Evaluation d'un algorithme de détection de mouvement par approche markovienne sur composants reconfigurables dynamiquement

Nassima BOUDOUANI, Lounis KESSAL, Didier DEMIGNY

ETIS UPRSEA CNRS 8051, ENSEA et Université de Cergy Pontoise

ENSEA, 6 av. du Ponceau, 95014 Cergy Pontoise cedex

boudouani@ensea.fr

Résumé – Cet article décrit une implantation temps réel sur FPGA reconfigurables dynamiquement d'un algorithme de détection de mouvement basé sur les champs de markov.

Abstract *– This paper describes a real-time impementation on dynamically reconfigurable FPGA of a motion detection algorithm based on Markov Random Field.*

1. Introduction

L'apparition des FPGA offrant à la fois un temps de reconfiguration réduit et la possibilité de reconfigurer le composant au cours de son fonctionnement marque l'émergence d'un nouveau paradigme : vers un matériel aussi malléable que le logiciel. Cette technique appelée reconfiguration dynamique procure de la souplesse aux applications nécessitant de la puissance de calcul.

Dans cet article nous nous sommes intéressés à l'implantation sur composant reconfigurable dynamiquement d'un algorithme de détection de mouvement basé sur une approche markovienne. C'est un algorithme qui a été développé à l'IRISA à Rennes[1], repris pour être simplifié et implémenté sur machines parallèle, DSP et réseaux résistifs analogique [2][3][4] par le LIS-Grenoble et récemment implémenté sur des DSP C80, C60 et PentiumII[5] au LIS-Paris. Pour notre travail nous avons utilisé la version de cet algorithme établie au LIS de Grenoble.

Le but de notre travail est d'étudier la faisabilité de cette implantation et d'en déduire des connaissances sur les exigences architecturales qu'induisent les algorithmes basés sur une modélisation markovienne connue pour leur robustesse mais aussi pour leur complexité calculatoire.

Nous avons utilisé comme support à notre travail l'architecture ARDOISE (Architecture reconfigurable Dynamiquement Orientée Image et Signal Embarquée)[6].

2. La détection de mouvement par modélisation markovienne

En faisant l'hypothèse d'un capteur fixe et un éclairement stable de la scène, toute variation temporelle de l'intensité est liée au mouvement d'un objet.

Les informations de mouvement ou observations issues de la séquence d'image sont définies en chaque site s=(x, y) par : $o_t(s) = |I_{t+1}(s) - I_t(s)|$.

La détection de mouvement vise à trouver la meilleure estimation de l'étiquette e(s) (fixe ou mobile) étant donné l'observation $o_t(s)$. L'estimateur utilisé est celui du maximum a posteriori (MAP).

Maximiser la probabilité a posteriori revient à rechercher le minimum d'une fonction d'énergie globale $U(o_s, e_s)$, cette opération est effectuée par l'algorithme itératif de relaxation déterministe ICM (Iterated Conditional Modes).

Le schéma synoptique suivant résume le fonctionnement de l'algorithme.

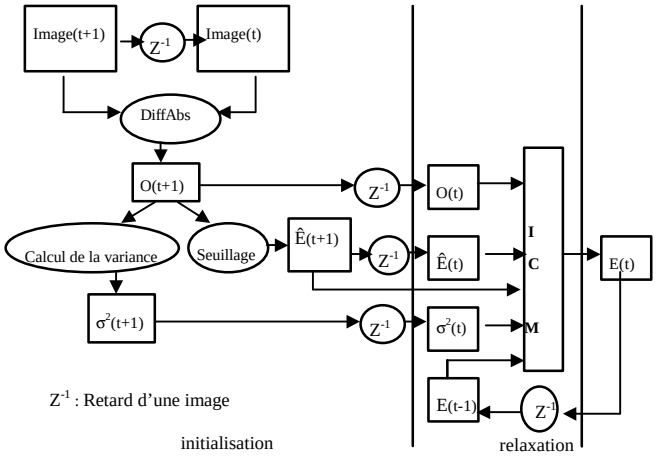

FIG. 1 : schéma synoptique du détecteur de mouvement

L'énergie $U(o_s, e_s)$ est composée de deux termes :
$U_m(e_s)$: terme d'énergie lié au modèle
$U_a(o_s, e_s)$: terme d'énergie d'attache aux données.

$$U_m(e_s) = \sum_{c \in C} V_c(e_s, e_r)$$

$$U_a(o_s, e_s) = \frac{1}{2\sigma^2}\left[o_s - y(e_s)\right]^2$$

$$y(e_s) = \begin{cases} 0 \text{ si } e_s = 0 \text{ (fond)} \\ a \text{ si } e_s = 1 \text{ (mouvement)} \end{cases}$$

Le calcul de l'énergie U_m se fait suivant un voisinage spatio-temporel d'ordre 2 (FIG.2).

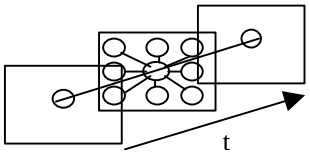

FIG. 2

Le potentiel de la clique spatiale est :

$$V_c(e_s, e_r) = -b_s \text{ si } e(s) = e(r_s)$$
$$= +b_s \text{ si } e(s) \neq e(r_s)$$

Les deux potentiels associés aux cliques temporelles (passées et futures) sont :

$$Vc(e_s^t, e_s^{t-1}) = -b_p \text{ si } e_s^t = e_s^{t-1} \qquad Vc(e_s^t, e_s^{t+1}) = -b_f \text{ si } e_s^t = e_s^{t+1}$$
$$= +b_p \text{ si } e_s^t \neq e_s^{t-1} \qquad = +b_f \text{ si } e_s^t \neq e_s^{t+1}$$

Le choix des valeurs des coefficients β_i et α a fait l'objet de l'étude menée dans [2], ils sont fixés à :

$\beta_p = 10$, $\beta_s = 20$, $\beta_f = 30$, $\alpha=20$.

3. Spécificités de l'algorithme et calcul de l'ICM

3.1 Minimisation de l'énergie

D'après l'étude faite dans [5], minimiser l'énergie U d'un site revient à calculer l'énergie U_1 associée l'étiquette e = 1 et U_0 associé à l'étiquette e = 0 ; puis prendre comme nouvelle étiquette du site celle qui donne la plus petite énergie.
Ainsi U_{m1}, U_{a1} correspondent à e=1 et U_{m0}, U_{a0} à e=0.
Nous aboutirons donc à la formulation suivante des énergies:

$$U_{m1} = (8 - 2s_1)\, b_s + (1 - 2p_1)\, b_p + (1 - 2f_1)\, b_f$$
$$U_{m0} = -U_{m1} \quad (1)$$
$$U_{a1} = \frac{1}{2\sigma^2}[o - \alpha]^2$$
$$U_{a0} = \frac{1}{2\sigma^2} o^2$$

s_1, p_1, f_1 représentent respectivement le nombre de sites de la clique spatiale, et des cliques temporelles passées et futures dont l'étiquette e est à 1.
Le test que nous devons faire pour décider de l'étiquette à attribuer à un site s est :

$$Si\ (U_{m1} + U_{a1}) < (U_{m0} + U_{a0}) \text{ alors } e(s) = 1 \text{ sinon } e(s) = 0$$

En utilisant la propriété (1) nous aboutissons au test suivant :

$$Si\ 4\sigma^2 U_{m1} < 2\alpha o - \alpha^2 \text{ alors } e(s) = 1 \text{ sinon } e(s) = 0$$

Les valeurs de U_{m1} étant multiples de 10 nous pourrons encore simplifier notre test en considérant les même valeurs de l'énergie U_{m1} divisées par 10, et en remplaçant a par sa valeur, nous aboutirons au test suivant :

$$Si\ \sigma^2 U_{m1} < o - 10 \text{ alors } e(s) = 1 \text{ sinon } e(s) = 0$$

3.2 Calcul de l'ICM, mise à jour des sites, balayage de l'image et implications sur la gestion des données

La schéma de la FIG.3 illustre le calcul de la partie ICM

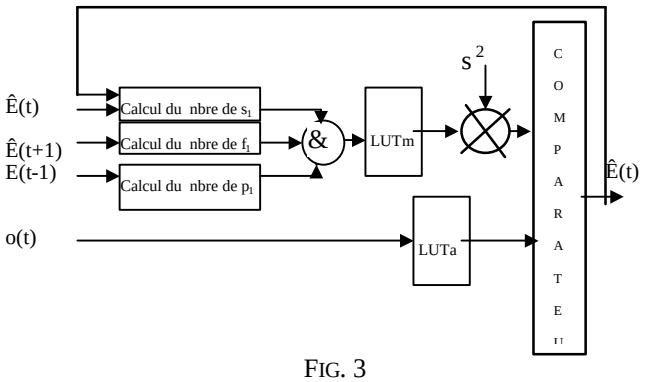

FIG. 3

Cette implantation fait intervenir deux LUT (Look Up Table) dans le calcul des énergies ; la LUTm contient toutes les valeurs possibles de U_{m1} et est adressé avec $s_1\&p_1\&f_1$, et la LUTa contient les différentes valeurs que peut prendre o-10 et est adressé avec la valeur de l'observation o.
le calcul du terme U_{m1} introduit la notion de voisinage spatio-temporel[FIG.2].
L'aspect temporel est vite réglé, car au-delà du stockage préalable des données (E(t-1) et Ê(t+1)), il n'impose aucun ordre ni organisation en mémoire ;
Par contre l'aspect spatial utilisant un voisinage de 3x3 implique un stockage préalable de 3 données dans le même emplacement mémoire.
Le balayage de l'image est un balayage chaîné [FIG.4], ce mode de balayage permet d'accélérer la propagation des contraintes markoviennes et induit le nombre d'itérations minimum avant la convergence, ce nombre a été estimé à 4 [2].

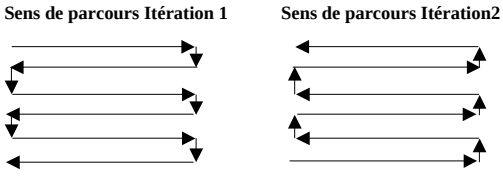

FIG. 4 : balayage chaîné

La mise à jour des sites est « sites récursifs »[1], elle suppose un stokage préalable et une gestion de mémoire assez ardue.
En effet la valeur originelle d'un site B issue de l'étape d'initialisation doit contribuer à son propre traitement de relaxation ainsi qu'au traitement du site A qui le précède d'une ligne. Sa valeur après relaxation doit aussi être utilisée pour le traitement du site D voisin (ce qui est montré par la boucle de retour de FIG.3), ainsi que pour la relaxation du site C qui le suit d'une ligne. Ce mode de fonctionnement impose un stockage préalable en mémoire qui consiste à dupliquer le stockage d'un pixel en trois emplacements

différents et nous ramène à proposer deux solutions quant au fonctionnement des mémoires en vue d'optimiser la rapidité globale des accès.

La première solution consiste à utiliser une seule mémoire en lecture/écriture et donc à faire deux écriture en mémoire pour chaque lecture, ce qui fait trois accès mémoire pour chaque traitement. Ce mode d'utilisation des mémoires est pénalisant, sauf si nous arrivons à masquer les trois accès mémoire par un traitement qui dure au moins trois temps_accès_mémoire.

Avec cette solution aucune reconfiguration de FPGA n'est à prévoir entre les 4 itérations de l'ICM, puisque nous utilisons constamment les mêmes modules de traitement et d'adressage.

La deuxième solution consiste à utiliser en permanence deux mémoires l'une en lecture/écriture que nous utilisons pour le traitement de l'itération présente et l'autre en écriture (en deux emplacement) qui nous servira pour le traitement de l'itération suivante. Le rôle de ces deux mémoires sera inversé lors du prochain traitement. Cette solution suppose une reconfiguration du FPGA pour chaque itération de l'ICM.

Pour masquer le temps d'accès mémoire nous devons avoir un traitement qui dure au moins deux temps_accès_mémoire.

Si nous arrivons à stocker deux données à chaque emplacement, masquer le temps d'accès mémoire revient à traiter deux pixels dans une durée au moins égale à deux temps_accès_mémoire.

En plus de l'amélioration en temps de traitement que peut procurer cette solution, elle donne plus de souplesse quant au choix du nombre d'itérations à faire, ainsi nous pourrons faire moins d'itérations si nous estimons que c'est suffisant vis à vis de la qualité de traitement.

C'est cette derniere solution que nous avons choisi pour l'implantation de l'algorithme sur Ardoise.

4. Implantation de l'algorithme sur l'architecture ARDOISE

L'architecture ARDOISE [6] est composée de trois modules identiques : Un module Uc permet de faire les calculs requis par les algorithmes, et deux modules GTI permettent de découpler l'acquisition et le traitement.

Chaque module comporte un composant FPGA AT40K40 (45Kportes,144*32*4bits de Ram interne, avec un temps de reconfiguration ~1ms) lié à deux mémoires 256*32kbits.

La gestion des configurations est faite par un DSP à travers un contrôleur FPGA.

L'algorithme se décompose en deux parties bien distinctes qui sont l'initialisation et la relaxation.

La première permet de faire le calcul : des observations, de l'estimation des étiquettes et calcul de la variance de l'image t+1. La seconde fait le calcul de l'ICM de l'image t, tout en se servant de l'estimation des étiquettes de l'image t+1 pour fournir à la fin des quatre itérations la carte des objets mobiles.

Nous avons supposé deux reconfigurations de l'UC pour la partie initialisation :

La première correspond au calcul de la différence, valeur absolue, seuillage et moyenne, quant à la deuxième elle nous permet de calculer la variance et de structurer l'organisation des données décrite précédemment et nécessaire à la partie relaxation.

Pour accélérer les calculs de la 1ère et 2ème configurations, nous avons utilisé un parallélisme de donnée d'ordre 4, ce qui nous ramène à un fonctionnement en SPMD.

La partie relaxation est faite au bout de quatre reconfigurations du FPGA, correspondant aux quatre itérations prévues, et est composé d'un module 'adressage chaîné', et d'un module de calcul (FIG.3) dans lequel nous avons utiliser les Rams internes du FPGA comme LUT.

5. Résultats

Le tableau suivant représente les résultats en temps de traitement et de nombre de portes logiques de l'implantation des différentes configurations sur l'AT40K40.

TAB. 1

configuration	Temps d'exécution en ms pour des images 512*512	Nb de portes logiques
Config1 : DiffAbs, seuillage Moyenne	4.36 ms	34928
Config2 : Variance, organisation des données	4.45 ms	43982
ICM (1 itération)	8.12 ms	30 rams internes et 1872 portes logiques

Notons que les temps affichés dans ce tableau reposent sur l'évaluation des temps critiques de chaque configuration.

Tel que présenté dans le TAB.1, si nous considérons un nombre d'itérations de 4 pour l'ICM, et un temps de reconfiguration du FPGA de 1ms nous arriverons à un temps de traitement global de 47ms pour une image 512x512 soit une cadence de 21 images/s .

A partir de là trois solutions s'offrent à nous pour améliorer ce temps :

- Réduire le nombre de configurations sur le FPGA de calcul, en attribuant le calcul de la variance au DSP dédié jusqu'à présent à la gestion des configurations, ou considérer cette variance constante pour l'ensemble de la séquence[1].
- Utiliser le parallélisme pour le traitement de l'ICM comme décrit dans [5], et donc faire un balayage séquentiel de l'image.
- Utiliser le pipeline et avoir constamment dans le composant toutes les configurations nécessaires à notre algorithme.

Les résultats spécifiés dans TAB.2 représentent les temps de traitement obtenus avec quelques réalisations de cet algorithme.

TAB. 2

Laboratoire	architecture	Nbre de processeurs	Taille de l'image	Cadence (image/s)
LIS Grenoble	CNAP	256	128	3
LIS Grenoble	M96002	1	128	15
LIS Paris	C80	1+4	256	18
LIS Paris	Pentium II	1	256	55
LIS Paris	C6203 (estimation)	1	512	37
ETIS (parallélisme dans l'ICM)	Ardoise (AT40K40)	1	512	55
ETIS	Ardoise (AT40K40 + DSP)	1	512	25
ETIS (sans parallélisme)	Ardoise (AT40K40)	1	256	83

En faisant collaborer le DSP dans le calcul de la variance nous arrivons à traiter des images 512x512 en 39ms, et avec un traitement parallèle de deux pixels de la partie ICM nous arrivons à un temps de 18ms/image ce qui fait un traitement de 55 images 512x512 par seconde, ceci nous laisse largement le temps de faire plus d'itérations de l'ICM si nous estimons que le résultat n'est pas satisfaisant avec 4.
La dernière ligne représente les résultats obtenus sur des images de taille 256x256 sans parallélisme et avec un balayage chaîné.

La solution pipelinée n'a pas été envisageable sur l'AT40K40 par manque de ressource de calcul. Mais est parfaitement faisable sur de plus gros composants comme le Virtex xv1000 sur lequel nous arrivons à un traitement de 37 ms pour des images 512x512 avec deux étages de pipeline et un balayage chaîné de l'image.

D'après le TAB.2, les performances obtenues en utilisant un FPGA sont largement satisfaisantes, elles dépassent les performances obtenues sur le Pentium II à 400Mhz et sur le DSPC80 pour des tailles d'images de 256x256, et ce sans utiliser de parallélisme dans la partie ICM.
Avec un parallélisme de deux pixels dans le calcul de l'ICM, nous concurrençons la solution DSPC60.

6. Conclusion

Dans cet article nous avons étudié l'implantation sur FPGA d'un algorithme de détection de mouvement basé sur une modélisation markovienne, la difficulté principale de cet algorithme est liée au choix du mode de balayage de l'image et à la mise à jour des sites, son implantation nous a ramené à utiliser constamment les ram internes du FPGA pour le calcul des énergies.

Les résultats de nos estimations montrent que l'utilisation des FPGA à reconfiguration dynamique pour ce genre d'algorithmes est tout à fait possible et permet d'atteindre des fréquences de calcul supérieures aux réalisation sur Pentium et DSP.

Ainsi le choix du type d'implantation de cet algorithme ne dépend que du contexte système. Si aucune contrainte sur la puissance consommée ou sur l'encombrement n'est imposée une solution à base de processeurs PentiumII est suffisante pour des images 256x256, mais pour des réalisations temps réel embarquables et pour des tailles d'images supérieures, la solution FPGA semble la plus appropriée surtout qu'elle utilise peu de portes logiques ce qui permet d'associer plusieurs algorithmes dans un même composant, alors que plusieurs DSP seraient nécessaires à la réalisation d'un tel système.

Références

[1] P.Lalande, P.Boutemy, Détection de mouvement dans les séquences d'images selon une approche markovienne. Application à la Robotique sous-marine, Thèse de doctorat de l'université de Rennes I, 1990.

[2] Alice Caplier, Modèle markovien de détection de mouvement dans les séquences d'images : Approche spatio-temporelle et mise en œuvre temps réel. Thèse de doctorat de l'Institut National Polytechnique de Grenoble(INPG), Decembre1995.

[3] C.Dumontier, Etude et mise en œuvre temps réel d'un algorithme de détection de mouvement par approche markovienne. Thèse de doctorat de l'Institut National Polytechnique de Grenoble(INPG), Novembre1996.

[4] A.Caplier, F.Luthon, C.Dumontier, Algorithme markovien de détection de mouvement, mise en place 'temps réel'. GRETSI 1995.

[5] Lionel Lacassagne, Détection de mouvement et suivi d'objets en temps réel, Thèse de Doctorat de l'Université Pierre et Marie Curie Juin2000.

[6] L.Kessal, D.Demigny, N.Boudouani, R.Bourguiba Reconfigurable hardware for real time image processing.