الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Master Académique

Thème :

Application d'un système immunitaire artificiel dans la reconnaissance des surfaces

Encadré Par :

Dr Kouahla Mohamed Nadjib

Présenté par :

Delloul Ali

Rahal Radouane

Juin 2015

Remerciement

Nous nous adressons en premier lieu nos sincères remerciements à notre encadreur Dr Kouahla Mohamed Nadjib, nous le remercions de nous avoir encadré, orienter, aider et conseiller. Nous remercions la présidente et les membres du jury pour avoir accepter de juger ce modeste travail. Nous remercions toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé nos réflexions et ont accepté à nous rencontrer et répondre à nos questions durant notre recherche. À tous ses intervenants nous présentons notre respect et gratitude.

Résumé

Le travail réalisé concerne la reconstruction 3D par Supershapes, rentre dans le cadre d'un projet de Reconstruction 3D de scènes réelles. Le problème abordé est la représentation d'un nuage de points 3D obtenus par la numérisation d'un objet ou d'une scène réelle. L'algorithme de reconstruction produit une surface (ou une combinaison de surfaces) représentant la forme de l'objet ou de la scène par une ou plusieurs supershapes. On utilise les R-fonctions pour combiner les équations implicites de plusieurs supershapes en une seule équation qui représente l'objet final. Les systèmes immunitaires artificiels et les algorithmes génétiques sont utilisés pour optimiser les fonctions de coût qui permet de déterminer les paramètres des supershapes pour représenter au mieux le nuage.

Sommaire

Liste des figures

Liste des tableaux

Introduction Générale

Introduction :

La reconstruction d'un modèle 3D consiste à convertir un nuage de points en une surface qui est représentée sous la forme d'un maillage polygonal ou d'une surface implicite. Ce processus de reconstruction fait l'objet de travaux de recherche en informatique graphique, en modélisation géométrique et en géométrie algorithmique.

L'objectif de la reconstruction est de reproduire un modèle de surface continu à partir d'un nuage de points qui respectent autant que possible les propriétés géométriques de la surface de l'objet. Nous nous intéressons à la représentation d'un objet 3D par une ou plusieurs surfaces. C'est-a-dire que nous essayons de trouver la meilleure solution pour représenter le nuage de points. Ce problème correspond à une optimisation non linéaire et il existe différentes méthodes :

- les méthodes déterministes
- les méthodes stochastiques

Parmi les méthodes déterministes, on retrouve par exemple l'algorithme de Levenberg Marquardt [1][2] ou l'algorithme des gradients conjugués [3]. Et pour les méthodes stochastiques on retrouve les algorithmes génétiques (GA) ou les techniques intermédiaires comme par exemple le recuit-simulé [4].

Dans ces dernières années, il y a aussi les systèmes immunitaires artificiels qui se présentent comme une nouvelle métaphore puissante dans les méthodes d'optimisation inspirée par le mécanisme de défense humain. On s'inspire des résultats de deux méthodes : une méthode déterministe proposée par Fougerolle et al. [5] et une méthode stochastique proposée par Bokhabrine et al. [6] qui a utilisé un GA pour reconstruire des objets 3D par une ou deux surfaces.

Le document est composé des parties suivantes :

Chapitre 1 présente un état d'art sur les supershapes, les méthodes d'approximation d'un nuage de point par une surface, les R-fonctions, chapitre 2 présente les systèmes immunitaires et les algorithmes génétiques.

Pour le chapitre 3, on présente la structure et l'implementation de notre algorithme génétique, système immunitaire artificiel et les générateurs des variables aléatoires utilisés. Et enfin les résultats d'exécution sont présentés dans le chapitre 4. Nous terminons par une conclusion et quelques perspectives de recherche.

Chapitre 1

Les Supershapes

&

R-fonctions

1.1 Introduction

Les supershapes sont des récentes primitives peu utilisé à la reconstruction des surfaces, ils sont seulement appliqués pour la modélisation de pièces mécaniques [7] et la représentation de fleurs [8]. Ce qui justifient pourquoi ils sont mal connues inversement les super quadriques [8] sont des primitives standards qui ont été largement étudiées et appliquées depuis une vingtaine d'années. En commun la reconstruction des surfaces par les supershapes ou les super quadriques nécessite le développement d'une fonction de coûts appropriés qui n'a pas été encore proposé.

1.2 Supershapes ou surfaces de Gielis

Les Supershapes ont récemment été introduites par Gielis [8] comme une extension des super quadriques avec multiples symétries [7]. Le terme supershape est une extension 3D pour désigner la surface définie paramétriquement par l'équation (1):

$$\begin{pmatrix} x(\theta,\phi) \\ y(\theta,\phi) \\ z(\theta,\phi) \end{pmatrix} = \begin{pmatrix} r_1(\theta)r_2(\phi)\cos(\theta)\cos(\phi) \\ r_1(\theta)r_2(\phi)\sin(\theta)\cos(\phi) \\ r_2(\phi)\sin(\phi) \end{pmatrix} \quad (1)$$

$$r(\theta) = \frac{1}{\sqrt[n_1]{\left|\frac{1}{a}\cos\left(\frac{m\theta}{4}\right)\right|^{n_2} + \left|\frac{1}{b}\sin\left(\frac{m\theta}{4}\right)\right|^{n_3}}}$$

La figure 1.1 illustre un exemple de supershape 3D

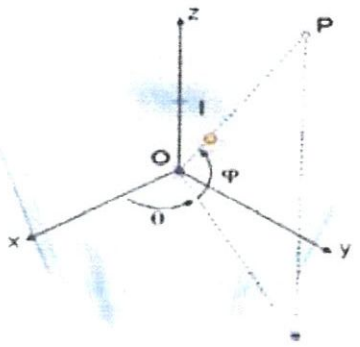


Figure 1-1: Interprétation géométrique d'une supershape

- Les paramètres a > 0 et b > 0 contrôlent les dimensions du polygone.
- Les paramètres m et M correspondent aux nombres de symétries.
- Les paramètres $\{n_1, n_2, n_3, N_1, N_2, N_3\}$ contrôlent la forme.

À partir de l'équation (1), on peut définir une fonction de potentiel par :

$$f(x,y,z) = 1 - \frac{x^2 + y^2 + r_1^2(\theta)z^2}{r_1^2(\theta)r_2^2(\phi)}$$

Fougerolle et al.[6] ont proposé une forme isotrope de cette fonction, définie par :

$$F(x,y,z) = 1 - \frac{1}{r_2(\theta)}\sqrt{\frac{x^2 + y^2 + z^2}{cos^2\phi(r_1^2(\theta) - 1) + 1}}$$

A partir de la valeur de la fonction F(x,y,z) on peut déduire la position du point P par rapport à la surface de la supershape, comme l'illustre la figure 1.2 :

- Si F (x, y, z) = 0 alors le point P est sur la surface de la supershape.
- Si F (x, y, z) > 0 alors le point P est à l'intérieur de la supershape.
- Si F (x, y, z) < 0 alors le point P est à l'extérieur de la supershape.

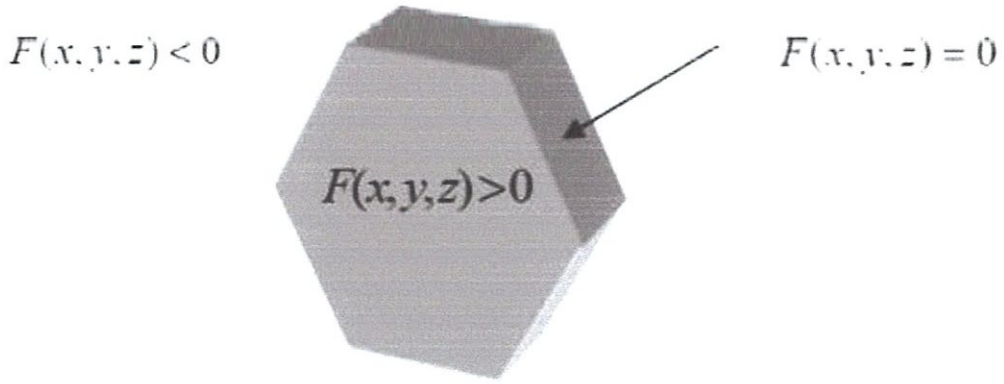


Figure 1-2: Signe de la fonction potentiel d'une supershapes

La figure 1.3 présente des exemples de supershapes 3D pour les différents jeux de paramètres de forme et de symétrie mentionnés dans le tableau 1.1.

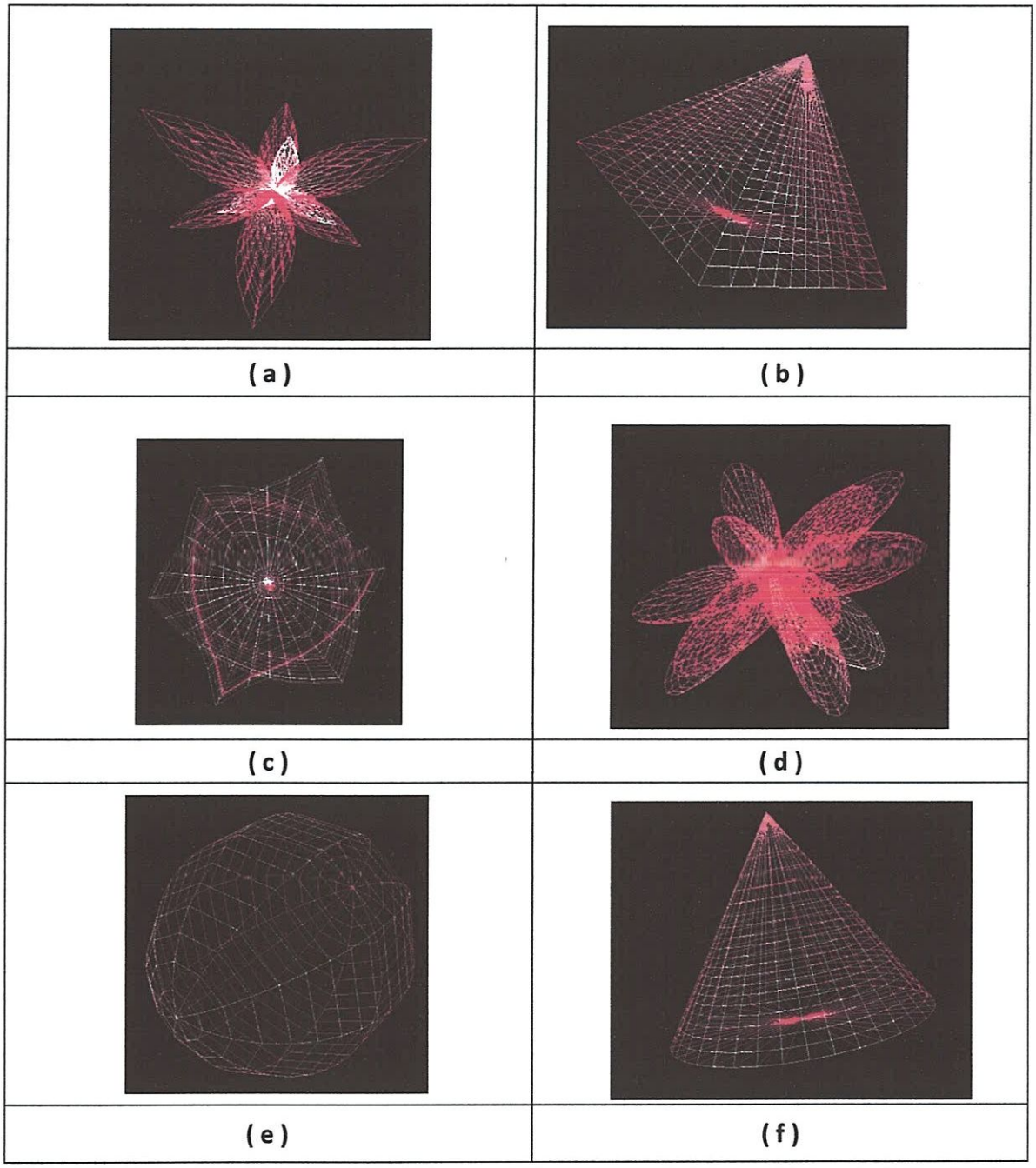


Figure 1-3 : Exemples de supershapes 3D

Figure	n_1	n_2	n_3	m	N_1	N_2	N_3	M
(a)	1.0111	0.873	-2.771	6	1.092	0.869	-2.779	6
(b)	1	1	1	4	1	1	1	4
(c)	1.072	1.023	0.714	3	1.007	0.968	1.438	4
(d)	2	-4.1	2	9	2	-5.1	1	9
(e)	2.066	1.968	3.397	1	2.174	2.303	4.866	2
(f)	2	2	2	4	1	1	1	4

Tableau 1-1: Paramètres des supershapes illustrées en figure 1.3.

L'une des propriétés intéressantes des supershapes est d'être définie à la fois sous forme paramétrique et sous forme implicite. La forme implicite permet d'obtenir des informations sur la position d'un point P par rapport à la surface ce qui nous sera utile pour le calcul d'erreur. La forme paramétrique permet un affichage direct sans qu'il soit nécessaire d'utiliser des algorithmes de polygonisation de surface comme les marching cubes [9].

Les supershapes ont rarement été appliquées dans le domaine de reconstruction de surface, peut-être parce que la fonction du potentiel est encore mal connue et que les super quadriques sont utilisés depuis 1981, elles sont considérées comme des primitives de base. Les supershapes n'ont été utilisés que par Gielis en 2003 dans la représentation de fleurs [7] et par Fougerolle et al.[5] en 2005 dans la modélisation de pièces mécaniques.

1.3 Approximation d'un ensemble de points par une supershape

La méthode que nous utilisons s'inspire de celle proposée par Fougerolle et al.[5], qui est une adaptation de la méthode proposée pour les super quadriques par Solina et Bajcsy [10]. On définit une fonction d'erreur F(A) comme la somme des carrés d'une R-fonction évaluée en chaque point du nuage. Pour un ensemble de N points, il s'agit donc de trouver les 17 paramètres d'une supershape notes:

$$A = \{m, n_1, n_2, n_3, M, N_1, N_2, N_3, t_x, t_y, t_z, \phi, \theta, \psi, s_x, s_y, s_z\}$$

Pour modéliser les données et qui minimisent la fonction d'erreur définie par:

$$F(A) = \sum_{i=1}^{N} F(x_i)^2$$

Avec:

- F(p) est la valeur de la fonction potentielle du point P(pouvant être la distance euclidienne ou radiale) ou une R-fonction si plusieurs supershapes sont utilisées.
- $t_x, t_y \; et \; t_z$ sont les paramètres de translation.
- $\phi, \theta \; et \; \psi$ sont les angles de rotation suivant les axes (Ox), (Oy) et (Oz) respectivement
- S_X, S_Y et S_Z sont les coefficients de mise à l'échelle.

La fonction de coût proposée par Fougerolle et al [6]. Pour une supershape est définie par :

$$F(A) = s_x s_y s_z \sum F^2(P_i)$$

Le terme S_x, S_y, S_z détermine la supershape de volume minimal, ce qui permet de reconstruire des données incomplètes.

1.4 R-fonctions

1.4.1 Arbre CSG

Dans cette partie, nous nous intéressons à la reconstruction des objets solides, pouvant résulter d'opérations booléennes entre plusieurs primitives élémentaires. Si ces primitives sont des volumes on se retrouve dans le cas des approches de type CSG (Géométrie constructive des solides) où un objet est représenté comme une composition de plusieurs primitives telle que des cubes, cylindres, sphères, etc. La figure 1.4 montre un exemple d'arbre CSG correspondant à un objet issu d'une combinaison entre des objets simples par des opérations booléennes : intersection (and), union (or) et différence (sub).

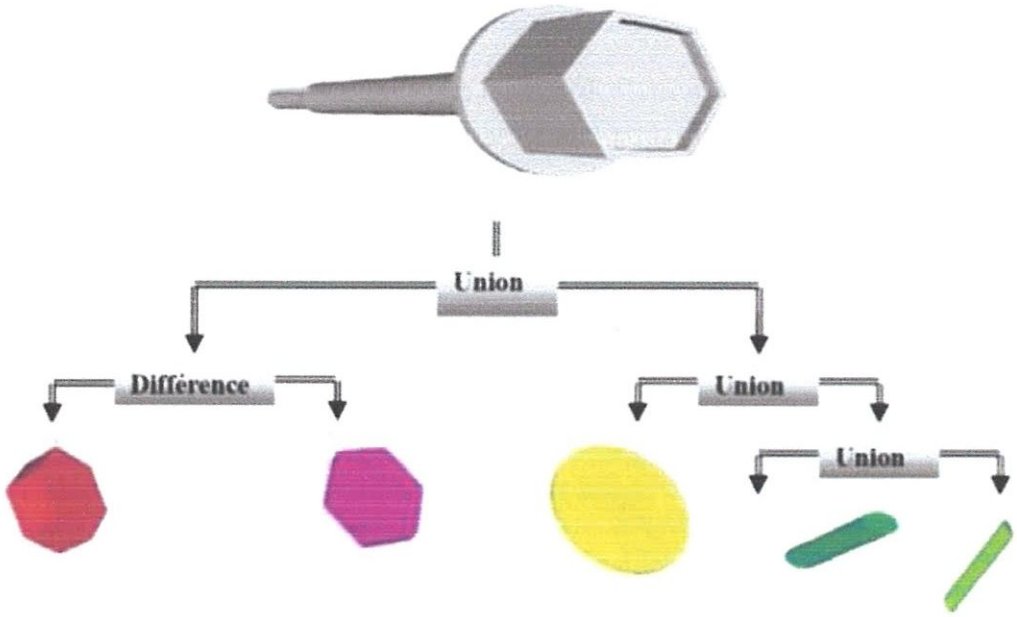


Figure 1-4: Exemple d'arbre CSG

Les objets réels sont souvent trop complexes pour être représentés par une seule supershape. Nous utilisons les R-fonctions pour passer de prédicats booléens entre des supershapes à des équations. Ceci permet de fusionner les supershapes créées en un seul objet en combinant leurs équations implicites en une seule équation implicite. Nous présentons tout d'abord les R-fonctions binaires (qui sont les plus utilisées) et leur extension aux R- fonctions n-aires dites R-fonctions en chaînes qui réduisent la profondeur des arbres CSG.

1.4.2 Théorie des R-fonctions

La théorie des R-fonctions a été introduite par V.L. Rvachev à la fin des années soixante [11]. Les principaux passages du livre initialement écrit en russe, ont été traduits en anglais par Shapiro [12]. Une R-fonction est une fonction à valeur réelle dont une propriété est complètement déterminée par la propriété correspondante de ses arguments (Telle que le signe par exemple). En particulier, certaines fonctions réelles à variables réelles possèdent la propriété que leur signe est complètement déterminé par le signe de leurs arguments. Par exemple, le signe de la fonction f(x) = xy est complètement déterminé par le signe de ses arguments alors que pour la fonction f(x) = cos(xy), non seulement le signe, mais également l'amplitude des arguments influencent le signe de la fonction [5].

Il existe différents types de R-fonctions qui sont notés: R_p , $R_\propto$ et R_0^m . On appelle R-fonctions en chaîne ou R-fonctions n-aires, les R-fonctions qui ont plus de deux arguments.

-La R-fonctions $R_\propto$

Définit par:

$$R_\propto = \frac{1}{1+\propto}\left(x + y \pm \sqrt{x^2 + y^2 - 2 \propto xy}\right)$$

Où : x et y sont les deux fonctions implicites à combiner.

α(x, y) est une fonction arbitraire telle que α(x, y) ∈] − 1, 1]. La R-fonction la plus simple et la plus utilisée est celle avec α(x,y) = 1 et correspond aux fonctions max(x, y) et min(x, y)

Les R fonctions $R_p et\ R_0^m$.

Ces deux fonctions sont deux branches de R-fonctions avec des propriétés différentielles garanties introduites par Rvachev pour résoudre le problème de la perte de différentiabilité le long de la ligne x = y. $R_p et\ R_0^m$ sont respectivement définies par :

$$R_0^m = \left(x + y \pm \sqrt{x^2 + y^2}\right)(x^2 + y^2)^{\frac{m}{2}}$$

$$R_p = x + y \pm (x^p + y^p)^{1/p}$$

Une des R-fonctions les plus utilisées est la fonction R_p pour p = 2 définie par :

$$R_{p=2} = x + y \pm \sqrt{x^2 + y^2}$$

-Les R-fonctions n-aires :

Les R-fonctions ne prennent en compte que deux arguments. Rvachev propose les R-conjonctions et R-disjonctions n-aires pour prendre en compte un nombre d'arguments supérieur à deux (Figure 5(b)). Les équations suivantes correspondent respectivement à une R-conjonction et une R-disjonction multiple avec les arguments xi, i = {1....p}. Le paramètre m est un entier positif ou nul et correspond à celui utilisé pour la R-fonction $\boldsymbol{R_0^m}$.

$$\bigwedge_{i=1}^{i=n^{(m)}} x_i \equiv \sum_{i=1}^{n} (-1)^m x_i^m (x_i - |x_i|) + \prod_{i=1}^{n} x_i^m (x_i + |x_i|)$$

$$\bigvee_{i=1}^{i=n^{(m)}} x_i \equiv \sum_{i=1}^{n} x_i^m (x_i + |x_i|) - \prod_{i=1}^{n} x_i^m (-1)^m (|x_i| - x_i)$$

L'intérêt des R-fonctions n-aires est de réduire la profondeur des arbres CSG pour les objets complexes où on a une série d'opérations booléennes de même type.

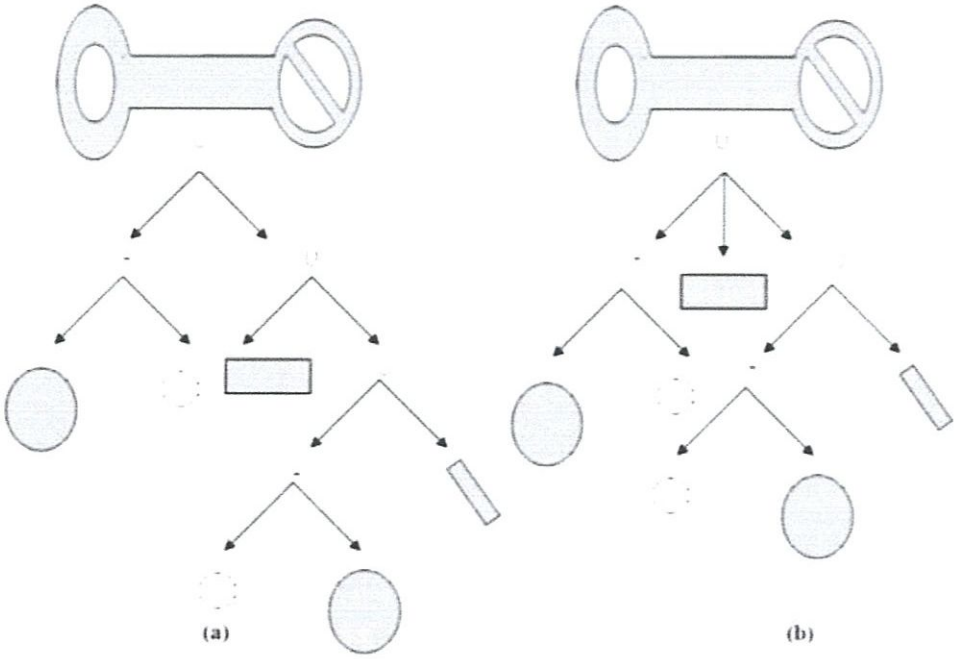


Figure 1-5: Exemple de décomposition en arbre

1.5 Conclusion

Pour ce chapitre nous avons présenté quelques modèles des supershapes avec ces paramètres ensuite l'Approximation d'un ensemble de points par une supershape (la fonction d'erreur et du coût) enfin la théorie des R-fonctions. Le chapitre suivant présente le principe des systèmes immunitaires artificiels et les algorithmes génétiques.

Chapitre 2

Les systèmes Bio-Inspirés

2.1 Introduction

Le système immunitaire artificiel et les Algorithme Génétiques sont deux autre approches qui peut être appliquée à la reconnaissance des formes [13] [14] [15] [16] [17], mais aussi à la robotique [18], la détection des intrusions dans les réseaux informatiques [19], l'apprentissage machine [20] ... etc. Les fonctions du système immunitaire artificiel s'inspirent du système immunitaire naturel, notamment en ce qui concerne : les cellules responsables de la réponse ainsi que la réponse immunitaire. Ces facteurs peuvent contribuer à la construction d'un système proportionnellement complet destiné à la reconnaissance des formes. Pour la deuxième approche ; les premiers travaux sur les algorithmes génétiques (GA) ont commencé dans les années cinquante lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis entre 1960 et 1970, John Holland [21] sur les bases des travaux précédents dans le cadre d'optimisation mathématique, développa les principes fondamentaux des algorithmes génétiques.

2.2 Le système immunitaire artificiel

Au cours de ces dernières années, un intérêt croissant a côtoyé le domaine des systèmes immunitaires artificiels et leurs applications. Cette discipline utilise des idées glanées de la nature, afin de développer des outils informatiques pour résoudre des problèmes du monde réel.

2.2.1 Le système immunitaire naturel

Le système immunitaire biologique constitue une arme contre des intrus dans un corps donné. Pour ce faire, il existe plusieurs cellules qui contribuent à éliminer ces intrus nommé antigènes. Ces cellules participent pour ce qu'on appelle une 'réponse immunitaire biologique'.

2.2.2 Le système immunitaire artificiel (AIS)

Le système immunitaire naturel est assez compliqué pour qu'une simulation artificielle soit réalisée d'une façon complète. Par contre, dans [22] l'auteur a réussi à simuler les fonctions les plus pertinentes dans un système immunitaire biologique pour que l'artificiel hérite le maximum des fonctionnalités naturelles dans le domaine de la reconnaissance des formes.

Ou plus simplement, selon Timmis [20] : « un système immunitaire artificiel est un système informatique basé sur des métaphores du système immunitaire naturel ».

L'AIS étaient composés typiquement de trois algorithmes intelligents, nommés : la sélection négative, le modèle du réseau immunitaire et la sélection clonale.

2.2.2.1 La sélection négative

Au départ, l'algorithme sélectionne un ensemble des éléments du self déjà reconnue par le système, après il va générer aléatoirement des détecteurs et les placer dans un ensemble P et déterminer la familiarité entre eux avec les éléments du self qui déjà existent. Si un de ces détecteur a été détecté qu'il y a un des éléments de p est semblable à un des éléments de self le système rejette automatiquement le détecteur, sinon il va l'ajouter sur un nouveau ensemble.

2.2.2.2 Le réseau immunitaire

Le biologiste Jerne a proposé une théorie des réseaux immunitaire pour expliquer certaines des propriétés émergentes observées du système immunitaire telle que la mémorisation. La prémisse de la théorie des réseaux immunitaires, c'est que tout récepteur de lymphocytes au sein d'un organisme peut être reconnu par un sous-ensemble du répertoire général. Les récepteurs de ce jeu de reconnaissance ont leur propre série de reconnaissance et ainsi de suite, un réseau d'interactions immunitaires est formé. Les réseaux immunitaires sont souvent désignés comme des réseaux idiotypiques [20]. En l'absence d'antigène étranger, Jerne a conclu que le système immunitaire doit afficher un comportement ou l'activité résulte d'interactions avec lui-même, et de ces comportements immunologiques des interactions telles que la tolérance et de la mémorisation.

2.2.2.3 La sélection clonale

La sélection est inspire aussi du monde réel (système immunitaire naturel), la sélection clonale a été utilisé comme source d'inspiration pour le développement de systèmes immunitaires artificiels qui effectuent des tâches d'optimisation et de reconnaissance des formes. Clonale proposé est une approche basée population, compétitive où des anticorps sont en compétition pour l'identification (l'optimisation) d'un antigène et coopérative où toute la population coopère pour présenter la solution finale. Le job c'est créer aléatoirement un

ensemble initial d'anticorps. A par exemple et sélectionner les anticorps rassemblable avec l'ensemble des motifs qui existe déjà après Générer des clones d'un sous-ensemble d'anticorps A selon la plus grande affinité: le nombre de clones d'un anticorps est proportionnel à son affinité. Après faire la mutation des clones de A et placer une copie de plus haut rassemblable anticorps dans un autre ensemble (mémoire M) et enfin remplacer les anticorps qui ne sont pas similaires par des anticorps généré aléatoirement.

La figure suivante explique les étapes de la sélection clonale

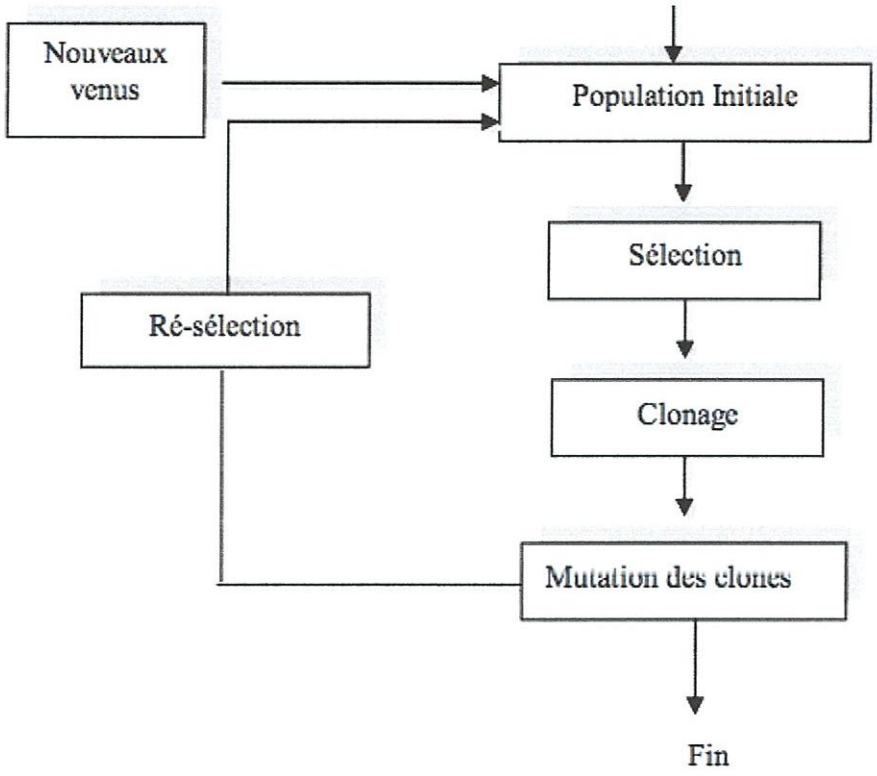


figure 2-1 Structure générale de sélection clonale

2.3 Algorithme génétique

Un algorithme génétique est utilisé pour optimiser une fonction de coût. Un GA a besoin des deux fonctions suivantes :

- Une fonction de codage des données du problème sous la forme d'une séquence de bits, ou se forme d'une liste d'entier. Si le codage est efficace, ou du moins bien adapté au problème alors les algorithmes génétiques donneront de bons résultats, sinon leur efficacité ne sera pas médiocre.
- Une fonction de coût U(x) permettant de calculer l'adaptation au problème considéré d'une séquence de bits ou d'entiers x donnés. Cette fonction correspondra au critère à optimiser sur l'espace des données [23].

2.3.1 Codage des données

Historiquement, le codage utilisé par les algorithmes génétiques était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace d'états. Ce type de codage a pour intérêt de permettre la création d'opérateurs de croisement et de mutation simples. Cependant, ce type de codage n'est pas toujours pratique. En effet, deux éléments voisins en terme de distance de Hamming, ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un codage de Gray. Toutefois, pour des problèmes d'optimisation dans des espaces de grande dimension, le codage binaire peut rapidement devenir mauvais.

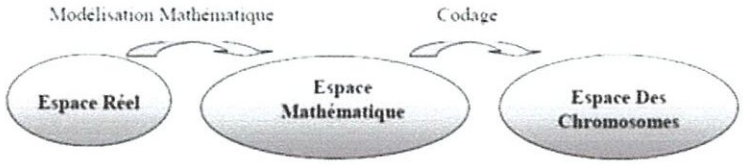


Figure 2-2: Schéma de codage

Les algorithmes génétiques utilisant des vecteurs réels connus sous le nom des algorithmes évolutionnistes évitent ce problème en conservant les variables dans le codage de l'élément de population sans passer par le codage binaire intermédiaire. La structure du problème est conservée dans le codage. Les GA sont des processus itératifs qui suivent le schéma général suivant :

```
Algorithme évolutionniste ( ) ;
1 t := 0 ;
2 Initialiser la population P(t)
3 Evaluation (P(t)) ;
4   Repeat
5           t := t +1;
6           Sélection parent(P(t))
7           Croisement (P(t))
8           Mutation(P(t))
9           Recombinaison (P(t))
10          Evaluation (P(t))
11          Reproduction (P(t))
12  Until Test-critère d'arrêt (P(t))
```

Figure 2-3: Schéma général des algorithmes évolutionnistes.

On génère d'abord un certain nombre de bits qui code les différents éléments de la population initiale. Nous appelons ces séquences de bits les chromosomes des éléments de la population

2.3.2 Chromosomes

Chaque chromosome de la population est une solution du système. Une solution peut être construite à partir d'une ou plusieurs supershapes donc la taille de notre chromosome est un multiple de 17.

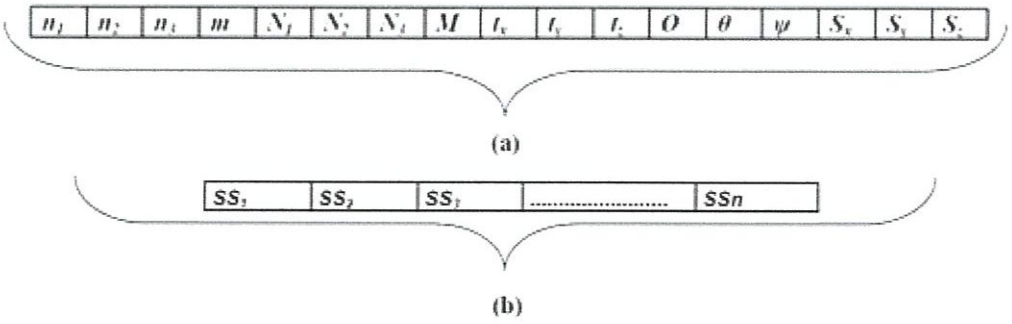


Figure 2-4: Structure : (a) d'un Gène (une supershape).(b) d'un chromosome

Pour chaque point P_i par l'équation

$$Error = \sum_{i=1}^{N} RF^2(P_i)$$

2.3.3 Croisement

L'opérateur de croisement consiste à prendre deux chromosomes parents P_1 et P_2 repérés dans la population par leurs positions pos_1 et pos_2 respectivement et à échanger en partie leurs informations pour obtenir deux gènes fils C_1 et C_2. On définit le nombre de points de coupure et on fait la permutation entre les deux gènes parents comme l'illustre la figure 3.3 :

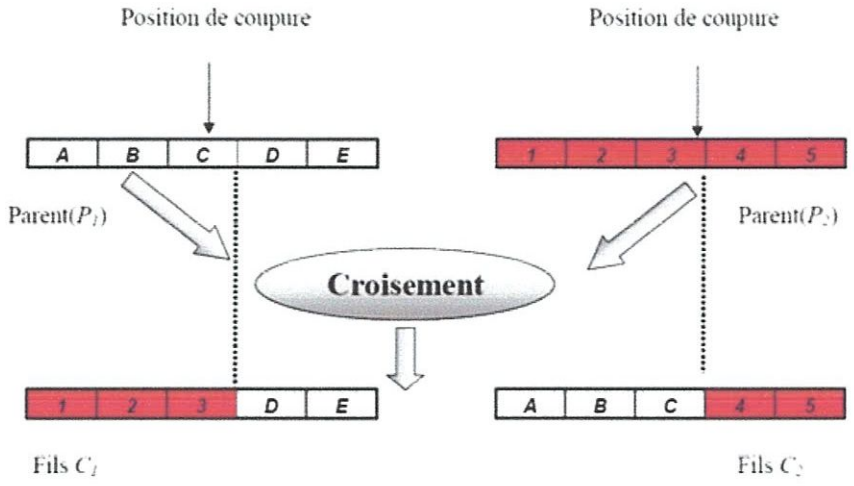


Figure 2-5: L'opérateur de croisement

2.3.4 Mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. L'opérateur de mutation consiste généralement à tirer aléatoirement un gène et à le remplacer par une valeur aléatoire. Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale.

Dans notre cas, la mutation est appliquée aux résultats de croisement c'est-à-dire aux chromosomes C_1 où C_2 dans notre exemple. On doit générer aléatoirement une valeur pmut pour chaque chromosome résultant de croisement. Si *Pmut* est inférieure à la constante de mutation (PROBAMUTATION), on affecte une perturbation aléatoire de certaines valeurs de gène. Pour les paramètres de transformation (translation, rotation et mise à échelle) on ajoute une petite valeur signée aux paramètres concernés et pour les paramètres de forme et de symétrie on génère une nouvelle valeur aléatoire (Figure 3.4).

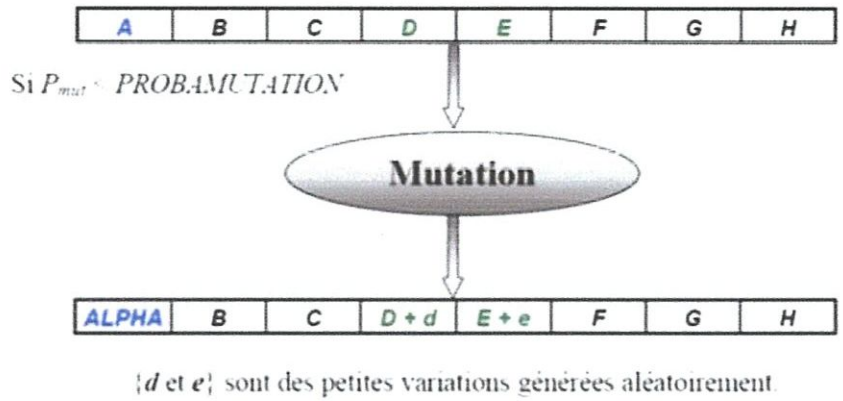


Figure 2-6: L'opérateur de mutation

2.3.5 Sélection

La sélection consiste à conserver les meilleures solutions. On récupère le même nombre de chromosomes qu'avant le croisement et la mutation. A chaque itération, la taille de la population ne varie pas, mais la population est de plus en plus adaptée. On notera donc qu'au fur et à mesure que le nombre de générations augmente, les meilleurs éléments sont de plus en plus représentés aux dépends des autres individus.

L'algorithme général est illustré en figure 3.5. En résumé : on initialise la population des chromosomes, puis on applique les opérateurs de croisement, de mutation et de sélection. Ce processus est répété jusqu'à ce qu'on atteigne l'un des critères d'arrêt suivant :

- Une erreur plus faible qu'un seuil arbitraire.
- Un nombre maximal d'itérations sans amélioration de la solution. On considèrera alors que l'algorithme a converge
- Un nombre maximum d'itération pour borner le temps d'exécution.

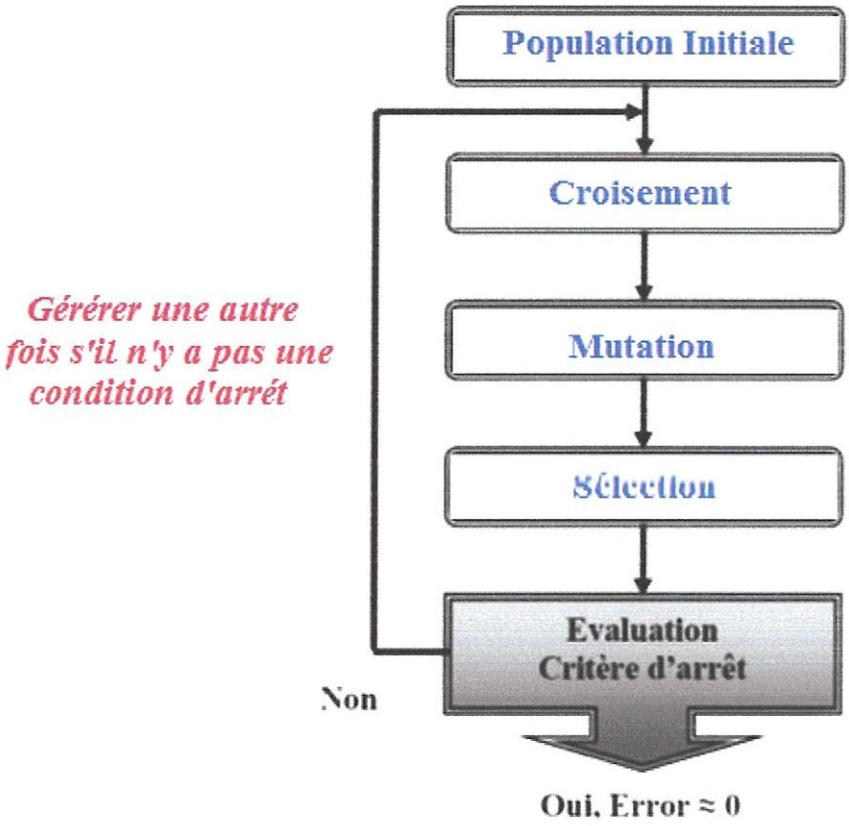


Figure 2-7: *Diagramme général d'un algorithme génétique.*

Les détails pratiques de chaque étape de cet algorithme sont présentés dans la section implémentation.

2.4 Conclusion

Ce chapitre fait absolument une présentation générale des systèmes immunitaires et les algorithmes génétiques, ou on donne les grandes lignes de chaque algorithme. Il faut surtout retenir les idées suivantes :

La sélection clonale d'un système immunitaire artificiel et le codage employé sont très importants. Il fait le lien entre les individus et les solutions du problème. Ainsi pour l'algorithme génétique régit les algorithmes de reproduction, en particulier le croisement et la mutation. Une bonne mise en équation de la fitness est utile pour une évaluation des individus spécifique au problème.

Chapitre 3

Conception

Chapitre 3 : Conception

3.1 Introduction

Pour notre implémentation, le programme est développé en C++ et OPENGL. Il se décompose de deux classes principales : population (.cpp/.h) et Gene (.cpp/.h).

3.1 Fonctions de la classe Gene

3.1.1 Fonction Initrandom

void InitRandom(int n, double x,double y, double z, double r)

Elle sert à initialiser les 17 paramètres de chaque gène (les 17 de chaque supershape). Elle a comme paramètres (n, x, y, z, r) tels que :

- n : le nombre de supershapes par chromosome.
- x, y, z : correspondent aux coordonnées des supershapes.
  r : le rayon des supershapes, il correspond à la distance maximale entre le barycentre et chaque point du maillage.
- Les paramètres de forme $n_1, n_2, n_3, M, N_1, N_2$ et N_3 sont générés aléatoirement entre 1 et 1000.
- Les paramètres de symétrie m et M sont générés aléatoirement entre 2 et 16.
- Les paramètres de translation t_x, t_y, t_z , de rotation ϕ, θ, ψ et de mise à l'échelle s_x, s_y, s_z sont générés aléatoirement par une loi uniforme dans des intervalles particuliers.

3.1.2 Fonction Evaluate

void Evaluate(std::vector<Vector3d> Plist)

Elle consiste à calculer la fonction implicite pour chaque point du maillage (stockés dans Plist) pour tous les gènes du chromosome. Pour chaque chromosome on doit évaluer et stocker la R-fonction. L'erreur totale de ce chromosome, notée Error, peut être calculée en utilisant la formule suivantes. Le résultat de ce calcul est stocké pour chaque chromosome.

$$F(A) = \sum_{i=1}^{N} RF^2(P_i)$$

Pour des raisons pratiques, certaines informations (comme la fonction implicite, la R-

fonction et l'erreur totale) sont stockées au sein du chromosome mais s'interviennent pas dans l'algorithme génétique.

3.1.3 Fonction Mutate

void Mutate()

Cette fonction génère une valeur *Pmut* par une loi uniforme. Si Pmut est inférieure à un seuil PROBAMUTATION, on doit muter le chromosome. La mutation consiste à ajouter une petite variation à certains paramètres du chromosome muté.

- Pour les paramètres de forme on génère une valeur comprise entre 1 et 1000

- Pour les paramètres de symétrie on génère une valeur comprise entre 2 et 16.

- Pour les paramètres de translation et rotation, on rajoute des petites variations signés aux valeurs initiaux par exemple pour les paramètres de rotation (2°), les paramètres de translation (0.01) et les paramètres de mise à l'échelle (0.1).

Enfin, on doit indiquer pour cette génération que ce chromosome est muté en positionnant le champ ModifiedM à True pour éviter qu'un chromosome ne soit muté plusieurs fois au cours d'une même génération.

3.2 Fonctions de la classe population

3.2.1 Création de la population

void Setpopul(int n, double x,double y, double z, double r)

On définit une population par les paramètres (n,x,y,z,r), il s'agit des mêmes paramètres que ceux utilisés pour la classe gène

Les chromosomes de la population sont créés par l'appel à la fonction Initrandom (n, x, y, z). Ensuite ils sont stockés dans un tableau de taille n noté *TabGene*.

3.2.2 Tableau de points de coupure

void SetSplit (int NSS)

On génère aléatoirement les positions entre (1, 15) pour tous les NSS qui

correspondent au nombre minimum des supershapes des deux chromosomes avant le croisement parce que chaque supershape à 17 paramètres notée de 0 à 16 alors on ne peut pas définie les points de coupure à la position 0 car on ne trouve pas le première partie de permutation et à la position 16 on doit permuter tout le gène. Ce tableau consiste à définir pour chaque gène (supershape) du chromosome une position de coupure pour appliquer l'opérateur crossover.

3.2.3 Le crossover

void Crossover(int Pos1, int Pos2)

C'est l'opération de croisement de deux chromosomes différents. Le crossover consiste à permuter les paramètres des supershapes de deux chromosomes récupérés du tableau *TabGene* par leurs positions Pos1 et Pos2. Le crossover n'est appliqué que sur des chromosomes qui n'ont jamais été croisées.

Après une opération de croisement, l'indicateur ModifiedC est positionné à True pour garder une Trace de l'opération, cela évite de réappliquer l'opérateur sur des chromosomes déjà croisés.

3.2.4 La mutation

void MutatePop ()

Pour tous les chromosomes stockés dans TabGene ont fait appel à la fonction Mutate.

3.2.5 Evaluation

void Evaluation (std::vector<Vector3d> PointList)

L'évaluation de la population consiste à évaluer tous les chromosomes du tableau TabGene par la fonction Evaluate.

3.2.5 Selection

void Select()

Cette fonction permet de trouver la meilleure solution qui représente le maillage. Le principe de la sélection est de trouver les n meilleurs chromosomes de la population qui ont la plus petite erreur et les conserver dans la structure TabGene.

3.3 Générateur de variables aléatoires

Une variable aléatoire est une variable dont la suite des valeurs n'est pas prédictible et ceci quelque soit la longueur de la série générée. Il n'est donc pas possible de trouver une équation permettant de prédire le reste de la série à partir de celles déjà générées. Toute variable aléatoire suit une loi de probabilité qui définit la répartition des valeurs au sein d'un intervalle. Le hasard vrai n'existe que dans la nature. Il est trés difficile de le reproduire sur ordinateur, sauf en le reliant à un dispositif physique de mesure d'un phénomène naturel. Ces dispositifs sont très complexes et coûteux. En revanche, il existe des algorithmes basés sur l'heure système (l'horloge) qui fournissent un pseudo-hasard suffisant pour les applications courantes. L'horloge est utilisée dans les générateurs aléatoires des langages de programmation les plus courant. Des calculs successifs sur les valeurs précédemment générées et sur cette heure système fournissent donc les valeurs des variables aléatoires [4].

3.3.1 Loi uniforme G.A.U

La loi uniforme définit une répartition uniforme des valeurs dans un intervalle, c'est à dire que toutes les valeurs de l'intervalle ont la même probabilité d'apparaître dans une longue série (Figure 4.1). La fonction de densité de la loi uniforme est définie par la fonction f(x) sur]a,b[telle que :

$$f(x) = \frac{1}{b-a}$$

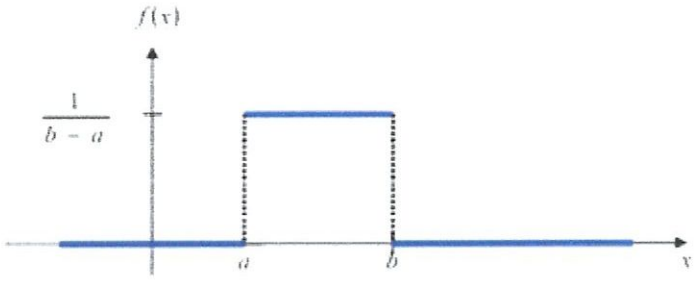


Figure 3 -1: fonction de densité de la loi uniforme

Dans la partie suivante on cite quelques fonctions qui font appel à la ligne de code suivante qui permet de générer une valeur val par G.A.U entre [0,1] :

double val = ((double) rand( )) / (double)(RAND_MAX);

On fait appel au G.A.U dans les opérations suivantes :

- Initialisation de la population.
- Génération de la variable aléatoire *Pmut* pour indiquer si un chromosome mute ou non.
- Définition du nombre de point de coupure et leurs positions.
- Génération de la paire aléatoire à croiser.
- Génération des supershape à croiser.

3.3.2 Loi normale G.A.G

La fonction de densité d'une loi normale, loi de Gauss " G.A.G ", ou de Laplace-Gauss, est définie par :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-u)^2}{2\sigma^2}}$$

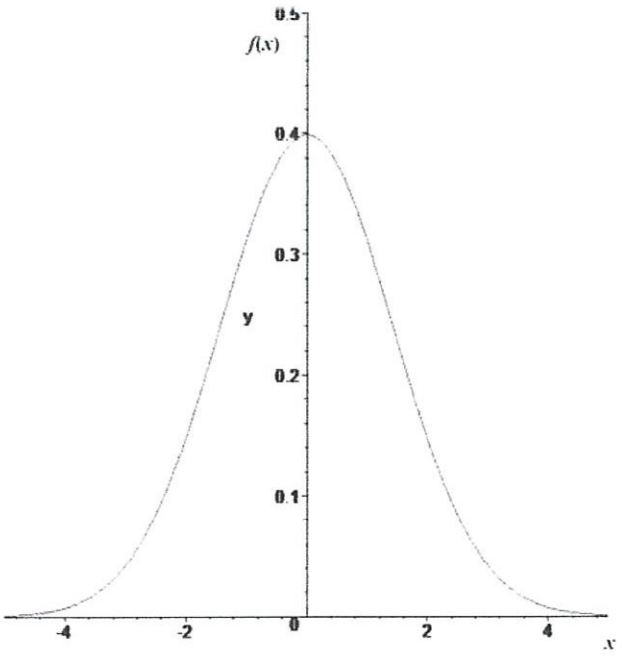


Figure 3-2: fonction de desité de loi normale

Le code de cette loi est le suivant :

U1= (double) rand() / (double)(RAND_MAX);

U2= (double)(rand()+1) / (double)(RAND_MAX+1);

X= cos(2*PI*U1)*sqrt(-2*log(U2)

Le générateur gaussien est utilisé dans la fonction de mutation des gènes pour muter les paramètres de transformation des gènes : coefficients de translation, coefficients de rotation et coefficients de mise à l'échelle.

En effet, on veut généralement tester si la faible variation autour des valeurs actuelles permet d'améliorer notre situation. La loi gaussienne est par conséquent la loi la plus adaptée car les fortes variations apparaissent avec une plus faible probabilité.

3.4 Sélection Clonale pour le Système immunitaire

void Clone () ;

C'est l'opération de clonage. Il consiste à sélectionner les gènes qui ont une forte affinité antigénique c'est-à-dire un taux d'erreur minimale (les 10 premiers). Après on génère le reste de la population.

3.5 Initialisation des paramètres

L'utilisateur fixe librement certain paramètres du processus comme par exemple :

- La taille de la population (500 chromosomes).
- Le nombre de point de coupure (1 seule point).
- La probabilité de mutation (0.5).
- Le nombre de répétition de l'erreur (10).
- L'erreur globale (1).

Les autres paramètres sont tous générés aléatoirement.

Les objets qu'on veut représenter sont exportés en VRML par la fonction Export **ToVRML**. Une fois que la supershape est créé la quantité de la RAM utilisée est constante. Dans cette partie on présente les résultats obtenus pour des formes simples et des formes complexes représentés par une seule supershape.

3.6. VRML:

Le choix du VRML semble judicieux. Ce langage repose déjà sur une certaine notoriété et possède une documentation largement fournie [24] [25] [26] Proposant d'une part une application standalone permettant d'afficher les modèles 3D, d'autre part la possibilité d'utiliser les librairies du moteur 3D en C++ grâce au kit de développement fourni (SDK).
Le but est donc de développer des outils de programmation pour traiter des données 3D et de les visualiser sur un PocketPC. Il faut aussi avoir des méthodes pour sélectionner, simplifier et convertir des données 3D en modèles VRML.
L'un des composants logiciels doivent être créés :
Le VRMLserver

- Traiter les données 3D, créer les cartes sous forme de fichiers VRML.
- Permettre la sélection de régions d'intérêt.
- Programmé en C sous Windows (comme l'algorithme Génétique).

Chapitre 4

Tests & Résultats

Chapitre 4 : Tests et résultats

4.1 Introduction

On s'intéresse dans un premier temps à des supershapes unitaires centrées sur l'origine, ceci nous permet d'analyser le comportement de notre algorithme pour les paramètres de forme et de symétrie. Nous présentons les résultats des Algorithmes Génétiques, ainsi que les influences des paramètres sur la performance du système de reconnaissance des formes.

4.2 Résultat d'exécution

La figure 4-1 montre nos résultats de reconstruction et les valeurs numériques sont regroupées dans le tableau 4-1. Dans ce tableau la notation X/Y correspond à <valeur reconstruite>/<valeur d'origine>.

n_1, n_2, n_3, m, N_1, N_2, N_3, M sont les paramètres de forme et de symétrie de la supershape.
Nb : Nombre d'itérations
Erreur : la valeur calculée par la fonction du cout (page 12, chapitre 1)
IM : optimisation avec un système immunitaire.
AG : optimisation avec un algorithme génétique.

Figure	Optimisation avec :	n_1	n_2	n_3	m	N_1	N_2	N_3	M	Nb	Erreur
(a)	IM	393.303/2.2	498.070/1.7	598.303/3	0.06006/8	525.346/3	493.069/3	493.32/2	0/3	7	0.54
	AG	836.34/2.2	653.297/1.7	747.285/8	8/8	628.382/3	619.349/3	683.84/2	8/3	15	0.70
(b)	IM	397.13/1	387.29/1	447.222/1	4/4	571.4/1	571.003/1	526.979/1	3.88086/4	3	0.20
	AG	589.6931/1	640.851/1	691.766/1	4/4	797.316 /1	700.577/1	735.852 /1	4/4	19	0.55
(c)	IM	583.991/2	103.909/2	119.691/2	6.44472/4	959.116/1	892.389/1	836.31/1	4/4	3	0.97
	AG	364.386/2	468.442/2	3.58224/2	11.9814/4	641.156/1	595.714 /1	684.571 /1	735.852/4	3	0.70

Tableau 4-1: Paramètres des supershapes illustrées

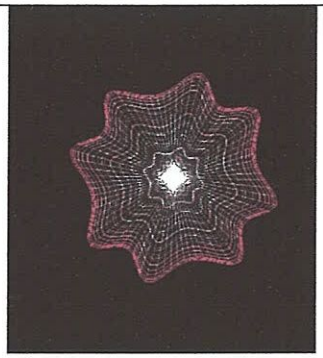
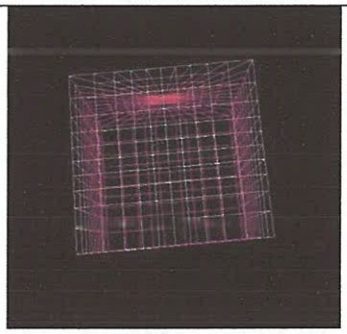
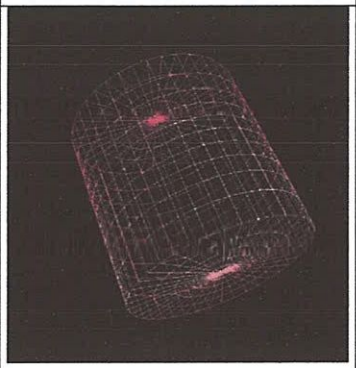
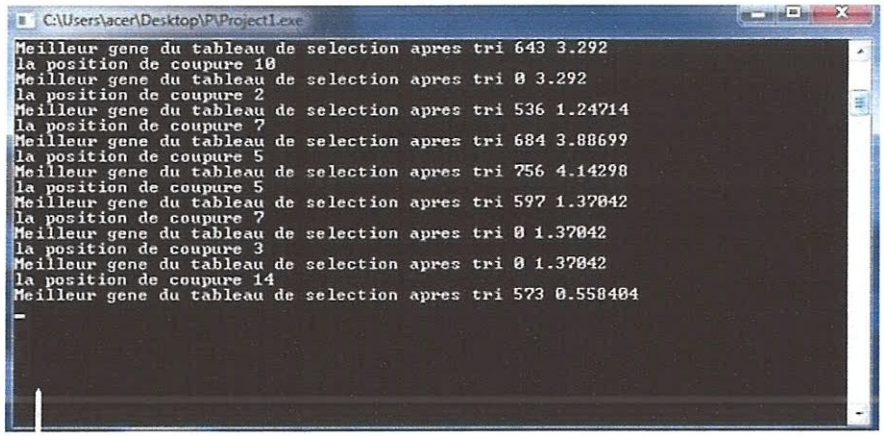
Figure	Supershape Origine	Reconstruction avec Algorithme Génétique	Reconstruction avec Système immunitaire
(a)			
(b)			
(c)			

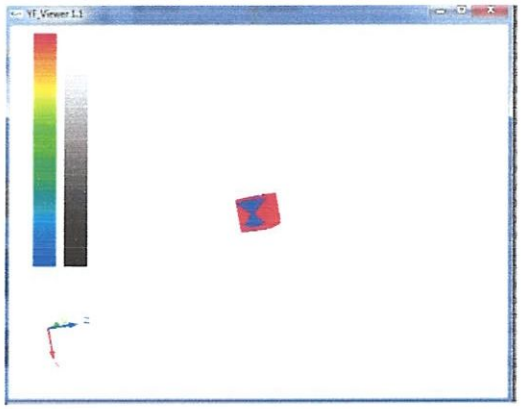
Figure 4-1: reconstruction des supershapes du tableau précédent

Sur chaque figure le nuage de points de synthèse est en rouge et la surface reconstruite par la supershape est en bleu en utilisant les algorithmes génétiques et en vert en utilisant le système immunitaire artificiel. On constate que notre algorithme permet de représenter fidèlement les données.

Cet algorithme donne des résultats aussi précis que ceux obtenus par la méthode déterministe avec un temps d'exécution acceptable (10 à 15 min pour les objets complexes) alors que pour des objets simple comme la sphère, le cube ou le cylindre la meilleure solution est trouvée des la fin de la deuxième itération au maximum.

Pour la figure 4-2, les interfaces d'exécution d'une forme complexe sont affichées et toutes les informations sont enregistrées dans un fichier Logfile.





LogFile.txt

```
 Ce gene il n'est pas male  ni= 522.223 850.944 406.612 8 824.725 770.974 697.62 8  |
tra=0 0 0  |  rot=0 0 0  |  s=1 1 1  | E = 00x475410RF=0  Total Error=1.82723

 Ce gene il n'est pas male  ni= 872.286 675.272 450.484 8 472.71 427.71 318.929 8  |
tra=0 0 0  |  rot=0 0 0  |  s=1 1 1  | E = 00x475410RF=0  Total Error=7.00599

 Ce gene il n'est pas male  ni= 773.718 573.321 382.039 8 505.21 673.199 380.149 8  |
tra=0 0 0  |  rot=0 0 0  |  s=1 1 1  | E = 00x475410RF=0  Total Error=5.70243

 Ce gene il n'est pas male  ni= 773.718 573.321 382.039 8 505.21 673.199 380.149 8  |
tra=0 0 0  |  rot=0 0 0  |  s=1 1 1  | E = 00x475410RF=0  Total Error=5.70243

 Ce gene il n'est pas male  ni= 544.418 355.484 716.858 8 554.753 630.577 452.344 7.83117
|  tra=0 0 0  |  rot=0 0 0  |  s=1 1 1  | E = 0.002496860x475410RF=0.00249686  Total
Error=3.62438

 Ce gene il n'est pas male  ni= 544.418 355.484 716.858 8 554.753 630.577 452.344 7.83117
|  tra=0 0 0  |  rot=0 0 0  |  s=1 1 1  | E = 0.002496860x475410RF=0.00249686  Total
Error=3.62438

 Ce gene il n'est pas male  ni= 739.724 776.188 927.682 8 819.572 696.949 936.981 8  |
tra=0 0 0  |  rot=0 0 0  |  s=1 1 1  | E = 00x475410RF=0  Total Error=2.70475

 Ce gene il n'est pas male  ni= 836.34 654.297 747.285 8 628.382 619.349 683.84 8  |
tra=0 0 0  |  rot=0 0 0  |  s=1 1 1  | E = 00x475410RF=0  Total Error=0.702034

 le meilleurs gene  ni= 836.34 654.297 747.285 8 628.382 619.349 683.84 8  |  tra=0 0 0  |
rot=0 0 0  |  s=1 1 1  | E = 00x475410RF=0  Total Error=0.702034
```

Figure 4-2: les interfaces des tests pour un cube

La figure suivante présente la variation de l'erreur de reconstruction obtenue à chaque itération avec le nombre d'itérations pour les différentes formes (a), (b) et (c) présentés en figure 4-3.

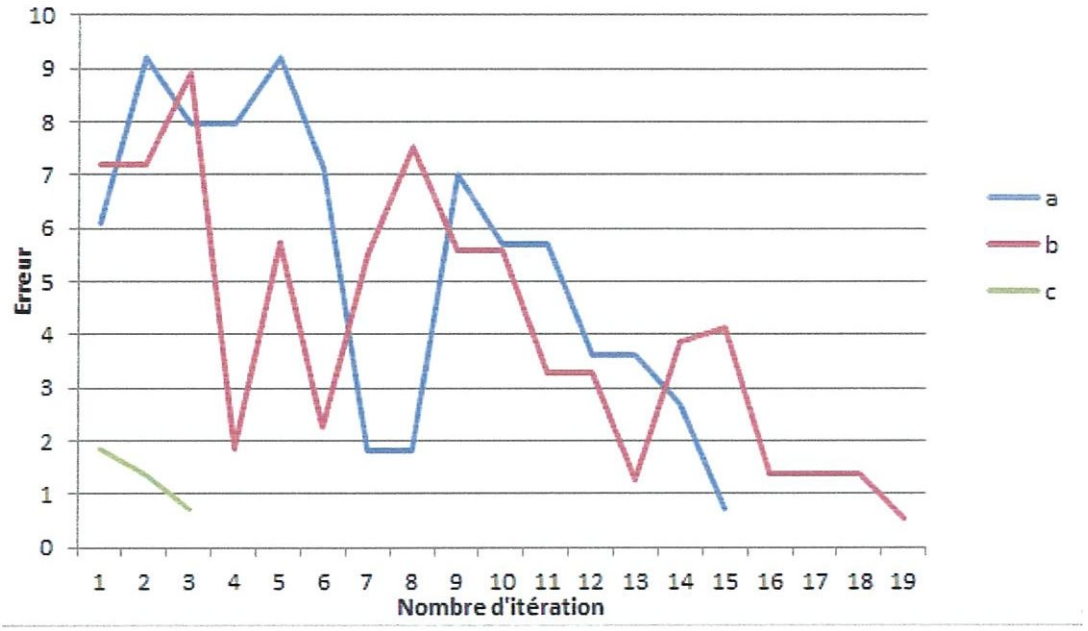


Figure 4-3: la courbe de convergence en utilisant l'AG.

Les formes asymétriques (présentent de fortes différences entre les coefficients n_1, n_2 et n_3 nécessitent un nombre d'itération plus élevé. Par exemple le résultat présenté en figure (b) nécessite 19 itérations. On peut également constaté visuellement que la forme n'est pas encore complètement reconstruite malgré la faible erreur de reconstruction. Ce phénomène se produit d'avantage pour les supershapes de forme étoilée où le ratio $\frac{n_1}{\max(n_2,n_3)}$ est important

La figure suivante 4.4 présente la variation de l'erreur de reconstruction obtenue pour chaque forme (a), (b) et (c) et une comparaison est faites entre l'utilisation d'un systèmes immunitaires artificiels et un algorithme génétique.

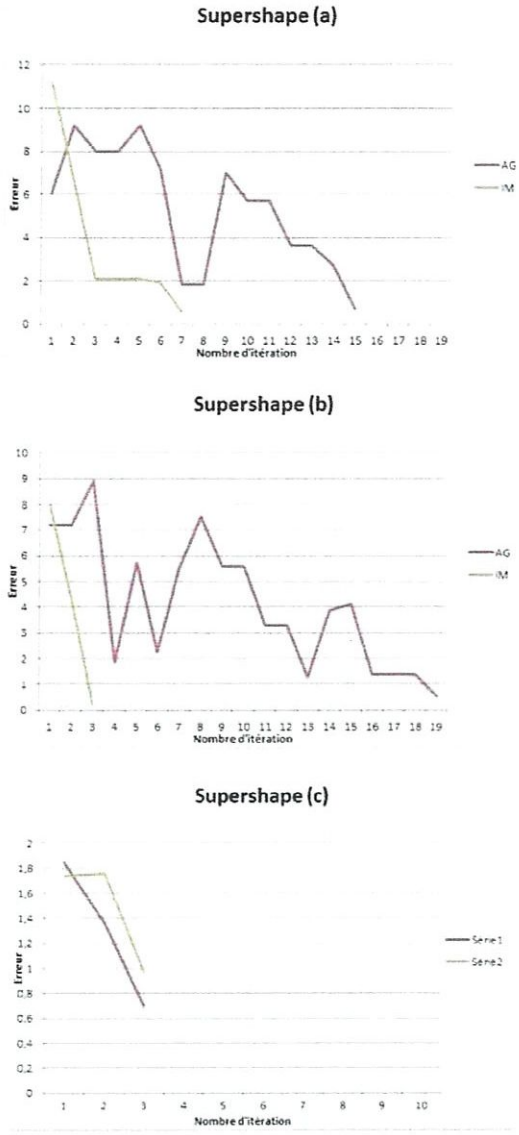


Figure 4-4: comparaison entre le AIS et GA

Pour les trois formes, afin d'y arriver à trouver une présentation optimale de notre supershape de départ, on voit bien que l'utilisation d'un système immunitaire artificiel est plus efficace que les algorithmes génétiques en terme temps (nombre d'itération est plus petit) et en terme de précision (le taux d'erreur).

Conclusion Générale

Conclusion :

Le travail proposé est une méthode de reconstruction des surfaces 3D par les supershapes Nous avons régler le problème par la combinaison des R-fonctions en une seule équation et un algorithme génétique et système immunitaire artificiels sont développées pour optimiser la fonction de coût. C'est une méthode stochastique qui permet de produire n'importe quel point de l'espace de représentation on obtient une solution définie de manière générale sans prendre en considération la densité, la topologie et la répartition des points de maillage.

La structure de notre algorithme permet de générer un nombre variable de supershapes c'est une extension du travail de Bokhabrine [6], mais malheureusement nous n'avons pas pu le tester en profondeur. Notre algorithme est adaptatif car le nombre de supershapes génères est indéfini. Les premières expériences réalisées pour la reconstruction de plusieurs supershapes semblent moins bons que ceux correspondant à une seule supershape. La solution obtenue est non optimale, alors on utilise plusieurs tests pour la régler mais on garde la taille de RAM constante à chaque itération et le temps d'exécution deviennent vite importants (10 à 15 mn pour une population de 500 chromosomes sur un maillage d'environ 2500 points). On pourrait apporter à cet algorithme des améliorations sur les tests exhaustifs de tous les paramètres êtres de rotation, La validation du comportement, La parallélisassions du code et l'étude et l'approfondissement des autres méthodes de croisement et de mutation. Le plus important est d'élargir le spectre d'essais de l'algorithme pour des objets plus complexe 3D (caractères) et optimiser le code pour accélérer l'exécution dans le cas d'un nombre important de chromosomes.

Références :

[1] Levenberg. K. A method for the solution of certain problems in least squares. Quart. Appl. Math, 2: 164–168, 1944.

[2] Marquardt, D. An algorithm for least-squares estimation of nonlinear parameters. SIAM J. Appl. Math, 11: 431–441, 1963.

[3] S.D,Booth and Fessler, J.A. Conjugate-gradient preconditionning methods for shift- variant PET image reconstruction IEEE Trans. Image Processing,vol.8(5) : 686–699, 1999.

[4] Kirkpatrick, S. Jr, D. G and Vecchi, M.P. Optimization by simulated annealing Science. 220 : 671–280, 1983.

[5] Fougerolle, Y.D. Modélisation et Reconstruction de surfaces par supershapes et R- fonction. Thèse de doctorat de l'université de Bourgogne, 2005.

[6] Bokhabrine, Y. Etude et comparaison d'algorithmes d'optimisation pour la reconstruction 3D par supershapes et R-fonctions. Rapport de DEA, 2006.

[7] Gielis, J., Beirinckx, B., and Bastiaens, E. Superquadrics with rational and irrational symmetry. In Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications :262–265, 2003. Seattle, Washington, USA.

[8] Barr,A. H, Superquadrics and angle-preserving transformations. IEEE Computer Graphics and Applications, 1(1)481–484, 1981

[9] Lorensen, W.and Cline, H. E. Marching cubes : A high resolution 3D surface construc- tion algorithm. Computer Graphics, 21(4) : 163–169, 1987.

[10] Solina, F. and Bajcsy, R. Recovery of parametric models from range images : The case for superquadrics with global deformations. The Visual Computer, 12(2) : 131–1476, 1990.

[11] Rvachev, V. L. Geometric Applications of Logic Algebra. Naukova Dumka. In Russian, 1967.

[12] Shapiro, V. Theory of R-functions and applications : A primer.Technical Report TR91-1219 Computer Science Department, Cornell University,Ithaca, NY., 1991.

[13] Deneche A. (2006). *Approches bios inspirées pour la reconnaissance de formes*, Thèse de Magister de l'université Mentouri, Constantine, Algérie.

[14] Deneche A., Meshoul S., Batouche M. (2005). Une approche hybride pour la reconnaissance de formes en utilisant un Système Immunitaire Artificiel, *Procéding des Journées d'informatique graphique,* Algérie.

[15] Emilie P. (2006). *Organisation du system immunitaire Felin*. Thèse doctorat vétérinaire, Ecole nationale de Lyon. Lyon

[16] Goodman D., Boggess L., Watkins A. (2002). Artificial immune system classification of multiple class problems. Journal *Intelligent Engineering Systems Through Artificial Neural*.

[17] Secker A., Freitas A., Timmis J. (2003). AISEC : an Artificial Immune System for E-mail Classification. Journal *IEEE Evolutionary Computation ECE'2003* pp 131-138.

[18] Jun J. H., Lee D. W., Sim K. B., (1999), Realization of cooperative and Swarm Behavior in Distributed Autonomous Robotic Systems Using Artificial Immune System. Procéding *IEEE, conférence Man and Cybernetics*.

[19] Kim J., Bently P. (2001), Towards an artificial immune system for network intrusion detection : an investigation of clonal selection with a negative selection operator, *Procéding de la conférence Congress on Evolutionary Computation*.

[20] Timmis J., (2000), *Artificial Immune Systems: A novel data analysis technique inspired by the immune network theory*, PhD, Département d'informatique, université de Wales UK

[21] Holland. H, J. Adaptation in Natural and Artificial Systems. An Introductory Ana- lysis with Applications to Biology. Control, and Artificial Intelligence, MIT Press, Cambridge, MA., 1975.

Watkins A. (2001). AIRS : *A resource limited artificial immune classifier*. Thèse de l'université de Mississipi du département d'informatique. Mississipi.

[22] Watkins A. (2001). AIRS : *A resource limited artificial immune classifier*. Thèse de l'université de Mississipi du département d'informatique. Mississipi.

[23] Aubazac, T. Optimisation de Syst`eme de Tri des Bagages dans une A´erogare. Laboratoire d'Informatique et Math´ematique Appliqu´ee. Ecole Nationale de l'Aviation Civile. Rapport de stage. Toulouse,1995.

[24] The Virtual Reality Modeling Language (VRML), ISO/IEC 14772-1:1997, 1997.

[25] The VRML Repository http://www.web3d.org/vrml/vrml.htm

[26] Ressources, tutoriels, cours VRML http://www.web3d-fr.com