUSION

Le prototypage rapide d'applications d'applications de vision temps-réel sur des machines multi-processeurs passe par le recours à des outils d'aide à la parallélisation dans la mesure où ceux ci permettent de libérer le programmeur des tâches lourdes de programmation bas niveau. En ce sens un outil tel que SynDEx, en automatisant les phases de placement-ordonnancement et de génération d'exécutif distribué sur une architecture MIMD à mémoire distribuée, constitue un gain de productivité significatif.
Toutefois, une analyse fine des applicatifs générés par la version 3.6 de SynDEx dans le cadre d'un algorithme réaliste de vision artificielle (l'Etiquetage en Composantes Connexes d'une image en niveaux de gris) met en évidence des différences significatives entre les performances théoriques calculées par l'outil et les performances réelles mesurées sur la machine cible. Ces différences peuvent être imputées essentiellement au surcoût résultant de la gestion dynamique des communications par les objets de l'exécutif distribué, surcoût non pris en compte par l'heuristique de placement-ordonnancement.
La version 4.0 de SynDEx, en cours de réalisation à l'INRIA devrait permettre d'éliminer presque totalement ce surcoût en s'appuyant sur un ordonnancement purement statique des

communications [10].
Pour notre part, les tests des premières briques de cette nouvelle version sur la machine multi-T800 Transvision ont déjà fourni des résultats encourageants et un portage sur la version T9000 de Transvision va être engagé. Une telle réalisation, en s'insérant dans la continuité du travail présenté ici sur T800, devrait permettre de concilier les avantages méthodologiques d'un outil tel que SynDEx avec les performances offertes par une architecture multi-T9000 dans le domaine de la vision artificielle à fortes contraintes temporelles.

Références

[CSS90] R. CYPHER, J.L.C. SANZ, and L. SNYDER. Algorithms for image componed labeling on simd mesh connected computers. In *IEEE Trans. Computers*, volume 32, Fevrier 1990.

[DBG91] J.P. DERUTIN, B. BESSERER, and J. GALLICE. A parallel vision machine: Transvision. In *Computer Architecture for Machine Perception - CAMP'91*, pages 241–251, Paris, Decembre 1991.

[ECC92] M. ECCHER. *Architecture parallèle dédiée à l'étude d'automates de vision en temps réel.* PhD thesis, Université de Franche Comté, Novembre 1992.

[ELS92] F. ENNESSER, C. LAVARENNE, and Y. SOREL. Méthode chronométrique pour l'optimisation du temps de réponse des exécutifs syndex. Rapport de recherche 1769, I.N.R.I.A. Institut National de Recherche en Informatique et en Automatique, Octobre 1992.

[GBGM91] P. LE GUERNIC, M. LE BORGNE, T. GAUTIER, and C. LE MAIRE. Programming real-time applications with signal. Rapport de recherche 1446, I.N.R.I.A. Institut National de Recherche en Informatique et en Automatique, Juin 1991.

[GIN95] D. GINHAC. Spécification et implantation d'un algorithme flots de données d'etiquetage en composantes connexes sur la machine multiprocesseurs à mémoire distribuée transvision. Mémoire de DEA d'Electronique et Systèmes, Université Blaise Pascal de Clermont-Ferrand, Juin 1995.

[LSSS91] C. LAVARENNE, O. SEGHROUCHNI, Y. SOREL, and M. SORINE. Syndex, un environnement de programmation pour applications de traitement du signal distribuées. In *Actes du treizième colloque GRETSI*, Juan Les Pins, Septembre 1991.

[NJR90] H.T. NGUYEN, K.K. JUNG, and R. RAGHAVAN. Fast parallel algorithms : from images to level sets and labels. In *Parallel Architectures for Image Processing*, volume 1246, pages 162–176, 1990.

[PRA93] S. PRAUD. Implantation d'un algorithme d'étiquetage en composantes connexes sur le calculateur fonctionnel. Mémoire de DEA, Université Paris Sud Centre d'Orsay, 1993.

[SEL72] SELKOW. One pass complexity analysis of digital picture properties. In *JACM*, volume 2, pages 283–295, Avril 1972.

10. A l'instar des processus de traitement