# L'algorithme des mouches: une Stratégie d'Evolution Individuelle appliquée en Stéréovision

# The Fly Algorithm: an Individual Evolutionary Strategy applied to Stereovision.

Jean Louchet
Ecole Nationale Supérieure de Techniques Avancées
32 boulevard Victor
75739 Paris cedex15, France
jean.louchet@ensta.fr
http://www.ensta.fr/~louchet

## Résumé

*Cet article présente l'application d'une stratégie d'évolution individuelle (variante d'algorithme génétique) à la résolution rapide approchée de certains problèmes d'analyse de scènes. L'exemple choisi est la détection d'obstacles par stéréovision. La population est un ensemble de points ("mouches") dans le champ des caméras. La fonction de performance favorise les mouches situées sur les surfaces apparentes des objets. L'algorithme utilise des opérateurs classiques de partage, mutation et croisement. Le résultat est un ensemble de points de la population, représentant les surfaces des objets. Des résultats expérimentaux sont présentés ainsi que les extensions en cours à la vision d'un robot mobile et au suivi d'objets en déplacement.*

## Mots Clef

Algorithmes génétiques, stratégies d'évolution, approche individuelle, vision robotique, détection d'obstacles.

## Abstract

*This paper presents an Individual Evolutionary Strategy devised for fast image analysis applications. The example problem chosen is obstacle detection using a pair of cameras. The algorithm evolves a population of three-dimensional points ('flies') in the cameras fields of view, using a low complexity fitness function giving highest values to flies likely to be on the surfaces of 3-D obstacles. The algorithm uses classical sharing, mutation and crossover operators. The final result is a fraction of the population rather than a single individual. Some test results are presented and potential extensions to real-time image sequence processing, mobile objects tracking and mobile robotics are discussed.*

## Keywords

Genetic Algorithms, Evolutionary Strategies. Individual Approach, Robot Vision, Obstacle Detection.

## 1 Introduction

### 1.1 Segmentation et Analyse de scènes

La voie classique en vision par ordinateur et analyse de scènes s'appuie sur l'extraction de primitives géométriques des images ("segmentation") par des calculs sur les pixels. On peut tenter de définir l'Analyse de Scènes comme le processus de reconstruction d'un modèle d'une scène tridimensionnelle, modèle qui sera exprimé en termes de primitives géométriques et d'attributs physiques (p.ex. photométriques), généralement en utilisant les résultats de la segmentation (en contours, régions, etc.) d'une ou plusieurs images, pour construire le modèle.

Ainsi en Stéréovision, les résultats de segmentation de deux ou plusieurs images prises de différents points de vue sont comparés et mis en correspondance pour exploiter les différences géométriques (souvent faibles) entre images et construire une représentation tridimensionnelle de la scène, par exemple à l'aide de primitives géométriques (polyèdres, etc.) . Ce résultat est ensuite exploité dans des applications telles que la planification, le suivi de routes, l'évitement d'obstacles... Cette approche "classique", qui a donné lieu à un travail considérable (voir par exemple [2], [6], [8]), requiert une puissance de calcul importante.

### 1.2 Espace dual et transformée de Hough

On peut voir l'approche de Hough [10] et de ses nombreux successeurs [1] comme une alternative à cette approche classique de la vision. L'idée générale de la transformation de Hough généralisée est de considérer la scène physique comme une collection d'objets, et de

définir l'*espace dual* comme l'espace des paramètres d'entrée d'un modèle capable de représenter chacun de ces objets. Le rôle de la transformation de Hough est de trouver quels points (vecteurs de paramètres) de l'espace dual, sont capables de donner l'explication la plus plausible des caractéristiques trouvées dans les images données. Pour cela, chaque pixel vote pour le sous-ensemble de l'espace dual constitué par tous les vecteurs de paramètres capables d'expliquer les caractéristiques de ce pixel. Lorsque tous les pixels ont voté, les vecteurs de paramètres recherchés sont probablement ceux qui ont recueilli les plus grands nombres de votes.

Cependant, malgré plusieurs réussites de cette approche (voir par exemple [13]), la transformation de Hough généralisée souffre d'un temps de calcul qui s'accroît rapidement avec la complexité des figures (images réciproques) dans l'espace dual. Les heuristiques destinées à améliorer sa rapidité sont d'un effet limité [16]. Elle devient pratiquement inutilisable lorsque la dimensionnalité de l'espace de recherche s'élève, principalement pour des raisons d'espace mémoire et de complexité de la tâche d'incrémentation de l'espace dual[1].

## 1.3 Résolution par Evolution Artificielle

Si l'on prend le point de vue de l'efficacité de la représentation et de l'exploration de l'espace dual, plutôt que de calculer partout dans cet espace les valeurs de vote puis de faire une exploration exhaustive de cet espace, il apparaît judicieux d'utiliser la philosophie d'approche des algorithmes d'évolution artificielle [9] que l'on peut considérer comme une heuristique efficace d'exploration de l'espace de recherche; elle permet également, dans notre cas, de ne calculer les valeurs de vote qu'aux points de l'espace dual où se trouvent réellement des individus de la population, et non sur la totalité de l'espace dual.

Le but de cet article est de présenter une application de cette approche à la modélisation de scènes tridimensionnelles. Nous avons choisi le problème de la stéréovision, qui consiste à construire un modèle tridimensionnel de la scène à partir des images fournies par deux caméras dont on connaît les paramètres géométriques. Les applications robotiques comme l'évitement d'obstacles et la planification de trajectoires n'exigent pas toujours une description géométrique exhaustive de la scène, qui conduirait à un coût calculatoire élevé dans les méthodes à base de segmentation, et pour laquelle les méthodes par vote sont généralement impraticables ou sans objet en raison de la dimensionnalité élevée de l'espace des paramètres.

Le principe général de l'évolution artificielle consiste, étant donné une fonction à optimiser, à considérer un ensemble arbitraire ("population") de points du domaine de définition de cette fonction ("individus"), puis à faire évoluer cette population (à l'image d'une évolution biologique), à l'aide d'opérateurs génétiques tels que mutations, croisements et sélection, le critère de sélection étant précisément la performance de chaque individu ("évaluation" ou "fitness") mesurée par la fonction à optimiser.

Le terme de "stratégie d'évolution" [12] se réfère aux algorithmes d'évolution artificielle où le codage ("gènes") est effectué à l'aide de nombres réels[2].

Dans l'approche classique, chaque individu de la population représente une solution potentielle du problème. A l'issue d'un certain nombre de générations, l'individu le plus performant (celui qui donne lieu à la valeur la plus élevée de la fonction) est retenu comme étant la solution du problème. L'approche "individuelle" [4] considère au contraire que la solution du problème n'est pas représentée par un seul individu de la population, mais par la population tout entière (ou du moins par une fraction importante de celle-ci).

Dans cet article, l'idée principale est d'utiliser cette dernière approche (Stratégie d'Evolution Individuelle) pour faire évoluer une population de points, de façon que ceux-ci se positionnent le mieux possible sur les surfaces visibles des objets de la scène. La notion d'approche "individuelle" implique qu'ici, le résultat de l'algorithme est exprimé non pas sous la forme d'un ensemble restreint de primitives complexes, mais par un ensemble plus fourni de primitives très simples: des points de l'espace. C'est ce parti-pris de simplicité[3] que nous tenterons d'exploiter, jusque dans l'expression de la fonction d'évaluation - dont la contribution à la charge totale de calcul est majeure.

## 1.4 Géométrie et fonction d'adaptation

Un individu (une "mouche") est défini comme un point de l'espace de coordonnées $(x, y, z)$. Comme nous utilisons deux caméras, la mouche se projette sur l'image de référence (normalement, l'image gauche) en $(x_L, y_L)$ et sur l'autre image (normalement, l'image droite) en $(x_R, y_R)$. Les paramètres de calibration des deux caméras sont supposés connus et permettent de calculer $x_L$, $y_L$, $x_R$, $y_R$ en fonction de $x$, $y$, $z$ par les for-

[1] Ce constat a justifié l'approche de Chellali [2] du problème de la stéréovision: celle-ci consiste (de manière analogue à un lancer de rayons) à exploiter les radiales issues de l'une des caméras et à chercher la meilleure correspondance sur l'épipolaire, bien que cette méthode intéressante souffre d'un manque de robustesse de la technique d'optimisation utilisée (descente de gradients).

[2] par opposition aux "algorithmes génétiques" proprement dits, pour lesquels les variables sont booléennes.

[3] En ce sens, notre travail se rapproche des travaux de Eberhart [5] sur les essaims de particules (particle swarms), un autre outil d'optimisation d'inspiration "vie artificielle", mais qui à notre connaissance n'a pas été encore appliqué en vision.

mules projectives:

$$\begin{pmatrix} x_L \\ y_L \\ 1 \end{pmatrix} \equiv F_L \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \quad \begin{pmatrix} x_R \\ y_R \\ 1 \end{pmatrix} \equiv F_R \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

où $F_L$ et $F_R$ sont les matrices projectives $(3,4)$ des caméras droite et gauche: par exemple,

$$F_L = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

où les $r_{ij}$ sont les éléments d'une matrice orthogonale (rotation) et les $t_i$ les termes de translation.

Si la mouche est située à la surface d'un objet, alors les pixels correspondants auront probablement les mêmes attributs dans les deux images (niveau de gris, couleur, etc.[4]). Inversement, si la mouche n'est pas sur la surface apparente d'un objet, la ressemblance des attributs de ses deux projections sera laissée au hasard de la texture de l'objet dans son alignement (voir figure 1). L'idée fondamentale de l'algorithme présenté ici est de traduire cette propriété sous la forme d'une fonction d'évaluation qui conduira le processus d'évolution d'une population de mouches initialisée aléatoirement.

La ffonction d'évaluation évalue le degré de ressemblance des voisinages des projections de la mouche dans chacune des deux images: cela donne des valeurs d'évaluation élevées pour les mouches posées sur la surface d'un objet.

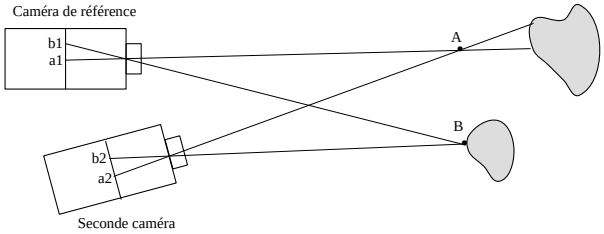


Fig. 1 : les pixels *b*1 and *b*2, projections de la mouche B, ont des niveaux de gris identiques. Les pixels *a*1 et *a*2, projections de la mouche A, n'ont pas nécessairement des niveaux de gris identiques car ils correspondent à deux points différents de la surface visible de l'objet.

Il découle de ces principes que la fonction d'évaluation, si elle ne prenait en compte que la dissimilarité des projections d'une mouche, donnerait des valeurs indésirablement élevées aux mouches situées devant un objet uniforme, même éloignées de sa surface. Pour éviter ce problème, la fonction d'évaluation, qui mesure la ressemblance des voisinages immédiats des pixels en jeu[5], doit inclure un terme de normalisation au numérateur:

$$ness(indiv) = \frac{G}{\sum_{(i,j)\in N} (L(x_L + i, y_L + j) - R(x_R + i, y_R + j))^2}$$

où:

- $L(x_L + i, y_L + j)$ est le niveau de gris de l'image gauche au pixel $(x_L + i, y_L + j)$
- $N$ est un petit voisinage orienté dans la direction épipolaire locale, sur lequel est mesuré le degré de ressemblance.

Le terme de normalisation $G$ mesure le contraste autour du pixel considéré de l'image de référence et permet d'éviter le double écueil de donner une évaluation élevée aux mouches situées devant un objet uniforme, et de ne donner une évaluation élevée qu'aux mouches situées devant un contour marqué. Les essais ont montré qu'on obtient un bon compromis en définissant $G$ par

$$G = \sqrt{\sum_{(i,j)\in N} (L(x_L + i, y_L + j) - L(x_L, y_L))^2}$$

De plus, la fonction d'évaluation est écrite de manière à s'affranchir de la composante continue de l'image par soustraction d'une moyenne glissante.

Ainsi, la fonction d'évaluation contient tous les calculs sur les pixels. Examinons maintenant les opérateurs de l'algorithme d'évolution.

## 1.5 Evolution Artificielle

La *population initiale* est créée dans le cône de vision de la caméra gauche, tronqué à une profondeur minimale arbitraire. Le *chromosome* d'un individu est le triplet $(x, y, z)$ qui contient ses coordonnées dans le repère, l'axe de visée de la caméra étant l'axe Oz. La distribution statistique des individus est choisie de manière à obtenir une distribution uniforme de leurs projections dans l'image gauche, et que les valeurs de $z^{-1}$ soient distribuées uniformément au-delà de la profondeur minimale. Ainsi la densité initiale de mouches décroît avec la profondeur.

[4] Cela est vérifié sur les surfaces lambertiennes où la rediffusion de la lumière incidente se fait de manière isotrope. La plupart des surfaces (hormis les surfaces brillantes) s'écartent suffisamment peu du modèle lambertien pour qu'il soit possible de compenser partiellement cet écart via une expression plus robuste de la fonction d'évaluation (voir ci-dessous). Les réflexions sur les surfaces brillantes peuvent donner lieu à la détection d'objets virtuels et à une interprétation 3-D erronée, quel que soit l'algorithme utilisé.

[5] Le dénominateur évalue le carré d'une distance entre les configurations de pixels autour des projections de la mouche dans les deux images. Ainsi, une valeur élevée de la fonction d'évaluation correspond à un individu dont les projections ont des voisinages ressemblants *et significatifs*.

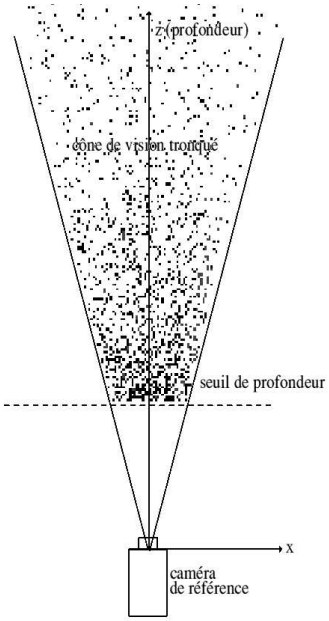


Fig. 2 : la population des mouches est initialisée à l'intérieur du cône de vision (tronqué) de la caméra de référence.

On suppose connus les paramètres de calibration des deux caméras. On peut donc, pour chaque mouche $(x, y, z)$, calculer les coordonnées de ses projections sur les images et calculer sa valeur d'évaluation.

Nous avons adopté des opérateurs génétiques de forme très classique.

La *sélection* utilise un algorithme de *ranking* rapide (classement des individus par évaluation décroissantes) et approché, à l'aide d'un calcul d'histogramme cumulé.

Pour éviter à la population de se concentrer sur un trop petit nombre de maxima (ce qui compromettrait l'essence même de la méthode), une fonction de *partage* 2-D permet de réduire les fitness des individus se projetant dans des zones trop peuplées. La présence d'une mouche de coordonnées $(x, y, z)$ réduit les valeurs de fitness de toutes les mouches qui se projettent dans son voisinage.

L'opérateur de *mutation* consiste à appliquer un bruit quasi-gaussien aux paramètres $x$, $y$ et $z$, avec une variance donnée.

Un opérateur de *croisement* sera introduit dans la section suivante.

## 2 Opérateurs évolutionnaires

### 2.1 Partage

L'exemple suivant montre un couple stéréo de synthèse[6], contenant quatre disques situés devant un mur plan vertical.

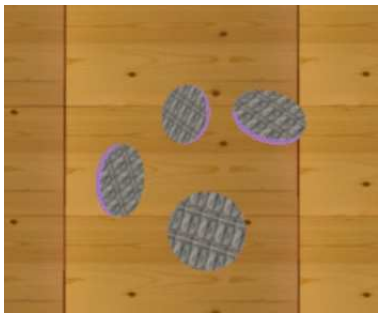

Fig. 3 : "Money", image gauche.

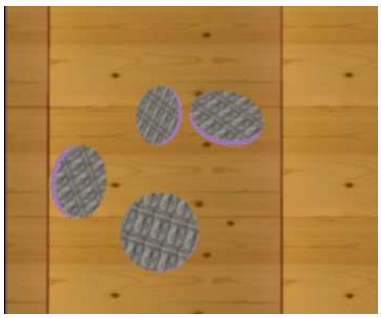

Fig. 4 : "Money", image droite.

Les Figures 5 and 6 montrent les projections des 30% meilleurs individus après 50 générations, sans partage.

La Fig. 5 est une vue de dessus, avec les axes $x_l$ (horizontal) et $1/z$ (vertical); la ligne pointillée horizontale en haut de l'image correspond à l'infini ($1/z = 0$); les points les plus proches de la caméra sont représentés plus bas. La Fig. 6 est une image de profondeurs obtenue avec la même population résultat, les niveaux de gris représentent les valeurs de la profondeur $z$ (les pixels plus foncés correspondent aux objets plus proches). Nous avons utilisé 5000 individus, un taux de mutation de 60%, et pas de croisement. Les 1500 (30%) meilleurs individus sont affichés après 50 générations. Les mouches se concentrent naturellement sur quelques pixels à gradient élevé. La qualité du résultat, déjà médiocre, se dégrade encore si on augmente le nombre de générations car les mouches continuent à se concentrer.

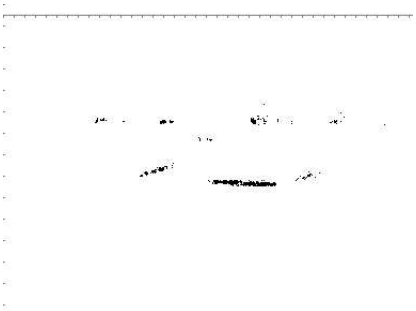

Fig. 5 : projection verticale (vue de dessus), sans partage

[6] images "Money", ©INRIA - projet Mirages

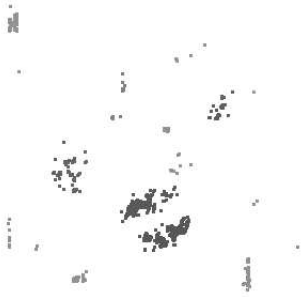

Fig. 6 : vue de face: image de profondeurs (proche = foncé), sans partage

Pour remédier à cet inconvénient, nous avons implanté un opérateur de partage classique [7] par subdivision de l'image en régions carrées et comptage du nombre d'individus se projetant dans chacun des carrés. La fonction d'évaluation de chaque mouche est alors décrémentée proportionnellement au nombre de mouches se projetant dans le même carré. Les figues 7 et 8 montrent les résultats avec une taille de carré ("rayon de partage") de 2 pixels et un coefficient de partage de 0,2; les figures 9 et 10 avec un rayon de partage de 3 et un coefficient 0,5. Dans le premier cas le mur et les quatre pièces sont détectés de manière acceptable, mais dans le deuxième cas le taux excessif de partage donne des résultats très bruités.

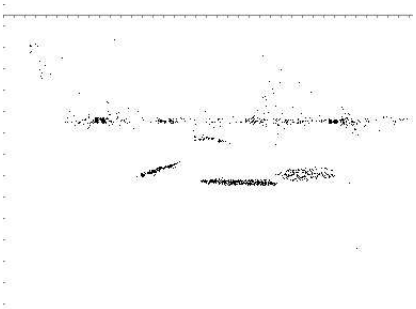

Fig. 7 : vue de dessus (avec partage)

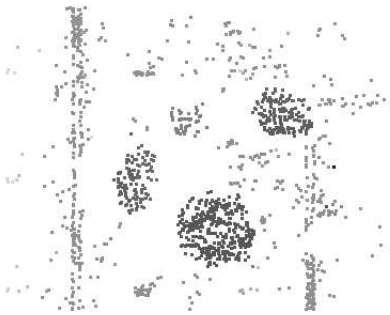

Fig. 8 : image de profondeurs (avec partage)

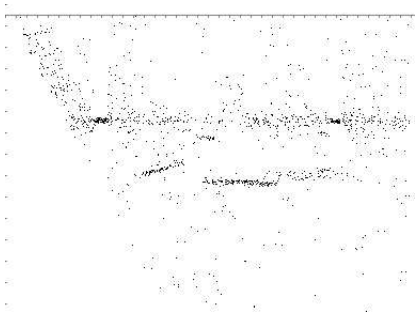

Fig. 9 : vue de dessus (avec partage excessif)

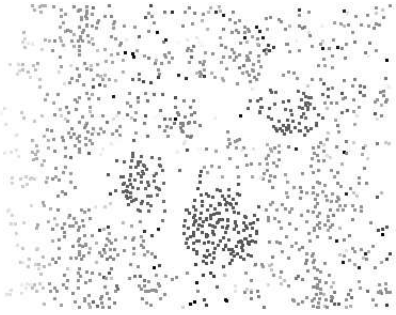

Fig. 10 : image de profondeurs (avec partage excessif)

## 2.2 Croisement

Les scènes du monde réel, surtout lorsqu'elles contiennent des objets manufacturés, comportent souvent des lignes droites et des surfaces planes. Nous avons traduit ce fait en termes d'heuristique, sous la forme d'un *opérateur de croisement barycentrique* qui construit le descendant de deux mouches comme un point placé aléatoirement sur le segment de droite qui relie ses parents: le descendant de deux individus de coordonnées $(x_1, y_1, z_1)$ et $(x_2, y_2, z_2)$ est l'individu de coordonnées $(x_3, y_3, z_3)$ définies par :

$$x_3 = \lambda x_1 + \mu x_2 \, ; \, y_3 = \lambda y_1 + \mu y_2 \, ; \; z_3 = \lambda z_1 + \mu z_2$$

où les poids $\lambda, \mu$ sont choisis avec une loi de probabilité uniforme sur l'intervalle $[0,1]$ $(\lambda + \mu = 1)$.[7]

Nous montrons (figues 11 à 16) les résultats sur 100 générations sur le même couple d'images, avec une population de 5000 individus, pour comparer l'effet de différents taux de croisement (rayon de partage 2, coefficient de partage 0,3, 40% affichés) :

- figures 11 et 12: mutation 60%, pas de croisement, CPU 8.47 s, 298959 évaluations, évaluation moyenne 2.891

[7] On considère généralement que cet opérateur de croisement est contractant, ce que l'on peut éviter en adoptant un intervalle plus grand. Cependant, dans notre application l'idée est qu'on obtient a priori une plus grande densité de points sur les bords (contrastés) des objets. Le but de l'opérateur de croisement est ici de remplir les surfaces ou les segments dont les bords ou les extrémités sont déjà détectés, et donc de privilégier la recherche à l'intérieur des segments, et non d'étendre ces surfaces ou segments vers l'extérieur. Il n'est donc pas en principe désirable d'utiliser des poids qui permettraient au barycentre d'être à l'extérieur des frontières de l'objet.

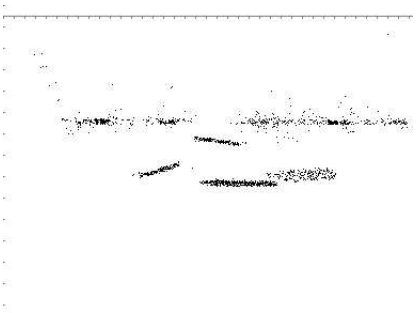

Fig. 11 ; vue de dessus

Fig. 12 : image de profondeurs

- figures 13 et 14: mutation 40%, croisement 20%, CPU 8.35 s, 298849 évaluations, évaluation moyenne 2.949

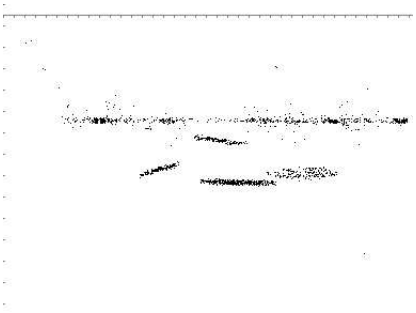

Fig. 13 : vue de dessus

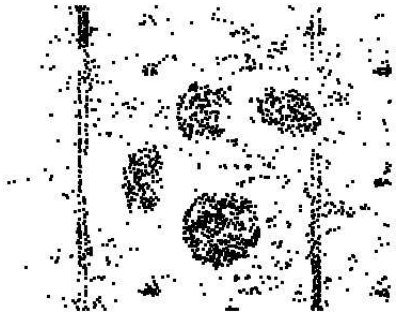

Fig. 14: image de profondeurs

- figures 15 et 16: mutation 20%, croisement 40%, CPU 8.53 s., 299049 évaluations, évaluation moyenne 2.903

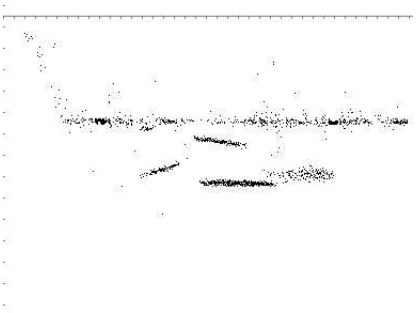

Fig. 15 : vue de dessus

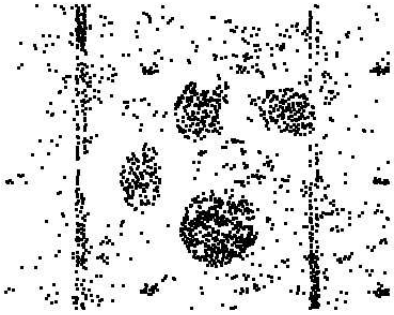

Fig. 16 : image de profondeurs

Dans la vue de dessus des figures 11, 13, 15, l'aspect flou de la quatrième pièce n'est pas imputable à l'algorithme, mais à la pièce qui est inclinée par rapport à la verticale et au mode d'affichage utilisé ici.

Les taux élevés de croisement tendent à remplir les espaces entre les objets détectés. On obtient une évaluation moyenne légèrement supérieure avec un taux de mutation de 40% et un taux de croisement de 20%, mais cela peut varier en fonction des images choisies et surtout de la taille de population, comme on le verra ci-après. L'effet du compromis entre taux de mutation et de croisement est beaucoup plus sensible avec les petites populations.

Le temps CPU indiqué est basé sur un PC i686 à 366MHz sous Linux, programmation en C non optimisée, et peut varier (typiquement de ±10%). Il inclut les entrées-sorties. Hors entrées-sorties, une génération de 5000 individus est traitée en environ 50 millisecondes, et le temps varie linéairement avec la taille de la population (soit 12 ms par génération pour 1000 individus).

Lorsqu'on utilise de petites populations (pour des raisons de temps de calcul), l'expérience montre qu'il faut augmenter le taux de partage pour avoir une bonne répartition de la population malgré la densité moyenne plus faible. Mais c'est surtout là que l'opérateur de croisement prend son intérêt et assure une détection des objets plus fiable, comme le montrent les figures 17 à 19, vues de dessus obtenues avec une population de 1000 individus, sur 100 générations, avec des taux de croisement différents (rayon de par-

tage 3; taux de partage 1,0; 80% affichés).

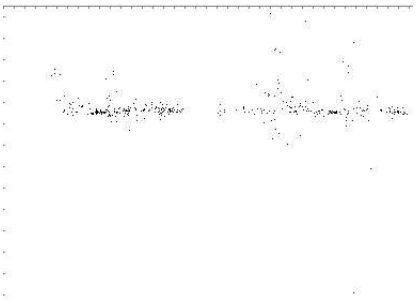

Fig. 17
mutation 60%
pas de croisement
CPU 2.04 sec.
58831 évaluations

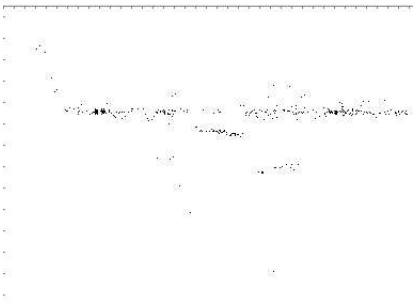

Fig. 18
mutation 40%
croisement 20%
CPU 2.01 sec.
58451 évaluations

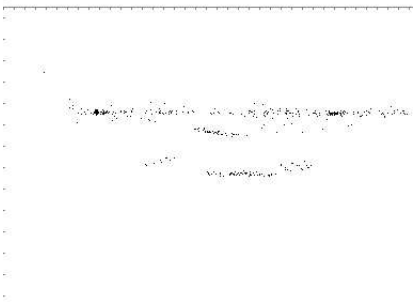

Fig. 19
mutation 20%
croisement 40%
CPU 2.03 sec.
58704 évaluations

## 3 Sensibilité aux paramètres et résultats de convergence

Pour plus de lisibilité nous présentons encore des résultats obtenus sur le couple d'images "Money", car elles correspondent à une scène simple dont les résultats sont facilement lisibles. Nous avons obtenu des résultats similaires avec des images de scènes naturelles, bien que moins lisibles en projection 2-D.

### 3.1 Taux de mutation et de croisement

Le graphique de la figure 20 représente l'évolution de la moyenne des valeurs d'évaluation sur l'ensemble de la population, en utilisant les mêmes images et valeurs de paramètres que dans les Figures 11 - 16.

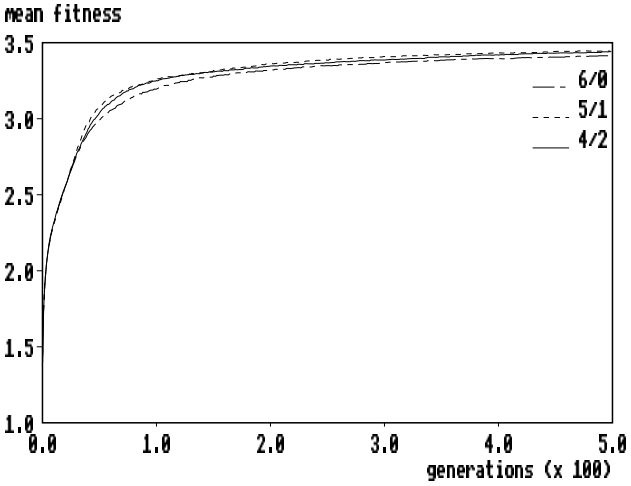


Fig. 20 : évaluation moyenne d'une population de 5000 individus en fonction du nombre de générations, pour trois combinaisons de taux de croisement et de mutation.

Les meilleures valeurs (courbe supérieure) sont obtenues avec un taux de mutation de 50% et un taux de croisement de 10%. La courbe la plus basse correspond à une évolution sans croisement. L'effet des mutations se fait sentir à partir de la trentième génération.

### 3.2 Nombre de générations

Les figures 21 à 24 (vues de dessus) montrent les résultats obtenus après 10, 50, 100 et 1000 générations, avec les mêmes paramètres (taux de mutation 50% et taux de croisement 10%).

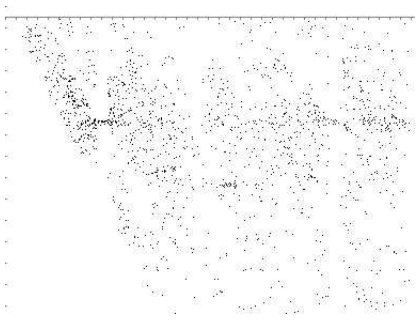

Fig. 21 : 10 générations

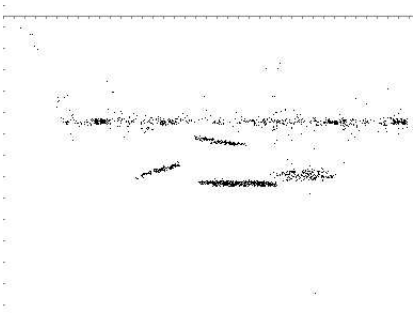

Fig. 22 : 50 générations

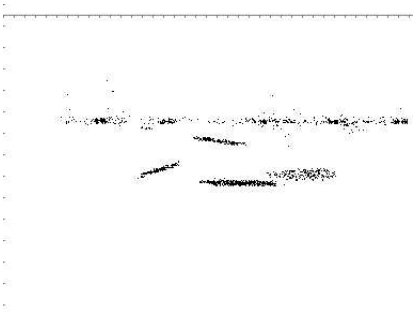

Fig. 23 : 100 générations

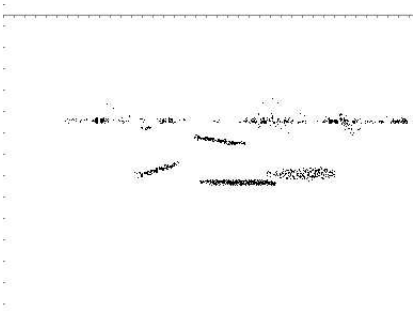

Fig.24 :1000 générations

Fig. 25 : image gauche

Fig. 26 : image droite

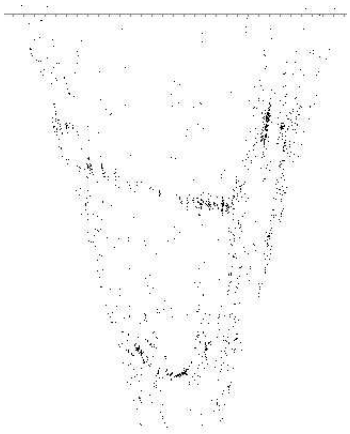

Fig. 27 : vue de dessus (image 768 × 576 )

## 3.3 Résultats sur images réelles

### 3.3.1 Stéréovision classique

Le couple d'images 760 × 560 des figures 25 et 26 est issu d'une caméra monochrome en mouvement de translation latéral. Nous avons utilisé des paramètres génétiques analogues aux précédents (5000 individus, 100 générations, mutations 40%, croisements 10%, rayon de partage 2, coefficient de partage 0.3).

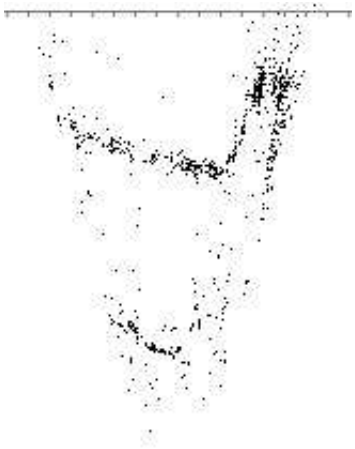

Fig. 28 : vue de dessus (image 384 × 288)

Sur les figures 27 et 28, on voit apparaître les deux faces de l'armoire, le début du mur à droite ainsi que le demi-cercle du tabouret. Il est intéressant de remarquer que la figure 28 est le résultat du traitement sur le couple stéréo à la résolution moitié: le résultat y apparaît meilleur, toutes choses égales par ailleurs (mêmes paramètres génétiques: 5000 individus, 100 générations, même temps de traitement; le rayon de partage a été modifié en proportion de la résolution) . Cela est probablement dû à la fenêtre de comparaison qui occupe une part relativement plus grande de l'image lorsque la résolution est inférieure, aboutissant à une corrélation plus sûre et sans doute aussi à une meilleure tolérance aux erreurs de calibration résiduelles.

### 3.3.2 Vision 3D axiale

Le couple suivant est issu de la même caméra, mais avec un mouvement de translation le long de l'axe de visée. La matrice de calibration a été modifiée en conséquence, sans autre changement dans l'algorithme ni ses paramètres. Le foyer d'expansion (FOE) est au centre de l'image.

Fig. 29 : image "arrière"

Fig. 30 : image "avant"

Nous utilisons les mêmes paramètres que précédemment (5000 individus, 100 générations). La relative complexité de la scène rend ici les résultats de la figure 31 plus difficiles à interpréter. mais l'image des profondeurs (Fig. 32) montre une détection correcte des boites des deux côtés, y compris la détection de l'objet virtuel issu de la réflexion spéculaire sur la table. La profondeur du mur et des autres objets est aussi estimée correctement. Le rail qui a servi de guide au déplacement de la caméra n'est pas détecté car les lignes de contrastes sont des épipolaires convergeant vers le FOE et ne donnent aucune information utile. Leur évaluation est basse en raison du terme de normalisation qui est calculé dans la direction épipolaire. De même, il reste une zone aveugle autour du centre de l'image car les déplacements apparents au voisinage du FOE sont trop faibles pour donner une information exploitable sur la distance des objets.

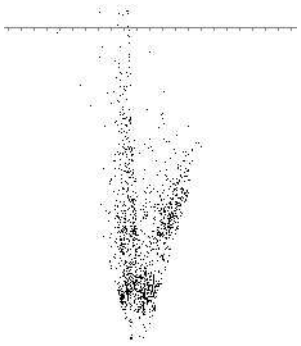

Fig. 31 : vue de dessus

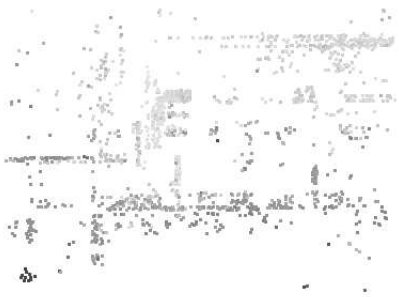

Fig. 32 : image de profondeurs

## 4 Vers une évolution en temps réel pour la poursuite d'objets mobiles

On considère souvent les algorithmes génétiques et les stratégies d'évolution comme lents et impropres aux applications temps-réel. Cependant :

- la vitesse n'est pas en soi le seul critère de l'applicabilité temps-réel. La capacité "temps réel" se réfère à l'aptitude à exploiter les données d'entrée à la cadence où elles sont fournies, pour que le système soit capable de réagir à la vitesse exigée par l'utilisateur final. Les stratégies d'évolution sont généralement capables d'adaptation, c'est-à-dire de résoudre un problème décrit par une fonction d'évaluation qui change durant l'exécution [15], ce qui n'est pas le cas général des méthodes d'optimisation;
- la vitesse d'exécution d'une stratégie d'évolution dépend fortement de la complexité calculatoire de la fonction d'évaluation - qui est plutôt simple dans le cas présenté.

C'est pourquoi nous travaillons actuellement à l'extension de l'algorithme aux séquences d'images et particulièrement aux séquences stéréo prises par un couple de caméras en mouvement, simulant un robot en déplacement. Les premiers résultats obtenus montrent que, si le mouvement est lent, la convergence est fortement accélérée si l'on exploite le résultat de l'algorithme au pas de temps précédent pour initialiser la nouvelle population. Pour traiter des mouvements un peu plus rapides, nous étendons le chromosome en l'enrichissant des coordonnées du vecteur vitesse des mouches dans le repère mobile lié au robot: cela permet à chaque mouche, maintenant riche de 6 gènes, de garder mémoire de sa vitesse dans le traitement d'une séquence de plusieurs images, de manière analogue à un processus markovien. Les opérateurs génétiques sont modifiés en conséquence. Le champ (non dense) des vecteurs vitesse trouvés permet d'ajuster une estimation du mouvement du robot par rapport à un repère lié à la scène, et les vecteurs vitesse incohérents avec ce déplacement global indiquent les objets en mouvement.

## 5 Conclusion

Nous avons décrit dans cet article une stratégie d'évolution rapide capable de fournir une description tridimensionnelle grossière d'une scène à partir d'un couple stéréo. Au contraire des approches classiques de la stéréovision, cette méthode ne nécessite aucune segmentation préalable et donne des résultats dont la précision s'améliore progressivement, ce qui semble être une propriété intéressante en Robotique par la variabilité intrinsèque du compromis vitesse/précision des résultats.

Dans l'introduction, nous avons tenté un parallèle avec la transformation de Hough dans laquelle chaque pixel vote pour un sous-ensemble d'un espace de paramètres. Ici c'est la population en évolution qui explore cet espace, chaque individu testant un prédicat au niveau pixel. Il n'y a pas de règle évidente disant laquelle des deux approches est la plus efficace, mais il semble que même dans notre exemple où une mouche n'est représentée que par 3 paramètres, une approche par vote pour remplir un espace de paramètres tridimensionnel serait beaucoup plus coûteuse. Le bénéfice de l'approche évolutionnaire est:

- traitement rapide[8] (en partie grâce à la recherche non exhaustive);
- segmentation préalable non requise, ce qui rend l'algorithme utilisable sur des images de scènes peu structurées;
- accumulation progressive de connaissance sur la scène, rendant possible d'exploiter les résultats à n'importe quel stade du déroulement de l'algorithme, et permettant un compromis ouvert entre qualité du résultat et temps de calcul sans modifier l'algorithme;
- compatibilité temps-réel, car la fonction d'évaluation peut être contrôlée par des paramètres externes [15] directement issus des capteurs (p.ex. dans les applications en robotique mobile) pendant l'exécution de l'algorithme.

Cependant, trois particularités importantes méritent d'être soulignées:

- la simplicité de la fonction d'évaluation peut surprendre. Les essais que nous avons effectués sur l'application à la stéréovision montrent par exemple que le coefficient de corrélation sur un voisinage donne des résultats de qualité inférieure pour un temps de calcul équivalent. En tout état de cause, le nombre d'appels à la fonction d'évaluation est égal à:
  $N \times P \times (\tau_m + \tau_c)$ où $N$ est le nombre de générations, $P$ l'effectif de la population, $\tau_m$ et $\tau_c$ les taux de mutation et de croisement; la rapidité d'exécution des évaluations est donc essentielle;
- à la différence des approches classiques en

[8] Le temps de traitement ne dépend pas directement de la taille d'image mais de la taille de population choisie.

traitement d'images où des modules standard de traitement sont connectés de manière spécifique à l'application, dans notre approche la fonction d'évaluation (et, dans une moindre mesure, les nuances dans les opérateurs de mutation et de croisement) contient toute la connaissance spécifique à l'application alors que l'architecture globale de l'algorithme reste indépendante de l'application;

- notre approche abandonne la lecture séquentielle en faveur d'une lecture aléatoire des pixels. On peut noter une concordance favorable avec l'apparition sur le marché de caméras CMOS qui permettent une lecture aléatoire et asynchrone des pixels directement sur la matrice de photo-détecteurs, et qui semblent donc idéalement adaptées à l'exécution de cette classe d'algorithmes dans des applications embarquées.

Dans notre travail actuel, nous introduisons le mouvement des caméras pour exploiter l'information accumulée avec le traitement des images précédentes et estimer la vitesse de déplacement de l'observateur. Nous prévoyons d'implanter cette méthode évolutionnaire d'analyse de scènes comme détecteur d'obstacles embarqué dans une application de robotique mobile, et de concevoir un système de planification de mouvement et d'évitement d'obstacles qui utilise directement en entrée le résultat de l'algorithme, sous la forme d'un nuage de points 3-D.

## 6 Bibliographie


[1] Dana H. Ballard, Christopher M. Brown, *Computer Vision*, Prentice Hall, 1982.

[2] Ryad Chellali, *La vision en robotique mobile: méthodes de reconstruction géométrique de l'environnement et ourtils d'aide à la navigation*, thèse de Doctorat, Université Pierre et Marie Curie, 1993.

[3] J.-P. Cocquerez, S. Philipp et al. *Analyse d'images : filtrage et segmentation*, Masson, 1995

[4] Pierre Collet, Evelyne Lutton, Frederic Raynal, Marc Schoenauer, *Individual GP: an Alternative Viewpoint for the Resolution of Complex Problems*, GECCO99, Orlando, Florida, July 1999.

[5] R.C. Eberhart, J. Kennedy, *A new Optimiser Using Particles Swarm Theory*, Proc. 6th Int. Symposium on Micro Machine and Human Science, Nagoya (Japon), IEEE service Centre, Piscataway, NJ, 39-43, 1995

[6] D. B. Gennery, *Modelling the Environment of an Exploring Vehicle by means of Stereo Vision*, PhD thesis, Stanford University, June 1980

[7] David E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison Wesley, 1989.

[8] R. M. Haralick, *Using Perspective Transformations in Scene Analysis*, Computer Graphics and Image Processing 13, 1980, pp. 191-221.

[9] J. H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, Univ. of Michigan Press, 1975

[10] P. V. C. Hough, *Method and Means of Recognising Complex Patterns*, U.S. Patent n°3 069 654, 18 December 1962.

[11] Evelyne Lutton, Patrice Martinez, *A Genetic Algorithm for the Detection of 3D Geometric Primitives in Images, 12th ICPR, Jerusalem, Israel, October 9-13, 1994 / INRIA technical report # 2210.*

[12] I. Rechenberg, *Evolution Strategy*, in J.M. Zurada, R.J. MarksII, C.J. Robinson, Computational Intelligence imitating life, IEEE Press (1994) 147-159

[13] John O'Rourke, *Motion Detection using Hough technique*, IEEE conference on Pattern Recognition and Image Processing, Dallas 1981, pp. 82-87.

[14] G. Roth and M. D. Levine, *Geometric Primitive Extraction using a Genetic Algorithm*, IEEE CVPR Conference, pp. 640-644, 1992.

[15] Ralf Salomon and Peter Eggenberger, *Adaptation on the Evolutionary Time Scale: a Working Hypothesis and Basic Experiments*, Third European Conference on Artificial Evolution, Nîmes, France, October 1997, Springer Lecture Notes on Computer Science no. 1363, pp. 251-262.

[16] Wei Pan, *A Comparative Study of Fast Hough Transform*, `http://www.stat.wisc.edu/~wpan/cs766/cs766.html`