Dix-septième colloque GRETSI, Vannes, 13-17 septembre 1999

Implémentation temps réel d'algorithme de détection de mouvement par champs de Markov sur RISC et DSP C6x

Lionel Lacassagne[1,2], Frantz Lohier[1], Maurice Milgram[1], Patrick Garda[1]

[1]Laboratoire des Instruments et Systèmes
Université Pierre et Marie Curie
4 place Jussieu
75252 Paris Cedex 05

[2]Imasys
« Le Quadral »
23 bis rue E. Nieuport
92150 Suresnes

lionel | lohier | milgram | garda @lis.jussieu.fr

lionel@imasys.fr

Résumé – Cet article décrit une implémentation temps réel d'un algorithme de détection de mouvement basée sur les champs de Markov. Il décrit aussi les optimisations architecturales appliquées aux RISC et au DSP C6x qui permettent d'atteindre le temps réel.

Abstract – This paper describes a real-time implementation of motion detection algorithm based on Markov Random Field. It introduces architectural optimizations applied to RISC and DSP C6x that let us to achieve real time processing.

1. Introduction

La détection de mouvement est la base de tout système de suivi d'objet, ou de compression d'image. Les algorithmes de détection de mouvement se classent en deux grandes classes : flot optique et corrélation. Ces méthodes ont été largement explorées, chacune ayant ses avantages et ses inconvénients. La première et la plus ancienne de toutes les méthodes est une simple différence d'image, où une variation notoire du niveau de gris est associée à un mouvement.

En parallèle se sont développés des algorithmes à base de champs de Markov (MRF), dans de nombreux domaines en traitement d'images (détection de contours, segmentation de régions restauration d'images bruitées). Leur rôle étant d'améliorer la qualité des résultats [6].

Si leurs avantages majeurs sont la robustesse et la qualité des résultats, leur principal inconvénient est leur lenteur d'exécution due au grand volume de calcul. Cela a conduit à étudier de nombreuses solutions pour accélérer leur exécution, telles des machines massivement parallèle ou des machines dédiées [1][2][5].

Dans cet article, nous décrivons une utilisation des champs de Markov visant à améliorer une image de différence. Puis en expliquant les différents types d'optimisations, nous montrons comment atteindre le temps réel sur processeurs RISC et DSP.

2. Méthode markovienne de détection de mouvement.

Le but du processus markovien est d'améliorer la qualité de l'image de différence, grâce à un algorithme de relaxation. Le modèle d'énergie utilisé a été introduit par le LIS-Grenoble [3][4].

Soient à l'instant t, I_t, l'image en niveaux de gris et O_t, l'observation c'est à dire la différence absolue de deux images consécutives. Ces observations, une fois binarisées (seuillage), servent d'estimation au champ d'étiquettes $\hat{E}_t$. L'ICM (Iterated Conditional Modes, 4 passes par image) est utilisé pour déterminer le champ d'étiquettes relaxées E_{t-1} à partir de $\hat{E}_t$, $\hat{E}_{t-1}$ et $\hat{E}_{t-2}$. La relaxation étant déterministe, l'algorithme converge en quelques itérations, vers le premier minimum local, d'où l'importance de l'initialisation. La simple différence d'image peut être remplacée par un calcul de maximum de vraisemblance [7] avec ou sans intervalle de confiance.

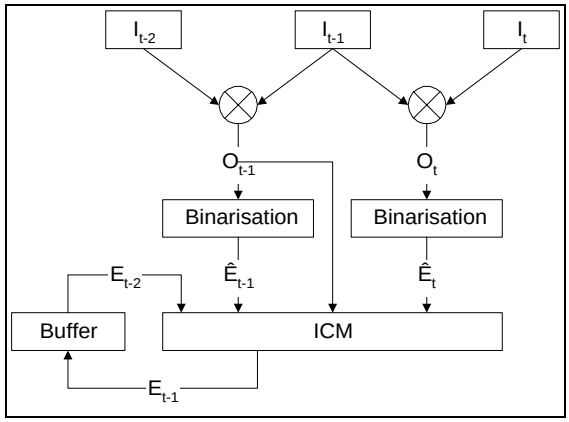

Figure 1

2.1 Modèle d'énergie et clique considérée

Le modèle d'énergie u_m est la somme de deux énergies : u_m, énergie de modèle, qui exprime la relation entre pixels voisins et u_a énergie d'adéquation, ou d'attache aux données qui empêche le modèle de trop dériver par rapport à l'initialisation.

Nous avons donc :

$$u(o_s, e_s) = u_m(e_s) + u_a(o_s, e_s) \quad (1)$$

$$u_m(e_s) = \sum_{c \in C} V_c(e_s, e_r) \quad (2)$$

$$u_a(o_s, e_s) = \frac{1}{2\sigma^2}\left[o_s - \Psi(e_s)\right]^2 \quad (3)$$

$$\Psi(e_s) = \begin{cases} 0 \text{ si fond} \\ \alpha \text{ si mouvement} \end{cases} \quad (4)$$

La clique associée à l'énergie u_m est une clique spatio-temporelle (Figure 2) d'ordre 2.

Le potentiel de la clique spatiale V_c est :

$$V_c(e_s, e_r) = \begin{cases} -\beta_s \text{ si } e_s = e_r \\ +\beta_s \text{ si } e_s \neq e_r \end{cases} \quad (5)$$

Les deux potentiels associés aux cliques temporelles (passée et future) sont :

$$V_p(e^t{}_s, e^{t-1}{}_s) = \begin{cases} -\beta_s \text{ si } e^t{}_s = e^{t-1}{}_s \\ +\beta_s \text{ si } e^t{}_s \neq e^{t-1}{}_s \end{cases} \quad (6)$$

$$V_f(e^t{}_s, e^{t+1}{}_s) = \begin{cases} -\beta_s \text{ si } e^t{}_s = e^{t+1}{}_s \\ +\beta_s \text{ si } e^t{}_s \neq e^{t+1}{}_s \end{cases} \quad (7)$$

En reprenant le modèle d'Ising (spin-up, spin-down), ces énergies peuvent s'écrire sous la forme :

$$u_m = -\beta_s \sum_{c \in C} x_s x_r - \beta_p x^t{}_s x^{t-1}{}_s - \beta_p x^t{}_s x^{t+1}{}_s \quad (8)$$

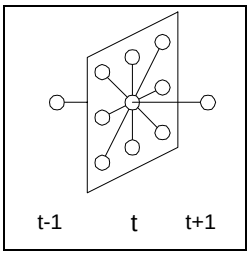

Figure 2

La mise à jour des étiquettes est « site-récursive » car elle donne les meilleurs résultats.

Afin de faire disparaître la trace de l'image I_{t-1}, il convient de prendre une valeur de β_f supérieure à β_p. Les coefficients vérifient donc l'inégalité suivante :

$$\beta_p < \beta_s < \beta_f \quad (9)$$

Typiquement les valeurs retenues sont :

$$\beta_p = 10, \beta_s = 20, \beta_f = 30, \alpha = 20 \quad (10)$$

3. Optimisations

Pour accélérer le traitement, il faut tenir compte de l'architecture de l'unité de traitement des processeurs cibles, à savoir un grand nombre de registres et un traitement pipeline des instructions pour les calculs proprement dit, et de la hiérarchie mémoire c'est à dire de l'accès aux données via les mémoires caches.

3.1 Les techniques d'optimisations

La technique la plus utilisée est le déroulement de boucle (loop-unrolling, Figure 3). Elle consiste dans le cas le plus simple à dupliquer le corps de la boucle afin d'avoir une meilleure utilisation de l'unité de calcul. En fonction de l'algorithme, et c'est le cas ici, il est possible de lancer des calculs en «parallèle». Malgré cela le pipeline de l'unité de traitement n'est pas toujours plein. Le software pipelining y remédie en exécutant dans le corps de boucle des instructions appartenant à d'autre itérations.

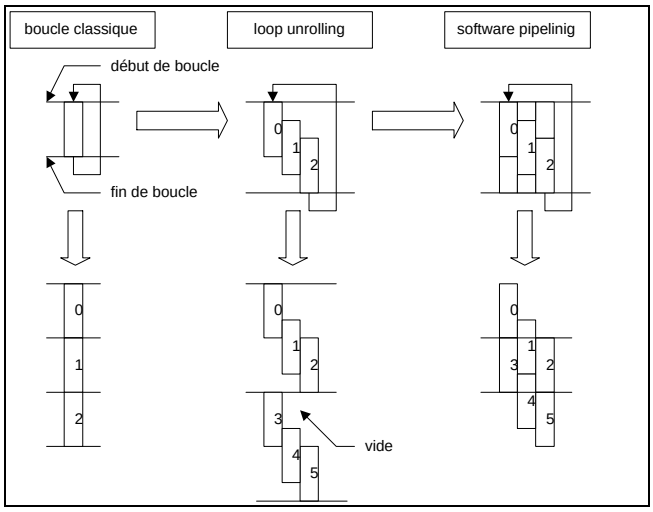

Figure 3

3.2 Particularités de l'algorithme

Entre deux itérations successives, il y a un recouvrement de deux colonnes. Le calcul de l'énergie spatiale est donc découpé en trois colonnes.

Le calcul des différents termes de l'énergie du modèle se fait par LUT (Look Up Table).

Nous étiquetons les sites de fond/mouvement à 0/1 plutôt que -1/+1. Au lieu de comparer la valeur de chaque site au site central, nous comptons le nombre de sites à 1 (pour l'image passée, spatiale et future). L'énergie spatiale vaut donc :

$$u_1 = (8 - 2s_1)\beta_s, \; u_0 = -u_1 \quad (11)$$

Et donc si

$$um_1 + ua_1 < um_0 + ua_0 \quad (12)$$

le site passe à 1 sinon il passe à zéro. Et ce, *sans se soucier* de son état précédent. Le test peut être simplifié en regroupant d'un côté l'énergie de modèle et de l'autre l'énergie d'adéquation. Cette modification, a-priori mineure, fait gagner 1 cycle sur 6 au C6x, soit 17% !

4. Implémentation sur RISC et DSP

L'algorithme a été implémenté sur Intel Pentium II-400 MHz, et estimé, en fonction du nombre de cycles du cœur de boucle sur le plus puissant des TI C6x, le C6202-250 MHz.

4.1 Optimisations utilisées

Sur Pentium, le nombre de registres est insuffisant pour faire efficacement du loop-unrolling. C'est donc la version à 8 accès par point qui est implémentée avec utilisation de LUTs.

Sur C6x, le nombre de registre étant suffisant, nous pouvons véritablement effectuer des calculs en parallèle. Nous appliquons un Loop-unrolling d'ordre 3 (la Figure 4 donne le diagramme temporel des calculs, les indices correspondent au retard par rapport à l'indice courant)

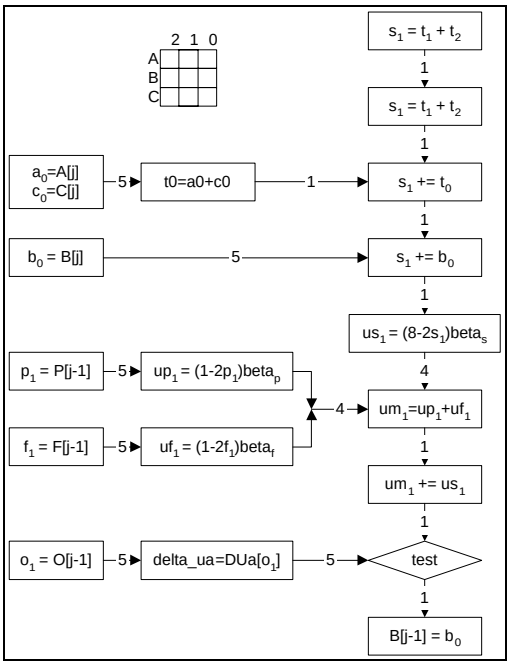

Figure 4

Par point, et sur un C6202 à 250 MHz, nous estimons la complexité et la puissance de calcul à :

Tableau 1: complexité

techniques d'optimisation	aucune	LU	SP
instructions	22	66	66
cycles	15	25	15
ipc	1.47	2.64	4.40
% charge	15	26	44
MIPS	367	660	1100

5. Résultats et analyse

Le tableau ci-dessous donne les temps de traitement pour l'ICM. Le temps du C6x est une estimation basée sur le nombre de cycle de la boucle, sans prendre en compte les accès DMA.

Tableau 2 : résultats

Laboratoire	architecture	nb proc	fréquence (MHz)	taille image	nb image/s
Lis Grenoble [4]	CNAPS	256	20	128	3
Lis Grenoble [3]	M 96002	1	40	128	15
Lis Paris [8]	C80	1+4	60	256	18
Lis Paris	Pentium II	1	400	256	40
Lis Paris	C6202 (simulé)	1	250	512	47

En remplaçant des tableaux d'octets (0/1) par des tableaux de bits, nous devrions traiter des images 512x512 en 14 ms, contre 21ms actuellement.

Malgré sa puissance fantastique, le C6x n'est utilisé qu'à 44% (55% avec un tableau de bit) de ses performances crêtes, ces limitations proviennent de l'algorithme lui-même, mais aussi de l'architecture du DSP.

6. Conclusion

Dans cet article, nous avons appliqué des méthodes d'optimisations du calcul scientifique à un algorithme de détection de mouvement par champs de Markov. Grâce à ces optimisations, un algorithme de relaxation comme l'ICM, qui est a-priori complexe, se ramène à une simple convolution par LUT.

Cela permet donc de faire de la détection de mouvement fiable par MRF en temps réel.

Un pentium II est capable de traiter à la cadence vidéo des images 256x256 et un C6202 devrait être capable de traiter des images 512x512. A notre connaissance ces résultats constituent une première mondiale.

De plus, comme les cadences dans ces formats d'images sont plus que temps réel, il reste du temps pour ajouter un algorithme de suivi d'objets, et donc avoir une chaîne complète de traitement temps réel sur des monoprocesseurs.

7. Références

[1] A.Bellon « Détection et suivi de véhicule en mouvement, implémentation parallèle sur un système à mémoire distribuée. » thèse LASMEA Clermont Ferrand, Oct 1996.

[2] R. Azencott « Simulated annealing : parallelization techniques » J.Whiley 1992.

[3] A. Caplier, F. Luthon, C. Dumontier « Algorithme markovien de détection de mouvement, mise en œuvre 'temps réel' » GRETSI 1995.

[4] A. Caplier « Modèles markoviens de détection de mouvements dans les séquences d'images : approches spatio-temporelle et mise en œuvre temps réel » thèse INPG Grenoble 1995.

[5] J.P.Cocquerez, S.Philipp « Analyse d'images : filtrage et segmentation », Masson 95. Temps de calcul sur Sparc5 et CM2 (INRIA-Sophia), p220.

[6] F.Heitz, P.Bouthémy « Multimodal Estimation of Discontinuous Optical Flow using MRF » Trans on PAMI-15,N-12 Dec 1993.

[7] Y.Z.Hsu, H.H. Nagel, G.Rekers « New likelihood test methods for change detection in image sequences » CVGIP 26, 73-106 (1984).

[8] F. Lohier et al. Generic programming methods for the real time implementation of MRF based motion detection algorithm on a multiprocessors DSP with multidimentional DMA. Gretsi 1999.