Implémentation temps réel sur GPU d'une architecture de rétine biologique à base d'automates cellulaires.

François Devillard[1],Bernard Heit[2],Stéphane Gobron[3]

[1]Université de Nancy - IUT de Saint-Dié des Vosges
11 rue de l'Université, F-88100 Saint-Dié-des-Vosges, France

[2]Université de Nancy - ESSTIN
2 rue Jean Lamour, F-54500 Vandoeuvre-lès-Nancy cedex, France

[3]Laboratoire Mouvement et Perception, Université de Marseille 2
163, avenue de Luminy, F-13288 Marseille cedex 9, France

francois.devillard@iutsd.uhp-nancy.fr, bernard.heit@esstin.uhp-nancy.fr,
stephane.gobron@univmed.fr

Résumé – Cet article présente une implémentation temps réel sur *PC* (*Personal Computer*) d'une modélisation architecturale de la rétine biologique. Nous proposons une modélisation pipelinée à base d'*AC* (*automates cellulaires*). Dans ce travail, la principale innovation est la méthode de calcul basée sur la programmation d'un *PC* en exploitant les ressources calcul des *GPU* (*Graphical Processor Unit*) de la carte graphique en *OpenGL shading language* (*GLSL*). Nous démontrons que la programmation directe des *GPU* permet de réduire d'un facteur supérieur à dix les temps de calcul par rapport à une solution de calcul uniquement menée par la *CPU* (*Central Processor Unit*).

Abstract – This article shows how the architectural modelization of biological retina allows real-time performances performances, on standard widespread computing systems. We propose a pipelined model of artificial retina based on cellular automata (*CA*). In this work, the main innovation is the computing method based on the programming of a personal computer graphical card using *OpenGL shading language* (*GLSL*). The last section demonstrates the efficiency of our model through numerical and graphical results. We lay particular emphasis on the fact that our direct implementation of the *Graphical Processor Unit* (*GPU*) provides computation power more ten times faster than conventional programming.

1 Introduction

Aujourd'hui les connaissances acquises dans les sciences du vivant [5] conduisent à reconsidérer les approches classiques et à s'intéresser aux traitements que nous offrent les systèmes visuels biologiques pour s'en servir de modèles pour concevoir des dispositifs innovants de vision par ordinateur. En nous appuyant sur la neurobiologie, nous avons développé un modèle simplifié d'architecture de la rétine qui tire profit des mécanismes du système visuel. Ses principales caractéristiques sont de faciliter la mise en évidence des principaux indices visuels dans les séquences d'images mais au prix d'un coût de calcul élevé ce qui pose le problème de l'accélération des traitements. La solution proposée répond aux besoins temps réels nécessaires aux applications embarquées où l'on recherche un opérateur homogène, performant, évolutif et s'accommodant aux différentes caractéristiques des images naturelles.

Dans cet article, nous présentons une méthode d'intégration innovante apportant une réponse au traitement d'un volume important d'informations alliant flexibilité et portabilité. Notre travail présente l'implémentation d'une modélisation des couches rétiniennes à base d'automates cellulaires sur calculateur *PC*. Les *GPU* de la carte vidéo sont chargés du calcul des couches cellulaires.

1.1 Positionnement de nos travaux

Le domaine scientifique de la vision biologique apporte, depuis plus de vingt ans, beaucoup d'idées innovantes pour développer des architectures dédiées aux rétines artificielles électroniques associant intimement un capteur d'image et une architecture parallèle de traitement. Un rapport très complet sur les rétines artificielles a été rédigé par A. Moini [8]. Un état de l'art classifiant les rétines artificielles est présenté par A. Pinna [9] où il positionne ses travaux sur une rétine connexionniste.

Nos travaux [4] se différencient par le fait qu'ils ne s'apparentent pas à une rétine artificielle électronique. Ils utilisent des composants grand public performants et en constantes évolutions que sont les *GPU*. Les *GPU* des cartes graphiques de nos ordinateurs proposent des architectures parallèles et programmables, propices aux modifications de programmes entraînant un faible coût de développement et d'équipement.

1.2 Plan de l'article

Cet article est organisé selon le plan suivant :

– La section 2 présente l'organisation des couches cellulaires constituant la rétine et les principaux flux d'informations. Nous mettons en évidence les principales

propriétés reproduites dans notre modélisation. L'algorithme y est présenté, sa complexité est évaluée.
- La section 3 présente l'intégration de l'algorithme sur un calculateur PC assisté de ses GPU. La répartition des phases algorithmiques du traitement est expliquée.
- La section 4 donne les performances obtenues qui sont évaluées en fonction des réglages de paramètre des couches. Les temps de calcul en sont déduits. Enfin les performances sont comparées à la solution non optimisée.
- La dernière partie conclut et donne des perspectives d'implémentation [3].

2 Modélisation de la rétine

La perception visuelle [5] commencent par le traitement des images mises au point sur la rétine [6] et [7]. Plus qu'un simple damier de photo-récepteurs, la rétine comporte plusieurs couches de neurones qui effectuent les premières analyses spatiales et temporelles de l'information. Cette image rétinienne sort au niveau des cellules ganglionnaires dont les axones forment le nerf optique. Cette première analyse de l'information visuelle se fait grâce aux propriétés particulières des cellules rétiniennes et surtout par leur organisation en champs récepteurs [1] modélisés par Marr [6] et Mead [7].

Entre les cônes et les cellules ganglionnaires de la rétine, il existe plusieurs classes de cellules organisées en deux couches fonctionnelles successives qui sont les couches plexiformes externe et interne (respectivement PLE et PLI).

2.1 Architecture de la rétine biologique

Vu du cortex cérébral, le flux d'information transmis par le nerf optique est composé de trois voies spécialisées. Différents types de cellules ganglionnaires définissent ces trois trajets [10] :
- La voie **koniocellulaire (K)** présente de nombreuses caractéristiques qui l'amènent à être sensible à la luminance et à la chrominance.
- La voie **magnocellulaire (M)** réalise un premier traitement essentiel dans la perception du mouvement et l'information scotopique avec peu de redondances.
- La voie **parvocellulaire (P)** effectue un pré-traitement qui extrait les contrastes chromatiques caractérisant les objets.

Ces différentes fonctionnalités sont obtenues par le traitement d'une succession de couches de cellules similaires et de réponses spatio-temporelles variables. De ce fait, seule une modélisation paramétrable d'une couche de cellules permettra de reproduire les principales fonctions de la rétine, tout en conservant l'architecture biologique.

Nos travaux actuels portent exclusivement sur une version achromatique des voies P et M rétiniennes décrites ci-dessus pour des conditions d'éclairement photopique. Les principales propriétés biologiques sont modélisées par un AC qui est constitué de *cellules* dont chacune est influencée par ses voisines immédiates.

2.2 Modélisation des couches cellulaires

Chaque couche rétinienne est obtenue par un maillage carré de cellules et ces dernières sont modélisées par des opérateurs de filtrage au comportement fréquentiel spatio-temporel de type passe-bas [2]. Chaque cellule constitue un opérateur élémentaire soumis à deux composantes de paramétrage :
- **Globale** qui spécialise la couche d'appartenance (cônes, horizontales ...).
- **Locale** qui simule les adaptations rétiniennes des cellules (luminance, contraste ...).

L'opérateur représentant la cellule doit donc être contrôlable individuellement. Les deux paragraphes suivants décrivent successivement les méthodes de filtrage et d'intégration retenues.

2.3 Algorithme de l'opérateur cellulaire

L'opérateur traduisant l'activité d'une cellule rétinienne est un filtre cascadant l'action d'un filtre spatial à *Réponse Impulsionnelle Finie* (RIF) et d'un filtre temporel à *Réponse Impulsionnelle Infinie* (RII). La description du traitement cellulaire, pour simplifier le formalisme, est donnée pour une dimension spatiale discrètisée x. Le temps est représenté par l'indice discrètisé k. Les dimensions sont à échelles unitaires. Les équations 2 et 3 traduisent la réponse d'une cellule à la position x à l'instant k. Les signaux d'entrée et de sortie de la cellule sont désignés par $e_{x,k}$ et $s_{x,k}$ dans les équations 1 et 4 .

L'équation 1 échantillonne l'entrée e. Les équations 2 et 3 réalisent respectivement un filtrage de type RIF puis RII. Le filtrage spatial RIF de l'équation 2 utilise trois coefficients de somme unitaire soit $[\alpha_{x,k}\,;1-2\cdot\alpha_{x,k}\,;\alpha_{x,k}]$ pour notre représentation monodimensionnelle. Le coefficients local $\alpha_{x,k}$ traduit le couplage entre la cellule x et ses deux voisines immédiates en $x-1$ et $x+1$. Il est réglé par défaut à 0.25 et peut génèrer ainsi une réponse spatiale pseudo-gaussienne proche de la modélisation rétinienne de Marr [6]. Ce noyau de filtrage situé à la position x est appliqué $n_{x,k}$ fois pour simuler l'étalement et par la même la zone d'influence (de rayon de $n_{x,k}$) sensibilisant les connexions dendritiques de la cellule.

$$e'_{x,k} \leftarrow e_{x,k} \tag{1}$$

$$e'_{x,k} \leftarrow (1-2\cdot\alpha_{x,k})\cdot e'_{x,k} + \alpha_{x,k}\cdot(e'_{x-1,k}+e'_{x+1,k}) \tag{2}$$

L'équation 3 représente la composante temporelle de filtrage RII de chaque cellule. Elle est paramétrée par le coefficient local $r_{x,k}$. r simule l'effet intégrateur de la réponse provenant des effets de résistance et capacité membranaire [10]. La valeur de r varie dans l'intervalle $[0,1[$ pour stabiliser la réponse du filtre.

$$s'_{x,k} \leftarrow (1-r_{x,k})\cdot e'_{x,k} + r_{x,k}\cdot s'_{x,k-1} \tag{3}$$

$$s_{x,k} \leftarrow s'_{x,k} \tag{4}$$

La couche traduite en deux dimensions spatiales reprend ce principe de filtrage dans un maillage cellulaire carré et adapte par conséquent les coefficients du filtre RIF. Le filtre RII conserve la même équation étendue à y comme seconde dimension spatiale. Le filtre RIF utilise un masque à cinq coefficients ; un central et quatre

autres égaux entre-eux placés verticalement et horizontalement dans le voisinage immédiat autour du central. Leur somme est toujours unitaire afin d'obtenir un gain en basse-fréquence égal à un.

En considérant chaque cellule comme un opérateur indépendant, la complexité de calcul est fonction de la dimension des couches cellulaires utilisées (la taille d'image). Les couches de régularisation spatiale et les couches de basse résolution (voie M) nécessitent des champs de récepteur très étalés et par conséquent une valeur de n importante. le temps de calcul évolue comme une fonction de n. La solution classique sur ordinateur PC ne suffit plus comme le montre notre article [3].

2.4 Conception de l'AC

Quelle que soit la complexité d'un phénomène, il résulte toujours d'une somme de sous-phénomènes à comportements limités. Les AC proposent une réciproque à cette hypothèse [12]. Chaque cellule d'automate représentant une cellule rétinienne est régie par les quatres règles de fonctionnement suivantes, elle-mêmes définies par leur état et conditions d'activation :

Règle 1. Acquisition de e (équation 1) - pour $i = 0$

Règle 2. Calcul filtre RIF(équation 2) - pour $i \in [1, n_{x,k}[$

Règle 3. Calcul filtre RII(équation 3) - pour $i = n_{x,k}$

Règle 4. Synchronisation de l'automate et Echantillonnage de la sortie s (équation 4) - pour $i > n_{x,k}$

L'AC est initialisé par une image e à l'entrée (pour $i = 0$). Il parcourt les quatres états en autant d'itérations qu'il faudra pour que toutes ses cellules atteignent l'étape 4 (synchronisation de toutes les cellules). La variable i comptabilise les itérations réalisées. Lorsque l'AC est synchronisé l'image s est disponible et une nouvelle image peut-être à nouveau traitée (définissant le cycle vidéo et celui de l'AC).

3 Implémentation

La figure 1 décrit l'environnement utilisé pour nos tests et détaille la répartition et les phases de traitements sur les différents opérateurs. Puis, la figure 2 présente quelques résultats obtenus en sortie de rétine.

3.1 Environnement

Le résultats sont obtenus avec un PC équipé de :
- Une CPU de type AMD Athlon^{TM} $64X2$ Dual Core 4400+ - 2.21 GHz.
- Une mémoire vive de 2 Go.
- Une carte vidéo NVidia^{TM} GeForce 7800.
- Le système d'exploitation MS-Windows^{TM} 2000.
- Et la suite logicielle $C++$ and $GLSL$ languages de MS-Visual Studio^{TM} 2005.

Par ailleurs pour l'entrée vidéo, nous utilisons une webcam de type $CREATIVE^{TM}$ model N^o : $PD1030$ (640×480 pixels/image à 15 images/s) - Port $USB11$.

Les images acquises par la webcam sont chargées par interruptions. Les images de type *bitmap* sont décrites et transmises à la carte graphique sous forme de textures par OpenGL^{TM} 1.4 shading language [11]. La CPU initialise et gère les cycles de l'AC. Plus de détails sur la réalisation sont disponibles dans la revue [4].

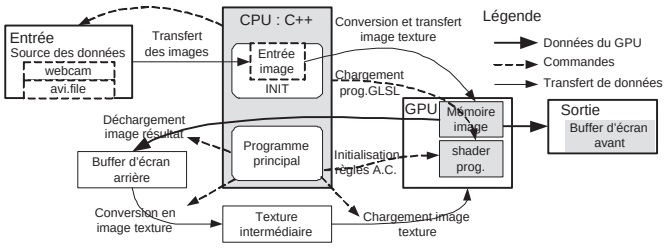

FIG. 1 – Schéma de principe de l'application.

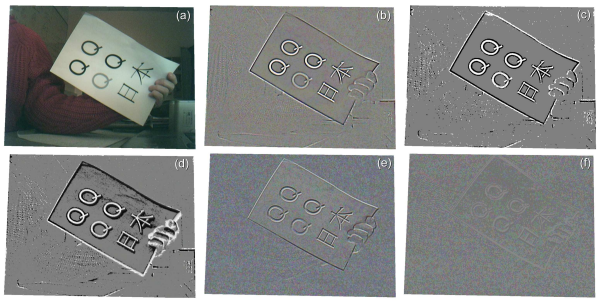

FIG. 2 – Résultats temps réels - Captures d'écran. (a) Image webcam, (b) sortie PLE, (c) PLE accentuées, (d) amacrines et (e) et (f) sorties PLI avec et sans mouvement.

4 Résultats

Dans notre environnement, l'ajout d'instructions de mesure perturbe le cycle d'exécution et fausse la précision pour la mesure de temps courts ($< 1ms$). Nous avons donc relevé tous les temps intermédiaires de l'algorithme pour différentes conditions expérimentales. De ces résultats, nous estimons le temps de calcul du cycle de l'AC puis celui de la CPU servant à la gestion des commandes et des échanges de données avec les GPU.

4.1 Temps de calcul de la PLE

L'algorithme est testé pour une voie rétinienne complète (simulation de la voie P). La fréquence de la webcam est d'environ 15 images/s. Les images sont de type VGA (640×480 pixels en luminance). La figure 3 donne les temps d'exécution de la PLE en fonction de $n_c + n_h$. Ces coefficients (voir section 2) ont une influence fonction de n sur le temps d'exécution.

On a matérialisé à titre indicatif en pointillés rouges sur la figure 3 les constantes $67ms$ (période de la webcam) et $20ms$ (période de trame vidéo standard). Le traitement de notre algorithme est plus rapide que la webcam utilisée. Si on compare les deux courbes en bleu continu, on constate pour n donné, une marge de progression encore possible en utilisant une vidéo plus rapide. Théoriquement jusqu'à n égal à 40, notre système supporte la cadence d'une trame

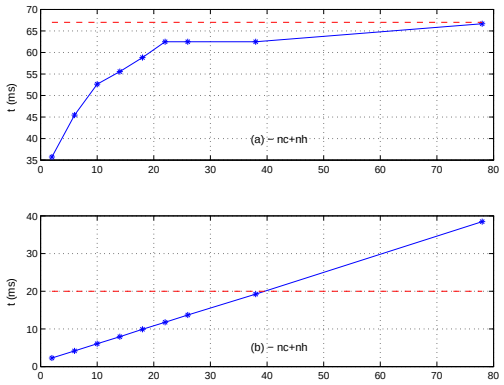

FIG. 3 – Temps d'exécution de la PLE. (a) Entrée webcam. (b) Entrée fichier type AVI.

standard (2 trames/image). Des simulations sans GPU ont été menées dans les mêmes conditions. La PLE (pour $n_c = n_h = 1$) appliquée sur la webcam avec GPU est calculée en $36ms$, le temps est décuplé sans. Le rapport s'accroît encore avec l'augmentation de n.

A noter que sans revoir notre conception, le passage à des images plus importantes est possible. Leur taille se limite à celle des buffers de la carte graphique (4096×4096 pixels RVB). Pour une taille N_{xy} pixels, le temps de traitement évolue comme une fonction de N_{xy}.

4.2 Temps d'exécution du cycle de l'AC

Les mesures faites permettent de calculer les temps d'exécution de chaque couche. En divisant leur temps d'exécution par leur nombre respectif d'itérations n, on observe que chaque cycle de l'AC est composé :

- D'un lapse de temps incompressible du aux contrôles et échanges entre la CPU et GPU.
- Du temps de calcul des n itérations par les GPU.

Pour n important, la limite du rapport temps d'exécution sur n tend vers le temps du calcul d'une seule itération. On obtient environ 460 μs/itération. Pour le traitement de fichiers AVI, on approche le temps d'exécution T de la PLE ou d'une couche par la relation :
$T(ms) = 0.46 \cdot n + 1.35$
Pour le flux webcam, la valeur $1.35ms$ est remplacée par $8.25ms$ (augmentation de la charge CPU). Les GPU conservent le même temps de cycle d'AC.

5 Conclusions et perspectives

Nous avons présenté une solution originale pour implémenter un algorithme modélisant une architecture fonctionnelle de la rétine biologique. La solution utilisée est un PC standard équipé d'une carte vidéo graphique équipée de puissants GPU pour accélérer les affichages $2D$ et $3D$. Le choix des AC se prête bien à l'intégration de réseaux cellulaires tels que des couches rétiniennes. Les connexions cellulaires en maillage carré se prêtent bien respectivement à l'architecture de la carte graphique et les opérations cellulaires simples aux jeux d'instructions (limités) des GPU. Nous atteignons actuellement des performances permettant d'envisager le calcul d'une voie en temps réel vidéo.

D'un point de vue modélisation, l'organisation cellulaire permet une évolution adaptative de la rétine implémentée [3] où le paramétrage individuel des cellules est modifiable en fonction de la position de la cellule ou de la nature de la scène traitée. Une séparation des chrominance est aisément réalisable à court terme. L'utilisation des AC nous amène à pouvoir simuler de nombreux comportements identifiés initialement par S. Wolfram [12] et apporte ainsi une solution méthodologique puissante et originale. De plus ils permettent une implémentation efficiente sur les cartes graphiques. L'augmentation des performances des calculateurs et l'utilisation des ressources matérielles et logicielles des cartes graphiques nous laisse penser que dans un avenir proche nous pourrons, à partir d'un calculateur de type PC, intégrer des architectures complexes rétiniennes d'inspiration biologique.

Références

[1] E.H. Adelson. *New cognitive neurosciences.* MIT Press - M.S. Gazzaniga, 2nd edition, 2000.

[2] W.H.A Beaudot. *Le traitement neuronal de l'information dans la rétine des vertébrés - Un creuset d'idées pour la vision artificielle.* PhD thesis, Institut National Polytechnique de Grenoble, France, 1994. 228 pages.

[3] F. Devillard, S. Gobron, F. Grandidier, and B. Heit. Implémentation par automates cellulaires d'une modélisation architecturale de rétine biologique. In *READ'05.* Institut National des Télécommunications, Evry, France, 1-3 Juin 2005.

[4] S. Gobron, F. Devillard, and B. Heit. Retina simulation using cellular automata and gpu programming. *Machine Vision and Applications - 0932-8092 ISSN*, Janvier 2007.

[5] J-M. Jolion. Les systèmes de vision. *Hermès Science Publications*, 2001.

[6] David Marr. *Vision.* W. H. Freeman and Company, New York, 1982.

[7] C.E. Mead and M.A. Mahowald. A silicon model of early visual processing. *Neural Networks*, 1(1) :91–97, 1988.

[8] A. Moini. Vision chips or seeing silicon. Technical report, Adelaide University, March 1997.

[9] A. Pinna. *Conception d'une rétine connexionniste : du capteur au système de vision sur puce.* PhD thesis, Université Pierre et Marie Curie - Paris VI, France, 2003.

[10] J.-F. Risse. *Exploration de la fonction visuelle - Applications au domaine sensoriel de l'oeil normal et en pathologie.* Masson, Société Française d'Ophtalmologie, 1999.

[11] R.J. Rost. *OpenGL Shading Language, 2nd Edition.* Addison Wesley Professional, 2006.

[12] S. Wolfram. *A new kind of science.* Wolfram Media Inc., 1st edition, 2002.