Application de l'outil d'analyse d'architectures multiprocesseurs Espion, au filtrage de Kalman 2-D rapide

Application of the Multiprocessor Architecture Analysis Tool Espion, to Fast 2-D Kalman Filter

Olivier SENTIEYS
LASTI-École Nationale Supérieure
de Sciences Appliquées
et de Technologie
ENSSAT, 6 rue de Kerampont
22305 Lannion Cedex
Tél. 96 46 50 30, Fax 96 37 01 99

Ingénieur diplômé de l'ENSSAT en 1990. Moniteur de l'enseignement supérieur, il prépare actuellement sa thèse sur l'expertise de la mise en œuvre parallèle d'algorithmes de traitement du signal et de l'image, sur réseaux multi-processeurs.

Eric MARTIN
LASTI-École Nationale Supérieure
de Sciences Appliquées
et de Technologie
ENSSAT, 6 rue de Kerampont
22305 Lannion Cedex

Agrégation de génie électrique à l'ENS de Cachan en 1984, maître de conférence à l'ENSSAT depuis 1988. Son domaine d'intérêt porte sur les outils de CAO en architectures dédiées aux applications de traitement du signal et de l'image. Il dirige au LASTI le groupe de recherche en architectures dédiées.

Hélène DUBOIS
LASTI-École Nationale Supérieure
de Sciences Appliquées
et de Technologie
ENSSAT, 6 rue de Kerampont
22305 Lannion Cedex

A reçu le diplôme d'ingénieur ENSERB en 1985 puis a soutenu une thèse de l'Université de Rennes 1 en janvier 1991 sur l'analyse des systèmes multi-processeurs. Elle est actuellement maître de conférence à l'ENSSAT. Ses travaux de recherche portent sur l'étude des performances des architectures multi-processeurs pour le domaine du traitement du signal.

Jean-Luc PHILIPPE
LASTI-École Nationale Supérieure
de Sciences Appliquées
et de Technologie
ENSSAT, 6 rue de Kerampont
22305 Lannion Cedex

Il a passé sa thèse en 1984, à l'université de Rennes 1. Maître de conférence à l'ENSSAT, ses domaines d'intérêt sur l'inter-action algorithme-architectures dans le domaine du traitement numérique du signal.

Application d'Espion au filtrage de Kalman 2-D rapide

Michel CORRAZA
LASTI-École Nationale Supérieure
de Sciences Appliquées
et de Technologie
ENSSAT, 6 rue de Kerampont
22305 Lannion Cedex

Professeur des Universités à l'ENSSAT. Responsable du Laboratoire d'Analyse des Systèmes de Traitement de l'Information. Domaine de recherche : Sûreté de fonctionnement, Architectures numériques, Traitement du signal.

RÉSUMÉ

Le problème de la mise en œuvre parallèle des algorithmes de traitement d'image, lié au choix d'une architecture cible, est crucial dans toutes les applications sous contrainte de temps réel. Cette mise en œuvre s'articule en une phase de représentation parallèle de l'algorithme, suivie d'une seconde phase d'implémentation sur une architecture parallèle. La restauration d'images en est un exemple convaincant de par sa complexité temporelle. Nous présentons dans cet article, une méthodologie de choix des réseaux multiprocesseurs MIMD, reposant sur une mesure objective des performances. Notre méthode est ici appliquée à un algorithme de filtrage de Kalman rapide pour la déconvolution.

MOTS CLÉS

Temps réel, multiprocesseurs, MIMD, méthodologie de conception, évaluation de performances, traitement d'image, filtrage de Kalman.

ABSTRACT

The parallel implementation of image processing algorithms, and the choice of target architectures is a very crucial problem in all applications under real time constraint. This design will be done in two stages : first the parallel presentation of the algorithm and then its implementation on a parallel architecture. Image restoration is a convincing example because of its temporal complexity. In this article we present a methodology depending on objective performance measures in order to design MIMD multiprocessors networks. Our method is then applied to a fast Kalman filter algorithm for deconvolution.

KEY WORDS

Real time, multiprocessors, MIMD, design methodology, performance estimation, image processing, Kalman filtering.

1. Introduction

La parallélisation d'algorithmes devient de plus en plus nécessaire dans les domaines où l'exécution doit se faire sous une contrainte de temps réel. C'est le cas, entre autre, des algorithmes de traitement du signal ou de l'image, où le temps de calcul élevé est du à deux phénomènes : d'une part à la grande quantité de données à traiter et d'autre part à la complexité des calculs. Nous nous sommes intéressés dans un pemier temps aux algorithmes de prétraitements des données. Ils sont caractérisés par des calculs relativement simples et répétitifs sur un grand volume de données. Ce choix nous a permis de porter nos efforts sur la prédiction de performances plutôt que sur la recherche de méthodes de parallélisation.

En effet, deux exploitations du parallélisme peuvent s'adapter à des problèmes comme le filtrage de Kalman : la parallélisation du corps de l'algorithme (parallélisation de tâches distinctes) et la parallélisation spatiale sur les données (parallélisation de tâches identiques). Nous étudierons la deuxième méthode qui s'intègre dans le cadre de notre recherche et de notre outil Espion [1]. Celui-ci s'appuie sur une méthodologie et permet une évaluation de performance de la mise en œuvre d'un algorithme parallélisé par le partage des données, sur un réseau multiprocesseurs.

Le problème de l'implémentation d'un algorithme se résume alors à trois problèmes : le choix du processeur de base, le choix du réseau d'interconnexion et celui de la découpe des données sous contraintes de temps d'exécution et de coût minimum.

La restauration d'images est un problème qui intervient dans de nombreuses applications. En effet quelle que soit l'origine des images numériques (images photographiques, satellites, scanner...), il est important d'en améliorer la qualité en vue d'un traitement ou d'une interprétation ultérieure. Le flou est un des phénomènes les plus fréquents avec le bruit de mesure car il est la conséquence du bougé de la caméra lors de sa prise de vue. Aussi l'intérêt d'atténuer le bruit ou le flou est évident quel que soit l'algorithme de traitement qui suivra (reconnaissance humaine ou automatique, extraction de contours...).

Ces algorithmes, qui peuvent être apparentés à une déconvolution, présentent une complexité calculatoire telle que leur mise en œuvre en temps réel ne peut pas être envisagée

sur les machines séquentielles classiques. Aussi est-il nécessaire de les paralléliser pour pouvoir soutenir la contrainte de temps (25 images par seconde pour la cadence vidéo ou quelques images par seconde pour le contrôle industriel) d'un éventuel cahier des charges.

Dans cet article nous présenterons tout d'abord le filtrage de Kalman 2-D sous une forme parallèle d'après une méthode de changement de domaine [2]. Nous décrirons par la suite nos modèles d'évaluation des architectures avant de les appliquer sur notre exemple pour plusieurs réseaux d'interconnexion.

2. Restauration d'image et filtrage de Kalman rapide

Les algorithmes de restauration seront différents selon le type de dégradation observée. Le modèle de cette dégradation peut être simplement un bruit additif (bruit de mesure ou de transmission) ou une convolution de l'image originale avec une fonction représentatice de la dégradation. Nous nous intéresserons ici au cas où cette fonction représente un bougé de la caméra, qui peut être horizontal ou vertical (voire les deux), ou une défocalisation de l'appareil photographique.

2.1. PRÉSENTATION DU PROBLÈME

On cherche ici à retrouver une image d'origine représentée par le signal $x(m, n)$ à partir d'une image dégradée $y(m, n)$ telle que :

$$y(m, n) = \sum_{(p, q)} c(p, q)\, x(m - p, n - q) + w(m, n)$$

où $c(p, q)$ est la réponse impulsionnelle, ou Point Spread Function (PSF), de la dégradation, et $w(m, n)$ un bruit additif de mesure blanc gaussien. Les bornes pour p et q dépendent de la dégradation : bougé horizontal et/ou vertical.

C'est un problème de déconvolution difficile car si l'on réécrit cette équation sous une forme matricielle ($Y = CX + W$), on fait apparaître une matrice C mal conditionnée. C'est-à-dire que certaines de ses lignes ou de ses colonnes peuvent être dépendantes apportant une solution non unique à ce problème. Il est donc nécessaire d'utiliser de méthodes autres que l'inversion pour le résoudre.

Les méthodes existantes sont de type optimisation au sens des moindres carrés (Filtrage de Wiener) ou de type maximum a posteriori (MAP) et ne donnent pas satisfaction soit au niveau du temps de calcul soit au niveau des résultats obtenus. J. W. Woods [3], [4] a étendu la méthode récursive du filtrage de Kalman au cas à deux dimensions pour la restauration d'images. Il s'est avéré que cette méthode donnait des résultats satisfaisants mais pour une complexité de calcul encore assez élevée ($O(I^4)$, $\mathbf{I} \times \mathbf{I}$ étant la taille de l'image). Plus récemment J. Biemond [2] a transposé le problème de déconvolution dans le domaine de Fourier en divisant la complexité du problème par I^2. D'autres travaux [5] améliorent cette méthode en périodisant l'image afin de faire disparaître certains inconvénients dus au changement de domaine.

Dans un premier temps, nous rappelerons brièvement l'algorithme de filtrage de Kalman rapide selon la méthode de Biemond, avant d'en expliquer l'implémentation sur différentes machines parallèles. Notre démarche se situe en aval de la présentation parallèle de l'algorithme, visant le respect des contraintes temps réel. Nous ne chercherons pas ici à optimiser cette forme de représentation.

2.2. MISE EN FORME DU MODÈLE POUR LE FILTRAGE DE KALMAN

L'algorithme est basé sur une modélisation de l'image par un modèle semi-causal autorégressif (Modèle AR) 2-D. On définit arbitrairement sur une image trois zones : le passé, le présent et le futur (voir *fig.* 1).

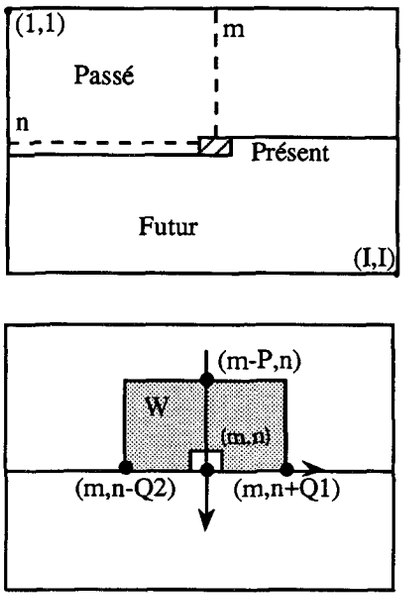

Figure 1. — Support semi-causal du modèle (NSHP).

Si $x(m, n)$ représente un pixel du présent de l'image d'origine, nous pouvons écrire l'équation du modèle AR 2-D :

$$(1) \quad x(m, n) = \sum_{(p, q) \in W} a(p, q)\, x(m - p, n - q) + u(m, n)$$

— W, définissant le support semi-causal du modèle NSHP (nonsymetric half-plane) appartenant aux régions passées et futures (voir *fig.* 1), est donné par :

$$W = \{(p, q) \mid 0 \leq p \leq P, -Q_1 \leq q \leq \leq Q_2 \text{ et } (p, q) \neq (0, 0)\}$$

— $u(m, n)$ est le bruit d'entrée du modèle, indépendant de $x(m, n)$, blanc par rapport à la variable m (dans le sens vertical) et coloré selon n, c'est-à-dire dépendant dans le

sens horizontal de ses voisins appartenant au support du modèle [2]. La fonction d'autocovariance de u est :

$$r_u(k, \ell) = \mathrm{E}[u(m, n)\, u(m+k, n+\ell)]$$

$$r_u(k, \ell) = \begin{cases} \beta^2 & \text{si} \quad (k, \ell) = (0, 0) \\ -\beta^2 a(0, \ell) & \text{si} \quad (0, \ell) \in \mathrm{W} \\ 0 & \text{sinon .} \end{cases}$$

L'observation $y(m, n)$, floue et bruitée par rapport à l'image d'origine $x(m, n)$, est définie par :

$$(2) \quad y(m, n) = \sum_{k=-K_1}^{K_2} \sum_{\ell=-L_1}^{L_2} c(k, \ell)\, x(m-k, n-\ell) + w(m, n)$$

où $c(k, \ell)$ représente la fonction du flou (PSF) et où $w(m, n)$ est le bruit de mesure supposé blanc, gaussien, de variance σ^2, et non corrélé avec les données.

L'application directe de la méthode du filtrage de Kalman 2-D aux équations (1) et (2) conduit à une complexité de calcul en $O(I^4)$, aussi est-il nécessaire de remanier ces équations afin de simplifier l'algorithme.

Tous les points d'une même ligne d'indice m ne peuvent être combinés dans un même vecteur $X = [x(m, 1), \ldots, x(m, I)]^T$. En réécrivant les équations (1) et (2), on obtient le système suivant :

$$(3)\text{-}(4) \quad \begin{cases} \sum_{p=0}^{P} A_p\, X(m-p) = B_0\, U(m) \\ \\ Y(m) = \sum_{k=K_1}^{K_2} C_k\, X(m-k) + W(m) \end{cases}$$

où :

- $Y(m) = [y(m, 1), \ldots, y(m, I)]^T$ est le vecteur d'observation,
- $U(m)$ $[u(m, 1), \ldots, u(m, I)]^T$ est le vecteur bruit du modèle ayant pour matrice d'autocovariance : $\Sigma_U(k) = \mathrm{E}[U(m+k)^T] = \beta^2 A_0\, \delta(k)$,
- $W(m) = [w(m, 1), \ldots, w(m, I)]^T$ est le vecteur bruit de mesure d'autocovariance :

$$\Sigma_W(k) = \sigma^2\, \mathbf{I}_{I \times I}\, \delta(k) .$$

- A_p $[p = 0, 1 \ldots P]$, avec $a(0, 0) = -1$, C_k $[k = -K_1 \ldots K_2]$, et $B_0 = \mathbf{I}_{I \times I}$, sont des matrices bandes de Toeplitz $I \times I$ définies sur la figure 2.

2.3. APPROXIMATION PAR DES MATRICES CIRCULANTES

Une matrice bande de Toeplitz T de dimension $(I \times I)$ peut être approchée par une matrice circulante T^c de même dimension dans laquelle chaque ligne est produite par un décalage circulaire vers la droite de celle du dessus et où la première ligne est un décalage circulaire vers la droite de la dernière. De telles matrices ont la propriété d'être facilement diagonalisables par : $\Lambda = \Phi^{-1}\, T^c\, \Phi$, où Λ est une

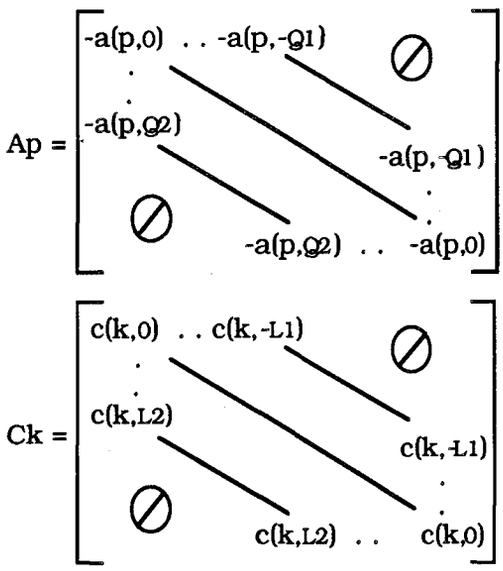

Figure 2. — Matrices du modèle de la PSF et du flou.

matrice diagonale $I \times I$ dont les éléments $\lambda(n, n)$ sont les valeurs propres de T^c et où Φ est la matrice unitaire des vecteurs propres de T^c.

Il en résulte donc que la matrice T^c peut être diagonalisée par simple transformation de Fourier (TFD ou TFR) sur la séquence cyclique $t(0) \ldots t(I-1)$ avec $t(I-m) = t(-m)$.

Cette diagonalisation peut être appliquée sur les matrices A_p et C_k des deux modèles. On obtient donc un nouveau système d'équations équivalentes aux équations (3) et (4) où les matrices $A'p$ et $C'k$ sont diagonales :

$$(5)\text{-}(6) \quad \begin{cases} \sum_{p=0}^{P} A'_p\, X'(p) = U'(m) \\ \\ Y'(m) = \sum_{k=-Q_1}^{Q_2} C'_k\, X'(m-k) + W'(m) . \end{cases}$$

On note par les variables X', Y', U' et W' les transformées de Fourier de la séquence formée par les coordonnées des vecteurs X, Y, U et W.

Les matrices étant diagonales on peut réécrire les équations afin d'obtenir I systèmes indépendants :

$$(7)\text{-}(8) \quad \begin{cases} x'(m, n) = -\sum_{p=1}^{P} \frac{a'_p(n)}{a'_0(n)}\, x'(m-p, n) \\ \qquad\qquad + \frac{1}{a'_0(n)}\, u'(m, n) \\ \\ y'(m, n) = \sum_{k=-K_1}^{K_2} c'_k(n)\, x'(m-k, n) + n'(m, n) \end{cases}$$

pour $m, n = 0, 1 \ldots I-1$,

avec des bruits n' et u' dans le plan de Fourier ayant pour fonction d'autocovariance :

$$\Sigma_{n'}(k) = \sigma^2 \, \mathbf{I}_{\mathrm{I}\times\mathrm{I}} \, \delta(k)$$
$$\Sigma_{u'}(k) = \beta^2 \, \mathrm{A}'_0 \, \delta(k)$$

$a'_p(n)$ et $c'_k(n)$, $(n = 0 \ldots \mathrm{I} - 1)$ sont les éléments de la diagonale des matrices A'_p et C'_k.

2.4. MODÈLE DYNAMIQUE

Pour calculer, à partir des équations (7) et (8), N filtres de Kalman en parallèle, il est nécessaire de les écrire sous une forme matricielle de type modèle d'état dynamique. On définit ainsi un vecteur d'état $Z'(m, n)$ de dimension $K = K_1 + K_2 + 1$ et tel que $K \geqslant P$:

$$Z'(m, n) = [x'(m + K_1, n), \ldots, x'(m - K_2, n)]^T .$$

Les équations du filtre peuvent donc être réécrites sous la forme :

$$Z'(m, n) = A'(n) \, Z'(m - 1, n) + B'(n) \, u'(m, n)$$

(9 *a*)-(9 *b*)

$$y'(m, n) = C'(n) \, Z'(m, n) + n'(m, n)$$

pour $m, n = 0, 1 \ldots \mathrm{I} - 1$.

La matrice $A'(n)$ et les vecteurs $B'(n)$ et $C'(n)$ sont respectivement de dimension $K \times K$, $K \times 1$ et $1 \times K$ et représentent le modèle AR 2-D et la PSF. Ils sont définis sur la figure 3. La matrice $A'(n)$ est une matrice de taille réduite dans la pratique ($K = 3, 5, 7$). En outre cette matrice, ainsi que le vecteur $B'(n)$ comportent peu de termes non nuls ou différents de 1 (P termes pour $A'(n)$ et un terme pour $B'(n)$), ce qui permettra de simplifier le calcul des équations du filtre de Kalman.

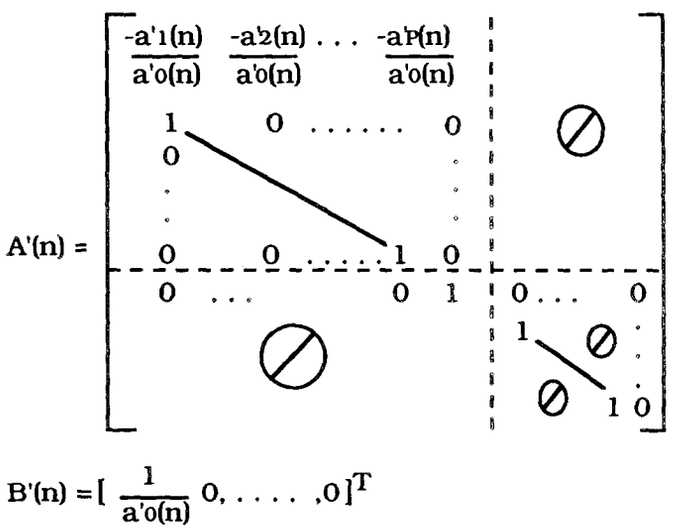

Figure 3. — Matrices du modèle et de la PSF dans le plan de Fourier.

2.5. LE FILTRE DE KALMAN RAPIDE

Ayant défini le modèle d'état de dimension K par le système (9) le n-ième filtre de Kalman sera défini par la meilleure estimation $\hat{Z}'(m, n)$ de $Z'(m, n)$ pour l'observation $y'(m, n)$. Le résultat étant bien connu, il aboutit à l'algorithme suivant :

$$\mathrm{K}a(m, n) = \mathrm{Q}(m - 1, n) \, \mathrm{C}'(n)^{\mathrm{T}} \times \times [\mathrm{C}'(n) \, \mathrm{Q}(m - 1, n) \, \mathrm{C}'(n) + \sigma_{n'}(n)]^{-1}$$

$$\mathrm{P}(m, n) = [\mathbf{I}_{\mathrm{I}\times\mathrm{I}} - \mathrm{K}a(m, n) \, \mathrm{C}'(n)] \, \mathrm{Q}(m - 1, n)$$

$$\mathrm{Q}(m, n) = \mathrm{A}'(n) \, \mathrm{P}(m, n) \, \mathrm{A}'(n)^{\mathrm{T}} + + \sigma_{u'}(n) \, \mathrm{B}'(n) \, \mathrm{B}'(n)^{\mathrm{T}}$$

$$\hat{\mathrm{Z}}'(m, n) = \mathrm{A}'(n) \, \hat{\mathrm{Z}}'(m - 1, n) + + \mathrm{K}a(m, n) \, [y'(m, n) - \mathrm{C}'(n) \, \mathrm{A}'(n) \, \hat{\mathrm{Z}}'(m - 1, n)]$$

où $P(m, n)$ est la matrice de covariance de l'erreur d'estimation $Z'(m, n) - \hat{Z}'(m, n)$.

Au fur et à mesure que le filtre est appliqué aux points de l'image dégradée, le gain de Kalman du filtre, $Ka(m, n)$, converge assez rapidement (une dizaine d'itération) vers une valeur $Ks(n)$ constante. On remarque que l'estimée du pixel $\hat{x}'(m, n)$ est contenue dans plusieurs vecteurs $\hat{Z}'(m, n)$.

$$\hat{Z}'(m - K_1, n) = [\hat{x}'(m, n), \ldots, \hat{x}'(m - K_1 - K_2, n)]^T$$
$$\hat{Z}'(m, n) = [\hat{x}'(m + K_1, n), \ldots, \hat{x}'(m, n), \ldots, \hat{x}'(m - K_2, n)]^T$$
$$\hat{Z}'(m + K_2, n) = [\hat{x}'(m + K_1 + K_2, n), \ldots, \hat{x}'(m, n)]^T .$$

On choisira donc le dernier cas pour mémoriser la valeur de $\hat{x}'(m, n)$ car c'est celui où l'erreur d'estimation est la plus faible.

$$\hat{x}'(m, n) = [0 \; 0 \ldots 1] \, \hat{Z}'(m + K_2, n) .$$

On obtient à la fin de l'algorithme I vecteurs estimés $\hat{X}'(m)$ des lignes de l'image dans le plan de Fourier sur lesquels on effectue une TFR inverse afin d'obtenir l'image résultante formée des lignes $\hat{X}(m)$ pour $m = 0 \ldots \mathrm{I} - 1$. Le schéma global du filtrage de Kalman est celui de la figure 4.

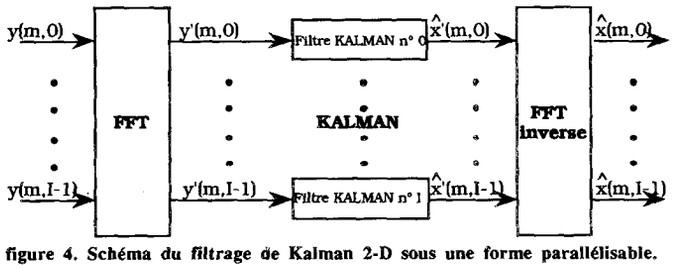

figure 4. Schéma du filtrage de Kalman 2-D sous une forme parallélisable.

2.6. COMPLEXITÉ DE L'ALGORITHME

Dans le cas d'un filtrage de Kalman implémenté directement sans transformation dans le domaine de Fourier, la

complexité de l'algorithme serait en $O(K^3 I^4)$. Mais en utilisant la diagonalisation des matrices circulantes les équations sont découpées en I équations parallèles demandant chacune des multiplications de matrices $K \times K$, L étant la taille de la PSF on peut considérer qu'elle est très petite devant I la taille de l'image. La complexité du nouvel algorithme ainsi défini serait donc en $O(K^3 I + I . \log_2 I)$ pour chaque ligne de l'image, où $I . \log_2 I$ provient du calcul de la TFR et de la TFR inverse alors que $K^3 I$ résulte du calcul d'un filtre de Kalman. La complexité totale sur l'image complète est donc en : $O[I^2(K^3 + \log_2 I)]$. On peut, par une première approximation, comparer K^3 à $\log_2 I$ dans le cas d'une image 256×256 dont le flou est obtenu à partir d'une moyenne locale 3×3 ($K = 3$, $I = 256$). On voit que le temps de calcul d'un filtre de Kalman est supérieur au temps de calcul d'une TFR sur la ligne (dans notre cas on obtient un facteur ≈ 4).

Une réduction du temps de calcul d'un filtre de Kalman monodimensionnel peut être mise en œuvre en considérant le fait que les matrices $P(m, n)$, $Q(m, n)$ et le vecteur $Ka(m, n)$ ne dépendent pas des observations. Ils peuvent donc être calculés à l'avance. On peut également remarquer que le gain du filtre $Ka(m, n)$ atteint une valeur constante après seulement quelques itérations. Ce gain $Ks(n)$ constant dû à la convergence de l'algorithme, ne dépend pas de l'image traitée, mais seulement du modèle auto-régressif 2-D et de la PSF. Il serait donc possible dans le cas d'une application temps réel de traitement d'une suite d'images de réaliser le calcul de $Ks(n)$ (gain statique de la ligne n) avant le traitement. On obtiendrait ainsi un algorithme de Kalman réduit sur une ligne :

$$\hat{Z}'(m, n) = A'(n)\,\hat{Z}'(m-1, n) + \\ + Ks(n)\,[y'(m, n) - C'(n)\,A'(n)\,\hat{Z}'(m-1, n)]\,.$$

En utilisant cette méthode le filtre de Kalman reste un algorithme dont le temps d'exécution ne dépend pas des données traitées (algorithme régulier), et sa complexité décroît en $O[I^2(K^2 + \log_2 I)]$.

En exploitant la particularité des matrices $A'(n)$, $B'(n)$ et $C'(n)$ (§ 2.4) on remarque que la complexité du filtrage de Kalman peut être encore réduite jusqu'à atteindre $O[I^2(K + \log_2 I)]$.

2.7. RÉSULTATS EXPÉRIMENTAUX

Nous avons commencé par tester l'algorithme sur un seul transputer. Les images étaient artificiellement rendues floues par moyennage sur des fenêtres de taille 3×3, 5×5, 7×7, pour obtenir des images plus ou moins floues selon les valeurs de $K = 3, 5, 7$.

Le modèle AR 2-D de l'image était défini comme suit [6] :

$$a(p, q) = \begin{cases} 1 & \text{si } p = q \\ \rho & \text{si } |p - q| = 1 \\ \rho^2 & \text{si } |p - q| = 2 \\ 0 & \text{ailleurs} \end{cases}$$

avec $\rho = 0{,}96$, et $\beta^2 = 0{,}64$ variance du bruit d'entrée du modèle.

Par ailleurs la fonction de flou $c(k, \ell)$ et le bruit de mesure $w(m, n)$ sont obtenus par simulation, ou par identification [6], [7].

On obtient donc les différentes matrices de Toeplitz Ap et Ck puis les matrices circulantes $A'p$ et $C'k$ définies au 2.2 et 2.3 sur lesquelles nous allons appliquer la méthode de filtrage de Kalman rapide du chapitre 2.5.

L'exécution de cet algorithme sur un Transputer T800-20 MHz [8] exige un temps de calcul bien trop élevé pour être utilisable dans le cas d'une application temps réel. Le tableau 1 les représente pour plusieurs valeurs de I dans le cas où $K = 3$.

Tableau 1. — Temps de calcul séquentiel pour différentes valeurs de I.

I	TFR lignes	Kalman colonnes complet	Kalman colonnes optimisé	TFR -1 lignes	Filtrage de Kalman rapide complet
64	0,19 s	2,66 s	0,5 s	0,19 s	0,88 s
128	0,91 s	7,74 s	2,01 s	0,91 s	3,83 s
256	4,14 s	25,16 s	8,06 s	4,14 s	16,34 s

Il apparaît donc immédiatement et même dans le cas du filtre de Kalman réduit, que la parallélisation est une condition nécessaire à ce type de traitement d'image en temps réel.

3. Parallélisation du filtre de Kalman

3.1. MODÈLES ESPION DE LA PARALLÉLISATION

ESPION [1] est un outil basé sur une méthodologie, permettant de rechercher l'adéquation optimale au sens de la contrainte temps/coût, entre un processeur et un réseau d'interconnexion formant une architecture parallèle, et un algorithme. Il le réalise à partir de l'émulation du code séquentiel de l'algorithme et de modèles simples de ces trois éléments. Il permet l'analyse prévisionnelle de performances d'un panorama complet d'architectures proposé par l'utilisateur, en se limitant à l'écriture séquentiel du code de l'algorithme.

La mise en œuvre parallèle d'un algorithme passe par les deux étapes suivantes : parallélisation et conception de l'architecture. Nous avons retenu pour la première, le cadre de la parallélisation par le partage des données. Cette orientation est bien adaptée aux algorithmes de traitement des images où le flot de données est important, avec cependant quelques difficultés concernant les algorithmes irréguliers (algorithmes pour lesquels le temps de calcul dépend de la donnée à traiter).

La conception d'une architecture multiprocesseurs repose sur le choix du nombre de processeurs à mettre en œuvre, et celui du réseau d'interconnexion leur permettant d'échanger les données à traiter. Ces échanges dépendent étroitement de la nature de l'algorithme (échanges locaux, de voisinage ou globaux) et du partage des données effectué. Nous montrons qu'il est possible d'évaluer le comportement de l'algorithme parallélisé à partir d'une émulation du code séquentiel, et de modèles simples du

processeur, de l'algorithme et du réseau d'interconnexion considérés. Cette approche offre l'avantage de pouvoir explorer une grande diversité de solutions, sans générer à chaque fois le code parallèle [9]. Nous ciblons pour cela une architecture de type MIMD, c'est-à-dire N processeurs possédant leur propre mémoire de programme et de données, interconnectés selon un réseau dont la topologie est figée [10]. Les communications entre les différentes unités élémentaires (processeur + mémoire + gestionnaire d'entrée/sortie) se feront par messages.

Nous nous placerons pour notre étude sous deux hypothèses :

- équi répartition spatiale des échanges : tout processeur P*i* ($i = 0 \ldots N-1$) communique à chaque P*j* ($j = 0 \ldots N-1, j \neq i$) le même nombre de données ;
- équirépartition temporelle des échanges : l'échéance dans le temps des transferts ne bloque en aucune manière le déroulement des calculs en cours.

Ces deux hypothèses peuvent facilement être levées en modifiant le modèle, mais elles en simplifient la compréhension. Nous montrerons que le filtrage de Kalman les respecte.

L'outil ESPION, qui automatise les étapes de calcul d'une méthodologie pour l'analyse de performances, peut être utilisé « à la main » en dehors de ce cadre informatique. Nous présentons cette méthodologie.

3.1.1. Modélisation de l'algorithme.

Un algorithme est caractérisé par $\mathbf{N_{Tâ}}$ tâches élémentaires séquentielles de temps **Tâ** dont **P** peuvent être effectuées en parallèle sur un réseau de **N** processeurs.

Le partage des données entre les processeurs peut impliquer la nécessité de communication entre ces processeurs : notons $\mathbf{N_{tc}}$ le nombre total de communications que requiert l'algorithme. On calculera facilement N_{tc} dans le cas de partage de données sur un algorithme régulier, par une simple analyse des indices de la donnée traitée.

On définit alors les paramètres suivants :

- $\mathbf{T_{seq}} = N_{Tâ} \cdot Tâ$, temps d'exécution lors de l'implémentation sur un seul processeur.

T_{seq} dépend à la fois du processeur et de l'algorithme à mettre en œuvre. La valeur de **P** dépend de la nature de l'algorithme (tâches entièrement parallélisables, possédant une tâche purement séquentielle, séquentialité entre les tâches).

- $\mathbf{T_i}$: temps de calcul idéal sur N processeurs, $T_i = T_{seq}/N$.
- $\mathbf{T_{cal}}$: Temps de calcul, $T_{cal} = T_s + T_p$, temps d'exécution lors de l'implémentation sur un réseau de N processeurs avec :

$\mathbf{T_s}$: temps de calcul de la partie séquentielle, $T_s = (1 - P) N_{Tâ} \cdot Tâ$,

$\mathbf{T_p}$: temps de calcul en parallèle sur N processeurs, $T_p = P \dfrac{N_{Tâ} \cdot Tâ}{N}$.

$$T_{cal} = \left[(1 - P) + \frac{P}{N} \right] \cdot T_{seq}$$

- $\mathbf{D_p}$: volume de données échangées pour tout processeur « i » communiquant avec tout processeur « j ». Dans l'hypothèse d'équirépartition spatiale D_p vaut :

$$D_p = \frac{\text{Nombre total de communications}}{\text{Nombre de processeurs}} = \frac{N_{tc}}{N}.$$

Ce paramètre dépend bien sûr de l'algorithme mais va également être fonction du partage des données entre les différents processeurs. Par exemple une découpe en lignes (chaque processeur possède I/N lignes de l'image) ne donnera pas toujours la même valeur de D_p que si le partage des données se fait en colonnes ou en carrés.

- **C** : nombre de communications par tâche élémentaire.

$$C = \frac{\text{Nombre total de communications}}{\text{Nb tâches}} \times \times \frac{\text{temps d'une communication}}{\text{temps d'une tâche}} = N_{tc} \frac{T_{com}}{T_{seq}}.$$

C, taux de communication, représente l'importance relative des communications à effectuer vis-à-vis des calculs, ces deux grandeurs étant ramenées à une même unité de temps. Nous verrons que C est un paramètre important pour juger de la possibilité de saturation des réseaux de communication entre processeurs.

3.1.2. Modélisation du processeur

Outre la rapidité d'exécution d'un algorithme $\mathbf{T_{seq}}$, représentant la puissance de traitement d'un processeur, il est nécessaire de faire intervenir la notion de rapidité à communiquer. Celle-ci sera considérée à travers deux paramètres, la vitesse de communication et le retard apporté aux calculs :

$\mathbf{T_{com}}$: temps pour communiquer une donnée sur le lien entre deux processeurs voisins, ce facteur dépend de la taille de la donnée (octet, entier, réel, nombre complexe ou tableau de données des types précédemment cités), et de la vitesse sur le lien.

$\mathbf{T_{nt}}$: taux de non-transparence. C'est le temps CPU nécessaire pour les échanges entre processeurs : initialisation du transfert, perturbations du DMA, basculements de contexte, constructeurs ajoutés (routage) pour organiser les échanges, etc... $T_{nt} \cdot T_{com}$ est un temps, assimilé à la pollution apportée par les communications.

$$T_{nt} = \frac{\text{temps CPU pris par la communication d'une donnée}}{\text{temps de transfert sur le lien de cette donnée}}$$

Nous modélisons T_{nt} par : $T_{nt} \cdot T_{com} = A \cdot N_{oct} + B$, où $\mathbf{N_{oct}}$ est la taille en données élémentaires (par exemple en octet, réel, nombre complexe, ...) du message circulant sur le lien. Ce paramètre est obtenu par une mesure expérimentale sur le processeur cible du comportement des processus gérant les échanges de données. B représente le temps d'initialisation du message, tandis que A représente le temps d'accès à chaque donnée (DMA, routage).

3.1.3. Modélisation du réseau de communication

Le réseau d'interconnexion reliant les différents processeurs élémentaires est caractérisé par les deux paramètres classiques suivants :

- $\mathbf{nb_{bus}}$: nombre de liaisons utilisables simultanément pour dialoguer entre 2 processeurs,
- $\mathbf{d_{moy}}$: distance moyenne d'un processeur à un autre. La distance unitaire est celle entre deux processeurs voisins.

$$d_{moy} = \frac{1}{N(N-1)} \sum_{\substack{\forall i, j \in 1, N \\ j \neq i}} \text{distance } (P_i, P_j) .$$

Ces deux paramètres apparaissent souvent dans la littérature sous une forme relativement équivalente [11] [12].

3.1.4. Mesures prévisionnelles

Nous avons défini trois mesures devant guider le choix d'un réseau : l'accélération, l'efficacité et la saturation. Si les deux premières sont bien connues (cas des architectures vectorielles), la troisième par contre apportera une connaissance importante : tout réseau hors saturation présente une capacité d'écoulement des communications aussi performante que le plus complexe d'entre-eux. Cette mesure importante doit conduire au choix objectif du réseau répondant au problème d'écoulement des données, sous contrainte de coût minimum.

3.1.4.1. Phénomène de saturation

La borne de saturation est définie comme la valeur de N pour laquelle le temps des échanges sur les liens devient supérieur au temps CPU du calcul.

On dit que le réseau est en saturation si le temps total d'exécution de l'algorithme dépend du nombre de communications.

On définit le temps total nécessaire aux échanges sur le réseau :

$$T_{ech} = \frac{\text{nombre total de comm}}{\text{nombre de possibilité de comm}} \times \text{temps moyen d'une comm}$$

$$T_{ech} = \frac{N D_p}{nb_{bus}} \times d_{moy} T_{com} = CPT_{seq} \frac{d_{moy}}{nb_{bus}} .$$

Hors saturation du réseau de communication, le temps d'exécution de notre algorithme sur le multiprocesseur est donné par :

$$T_{cal} = P \frac{T_{seq}}{N} + (1 - P) T_{seq} .$$

On peut ainsi définir le phénomène de saturation et dire que le réseau est saturé si :

$$T_{ech} > T_{cal}$$

$$\Leftrightarrow (1 - P) + \frac{P}{N} < \frac{CP\, d_{moy}}{nb_{bus}} .$$

NB : lorsque le rapport d_{moy}/nb_{bus} est constant, la saturation se fait par valeur supérieure, c'est-à-dire que le réseau sature pour $N > N_{sat}$, alors que si ce même rapport est en 1/N, la saturation aura lieu par valeur inférieure ($N < N_{sat}$).

Le tableau 2 montre pour quelques réseaux classiquement utilisés, les critères utilisés par ESPION ainsi que la borne de saturaton qui en découle.

Tableau 2. — Paramètres ESPION appliqués à des réseaux classiques [12].

Nom du réseau	dmoy	nbbus	borne de saturation
FULL-CONNECTED	1	N(N-1)	si p=1 jamais saturé $N < \sqrt{\frac{p(C-1)}{1-p}}$ si C < 1 jamais saturé
CROSSBAR	1	N	$C > 1 + \frac{1-p}{p} N$ si C < 1 jamais saturé
MULTIBUS //	1	B	$N > \frac{B\,p}{C\,p - B(1-p)}$ si $C < B\frac{1-p}{p}$ toujours saturé si $C > \frac{B}{p}$ toujours saturé
ANNEAU UNIDIRECTIONNEL UNIFILAIRE	$\frac{N}{2}$	N	$N > \frac{2\,p}{C\,p - 2(1-p)}$ si $C < 2\frac{1-p}{p}$ toujours saturé
ANNEAU BIDIRECTIONNEL BIFILAIRE	N pair: $\frac{N^2}{4(N-1)}$ N imp: $\frac{N+1}{4}$	2 N	$N > \frac{8\,p}{C\,p - 8(1-p)}$ si $C < 8\frac{1-p}{p}$ toujours saturé
CARRE MAILLE UNIDIRECTIONNEL	$\frac{N(\sqrt{N}-1)}{N-1}$	2 N	$\sqrt{N} > \frac{2}{C}$ si p = 1
CARRE MAILLE BIDIRECTIONNEL	$\frac{\sqrt{N}}{2}$	4 N	$\sqrt{N} > \frac{8}{C}$ si p = 1
HYPERCUBE $N = 2^n$	$\frac{N \log_2(N)}{2(N-1)}$	N log2(N)	$C > 2(N-1) \times [\frac{1-p}{p} + \frac{1}{N}]$ si p=1 {C < 2 jamais saturé; C > 2 toujours saturé} si p≠1 {C < 2 jamais saturé; $C > 2 \Rightarrow N < \frac{p(C-2)}{2(1-p)}$}

3.1.4.2. Efficacité et accélération

Les notions d'efficacité et d'accélération sont définies par les formules suivantes :

$$\text{efficacité} = \frac{\text{temps idéal sur N processeurs}}{\text{temps réel sur le réseau de N processeurs}}$$

$$\text{efficacité} = \frac{T_{seq}/N}{T_{réel}}$$

$$\text{accélération} = \frac{\text{temps d'exécution sur 1 processeur}}{\text{temps réel sur le réseau de N processeurs}}$$

$$\text{accélération} = \frac{T_{seq}}{T_{réel}} .$$

D'après la notion introduite précédemment, le temps réel d'exécution de l'algorithme après implémentation sur un réseau de N processeurs (Tréel) sera différent si la borne de saturation est, ou non, atteinte.

Hors saturation, le temps réel d'exécution de l'algorithme sur le réseau vaut :

$$T_{cal} = \left[(1 - P) + \frac{P}{N} \right] T_{seq}$$

l'efficacité de la mise en œuvre parallèle est identique pour tout réseau :

$$Eff_{hors\ sat} = \frac{1}{(1 - P) . N + P} .$$

En supposant les communications équiréparties, la saturation du réseau est obtenue lorsque le temps des échanges est prépondérant devant le temps nécessaire aux calculs. Le temps d'exécution de l'algorithme devient alors :

$$T_{réel} = T_{ech} = CPT_{seq} \frac{d_{moy}}{nb_{bus}}.$$

Après saturation, l'efficacité vaut :

$$Eff_{sat} = \frac{nb_{bus}}{NCP\ d_{moy}}.$$

3.1.5. Notion d'efficacité réelle

Les valeurs d'efficacités définies précédemment sont considérées dans le cas idéal où les communications ne provoquent aucun recouvrement sur les calculs. Notre modèle du processeur fait apparaître le taux de non-transparence des échanges T_{nt}. Si on en tient compte, il faut ajouter au temps de calcul une pollution représentant le temps CPU nécessaire à la gestion des échanges et de leur routage sur le réseau. Ce temps est donné par le terme $T_{nt} \cdot T_{com} \frac{D_p}{N_{oct}} d_{moy}$, où N_{oct} représente la taille des blocs de données élémentaires circulant sur les interconnexions du réseau, et où d_{moy} représente la notion de routage moyen d'un processeur i vers un processeur j. On rappelle que, par l'hypothèse d'équirépartition spatiale, un processeur i communique avec tous les autres processeurs j, $\forall j \in [0 \ldots N-1], i \neq j$.

Hors saturation le temps d'exécution réel de l'algorithme et son efficacité deviennent alors :

$$T_{exe} = \left[(1-P) + \frac{P}{N} \right] T_{seq} + \left\{ T_{nt} \cdot T_{com} \frac{D_p}{N_{oct}} d_{moy} \right\}$$

$$\text{efficacité} = \frac{1}{(1-P)N + P + \frac{T_{nt} \cdot T_{com}}{T_{seq}} \frac{D_p}{N_{oct}} d_{moy}}.$$

3.1.6. Méthodes de mesure et de calcul des paramètres du modèle

La première méthode à laquelle on peut penser pour déterminer les paramètres de l'algorithme parallélisé, est une méthode analytique globale : l'analyse de la syntaxe du code écrit dans un langage quelconque permettrait de dégager les opérations de calcul importantes (n multiplications, p additions...), et connaissant la durée élémentaire de ces opérations sur le processeur cible, on pourrait ainsi évaluer les deux paramètres T_{seq} et P.

Cette méthode, que nous avons expérimentée, donne des résultats très médiocres dans le cas d'un processeur complexe comme le transputer programmé en OCCAM [13] ; beaucoup trop de phénomènes non déterministes seraient en effet à prendre en compte pour qu'une telle évaluation soit de qualité. Cette méthode peut néanmoins être utilisée pour des processeurs type Processeur de Traitement de Signaux (PTS) programmés en assembleur.

Donc pour des raisons de simplicité et de qualité de la détermination des paramètres, nous avons choisi de mesurer ceux-ci par « espionnage » de l'exécution du code séquentiel de l'algorithme. Quelques instructions de mesure de temps sont ajoutées au code et permettent de mesurer précisément le temps T_{seq}, ainsi que le rapport P.

Le paramètre D_p peut être lui aussi déterminé de deux façons : mesure avec ESPION ou calcul direct. Pour les algorithmes réguliers, il découle directement de la forme de la découpe. Par exemple, pour un filtrage par convolution de l'image avec une fenêtre de taille 3×3 passant plusieurs fois sur l'image (itératif), une découpe en blocs de lignes nécessite un échange de I données, I étant la taille de l'image. Pour les algorithmes irréguliers, nous avons gardé l'idée de mesure sur l'exécution de l'algorithme par ajout d'instructions dans le code séquentiel. Notre outil ESPION permet dans ce cas de mesurer le paramètre D_p pour différentes découpes.

Le paramètre T_{com} ne pose pas de problèmes : il est directement lié à la vitesse de transmission sur le chemin de communication et à la taille de la donnée échangée ; par exemple pour des données de type pixel 8 bits et un lien transputer utilisé en bidirectionnel à 20 Mbits/s, le T_{com} vaut 0,85 µs (en tenant compte des bits ajoutés pour la trame et l'acquittement). Ces paramètres sont donnés par le constructeur mais peuvent être vérifiés par des mesures.

Pour déterminer le paramètre T_{nt}, on se heurte au même problème de complexité à modéliser un grand nombre de phénomènes, notamment dans le cas du transputer : initialisation, perturbation due à l'accès à la mémoire, basculement de contexte. La modélisation se fait donc également par mesure de temps sur l'exécution des processus participant à la gestion des échanges. Les mesures effectuées pour différentes tailles d'échanges permettent de tracer une courbe $T_{nt} = f(N_{oct})$. Dans le cas du transputer, cette courbe est une droite de la forme $A \cdot N_{oct} + B$.

Il est à noter que cette mesure n'est à faire qu'une fois pour une structure logicielle de gestion des échanges ; elle ne dépend pas de l'algorithme [1].

A partir des paramètres T_{nt}, T_{com} mesurés une fois pour toute, des paramètres nb_{bus} et d_{moy} donnés au tableau 2 pour quelques réseaux, et des paramètres estimés ou calculés à la main, ou mesurés par notre outil, on est capable, immédiatement et sans difficulté, d'évaluer la performance de la parallélisation. Nous appliquerons cette méthodologie à l'exemple du filtrage de Kalman bidimensionnel.

Nous avons mesuré les paramètres processeurs sur d'autres processeurs de type différents (TMS320C25, les versions C30 et C40 étant en cours d'évaluation). Les processeurs de traitement des signaux de première génération comme le C25 s'avèrent rapides en calcul mais incapables de communiquer des données en parallèle au calcul ($T_{nt} > 100\ \%$). De plus les langages de hauts niveaux ne sont pour l'instant pas très performants. Mais avec les nouvelles générations de PTS, et surtout le C40 de chez Texas Instruments, ayant les capacités de calculs d'un PTS et dépassant les capacités de communication d'un transputer, on peut s'attendre à des efficacités de parallélisation équivalentes au transputer mais pour des temps de calculs bien inférieurs.

3.2. ESPION ET LE FILTRAGE DE KALMAN 2-D

Du fait du passage des lignes de l'image dans le domaine de Fourier, les colonnes s'en trouvent décorrélées entre elles, les matrices représentatives du modèle de l'image et de la PSF, A*p* et C*k*, étant diagonalisées par Transformation de Fourier rapide. On voit donc apparaître I filtres de Kalman monodimensionnels indépendants à appliquer sur les I colonnes de l'image transformée (*fig.* 4).

Une manière d'effectuer la parallélisation est basée sur le fait que l'on peut calculer en parallèle les I filtres de Kalman et que l'on peut donc confier I/N colonnes de l'image à chaque élément d'un réseau de N processeurs (découpe en lignes). Cette méthode est conforme aux modèles présentés au 3.1.

L'algorithme complet est décrit ci-dessous, chaque Processeur Élémentaire (PE) se voit affecté les tâches suivantes :

- $\frac{I}{N}$ TFR monodimensionnelles sur $\frac{I}{N}$ lignes de l'image,
- $\frac{I}{N}$ Filtres de Kalman monodimensionnels sur $\frac{I}{N}$ colonnes de l'image transformée,
- $\frac{I}{N}$ TFR inverses monodimensionnelles sur les $\frac{I}{N}$ lignes de l'image résultante.

Nous voyons donc apparaître la nécessité de communiquer entre les PE, de façon régulière, selon le schéma de la figure 5 pour un exemple où N est égal à 4 :

Les échanges entre processeurs peuvent être traités parallèlement aux calculs selon la méthode de programmation classique :

Faire en séquentiel sur le Processeur ***Pi*** ($i = 0 \ldots N - 1$)
  Traitement de la ligne $\frac{Pi \,.\, I}{N}$ (ou colonne)
  Pour *ligne* (ou *colonne*) **de** *1* **à** $\frac{(Pi + 1) \,.\, I}{N} - 1$
    Faire en parallèle sur ***Pi***
    Envoi des blocs *bij* de la ligne *ligne-1*
    Traitement de la ligne *ligne*
  Envoi des blocs *bij* de la ligne $\frac{(Pi + 1) \,.\, I}{N} - 1$
Fin

Dans ce cas si on considère $I/N \gg 1$, le traitement de la première ligne peut être négligé, et on a alors $P = 1$. Il faut également remarquer que si I/N n'est pas entier, tous les processeurs n'auront pas le même nombre de lignes à traiter et auront un volume de données à échanger D_p différents. Pour l'étude suivante nous nous placerons dans une hypothèse d'équirépartition spatiale avec P égal à 1.

Pour les applications numériques nous prendrons dans la suite le cas où I, taille d'une ligne de l'image, vaut 128 et où K, dimension du modèle de la PSF, vaut 3.

On voit apparaître trois parties distinctes dans l'algorithme :

- La TFR sur les lignes avec communication des blocs de données *bij* du processeur *i* vers le processeur *j*.

Son temps d'exécution :

$$T_{seq\ TFR} = \frac{I^2}{2} \log_2 I\, T_{pap} = 0{,}9 \text{ s}.$$

$T_{pap} = 15{,}8\ \mu s$, temps de calcul d'un papillon de la TFR. Pour une quantité d'échanges par PE de : $D_p = \frac{I^2}{N^2}(N - 1)$.

- Le filtrage de Kalman sur les colonnes de l'image transformée avec à nouveau communication de PE*i* vers PE*j* des blocs *bij*.

Son temps d'exécution : $T_{seq\ KAL} = I^2\, T_{kal} = 2$ s.

$T_{kal} = 122{,}1\ \mu s$, temps de prédiction/estimation d'un pixel de l'image.

Pour une même quantité de communications : $D_p = \frac{I^2}{N^2}(N - 1)$.

- La TFR inverse sur les lignes possédant un temps de calcul identique à la première TFR et n'engendrant aucune communication.

Le temps séquentiel global de l'algorithme sur un seul processeur vaut :

$$T_{seq} = I^2 \log_2 I\, T_{pap} + I^2\, T_{kal},$$

le temps idéal sur un réseau de N processeurs valant donc $T_i = T_{seq}/N$.

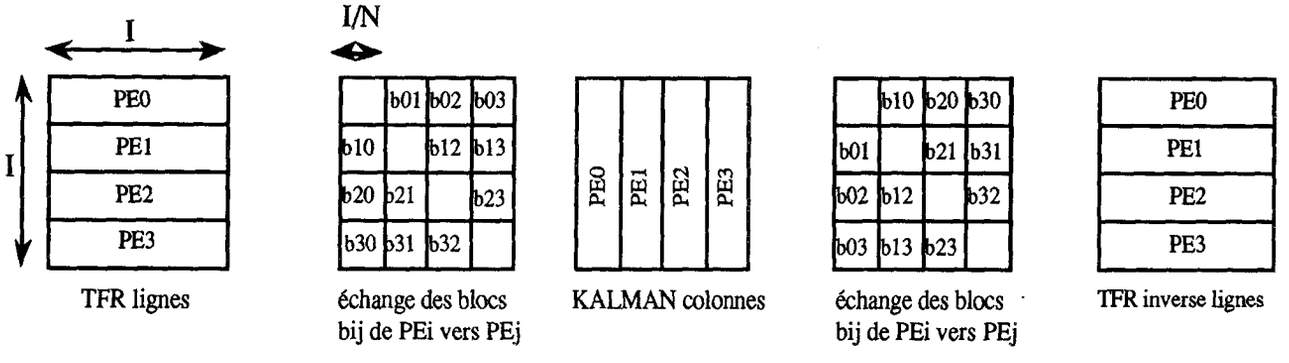

Figure 5. — Organisation des échanges et des calculs du filtrage de Kalman parallèle.

3.3. ÉTUDE DE LA SATURATION SUR UN ANNEAU BIDIRECTIONNEL

Les taux de communication C_1 et C_2 locaux aux deux premières parties vont engendrer chacun une saturation locale des communications, on voit donc immédiatement que c'est la partie Transformation de Fourier, dont le temps est le plus faible, qui va nous imposer une valeur de N minimale pour que l'algorithme dans sa globalité ne soit pas saturé. Les volumes de données communiquées étant égaux pour les deux parties.

Le taux de communication est calculé en fonction de D_p, T_{seq} et T_{com} : Temps de communication d'une donnée complexe (deux nombres réels 32 bits) sur les liens du transputer (6,8 μs).

Sur un anneau, la saturation apparaît lorsque $N > \frac{8}{C} - 1$, ce même facteur C variant entre la TFR lignes et le filtrage de Kalman colonnes.

3.3.1. TFR lignes

Dans le cas de la TFR sur les lignes, la saturation apparaît si :

$$N > \frac{8\,T_{seq}}{I^2\,T_{com}} = \frac{4\log_2(I)\,T_{pap}}{T_{com}}$$

Tableau 3. — Seuil de saturation de la partie TFR lignes.

I	32	64	128	256	512	1024
N sat	> 32	*56*	*65*	*74*	84	93

Les résultats du tableau 3 donnent le nombre de processeurs pour lequel la saturation est atteinte (N_{sat}) en fonction de I, taille d'une ligne de l'image. Quand N_{sat} est donné en italique on remarque que la condition $I/N \gg 1$ n'est plus vérifiée, il faudrait également prendre pour N_{sat} la valeur immédiatement supérieure qui soit une puissance de deux (I étant ici une puissance de deux).

L'efficacité vaut hors saturation sur la partie TFR lignes :

$$\text{efficacité}\,(I, N) = \frac{1}{1 + T_{nt} \cdot T_{com} \dfrac{IN^2}{4\,T_{seq}}} = \frac{1}{1 + T_{nt} \cdot T_{com} \dfrac{N^2}{2\,I \log_2 IT_{pap}}}$$

avec $T_{nt} \cdot T_{com} = (2{,}7 \cdot N_{oct} + 100)$ μs, N_{oct} représentant la taille des blocs transférés sur les liens (I/N données complexes dans notre cas). Cette valeur du taux de non-transparence, mesurée par émulation, est élevée car nous utilisons un routeur complexe intégrant des multiplexeurs programmés. On remarque que la valeur de N_{oct} influence peu T_{nt} pour N grand, ce qui montre que la partie routage et initialisation du transfert est bien supérieure à la partie communication (perturbations dues au DMA du gestionnaire des liens). Il est donc plus intéressant d'échanger des blocs assez longs et le moins souvent possible. Cette équation fait donc apparaître que plus le nombre de processeurs sera élevé, plus la partie « échanges non transparents » sera prédominante.

Tableau 4. — Efficacité locale de la partie TFR lignes.

N	2	4	8	16	32
Dp	4096	3072	1792	960	496
C	0,028	0,042	0,0485	0,052	0,0538
Tcal idéal	0,453	0,226	0,113	0,057	0,028
Tcal réel	0,47	0,25	0,15	0,12	0,142
Efficacité	96,29%	90,47%	75,55%	47,63%	19,97%
Accélération	1,93	3,62	6,05	7,62	6,39

Dans le tableau 4 on voit apparaître plusieurs paramètres fonctions du nombre de processeurs : D_p : nombre de données que chaque processeur aura à communiquer, C : taux de communication de la partie TFR lignes, T_{cal} idéal : temps de calcul idéal sur N processeurs et T_{cal} réel : le temps de calcul réel après implémentation sur l'architecture cible.

On remarque qu'au-delà de 16 processeurs, et même si la saturation n'est pas atteinte, le temps de calcul réel cesse de décroître et même commence à augmenter (voir tableau 4). Ceci est dû au temps que le processeur passe à router les messages sur le réseau et prouve donc bien que le transputer seul ne peut assurer les fonctions de CPU et de gestionnaire de réseau. On voit ainsi apparaître la nécessité de décharger le processeur d'une de ces deux fonctions, en fournissant au transputer un coprocesseur généré par notre outil de synthèse automatique d'unité de traitement Gaut [14], lui évitant ainsi d'effectuer les calculs critiques de la TFR ou du filtrage de Kalman.

3.3.2. Kalman colonnes

Dans le cas du filtrage de Kalman sur les colonnes, la saturation apparaît si :

$$N > \frac{8\,T_{seq}}{I^2\,T_{com}} = \frac{8\,T_{kal}}{T_{com}}$$

$$N > 144$$

L'efficacité vaut hors saturation :

$$\frac{1}{1 + T_{nt} \cdot T_{com} \dfrac{N^2}{4\,IT_{kal}}}.$$

Dans le cas de la partie filtrage de Kalman, les résultats (voir tableau 5) ne sont pas aussi critiques pour des valeurs élevées de N. Ceci est simplement dû au temps séquentiel deux fois plus élevé que dans le cas de la TFR.

Tableau 5. — Efficacité locale de la partie filtrage de Kalman colonnes.

N	2	4	8	16	32
Efficacité	98,29%	95,46%	87,24%	66,81%	35,58%
Accélération	1,97	3,82	6,98	10,69	11,39

3.4. EFFICACITÉ DE LA PARALLÉLISATION

Comme nous l'avons montré, la restauration d'une image par filtrage de Kalman peut se décomposer en trois parties distinctes :

- TFR 1D sur les I lignes de l'image avec échanges de portions de lignes de taille I/N entre processeurs,
- I filtrages de Kalman 1D sur chaque colonne de l'image dans le plan de Fourier avec échanges des portions de colonnes de taille I/N,
- TFR inverse 1D sur les I lignes de l'image résultante (sans échange).

Ce qui donne donc un temps d'exécution sur un seul processeur :

$$T_{\text{kalman 2D}} = \frac{I^2}{2}\log_2(I)\,T_{\text{pap}} + I^2\,T_{\text{kal}} + \frac{I^2}{2}\log_2(I)\,T_{\text{pap}}\,.$$

Le temps de calcul réel après implémentation sur le réseau en anneau donné ci-après implique une valeur d'efficacité dépendante de I et de N définie comme suit (voir tableau 6) :

$$T_{\text{cal}} = \frac{T_{\text{kalman 2D}}}{N} + 2 \times \left[T_{\text{nt}}\,T_{\text{com}}\,\frac{D_{\text{p FFT/KAL}}}{\text{taille}}\,d_{\text{moy}}\right]$$

$$\text{efficacité}\ (I, N) = \frac{1}{1 + 2\,T_{\text{nt}}\,T_{\text{com}}\,\dfrac{IN^2}{4\,T_{\text{kalman 2D}}}}\,.$$

Tableau 6. — Efficacité globale du filtrage de Kalman sur un anneau bidirectionnel.

N	2	4	8	16	32
Tcal réel	1,944	1,002	0,55	0,363	0,346
Efficacité	98,20%	95,24%	86,68%	65,71%	34,46%
Accélération	1,96	3,8	6,93	10,51	11,03

Le tableau 6 montre bien que la parallélisation permet d'approcher la contrainte temps réel des applications de restauration d'image (ici 3 images par seconde pour N = 16). Le choix d'un processeur plus performant permettra de diminuer encore le temps de calcul réel. Ces estimations ont bien entendu été vérifiées lors d'une implémentation réelle sur l'architecture. Les prévisions calculées à partir des modèles par ESPION diffèrent de moins de 5 % des mesures effectuées sur les cartes de transputers.

3.5. ÉTUDE SUR D'AUTRES RÉSEAUX

Nous avons complété l'étude par une prévision des efficacités de parallélisation toujours appliquée au transputer, mais sur des réseaux plus complexes et donc plus coûteux que l'anneau : l'hypercube, un processeur est relié à $\log_2 N$ autres PE et le réseau crossbar, un processeur est relié à tous les autres. On reprend donc la méthode appliquée à l'anneau avec les nouveaux paramètres des réseaux étudiés (voir tableau 2) et on en déduit les prévisions des tableaux 7 et 8.

Tableau 7. — Efficacité globale du filtrage de Kalman sur un réseau hypercube.

N	2	4	8	16	32
Tcal réel	1,944	1,002	0,532	0,301	0,190
Efficacité	98,20%	95,24%	89,67%	79,31%	62,72%
Accélération	1,96	3,8	7,17	12,7	20,1

Tableau 8. — Efficacité globale du filtrage de Kalman sur un réseau crossbar.

N	2	4	8	16	32
Tcal réel	1,944	0,990	0,509	0,268	0,147
Efficacité	98,20%	96,39%	93,70%	89,10%	81,28%
Accélération	1,96	3,85	7,5	14,25	26

On peut remarquer que si les efficacités sont bien meilleures, pour de grandes valeurs de N, le coût du réseau croît encore plus rapidement. En effet un transputer ne possédant que quatre liens bidirectionnels physiques, le réseau crossbar doit être géré extérieurement par d'autres composants pour des valeurs de N supérieurs à 4. De plus un hypercube de transputers n'est réalisable simplement que pour $N \leq 16$. On peut donc dire qu'un anneau, ne nécessitant que du câblage, avec des procédures de routage des données optimisées, donne des résultats satisfaisants pour le transputer.

3.6. BILAN DE LA PARALLÉLISATION : CHOIX D'UNE ARCHITECTURE

La figure 6 donne la mesure d'accélération de la parallélisation du filtrage de Kalman en fonction du nombre de processeurs pour plusieurs réseaux, la courbe idéale étant la droite Accélération = N. Elle donne également le temps de calcul du filtre dans les mêmes conditions. On pourrait ainsi, grâce à ces courbes choisir une architecture à base de transputers satisfaisant par exemple la contrainte temps réel de 5 images par secondes ($T_{\text{cal}} = 0{,}2$ s). On voit que plusieurs réseaux respectent cette contrainte, mais seul le réseau carré maillé (tout processeur est relié à quatre voisins) peut être directement implémenté pour $N > 16$ sans adjonction de composants réseaux. Cette remarque permet ainsi à l'utilisateur de choisir, pourquoi pas, un réseau de ce type.

Le cas du TMS320C25 [15] est encore plus simple puisqu'il ne possède qu'un seul lien bidirectionnel, le seul réseau envisageable simplement, sans composants externes, est l'anneau unidirectionnel. On s'aperçoit que, dans ce cas, l'efficacité de la parallélisation est très faible. Ceci est dû aux faibles performances de communication du Processeur et à la pauvreté du réseau. La figure 7 montre que si l'on veut respecter le temps réel précédent, une architecture simple à base de TMS320C25 n'est pas correcte. On pourra par contre envisager d'utiliser des processeurs de traitement du signal de générations postérieures comme le TMS320C30 (2 liens) ou le TMS320C40 (6 liens rapides).

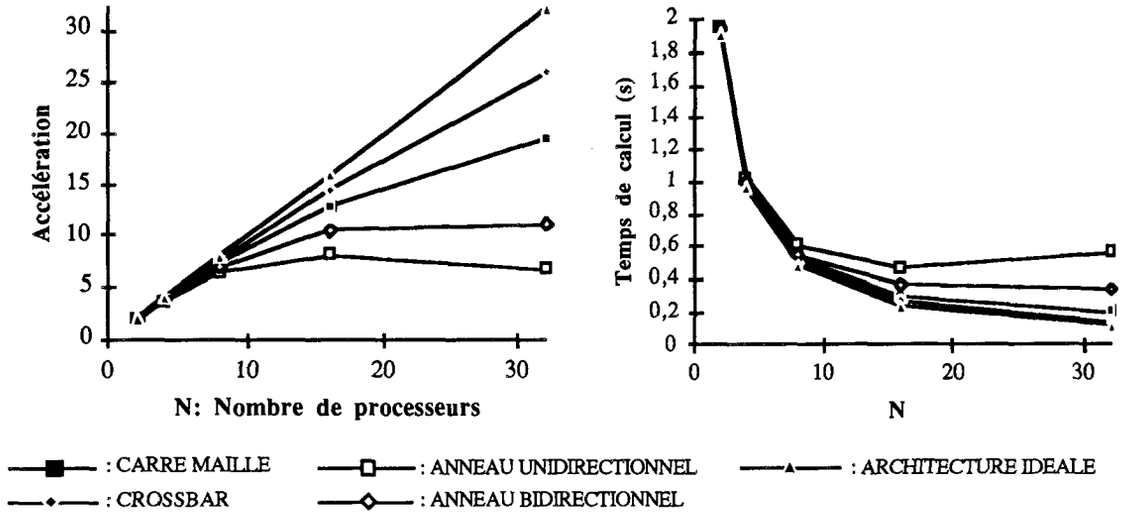

Figure 6. — Accélération et temps de calcul de l'algorithme sur transputers.

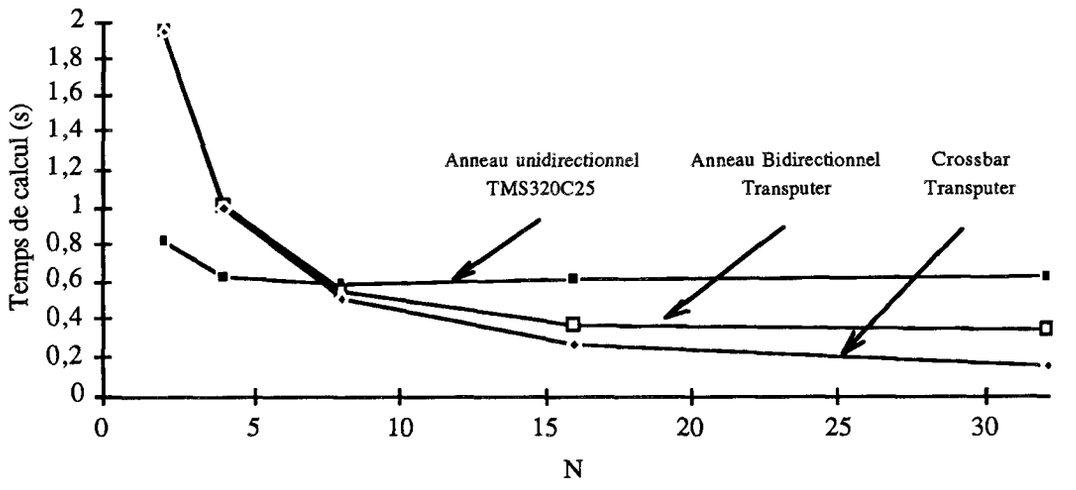

Figure 7. — Comparaison entre les solutions à base de transputers et de TMS320C25.

4. Conclusion

L'expertise des performances par ESPION se révèle bien plus intéressante que la simple approximation de l'efficacité par la complexité de l'algorithme ramenée au nombre de processeurs. De plus il permet l'évaluation d'un compromis efficacité/coût, à partir d'une bibliothèque de processeurs (transputer, processeurs de traitement de signal, ...) et de réseaux. Cette expertise peut s'appliquer à des machines existantes en fournissant par les critères d'ESPION une évaluation de leurs performances.

L'expertise précédente sur le filtrage de Kalman s'est révélée un bon exemple, mais d'autres sont en cours d'études (TFR 1-D, Filtrage 2-D, ...) afin d'automatiser l'outil et de lever les quelques hypothèses restantes. Une version d'ESPION dans le cas d'algorithmes irréguliers a même été étudiée [16].

Nos perspectives visent à une convergence entre nos analyses et notre outil de synthèse d'unité de traitement dédiée au traitement de signal, en vue de décharger les processeurs utilisés d'une partie pertinente des calculs vers un ASIC spécialisé afin de faire tendre les temps d'exécution vers les contraintes imposées par le temps réel.

Manuscrit reçu le 28 février 1992.

BIBLIOGRAPHIE

[1] H. Dubois, Analyse des systèmes multiprocesseurs : Application à la mise en œuvre sous contraintes d'algorithmes de traitement d'images, *Thèse Université de Rennes I, ENSSAT LASTI,* Lannion, janvier 1991.

[2] J. Biemond, A Fast Kalman Filter for Images Degraded by both Blur and Noise, *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. 31, n° 5, pp. 1248-1256, October 1983.

[3] J. W. Woods, Kalman Filtering in Two Dimensions, *IEEE Transactions on Information Theory,* vol. IT-23, n° 4, pp. 473-482, July 1977.

[4] J. W. Woods, Kalman Filtering in Two Dimensions : Further Results, *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. ASSP-29, n° 2, pp. 188-196, April 1981.

[5] L. Blanc-Feraud, M. Barlaud, P. Mathieu, Amélioration de la restauration d'images floues par un filtrage adaptatif de Kalman utilisant une image miroir. 2e Atelier Scientifique : Traitement d'images : du Pixel à l'interprétation Aussois, Savoie, avril 1988.

[6] M. Sabri, Filtrage et Restauration en traitement des images numériques : Recherche d'une mise en œuvre automatique, *Thèse Université de Rennes I, ENSSAT LASTI,* Lannion, mars 1991.

[7] J. Biemond, F. G. van der Putten, J. W. Woods, A Parallel Identification Procedure for Images with Noncausal Symetric Blurs, *Proceedings IEEE, ICASSP,* Tokyo, pp. 1489-1492, 1986.

[8] INMOST « The IMS T800 transputer », ingineering data, avril 1987.

[9] E. Martin, H. Dubois, O. Sentieys et J. L. Philippe, Définition de mesures objectives de performances pour la mise en œuvre parallèle d'algorithmes de traitement d'image. Treizième colloque GRETSI sur le traitement du signal et des images, septembre 1991.

[10] E. T. Fathi et M. Krieger, Multiple Microprocessor Systems : What, Why and When, *IEEE Computer,* pp. 23-32, March 1983.

[11] D. P. Agrawal et V. K. Janakiram, Evaluating the performance of multicomputer configurations, *IEEE Computer,* pp. 23-37, May 1986.

[12] T. S. Feng, A survey of interconnection networks, *IEEE Computer,* pp. 12-27, December 1981.

[13] INMOS « OCCAM reference manual », Prentice Hall 1988.

[14] E. Martin *et al.*, Synthèse d'architectures dédiées au traitement du signal temps réel. AFCET, Workshop synthèse et compilation d'architecture, Grasse, 23-24 janvier 1992.

[15] TEXAS INSTRUMENTS « Second-Generation TMS320 », Digital Signal Processor Products, Texas Instruments, 1989.

[16] J. L. Philippe, H. Dubois, E. Martin & M. Corazza, « Evaluation of parallel structure dedicated to undersea image processing : application to TOS project », International workshop on algorithms, Pont-à-Mousson, juin 90.