Détection de primitives géométriques bidimensionnelles dans les images à l'aide d'un algorithme génétique

Evelyne LUTTON *Patrice MARTINEZ*
INRIA - Rocquencourt
B.P. 105, 78153 LE CHESNAY Cedex, France
Tel : 33 1 39 63 55 23 - Fax : 33 1 39 63 53 30
email : evelyne.lutton@inria.fr

Résumé

Nous étudions l'emploi des algorithmes génétiques dans le cadre de l'extraction de primitives (segments, cercles, quadrilatères, etc ...) dans des images. Cette approche est complémentaire de l'approche classique par transformée de Hough, dans le sens où les algorithmes génétiques se révèlent efficaces là où la Transformée de Hough devient trop complexe et trop gourmande en espace mémoire, c'est-à-dire dans les cas où l'on recherche des primitives ayant plus de 3 ou 4 paramètres.

En effet, les algorithmes génétiques, employés en tant qu'outil d'optimisation stochastique, sont réputés coûteux en temps de calcul, mais se révèlent efficaces dans les cas où les fonctions à optimiser sont très irrégulières et de forte dimensionnalité. La philosophie de la méthode que nous présentons est donc très similaire à celle de la transformée de Hough, qui est de rechercher un optimum dans un espace de paramètres. Cependant, nous verrons que les implantations algorithmiques diffèrent.

Cette approche de l'extraction de primitives par algorithmes génétiques n'est pas une idée nouvelle : nous avons repris et amélioré une technique originale proposée par Roth et Levine en 1992. Nous pouvons résumer notre apport sur cette technique en trois points principaux :

• nous avons utilisé des images de distances pour "adoucir" la fonction à optimiser (aussi appelée "fitness"),

• pour détecter plusieurs primitives à la fois, nous avons implanté et amélioré une technique de partage de la population (technique de "sharing"),

• et enfin, nous avons appliqué quelques résultats théoriques récemments établis à propos des probabilités de mutations, ce qui nous a permis d'améliorer, notamment, les temps d'exécution.

Mots-Clés : Algorithmes génétiques, Extraction de primitives, Sharing, Transformée de Hough.

1 Introduction

L'extraction de primitives géométriques est une tâche importante en analyse d'images. Elle est par exemple une tâche de base en calibration de caméras, en reconstruction tridimensionnelle stéréo, ou en reconnaissance des formes. Cette exploitation des primitives géométriques est particulièrement importante pour la vision robotique dans des scènes d'intérieur, où la plupart des objets à analyser sont manufacturés. La description de tels objets à l'aide de primitives géométriques bidimensionnelles ou tridimensionnelles est naturellement adaptée.

Notre but est ici de présenter une alternative a la transformée de Hough [11], employée pour extraire des primitives. La transformée de Hough est une méthode extrêmement efficace pour la détection de lignes ou de primitives simples dans des images (voir à titre d'exemple [13, 19, 14]), cependant, elle atteint rapidement ses limites lorsque l'on recherche des primitives plus complexes.

Cette méthode consiste à détecter des maxima dans l'espace des paramètres qui décrivent la primitive recherchée. Prenons l'exemple de la recherche de points alignés dans une image : ces points sont par exemple des points de contour détectés dans l'image. Les droites de l'images peuvent être représentées par une équation du type $cos\theta x + sin\theta y = p$ (représentation polaire d'une droite), une droite est donc définie par deux paramètres : θ et p. Chaque point de l'image peut faire partie d'un certain nombre de droites, et si l'on se place dans l'espace des paramètres (θ, p), ces droites sont représentées par une courbe (une sinusoïde en l'occurence). On construit donc ce que l'on appelle un accumulateur, qui repésente l'espace des paramètres (θ, p) divisé en "cellules". Pour chaque point de l'image, on incrémente les cellules de l'accumulateur qui représentent toutes les droites qui peuvent passer par ce point. La recherche des cellules-maxima de l'accumulateur fournit les droites qui sont effectivement présentes dans l'image analysée.

D'un point de vue plus général, il existe deux techniques classiques équivalentes pour remplir l'accumulateur :

- *l'approche 1 à m*, où pour un point de l'image, on "trace" une courbe dans l'espace des paramètres (sur l'accumulateur), qui repésente tous les paramètres des primitives passant par le point en question, c'est la technique que nous venons de décrire dans l'exemple de la détection d'une droite,
- *l'approche m à 1*, appelée aussi *transformée de Hough randomisée* or *combinatoire*, où, pour chaque m-uplet possible de points dans l'image (pour chaque couple de points, si l'on recherche des droites), il correspond un point de l'espace des paramètres, qui représente la primitive qui passe par le m-uplet de points.

La détection effective des primitives, c'est à dire la recherche des maxima dans l'espace des paramètres, est faite par un parcours séquentiel direct de l'accumulateur. Si l'on recherche des cercles, il faut construire un accumulateur de dimension 3 (représentant par exemple les coordonnées du centre du cercle et son rayon), il est donc bien évident que cette technique devient rapidement gourmande en espace mémoire, en fonction de la complexité des primitives à extraire.

Pour résumer, la transformée de Hough est donc une technique très rapide et très précise pour les primitives géométriques simples, mais il devient rapidement difficile de stocker un accumulateur et de détecter des maxima sur celui-ci lorsque le nombre de paramètres à estimer augmente.

Ainsi, il paraît nécessaire de considérer des techniques efficaces d'optimisation si l'on veut traiter le problème de la détection de primitives géométriques complexes. Ce problème peut être directement formulé comme un problème d'optimisation : optimiser la position et la taille d'une primitive géométrique (ou de façon équivalente, optimiser les valeurs des paramètres la définissant), connaissant les points de contours détectés dans l'image. La fonction qui est implicitement optimisée dans la transformée de Hough est une fonction de comptage des coïncidences entre la primitive candidate et les points de contours présents dans l'image.

Un problème supplémentaire vient du fait que, lorsque la dimension de l'espace de recherche est grande, la fonction à optimiser peut devenir très irrégulière.

Lorsqu'une fonction possède un certain type de régularité, de nombreuses méthodes d'optimisation existent, la plupart du temps basées sur des calculs de gradient ou de gradient généralisé (voir par exemple [4]). Les méthodes fondées sur le gradient généralisé sont efficaces lorsque :

- un gradient peut être défini et calculé en tout point de l'espace des solutions (par exemple, des dérivées directionnelles),
- la fonction ne possède pas trop de maxima locaux, ou bien la valeur de la fonction sur ces maxima est significativement plus faible que la valeur du maximum absolu.

Pour des fonctions très irrégulières, il est nécessaire d'employer des méthodes différentes, la plupart du temps fondées sur un schéma stochastique.

L'un des algorithmes d'optimisation stochastique les plus connus est le *recuit simulé*. Cette technique d'optimisation globale se révèle puissante lorsque les fonctions à optimiser dépendent d'un grand nombre de paramètres. Elle est fondée sur une analogie avec le recuit physique des solides, qui consiste à chauffer un matériau à haute température, et à le faire refroidir très lentement afin de laisser le système atteindre son énergie minimale. Le point délicat est de ne pas refroidir le température T trop rapidement, afin d'éviter les minima locaux de l'énergie. La transposition de cette technique pour des problèmes d'optimisation est faite grâce à un parallèle entre les états du système physique et des états définis sur le système à optimiser, et grâce à une généralisation de la notion de température en un paramètre de contrôle pour le processus d'optimisation. La plupart du temps, on emploie l'algorithme de Metropolis : à la température T, le saut d'énergie E à E' se fait avec une probabilité 1 si $E' < E$ et avec une probabilité proportionnelle à $e^{(E-E')/T}$ sinon ([1, 16]).

Le principal inconvénient du recuit simulé est le coût en temps de calcul : la solution optimale n'est garantie que si la température est diminuée avec un taux logarithmique ([6]), nécessitant donc un grand nombre d'itérations. La plupart du temps, un taux linéaire est utilisé pour obtenir des temps de convergence raisonnables, mais pour certaines fonctions très irrégulières, il est nécessaire de faire décroître la température plus lentement.

Dans cet article, nous nous intéressons à l'emploi de techniques d'optimisation stochastique fondées sur une analogie avec l'évolution des systèmes biologiques, communément appelées Algorithmes Génétiques [9, 18, 17]. Dans la section 2 nous présentons les caractéristiques de l'algorithme génétique que nous avons développé pour la détection des primitives dans les images. En section 3, nous décrivons la méthode de sharing que nous avons employée, puis présentons des résultats de détection en section 4.

2 L'AG de détection des primitives

L'intérêt d'utiliser des algorithmes génétiques (AG) pour optimiser des fonctions irrégulières vient du fait qu'ils peuvent effectuer une recherche stochastique dans un espace très large, grâce à l'évolution d'une population de solutions (contrairement au recuit simulé qui fait évoluer une seule solution). Il faut cependant remarquer que les algorithmes génétiques, aussi bien en tant que technique d'optimisation stochastique que sous la forme de systèmes de classeurs ou de programmation génétique, ont été relativement peu employés dans le cadre de la vision robotique et de l'analyse d'images.

Nous décrivons ici une implantation séquentielle de la détection de primitives par AG, une version parallèle de cet algorithme est actuellement en cours d'implantation sur le calculateur KSR de l'INRIA.

L'AG que nous employons est classique, dans le sens où il fait évoluer (par sélection élitiste avec scaling, et par croisement et mutation) des solutions *codées* sous forme de chromosomes. La représentation discrète convient en général bien aux application en traitement des images, où l'on a à traiter des données initiales sous forme de matrices de pixels. Dans ce qui suit nous détaillons les modifications que nous avons apportées par rapport au schéma de Goldberg [7], ainsi que les composantes (codage des solutions et calcul de la fonction de fitness) spécifiques à notre application.

2.1 La représentation chromosomique : le codage des primitives

Classiquement pour l'emploi d'un AG en optimisation, le chromosome est directement la concaténation des codes binaires des composantes du vecteur représentant un point de l'espace de recherche. Cet espace doit donc être borné. Dans le cas où l'espace de recherche est "continu", il est échantillonné avec une certaine précision, le choix de la précision d'échantillonnage étant parfois délicat [20].

L'approche de Roth et Levine [17] est fondée sur le codage des primitives par un certain nombre de points de contour de la primitive (permettant de la définir uniquement). La représentation d'une primitive par un ensemble minimal de points rend l'étape d'extraction moins sensible au bruit et le codage trivial : un chromosome est la concaténation des coordonnées de ces points. Cependant cette représentation a un inconvénient majeur qui est la redondance : la même primitive peut être représentée par un grand nombre de chromosomes différents.

Nous avons donc préféré une représentation pour laquelle la correspondance primitive/chromosome est bijective, simplifiant ainsi la tâche de l'AG. Voici les codages que nous avons employés pour la détection des primitives suivantes :

- **Segment** : 2 points de l'image : $I = [0..x_{max}, 0..y_{max}]$, de coordonnées entières, représentant les extrémités du segment,
- **Cercle** : 1 point de I pour le centre du cercle, et un entier positif de $[0.. \max(x_{max}, y_{max})]$ pour son rayon,

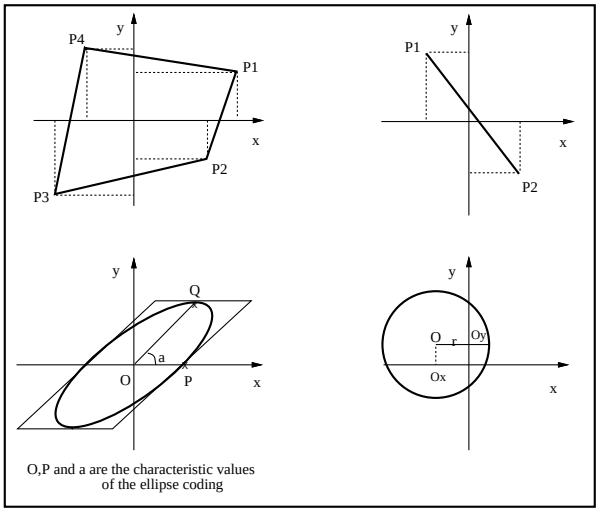


Figure 1: Codage graphique d'un segment, d'un cercle, d'un quadrilatère et d'une ellipse

- **Ellipse** : 2 point de I, le centre O et le point P, un réel positif a, entre 0 et $\frac{\pi}{2}$ (échantillonné sur 8 bits), représentant l'angle de rotation de l'ellipse. Les caractéristiques de ce codage classique en synthèse d'images sont détaillées dans [12] (voir figure 1),
- **Rectangle** : 2 points de I, coordonnées des sommet supérieur-gauche et inférieur-droit. Ce rectangle est parallèle aux axes de l'image. Pour différentes orientations, on adjoint un réel positif, entre 0 et $\frac{\pi}{2}$, représentant l'angle de rotation de la figure.
- **Quadrilatère** : 4 points de I, pour les 4 sommets.

2.2 Les opérateurs génétiques

La création d'un individu se fait par *sélection* (avec scaling) de deux "parent", le critère de sélection étant donné par la fonction de fitness, puis par application des opérateurs génétiques: le *croisement* et la *mutation*. Ces deux opérateurs sont appliqués aléatoirement. Pour le croisement, la probabilité de croisement est fixée entre 0.7 et 0.9. Nous avons testé deux types de croisement classiques, le croisement à un site et le croisement uniforme (qui donnent des résultats comparables pour nos expériences).

La probabilité de mutation est classiquement fixée à une valeur très faible tout au long de l'algorithme. Des résultats récents [5] sur la convergence des AG à population finie et probabilité de croisement constante assurent théoriquement une convergence vers l'optimum global, si la probabilité de mutation $p_m(k)$ suit une loi de décroissance en fonction de la génération k, qui ne décroît pas plus rapidement que:

$$p_m(k) = \frac{1}{2} * k^{-\frac{1}{M*L}}$$

M est la taille de la population, et L, la longueur des chromosomes.

Bien évidemment, ce taux de décroissance est très lent (voir figure 2), et un nombre infini de générations est nécessaire pour assurer la convergence de l'AG vers l'optimum global. D'un point de vue pratique, tout comme dans le cas des implantations du recuit simulé, on utilise un taux de décroissance plus rapide. Nous avons utilisé la décroissance donnée par la formule:

$$p_m(k) = p_m(0) * \exp(-\frac{k}{\alpha})$$

$p_m(0)$ est la probabilité de mutation initiale, α est calculé pour donner un taux final de mutation très bas (nous avons imposé 10^{-4}) :

$$\alpha = \frac{\text{Nb Max de Générations}}{\ln \frac{p_m(0)}{10^{-4}}}$$

La courbe en traits continus de la figure 2 représente la fonction de décroissance que nous avons adoptée, elle est dessinée pour une évolution sur 100 générations (avec une probabilité initiale de mutation de 0.25). A titre de comparaison, est tracée la courbe théorique (en pointillés) sur 100 générations et sur 1000 générations, pour une longueur de chromosome de 32 bits.

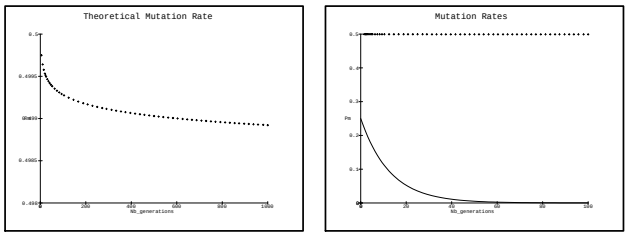


Figure 2: A gauche, la courbe théorique de Davis (pointillés) *sur 1000 générations*, à droite, comparaison avec celle que nous adoptons (en traits continus) *sur 100 générations*

Le taux de décroissance que nous proposons permet une large exploration de l'espace de recherche au début de l'algorithme, puis une convergence plus rapide à la

fin de l'évolution de l'AG, en comparaison avec la version laissant constante la probabilité de mutation tout au long de l'algorithme. Ce taux de décroissance relativement grossier est bien évidemment dépendant de la forme de la fonction à optimiser. Si cette fonction est trop irrégulière, un taux de décroissance moins rapide sera nécessaire, exactement comme dans le cas du recuit simulé.

2.3 Calcul de la fonction de fitness

En ce qui concerne le calcul de la fonction d'évaluation des primitives, nous avons préféré employer une image de distances au lieu de faire directement un comptage des coïncidences entre les points de l'image de contours et la trace de la primitive. En effet, pour tolérer les petites erreurs, il est souvent nécessaire d'effectuer un comptage sur une bande centrée sur la primitive, , voir figure 3, ce qui augmente le temps de calcul nécessaire pour une évaluation de la fonction de fitness.

En outre, la forme de la fonction de fitness calculée sur l'image originale est très irrégulière, et une primitive proche d'un contour réel mais qui ne coïncide pas avec celui-ci (parallèle par exemple), n'a pas plus d'informations qu'une primitive franchement éloignée de celui-ci. La convergence d'un AG dans ce cas risque d'être très lente, tout particulièrement lorsque les contours sont peu denses dans l'image.

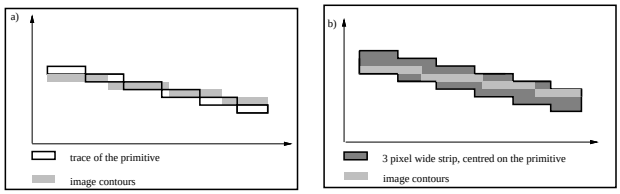


Figure 3: Calcul de la fonction de fitness sur des images de contours

Une image de distances est une image en niveaux de gris, calculée à partir de l'image des contours, qui donne en chaque pixel la distance de ce pixel au point de contour le plus proche. Ces images de distance sont assez largement employées en morphologie mathématique, et sont créées simplement par convolution de l'image des contours avec deux masques, [3]. Les distances ainsi calculées sont paramétrées par $d1$ et $d2$, voir figure 4, qui représentent les deux distances élémentaires entre pixels verticaux/horizontaux et diagonaux.

Nous employons les distances de Chamfer ($d1 = 3$ et $d2 = 4$), ou des distance plus "abruptes ($d1 = 10$ et $d2 = 14$), voir figure 5. L'ajustement des paramètres $d1$ et $d2$ dépend de la densité de contours dans l'image des contours, et l'on pourrait facilement imaginer une technique d'ajustement automatique de ces paramètres (nous ne l'avons pas encore implantée).

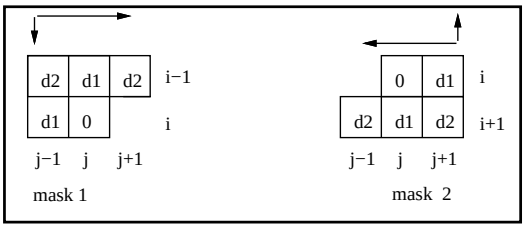


Figure 4: Masques de distances

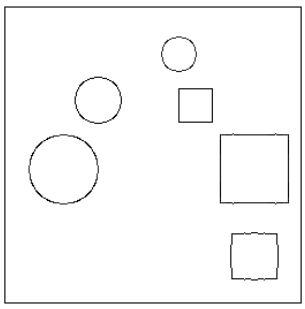
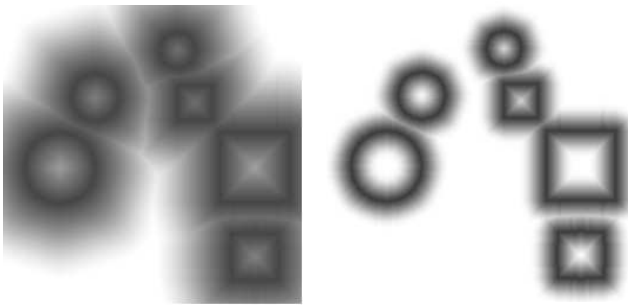


Image de contours



distance de Chamfer (3,4) distance (10,14)

Figure 5: Exemple d'images de distances sur des contours de synthèse

L'intérêt d'utiliser de telles images de distance est double : tout d'abord, la fonction de fitness est plus rapide à calculer (un calcul sur les traces des primitives est suffisant), et ensuite, la tolérance aux petites erreurs est améliorée. La fonction de fitness est une combinaison de deux termes : l'intensité moyenne des pixels de l'image de distances en coïncidence avec la

trace de la primitive (pour positionner la primitive), plus un terme de comptage des pixels de contour sur la trace (pour favoriser les primitives les plus longues).

2.4 Analyse de la population finale : extraction des solutions

Une des caractéristiques importantes de la convergence des AG est qu'il leur arrive de "visiter" plusieurs bons optima locaux, avant de converger vers l'optimum global. Plutôt que de rechercher directement le chromosome représentant l'optimum absolu de la population finale, une analyse plus poussée peut fournir des informations importantes, particulièrement si l'on a stoppé l'algorithme avant convergence complète.

Cela est intéressant tout particulièrement dans notre cas de figure, où l'on recherche plusieurs optima, locaux ou globaux. Nous avons donc développé une technique simple de classification, qui permet de localiser plusieurs optima s'ils sont présents dans la population finale. De tels comportements de convergence peuvent être favorisés et contrôlés grâce aux méthodes de "sharing", nous décrivons ces méthodes et l'emploi que nous en avons fait dans la section qui suit.

3 Emploi d'une technique de sharing

La méthode d'extraction de primitives, telle que nous venons de la décrire, ne permet de détecter qu'une primitive à chaque fois que l'on lance l'AG. Pour détecter toutes les primitives présentes dans l'image, il est nécessaire d'itérer le processus, en mettant à jour l'image de contours (en "éliminant" le contour correspondant à la primitive qui vient d'être détectée), en re-générant l'image de distance correspondant à cette nouvelle image de contours, et en lançant de nouveau l'AG sur ce nouvel environnement. Le processus est stoppé lorsqu'il n'y a plus de contours dans l'image, ou lorsque les meilleures primitives détectées ont une fonction de fitness inférieur à un certain seuil (taille trop petite, par exemple).

L'intérêt de détecter plusieurs primitives au cours d'un même AG est, dans ce cadre, évident. Pour cela, nous proposons d'employer une technique de sharing, complétée par la classification de la population finale que nous avons évoquée en section 2.4.

Les techniques de sharing [7, 8] simulent à l'aide d'opérateurs simples, le phénomène naturel de création de niches écologiques et de sous-espèces. En considérant schématiquement que les individus d'une même sous-population doivent partager les ressources locales, lorsqu'il y a surpopulation, les ressources locales diminuent, et les individus tendent à coloniser d'autres régions, créant ainsi des sous-espèces.

De nombreuses solutions ont été proposées, créant implicitement ou explicitement des niches. Plus précisément, ces approches peuvent être divisées en deux grands courants. Le premier, plus ancien, représente les techniques de maintien de la diversité (en interdisant la cohabitation d'individus trop proches les uns des autres) tout au long de l'évolution de l'AG, ce qui favorise dans une certaine mesure la création de sous-populations séparées. Le second courant utilise une modification de la fonction de fitness de façon à simuler effectivement un partage des ressources locales dans la population (Goldberg et Richardson en 1987 [8]). Cette approche est basée sur une notion de distance entre individus : la valeur de fitness d'un individu est diminué en fonction du nombre de ses voisins, par l'intermédiaire d'une fonction de sharing.

Ces méthodes de sharing on été étudiées assez précisément ces cinq dernières années, et leur capacité à détecter plusieurs optima a été démontrée (analyse par chaîne de Markov en 1993 [10], par exemple). D'une certaine façon, les techniques de sharing assurent que plusieurs optima serons "colonisés" s'ils existent dans l'environnement.

3.1 La fonction de sharing

La fonction de sharing permet de déterminer la dégradation de la valeur de fitness d'un individu, due à la présence d'un voisin à une certaine distance. Il faut donc définir une notion de distance sur notre espace de recherche, qui peut être directement calculée sur les chromosomes (distance génotypique), ou bien être définie sur l'espace de recherche lui-même (distance phénotypique).

Dans notre cas, nous avons préféré employer une distance entre primitives images, c'est à dire une distance phénotypique. En fait, les distances phénotypiques lorsqu'on peut les utiliser semblent les plus performantes [8].

On définit, à partir de la fonction distance, des voisinages flous, par l'intermédiaire de fonctions d'appartenance $sh()$. Nous employons la fonctions d'appartenance proposée par Goldberg et Richardson [8], qui dépend de deux constantes : σ_{share} qui règle la largeur du voisinage, et α qui règle sa forme, voir figure 6.

$$Sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha} & \text{si } d < \sigma_{share} \\ 0 & \text{sinon} \end{cases}$$

Le sharing est tout simplement implanté par modification de la valeur de fitness, en le divisant par un

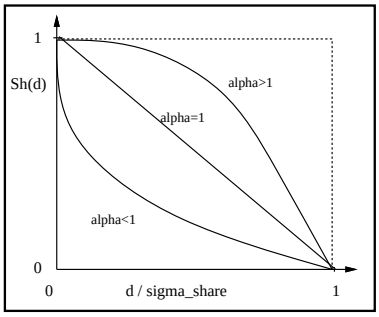


Figure 6: Forme des voisinages en fonction de α

comptage des voisins m_i :

$$\text{NewFitness}(i) = \frac{\text{Oldfitness}(i)}{m_i}$$

$$m_i = \sum_{k=1}^{N} Sh(d_{ik})$$

L'effet du sharing est de séparer la population en sous-populations de tailles proportionnelles à la hauteur du pic qu'elles colonisent. Goldberg et Richardson [8] on expliqué qu'un sharing est stabilisé lorsque $\frac{f_h}{m_h} = \frac{f_k}{m_k}$ ($\forall h, k$ avec h et k des optima locaux différents), voir figure 7.

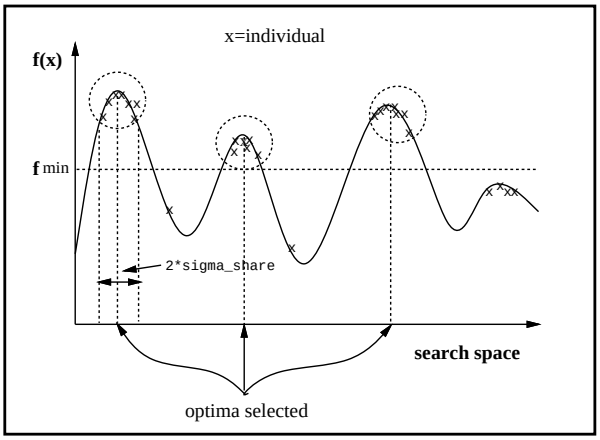


Figure 7: Répartition de la population après sharing

L'efficacité du sharing dépend principalement du réglage du paramètre σ_{share}, qui est une mesure de la séparation minimale tolérée entre deux pic détectés.

3.2 Sharing avec restriction des croisements

Si l'on observe l'évolution d'un AG avec sharing sur une fonction multimodale simple, on peut remarquer qu'un certain nombre d'individus on tendance à "errer" entre les pic colonisée par des sous-populations. Cela est dû au fait que le croisement d'individus de deux niches différentes ne permet que rarement d'obtenir un "bon" individu. Booker [2] a proposé de limiter le croisement des individus de différentes sous-populations, en modifiant l'étape de sélection de l'AG de la façon suivante :

- choix du premier parent (sélection sur la fonction de fitness classique),
- parcours de la population pour localiser la sous-population d'individus qui sont à une distance inférieure à σ_{cross} du premier parent (σ_{cross} est souvent pris égal à σ_{share}),
- si la taille de cette sous-population est supérieure à 1, on applique la sélection élitiste classique dans cette sous-populations, sinon, le second parent est sélectionné dans la population totale.

Goldberg et Richardson [8] ont prouvé que sur des fonctions multimodales simples, ce schéma améliore l'efficacité du sharing, fait que nous avons expérimentalement vérifié dans notre application.

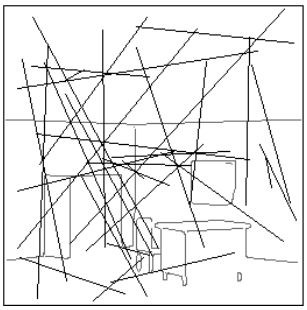


Figure 8: Population initiale aléatoire de segments

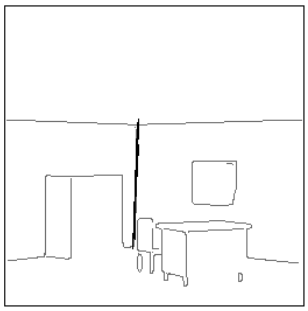


Figure 9: Convergence avec AG sans sharing (80 generations)

3.3 Une technique de sharing modifiée

Nous proposons un schéma de sharing modifié, fondé sur l'importance relative (en termes de fitness) des individus dans le voisinage de sharing. Ce schéma modifié est fondé sur la constatation qu'un individu d'une niche écologique ayant une valeur de fitness faible aura peu d'influence dans l'évolution de cette niche particulière, et aura tendance à disparaître au bénéfice

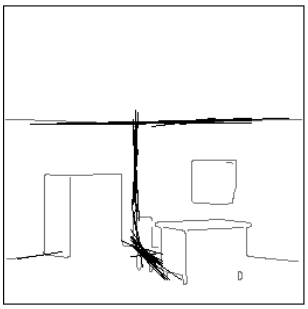


Figure 10: Convergence avec sharing

d'individus mieux adaptés. Goldberg et Richardson prennent en compte uniquement le nombre de voisins pour partager la valeur du fitness. Nous préférons tenir aussi compte de la "force" de ces voisins. Le sharing modifié que nous proposons est donc :

$$\text{NewFitness(i)} = \frac{\text{OldFitness(i)}}{\mu_i}$$

$$\mu_i = \sum_{k=1}^{N} (\text{OldFitness}(k) * Sh(d_{ik}))$$

μ_i représente une moyenne floue (une sorte de densité) dans le voisinage de l'individu i. Ainsi $\frac{fitness(i)}{\mu_i}$ devient une mesure de l'importance relative de l'individu par rapport à son entourage. En suivant le même raisonnement que dans [8], on peut dire que le processus d'évolution s'est stabilisé quand :

$$\frac{fitness(pic_h)}{\mu_{pic_h}} = \frac{fitness(pic_k)}{\mu_{pic_k}}$$

Ce schéma a tendance à équilibrer les sous-populations dans les pics, indépendamment des hauteurs des pics, à partir du moment où ceux-ci sont plus hauts qu'un certain seuil.

Cet effet particulier permet de coloniser plus de pics avec la même taille de population qu'avec le sharing classique, où le meilleur pic a tendance à attirer beaucoup plus d'individus, réduisant ainsi le nombre d'individus pouvant coloniser d'autres pics.

Ce comportement est très intéressant dans le cadre de notre application, et nous avons observé une nette amélioration des performances de l'AG avec ce sharing modifié, par rapport à la technique de sharing classique. Par exemple, pour une population de 100 individus (figures 8, 9, et 10), on peut détecter directement 4 à 7 optima locaux en une passe. Evidemment, l'AG avec sharing est plus coûteux en temps de calcul, mais permet de détecter plusieurs primitives à la fois, ce qui améliore nettement l'efficacité de la méthode d'un point de vue global.

4 Résultats

Nous présentons ici des résultats sur images de synthèse et sur images réelles pour quatre types de primitives :

- **segments** : figures 13 et 21, une passe de l'algorithme (qui localise 4 à 12 segments en même temps) prend 10 à 15 secondes sur une station Sparc II,
- **cercles** : figure 14, 70 à 80 secondes par passe,
- **ellipses** : figures 22 et 23,
- **rectangles** : figures 15 et 18.

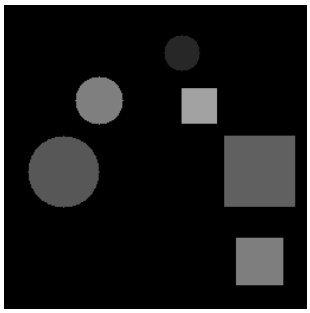


Figure 11: Image de synthèse

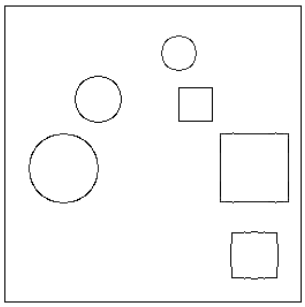


Figure 12: Contours

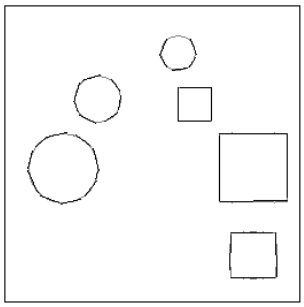


Figure 13: Segments détectés (continu) et contours (pointillés)

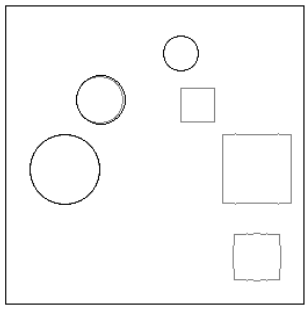


Figure 14: Cercles détectés (continu) et contours (pointillés)

5 Conclusion

La méthode que nous venons de présenter est complémentaire de la transformée de Hough dans le sens

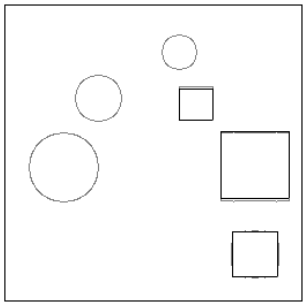


Figure 15: Rectangles détectés (continu) et contours (pointillés)

où pour des primitives simples (moins de 4 paramètres), les AG sont moins efficaces que la transformée de Hough. Les AG sont bien plus intéressants et simples à utiliser pour des primitives complexes (cercles, ellipses, quadrilatères, etc ...). L'adaptation de notre algorithme à d'autres types de primitives se fait très aisément, en modifiant la fonction de fitness et la fonction distance.

Dans le même ordre d'idées, on peut penser à des applications du type transformée de Hough généralisée, où l'on recherche dans les images des formes non paramétriques. La transformée de Hough généralisée construit un accumulateur représentant les translations et les rotations possibles de cette forme. Une fois encore la transformée de Hough se trouve habituellement limitée par le nombre de paramètres (on optimise les paramètres de translation *ou* de rotation *ou* de dilatation). Une approche par AG fournit un outil plus puissant, qui pourrait permettre d'optimiser des paramètres de déplacement et de déformation simultanément.

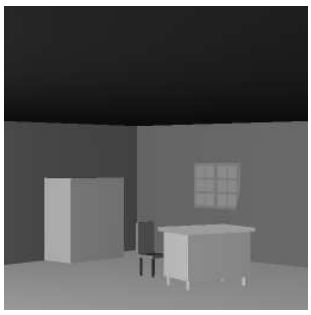


Figure 16: Image de synthèse

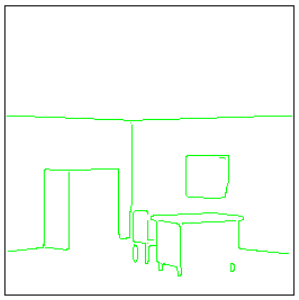


Figure 17: Contours

La formulation particulière des approches à base d'AG permet d'utiliser facilement des outils comme les images de distance, qui améliorent l'aspect de la

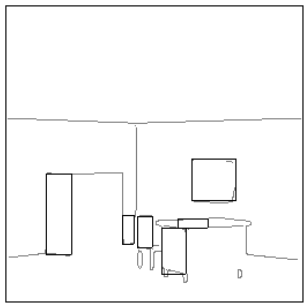


Figure 18: Rectangles détectés (continu) et contours (pointillés)

Figure 19: Image réelle

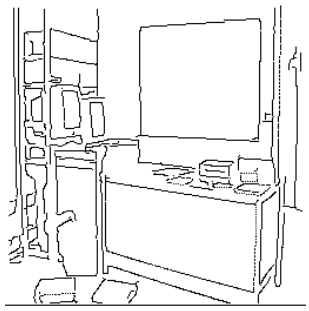


Figure 20: Contours

fonction à optimiser, et de ce fait diminuent les temps de convergence. Une différence de taille par rapport aux autres techniques d'optimisation est que l'on peut très facilement exploiter au sein d'AG des techniques comme le sharing, pour améliorer l'efficacité de la recherche d'optima sur des fonctions de fitness multimodales.

Le principal problème de telles approches reste cependant le réglage des paramètres, qui a une très grande influence sur la vitesse de convergence et la qualité des résultats. Sauf en ce qui concerne la probabilité de mutation où nous pouvons employer certains résultats théoriques, ce réglage est fait expérimentalement, et dépend du type de primitives. Ce problème du choix judicieux des paramètres constitue l'un des thèmes actuels de recherche sur les AG.

Notons enfin qu'un autre point à mentionner à l'avantage de ces techniques concerne la parallélisation, qui permet d'améliorer facilement et dans une mesure importante les performances.

Nous espérons avoir mis en évidence dans cet article l'attrait des approches génétiques pour les problèmes d'optimisation complexes que l'on rencontre en traitement d'images et en vision robotique. Nous ne prétendons absolument pas remplacer par des AG

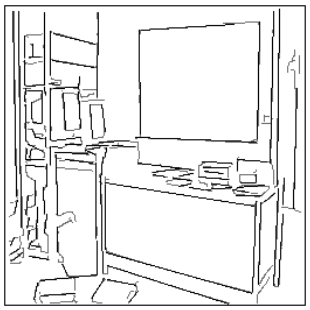


Figure 21: Segments détectés (continu) et contours (pointillés)

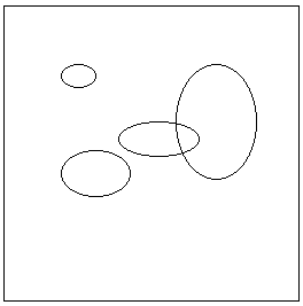


Figure 22: Contours de synthèse

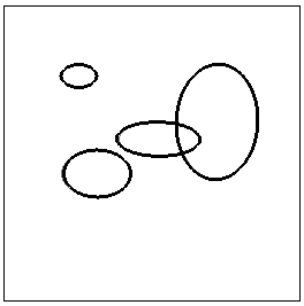


Figure 23: Ellipses détectées

(souvent trop coûteux et trop lents) des techniques bien connues qui on fait leurs preuves en traitement d'images, mais nous pensons que les approches génétiques doivent être considérées en tant qu'approches complémentaires, essentiellement sur des problèmes qui se révèlent trop complexes pour les techniques classiques.

References

[1] E. Aarts and P. Van Laarhoven. Simulated annealing : a pedestrian review of the theory and some applications. *AI Series* F30, NATO.

[2] L. B. Booker. *Intelligent behavior as an adaptatition to the task environment.* PhD thesis, University of Michigan, Logic of Computers Group, 1982.

[3] Gunilla Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27:321–345, 1984.

[4] A. Bihian C. Lemarechal, J.J. Strodiot. *On a bundle algorithm for nonsmooth optimization*, pages 245–282. Academic Press, 1881. Non-Linear Programming 4, Mangasarian, Meyer, Robinson, Editeurs.

[5] T. E. Davis and J. C. Principe. A Simulated Annealing Like Convergence Theory for the Simple Genetic Algorithm . In *Proceedings of the Fourth International Conference on Genetic Algorithm*, pages 174–182, 1991. 13-16 July.

[6] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6):712–741, November 1984.

[7] D. A. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning* . Addison-Wesley, January 1989.

[8] David E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications*, pages 41–49, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[9] J. H. Holland. *Adaptation in Natural and Artificial System* . Ann Arbor, University of Michigan Press, 1975.

[10] J. Horn. Finite Markov Chain Analysis of Genetic Algorithms with Niching . IlliGAL Report 93002, University of Illinois at Urbana Champaign, February 1993.

[11] P. V. C. Hough. A new method and means for recognizing complex pattern, 1962. U. S. Patent 3,0690,654.

[12] A. Van Dam J.D. Foley. Computer graphics - principles and practise.

[13] V. F. Leavers. The dynamic generalized hough transform for the concurrent detection of circles and ellipses. In *Progress in Image Analysis and Processing II.* 6th International Conference on IASP, 1991.

[14] Evelyne Lutton, Henri Maitre, and Jaime Lopez-Krahe. Determination of vanishing points using hough transform. *PAMI*, 16(4):430–438, April 1994.

[15] Evelyne Lutton and Patrice Martinez. A genetic algorithm for the detection of 2d geometric primitives in images. Research Report 2110, INRIA, November 1993.

[16] R. Otten and L. Van Ginneken. Annealing : the algorithm . Technical Report RC 10861, March 1984.

[17] G. Roth and M. D. Levine. Geometric Primitive Extraction Using a Genetic Algorithm . In *IEEE Computer Society Conference on CV and PR*, pages 640–644, 1992.

[18] Gerhard Roth and Martin D. Levine. Extracting geometric primitives. *CVGIP: Image Understanding*, 58(1):1–22, July 1993.

[19] Frank Tong and Ze-Nian Li. On improving the accuracy of line extraction in hough space. *International journal Of Pattern Recognition and Artificial Intelligence*, 6(5):831–847, 1992.

[20] J. Lévy Véhel and E. Lutton. Optimization of fractal functions using genetic algorithms. In *Fractal 93*, 1993. London.