Implémentation par automates cellulaires d'une modélisation architecturale de rétine biologique

F. DEVILLARD[1], S. GOBRON[2], F. GRANDIDIER[3] ET B. HEIT[3]

[1]Centre de Recherche en Automatique de Nancy UMR CNRS 7039 (CRAN - CNRS)
11, Rue de l'université, 88100 SAINT-DIE DES VOSGES, FRANCE
Tél.:+33(0)3.29.53.60.01, Fax: +33 (0)3.29.53.60.11, francois.devillard@iutsd.uhp-nancy.fr

[2]Equipe MoTrICE, Université Henri Poincaré, IUT St-Dié,
11, Rue de l'université, 88100 SAINT-DIE DES VOSGES, FRANCE
Tél.:+33(0)3.29.53.60.67, stephane.gobron@iutsd.uhp-nancy.fr

[3]Centre de Recherche en Automatique de Nancy UMR 7039 (CRAN - CNRS)
2, Rue Jean Lamour, 54500 VANDOEUVRE-LES-NANCY, FRANCE

***Résumé* - Cet article traite de l'implémentation sur calculateur classique d'une modélisation cellulaire de la rétine biologique via deux modèles d'automates cellulaires (2D et 3D). Les algorithmes utilisés dans cet objectif, pénalisants en temps de calcul, nécessitent la plupart du temps une architecture de traitement spécifique et par conséquent, une adaptation de l'algorithme. Notre solution alternative utilise les fonctionnalités de composants logiciels de synthèse d'images enfouissables en partie dans la carte graphique pour permettre la parallélisation des traitements cellulaires.**

Mots Clés - Rétine artificielle, adéquation algorithme architecture, traitement d'images, automates cellulaires, réseaux de cellules interactives.

1 Introduction

La rétine fovéale est constituée d'un *empilement* de couches de cellules traitant le signal lumineux pour en extraire des indices visuels (contrastes de chrominance statique et dynamique, luminance...). Dans l'objectif de modéliser son fonctionnement, une couche élémentaire de cellules nerveuses rétiniennes a été définie et sert de base à la conception d'une architecture de rétine utilisable en vision artificielle. Cette couche élémentaire est modélisée par des outils de traitement du signal et permet de reproduire une approche comportementale de la couche rétinienne biologique. Elle est composée d'un réseau de cellules de traitement paramétrables localement pour reproduire les mécanismes d'adaptation biologiques observables. L'assemblage des différentes couches de la rétine doit satisfaire à des contraintes d'implémentation temps réel dans le cadre d'applications de vision artificielle ; l'algorithme correspondant est décrit dans la section 2. Des précisions sont apportées pour une implémentation plus efficace sur des calculateurs programmables. La section 3 présente la méthode logicielle à base d'automates cellulaires utilisée pour la programmation de l'algorithme sous contrainte d'un flux de données temps réel provenant d'une source vidéo (*WebCam*). Cette solution nous apporte de nouvelles possibilités de traitement qui complètent l'algorithme original. De plus, une proposition d'adaptation de l'algorithme est présentée pour enfouir des traitements dans la carte graphique. La section 4 donne les premiers résultats obtenus : images et temps de traitement. Enfin la dernière section conclut et donne les perpectives envisageables à court terme.

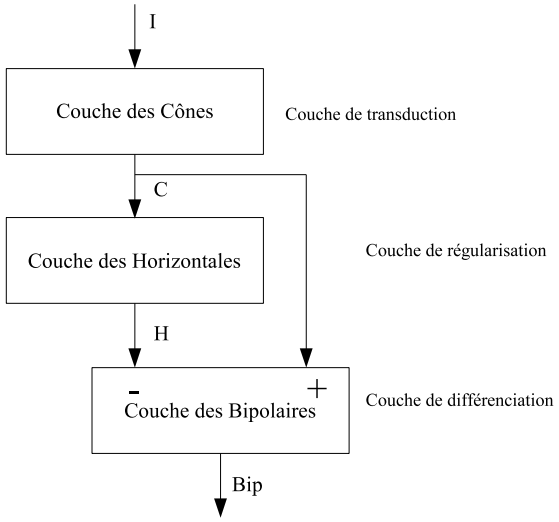

Figure 1: Schéma de l'architecture de la couche *PLE* de la rétine biologique photopique (vision diurne).

2 Architecture de la rétine

Les mécanismes de la perception visuelle [7] réduits à la couche plexiforme externe (*PLE*) du système parvocellulaire (zone fovéale de la rétine) sont schématisés dans la figure 1. L'acquisition de la scène observée est réalisée par une matrice de photorécepteurs (cellules de type cône) qui transforment l'énergie lumineuse I en énergie électrochimique C. Deux autres couches de cellules effectuent à la suite une première analyse du message par le rehaussement des contrastes présents dans l'image. La couche des cellules horizontale est une couche de régularisation estimant localement un référentiel de luminance H de la scène. La couche des cellules bipolaire différencie la réponse des cellules cône et horizontale pour fournir une réponse *Bip* mettant en évidence les contrastes caractérisant les objets observés.

Chaque couche est constituée d'un maillage plus ou moins diffus de cellules interagissant latéralement. Chaque cellule fournit une réponse, fonction d'une somme pondérée et non linéaire de ses entrées synaptiques. De plus, la réponse est une combinaison de filtrages passe-bas spatial et temporel. Par conséquent, chaque cellule peut être modélisée par un processeur ou filtre élémentaire à entrées multiples (les synapses cellulaires) et dont la sortie (l'activité cellulaire) est donné par un filtre passe-bas spatio-temporel [1].

L'association de filtres élémentaires travaillant en maillage permet de construire une couche de traitement cellulaire aux comportements comparables à celui d'une couche de la rétine et facilement applicable aux automates cellulaires (voir section 3).

2.1 Modélisation des couches rétiniennes

La mise en oeuvre de notre solution originale repose sur un filtrage spatio-temporel numérique distribué et adaptatif. La fonction de sortie de la cellule $s_{x,k}$ est donnée dans l'équation 1 (pour une seule dimension spatiale x). Elle traduit l'activité cellulaire pour une entrée $e_{x,k}$ à la position x et à l'instant k. Les périodes d'échantillonnage spatiale et temporelle sont unitaires pour simplifier la formulation.

$$s_{x,k} = (1-r_{x,k}) \cdot \left[b_{x,k} \sum_{i=-\lambda}^{\lambda} \frac{C_{i+\lambda}^{2\lambda}}{4^{\lambda}} \cdot e_{x+i,k} \right] + r_{x,k} \cdot s_{x,k-1} \quad (1)$$

avec λ représentant $n_{x,k}$ et $0 \leq r_{\lambda} < 1$.

Ce filtre linéaire et séparable est composé de structures de type *RIF* (Réponse Impulsionnelle Finie) et *RII* (Réponse Impulsionnelle Infinie) appliquées respectivement dans les domaines spatial et temporel. Les paramètres influençant la réponse spatiale de la cellule sont n et b (respectivement le couplage latéral entre cellules et le facteur de gain de cellule). Le coefficient b agit sur l'excitation de la cellule x comme un gain et le coefficient n traduit l'influence de la cellule x sur ses voisines –paramètre d'étalement fixant le diamètre de l'arborescence dendritique à $2n+1$.

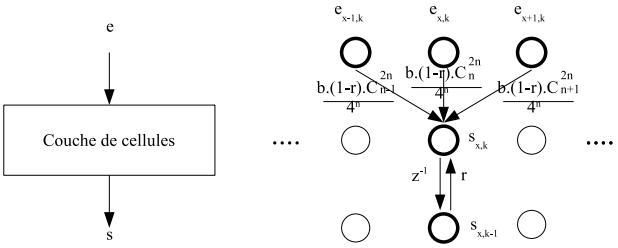

Figure 2: Graphe de fluence du traitement du processeur élémentaire équivalent à une cellule à synapses positionnée en x à l'instant k. Seules sont représentées les connexions avec le proche voisinage cellulaire. L'opérateur z^{-1} représente un retard pur et C la fonction statistique de combinaison.

L'intensité de cette influence, fonction de la distance, est obtenue à l'aide d'une fonction binomiale de manière à fournir une réponse impulsionnelle passe-bas, centrée et pseudo gaussienne [11]. Le paramètre influençant la réponse temporelle de la cellule est r. Il permet de reproduire l'effet intégrateur d'une constante de temps sur la sortie de la cellule (due à la résistance et à la capacité membranaire de la cellule) [12]. La figure 2 représente l'équation 1 graphiquement sous la forme d'un maillage.

La couche de cellules cône prend en entrée la succession d'images formant la séquence vidéo (1 pixel = 1 cône). Les cônes dans notre version d'algorithme sont considérés achromatiques. L'équation 1 régit le fonctionnement des couches de cellules cône et horizontale avec des paramétrages différents (nommés respectivement n_c et n_h). La couche de cellules bipolaire utilise également ce modèle mais prend en entrée la différence entre les couches cône et horizontale.

2.2 Les adaptations de l'algorithme

La mise en oeuvre de l'équation 1 nécessite une adaptation algorithmique et/ou architecturale pour obtenir une rétine optimisée en temps de calcul. En effet, la composante spatiale du filtrage est le principal consommateur de ressources calcul. L'équation 1 étendue à deux dimensions conduit naturellement à un calcul de convolution de masque de taille $(2n+1)^2$. D'un point de vue algorithmique cette opération peut être facilement optimisée par une décomposition en cascade de convolutions élémentaires (masque $b[3 \times 3]$ de l'équation 2). Par cette méthode, nous passons d'un coût $(2n+1)^2$ à $9n$ d'opérations de type multiplication-accumulation. D'un point de vue architectural, cette solution permet également de pallier à la limitation de l'architecture des processeurs du *GPU* qui n'optimise les opérations que dans un voisinage restreint autour du pixel traité. Comme l'indique le paragraphe précédent, ce filtre est séparable en deux composantes : une temporelle et une spatiale. Une structure pipelinée de filtrage

est possible. Cette optimisation accélère le temps d'exécution avec l'exploitation d'un cache instructions limité de processeur (*DSP*, *GPU*...).

$$b = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad c = \begin{bmatrix} 0 & a & 0 \\ a & 1-4a & a \\ 0 & a & 0 \end{bmatrix} \qquad (2)$$

L'utilisation du masque b nécessite une connexité de huit cellules. Les cellules des automates de synthèse d'images choisies pour la mise en oeuvre et décrites dans le paragraphe suivant opérent avec quatre connexités. Le masque bidimensionnel élémentaire b de l'équation 2, réponse impulsionnelle de l'équation 1 avec $n = 1$, $b = 1$ et $r = 0$, est remplacé par le masque c. Le gain de temps est important (quatre opérations Multiplication-Accumulation de moins avec le passage de b à c) et ceci sans dégrader les caractéristiques spectrales. L'écart quadratique entre les masques b et c est minimal pour $a = 0,175$. Dans ces conditions, la figure 3 montre la similitude des résultats obtenus lorsque n augmente.

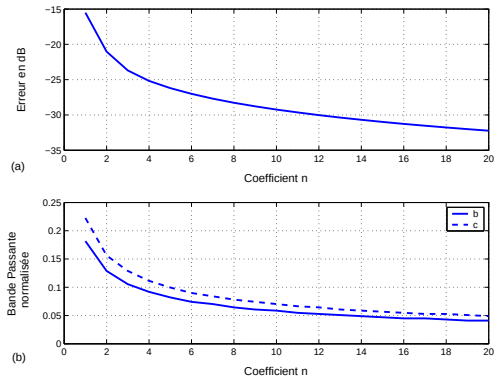

Figure 3: (a) Erreur sur la réponse impulsionnelle en dB et (b) Bande passante des masques en fréquence à $-3dB$ en fonction de n.

3 Implémentation

3.1 Cellules et automates cellulaires

Quelle que soit la complexité d'un phénomène, il résulte toujours d'une somme de sous phénomènes à comportement limités. Les automates cellulaires proposent une réciproque à cette hypothèse : soit un ensemble d'entitées appelées cellules ayant à la fois des propriétés géométriques communes et un réseaux entre cellules voisines, un changement d'état (liste finie de règles) est appliqué suivant une synchronisation commune.

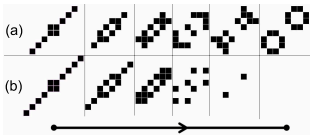

Figure 4: Exemple d'automate cellulaire booléen 2D connu sous le nom du *jeu de la vie*.

Pour illustrer cette définition, la figure 4 propose deux changements d'état d'un automate cellulaire booléen à deux dimensions dit du *jeu de la vie* proposé par Conwey [2] dès 1970 et illustré en synthèse d'images par Thalmann [15]. On peut observer qu'à partir d'hypothèses similaires tout en utilisant des règles identiques de changement d'état selon le voisinage direct, ce système converge vers des solutions radicalement différentes. Il est souligner que les automates cellulaires ne convergent pas systématiquement. Pour simuler tel ou tel comportement, le programmeur doit donc prendre soins d'étudier en profondeur cette problématique non triviale. Langton [9] a démontré que les automates présentent quatre états généraux : convergent, cyclique, cyclique asynchrone et chaotique.

Dans le même ordre d'idée, voici une liste non exhaustive de références couvrant à la fois le formalisme et les domaines d'applications que nous utiliserons dans le cadre de cette simulation :

Les notions de grammaire, formalisme informatique et entropie des automates cellulaires sont respectivement présentés par Rosenfied [13], Langton [9] et Kari [8]. Une approche couvrant les thèmes informatiques, physiques et mathématiques a été éditée par Wolfram [19]. En ce qui concerne l'analyse et la synthèse d'images, nous proposons les lectures suivantes : Turk [16] et Witkin [18] pour les problèmes relatifs au phénomènes de réaction/diffusion, ainsi que Fleischer [3]. Plus spécifiquement, les modèles présentés dans [5, 4] proposent l'étude des modèles surfaciques de phénomènes naturels appliqués à l'informatique graphique par automate cellulaire respectivement simple, hybride et généralisé.

En partant de l'hypothèse exposée en début de section, nous proposons de simuler certains comportements des cellules rétiniennes et en particulier le couplage cône/horizontale de la rétine en se basant sur des automates cellulaires. Pour mener à bien cette simulation –dont la source d'information est un flux provenant d'une *WebCam*– deux approches sont exposées sections 3.2 et 3.3 distinctes : la simulation directe en 2D, et l'approche d'un modèle de rétine naturelle tridimensionnelle.

3.2 Modèle 2D régulier : automates simples

Dans le cadre d'échanges d'informations dans une série d'images temps réel, la programmation d'automates cellulaires est facilement réalisable puisque l'adressage des voisins les plus proches est directe et que les règles simulant les filtres passe-bas sont triviales. En l'occurrence, on applique pour chaque cellule :

$$\forall C_{x,y} | C_{(t+1)} \leftarrow \; p \cdot C_{(t)} + (1-p) \cdot \sum_{i=0}^{n} C_{i(t)} \qquad (3)$$

avec p comme coefficient de lissage et C_i l'ensemble des deux, trois ou quatre cellules voisines.

Cette formule est appliquée en cascade pour le calcul du filtrage des cellules cône (n_c fois) puis des cellules horizontale (n_h fois) suivant les règles présentées section 2. La section 4.2 présente quelques exemple de résultats appliqués à différents automates (figure 7) et filtres/résolution (figure 8).

3.3 Réseaux cellulaires 3D irrégulier

Cette section décrit une première ébauche de simulation de rétine naturelle tridimensionnelle permettant à la fois une répartition des cellules et un échange d'information entre cellules plus réalistes. Dans cette optique nous nous basons en grande partie sur une récente généralisation de modèle d'automates cellulaires dynamiques [4] permettant l'interconnexion et la communication multiple entre des cellules non uniformément réparties dans l'espace.

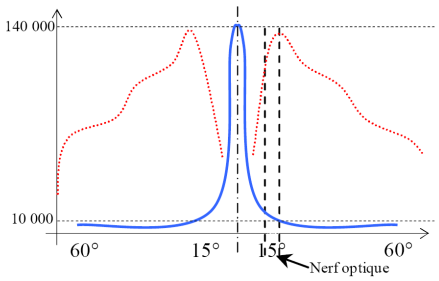

Figure 5: Approximation de la densité des photorécepteurs par mm^2.

Comme le montrent les travaux de Schreiber [17] la densité des photorécepteurs n'est pas répartie uniformément. En particulier, la répartition des cellules cône suit la courbe continue illustrée figure 5. On observe une concentration environ vingt fois plus élevée dans une région inférieure à 15 degrés autour de l'axe optique. Pour simuler ce phénomène, nous proposons une méthode basée sur des tirages aléatoires suivant la densité variant selon la position sur la sphère rétinienne. Pour faciliter les calculs on se place dans un repère sphérique (ρ, θ, ϕ) en utilisant la probabilité de sélection $P_{\theta,\phi}$ suivante :
si (θ ou $\phi < \frac{\pi}{12}$) **alors** $P_{\theta,\phi} = C_m$
sinon

$$P_{\theta,\phi} = \frac{(C_M - C_m).cos^2(12\theta).cos^2(12\phi) + C_m}{C_M} \quad (4)$$

avec le rayon approximatif de l'oeil, $\rho = 1,25cm$ et $C_{M,m}$ les concentrations maximum et minimum.

La figure 6 présente les résultats de cette équation pour un échantillonnage de 10^5 cellules. Les différences de concentration résultantes sont illustrées dans la section 4.3, figure 9.

Nous avons défini les positions spatiales des cellules, à présent il faut vérifier si pour toute position sur la rétine il est possible d'associer l'adresse

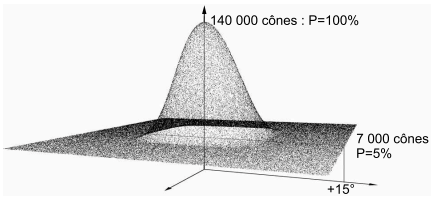

Figure 6: Concentration des cellules de la rétine pour un angle inférieur à 15 degrés (zone fovéale).

d'un pixel de la *WebCam*. Idéalement, il faudrait calculer le chemin parcouru par la lumière au travers de la cornée, l'humeur aqueuse, le cristallin et le corps vitré. Dans le modèle actuel nous proposons une simple projection indirecte résultant par :

$$WC_y = \frac{I_L}{2}(1 - \frac{12}{\pi}\phi), WC_x = \frac{I_H}{2} - \frac{6I_L}{\pi}\theta \quad (5)$$

avec $WC_{x,y}$ l'ensemble des pixels du flux vidéo de coordonnée (x, y), et $I_{L,H}$ l'une des trois largeur×hauteur de l'image (*c.f.* 160×120, 320×240 ou 640×480).

3.4 Gestion des automates par GPU

Depuis quelques années les besoins dans le monde du graphisme et principalement les industries du jeu vidéo, ont poussé les constructeurs à élaborer des cartes graphiques non seulement très puissantes mais programmables. De plus, la puissance de calcul cumulée des *GPU* est nettement supérieure à celle du *CPU*. L'idée est donc d'utiliser les cartes graphiques à des opérations autres que celles dédiées directement au graphisme. Nous travaillons actuellement sur le codage *GPU* de rétines artificielles sur une carte NVIDIA^{TM} de type *GeForce 6800GT*.

A court terme, nous souhaitons associer à une grille de pixels de la carte graphique une texture GL, et d'y associer les règles d'automates générant les lissages des cellules cône et horizontale. Ces opérations sont directement programmable par exemple avec *Pixel-Shader*^{TM}.

Dans un second temps, nous étudions un modèle plus complexe de calcul des échanges d'information entre cellules *naturelles* une fois encore avec la programmation de la carte graphique. L'irrégularité de la répartition spatiale des photorécepteur engendre un problème majeur : comment accéder au voisinage ? La solution que nous envisageons est d'associer à chaque pixel des adresses sur une série de champs d'adresses correspondant à son voisinage. Dans ce cas, et compte tenu des restrictions matérielles actuelles, chaque cellule ne peut avoir qu'un nombre restreint de voisins (huit au maximum). Nous travaillons actuellement sur l'élaboration d'un modèle permettant de remédier à ce problème.

4 Résultats

4.1 Moyens Techniques

Les deux logiciels d'automates cellulaires ont été développés en C++ sous Microsoft Windows 2000 avec MSVC++.NET. Le graphisme est géré par *vfw32.lib* pour les accès vidéo, *OpenGL* [14] pour la gestion graphique de base et Shader Designer^{TM} pour les accès à la carte graphique et au futur *GPU* [6] [10].

En ce qui concerne la configuration Hardware, nous avons utilisé un PC muni d'un processeur AMD Athlon 2600, d'une carte graphique GeForce FX 6800 GT, et 512 Mo de RAM. Les essais ont porté sur deux *WebCam* Logitech^{TM} et Creative^{TM} USB 1.1 –dont les performances sont finalement très similaires.

4.2 Application du flux direct

Figure 7: Différents types d'automates appliqués au traitement d'un flux temps réel d'images.

Un automate cellulaire de base se constitue d'un tableau à deux dimensions. Nous avons donc associé le pointeur de sortie *WebCam* à une texture OpenGL après traitement (voir section 3).

La figure 7 présente quatre traitements temps réel basés sur les automates cellulaires. La figure (a) illustre le résultat d'une couche de cellules cône $r = 70\%$. (b) propose une solution de décomposition de l'image couleur avec six paliers. Il suffit de diviser la somme des intensités *RVB* par 128 (*c.f.* $3.\frac{MAX_{RVB}}{6}$)) et d'y associer une palette de six couleurs. (c) montre les résultats d'un automate de détection de mouvement –soustraction [image$_{(t)}$ - image$_{(t-1)}$] et remise en paliers. Et enfin (d) présente les valeurs de sortie des cellules bipolaire. Dans cette dernière image nous avons volontairement augmenté le contraste pour visualiser les artéfacts dus à la compression *jpeg* de la caméra.

La figure 8 propose cinq filtres (lignes) effectués sur les trois séries (colonnes) de résolution vidéo. Ces tests ont été effectués avec des animations temps-réel, dont les différents résultats en *images par seconde* sont présentés table 1. Il est à noter que toute la charge de calcul a été dédiée à un processeur relativement mal adapté à ces opérations (par rapport au possibilité des *GPU*) et que le port USB était de faible débit (version 1.1). Selon des configurations similaires, la programmation *GPU* permettrait une augmentation moyenne de dix-sept fois de la vitesse des calculs (voir [6] et [10]).

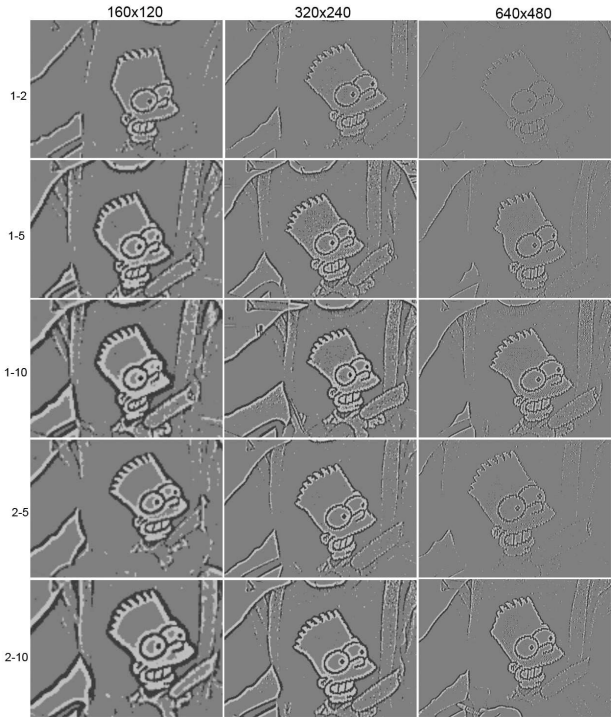

Figure 8: Variation des sorties bipolaires selon la résolution et les paramètres n_c et n_h.

$\mathbf{n_c}$ - $\mathbf{n_h}$	(160x120)	(320x240)	(640x480)
1 - 2	10,4	5,7	2,1
1 - 5	10,3	5,5	<2
1 - 10	9,8	4,1	<2
2 - 5	10,3	5,4	<2
2 - 10	9,1	4,0	<2

Table 1: Performances en images par seconde du traitement illustré figure 8.

La table 1 donne les valeurs des rendus en images par seconde de la figure 8. La notation inférieure (<) indique que les valeurs exactes ne peuvent pas être mesurées simplement, la notion de *temps réel* n'existe plus.

4.3 Résultats de la simulation non planaire

La figure 10 présente nos premiers résultats de simulation de rétine *naturelle* en 3D rendu temps-réel. (a) propose –avec une échelle de 1 cm par case– la sphère rétinienne ainsi que les voxels [4] englobant les cellules irrégulièrement réparties. La densité est de $7 \cdot 10^2$ à $1,5 \cdot 10^4$ cellules par mm^2 soit environ dix fois moins qu'une rétine biologique. Les voxels

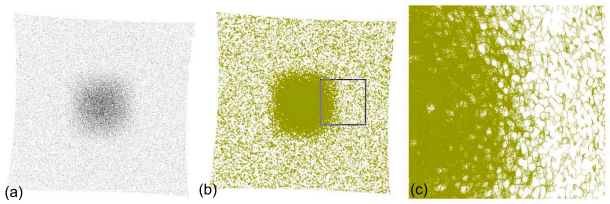

Figure 9: Répartition et connexions des cellules sur une partie (inférieur à 15 degrés par rapport à l'axe de la sphère rétinienne) : (a) cellules initiales ; (b) vue d'ensemble des connexions ; (c) cadrage entre 5 et 10 degrés

sont représentés en vue transparente en (b). La figure suivante (c) montre la distribution des cellules avec ϕ et θ inférieure à 15 degrés. (d) illustre les différentes couches de connexions entre cellules voisines ; on utilise ici une connectivité par sphère englobante de 3.10^{-5} m de rayon).

Après activation progressive des cellules de l'automate, l'image se recompose en (e) à l'envers (fond de l'oeil). La dernière image (f) montre qu'il est possible de composer en parallèle différentes règles (ici un automate de digression/fusion [16] [4] des cellules sous contrainte est simulé) sans altérer les temps de calcul.

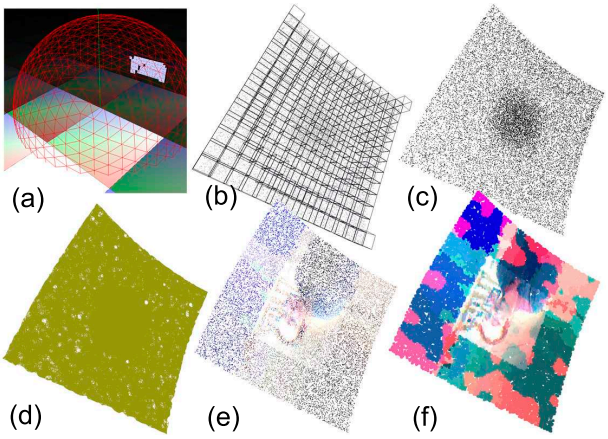

Figure 10: Ébauche de simulation de rétines 3D.

5 Conclusion et perspectives

Nous avons présenté une modélisation architecturale de rétine et son implémentation à base d'automates cellulaires de synthèse d'images. Les premiers résultats d'implémentation de la version de rétine $2D$ laisse envisager à court terme des résultats temps réel haute résolution pour un coefficient n variable dans les cas où l'on solliciterait les GPU ou plus simplement un CPU plus puissant. La version $3D$ de rétine présente des résultats intéressants au niveau de la couche des cellules cône (trois images par seconde pour 10^5 cônes). Le calcul des couches horizontale et bipolaire est réalisable à la suite mais réduirait conséquemment la cadence des images par seconde.

D'un point de vue modélisation, un certain nombre de traitement sont envisageables : l'organisation cellulaire permet une évolution adaptative de la rétine implémentée où le paramètrage des cellules serait localement et temporellement modifié en fonction de la position de la cellule rétinienne ou de la nature de la scène traitée. Une séparation des voies de chrominance est réalisable à court terme pour reproduire la perception des contrastes chromatiques de la voie parvocellulaire. Comme le montre la figure 7 une extension de la rétine à la couche plexiforme interne (PLI) est aussi possible. Dans le domaine des automates cellulaires de nombreux comportements ont été identifiés [19]. Nous sommes convaincus que ces résultats peuvent être appliqués à la simulation de la rétine et plus généralement du système visuel.

En résumé, les automates cellulaires apportent une solution à la fois puissante et originale pour l'implémentation de rétine artificielle d'inspiration biologique. L'augmentation des performances de calculateurs et l'utilisation des ressources matérielles et logicielles graphiques nous laisse penser que dans un avenir proche nous pourrons à partir d'un calculateur de type PC intégrer des architectures complexes de rétines artificielles d'inspiration biologique.

References

[1] W. 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.

[2] J.H. Conway. Game of life. *Scientific American 223*, pages 120–123, 1970.

[3] K.W. Fleischer, B.L. Currin D.H. Laidlaw, and A.H. Barr. Cellular texture generation. In *SIGGRAPH'95 Conf. Proc.*, pages 239–248, 1995.

[4] S. Gobron. Generilized surface cellular automata. *in reviewing process*, 2005.

[5] S. Gobron and N. Chiba. Crack pattern simulation based on 3d surface cellular automata. *The Visual Computer*, 17(5):287–309, 2001.

[6] M. Harris. Implementation of a cml boiling simulation using graphics hardware. In *CS. Dept, UNCCH, Tech. Report*, volume 02-016, 2003.

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

[8] J. Kari. Representation of reversible cellular automata with block permutations. *Mathematical Systems Theory*, 29(1):47–61, 1996.

[9] C.G. Langton. *Live at the edge of chaos, Artificial Life II.* ed. Langton et al., Addison-Wesley, 1992.

[10] S. Lefebvre, S. Hornus, , and F. Neyret. All-purpose texture sprites. In *INRIA Research Report 5209, INRIA*, 2004.

[11] D. Marr. Vision. *W. H. Freeman and Company, New York*, 1982.

[12] J.-F. Risse. Exploration de la fonction visuelle - applications au domaine sensoriel de l'oeil normal et en pathologie. *Société Française d'Ophtalmologie.*

[13] A. Rosenfeld. *Picture Languages.* Academic Press, New York, 1979.

[14] D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide v1.4.* Addison Wesley Professional, 2003.

[15] D. Thalmann. A *lifegame* approach to surface modeling and rendering. *The Visual Computer*, 2:384–390, 1986.

[16] G. Turk. Generating texture for arbitrary surfaces using reaction-diffusion. In *SIGGRAPH'91 Conf. Proc.*, volume 25, pages 289–298, 1991.

[17] W.F.Schreiber. *Fundamentals of Electronic Imaging Systems.* 1986.

[18] A. Witkin and M. Kass. Reaction-diffusion textures. In *SIGGRAPH'91 Conf. Proc.*, pages 299–308, 1991.

[19] S. Wolfram. *A new kind of science.* 2002.