

---

# L'algorithme des mouches dynamiques

## Guider un robot par évolution artificielle en temps réel

Jean Louchet\* — Maud Guyon\* — Marie-Jeanne Lesot\*  
Amine Boumaza\*\*

*\*Ecole Nationale Supérieure de Techniques Avancées  
32 boulevard Victor, 75739 Paris cedex15  
louchet@ensta.fr  
URL: <http://www.ensta.fr/~louchet>*

*\*\*INRIA Rocquencourt  
BP 75, 78153 Le Chesnay cedex  
amine.boumaza@inria.fr*

---

*RÉSUMÉ. L'algorithme des mouches est un algorithme rapide d'évolution artificielle conçu pour l'exploration de l'espace des paramètres dans certaines applications de reconnaissance des formes. Ici, la population est un ensemble de particules situées dans l'espace, et qui constitue une représentation collective ("approche parisienne") de la scène en trois dimensions. Chaque individu peut en outre être muni de paramètres de vitesse. L'évolution utilise une fonction d'objectif qui contient tous les calculs sur les pixels d'images, et utilise des opérateurs classiques (partage, mutation, croisement). La combinaison de l'approche parisienne, d'une fonction d'objectif simple et de l'asynchronisme de l'algorithme lui confèrent des propriétés temps-réel illustrées ici par une application à l'évitement d'obstacles en robotique mobile.*

*ABSTRACT. The fly algorithm is a fast artificial evolution-based technique devised for the exploration of parameter space in pattern recognition applications. It evolves a population of particles which constitutes a three-dimensional representation of the scene. Each individual may be fitted with optional velocity parameters. Evolution is controlled by a fitness function which contains all pixel-level calculations, and uses classical evolutionary operators (sharing, mutation, crossover). The combined individual approach, low complexity fitness function and asynchronous input and output allow fast processing and promising real-time capabilities, illustrated through an application to obstacle avoidance in mobile robotics.*

*MOTS-CLÉS: évolution artificielle, reconnaissance des formes, vision robotique.*

*KEYWORDS: artificial evolution, pattern recognition, robot vision.*

---

## 1. Introduction

Les méthodes de vote sur l'espace des paramètres comme la transformation de Hough généralisée [HOU 62] sont limitées par le nombre de paramètres et la taille de l'espace de recherche. On peut voir l'évolution artificielle [GOL 89, HOL 75, REC 94] comme une classe de techniques d'exploration de l'espace des paramètres qui permet de traiter avec succès certains problèmes de traitement d'images [FON 00, LUT 94, ROT 92, SER 99] mais souffrent souvent des calculs importants nécessaires lorsque chaque individu de la population à manipuler est en lui-même une description complexe de la scène. L'"approche parisienne" [COL 99, LOU 00, LOU 01] de l'évolution artificielle considère chaque individu comme une partie de la solution du problème : la solution est représentée par l'ensemble de la population, ou une large fraction de celle-ci. L'algorithme présenté dans cet article exploite un couple d'images stéréo en utilisant des points 3-D<sup>1</sup> ('mouches') comme individus, et fait évoluer cette population grâce à une fonction d'objectif (fitness) telle que les mouches convergent vers les surfaces visibles des objets de la scène<sup>2</sup> (section 2). Dans la section 3 nous présentons une extension aux séquences d'images stéréoscopiques par l'introduction de gènes supplémentaires représentant la vitesse de la mouche. Une application au guidage de robot mobile avec évitement d'obstacles est présentée dans la section 4.

## 2. Evolution des mouches

### 2.1. Géométrie et fonction d'objectif

Une mouche est définie comme un points 3-D de coordonnées  $(x, y, z)$ . Les coordonnées de ses projections sont  $(x_L, y_L)$  dans l'image gauche et  $(x_R, y_R)$  dans l'image droite. Connaissant les paramètres de calibration des caméras, il est aisé de calculer  $x_L, y_L, x_R, y_R$  à partir de  $x, y, z$  par géométrie projective [HAR 80, JAI 94]. Si la mouche se trouve à la surface d'un objet opaque, alors les pixels correspondants dans les deux images auront normalement le même niveau de gris<sup>3</sup> et des voisinages ressemblants (figure 1). Inversement, si la mouche n'est pas sur la surface d'un objet et que ce dernier est suffisamment hétérogène, la ressemblance

1. Les essaims de particules [MIL 94, EBE 95] utilisent aussi des points 3-D comme primitives, dans une démarche inspirée des systèmes de particules utilisés en infographie mais où les comportements collectifs sont utilisés comme outil d'optimisation. Notre approche utilise les opérateurs génétiques (sélection, mutation, croisement...) qui ne sont habituellement pas utilisés dans les techniques par essais de particules. On peut cependant noter plusieurs analogies entre les deux techniques : par exemple, vitesses aléatoires et mutations, évitement mutuel et partage.

2. Tang et Medioni [TAN 98] utilisent une méthode de "vote tensoriel" avec une distribution aléatoire de points 3-D qui n'évoluent pas, mais sont utilisés pour initialiser le calcul d'un champ dense de tenseurs.

3. Ou les mêmes composantes chromatiques dans le cas d'images en couleurs. Cette propriété est bien vérifiée avec les surfaces mates (lambertiennes) où la diffusion de la lumière est isotrope. Les réflexions spéculaires peuvent créer des objets virtuels (au sens de l'optique géométrique) et fausser l'interprétation 3-D, quel que soit l'algorithme de traitement stéréo.

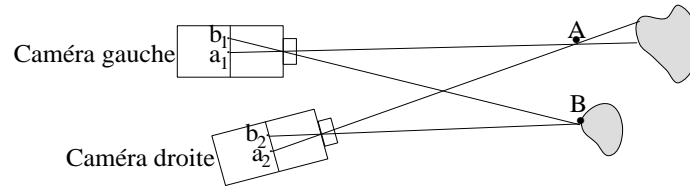
entre les voisinages de ses projections sera faible. Pour favoriser l'évolution de la population de mouches vers les surfaces apparentes des objets, nous exprimons cela sous la forme d'une fonction d'objectif qui caractérise la ressemblance des voisinages des projections de la mouche dans les deux images :

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

–  $(x_L, y_L)$  et  $(x_R, y_R)$  sont les coordonnées des projections droite et gauche de l'individu courant (cf. figure 1),

–  $L(x_L + i, y_L + j)$  est le niveau de gris de l'image gauche au pixel  $(x_L + i, y_L + j)$ , et de même avec l'image de droite.

–  $N$  est un voisinage introduit pour que la comparaison des projections soit plus discriminante.



**Figure 1.** Les pixels  $b1$  et  $b2$ , projections de la mouche  $B$ , sont fortement corrélés, à la différence des pixels  $a1$  et  $a2$ , projections de  $A$ , qui reçoivent leur illumination de deux points physiques différents de la scène

Dans le cas d'images en couleurs, la fonction d'objectif est la somme des trois termes correspondant à chaque canal chromatique.

Le facteur de normalisation  $G$  est une mesure locale du contraste, destinée à favoriser les mouches dont les projections sur les images ont une ressemblance *significative*, et donc ne sont pas trop uniformes. Nous avons déterminé empiriquement  $G$  comme la racine carrée du gradient de Sobel [JAI 94], et complété le dénominateur de la fonction d'évaluation de manière à l'affranchir des effets de la composante continue de l'image et des basses fréquences spatiales (qui peuvent être dues par exemple à une différence de calibration radiométrique des caméras).  $G$  est mis à zéro si la mouche sort du champ d'une caméra.

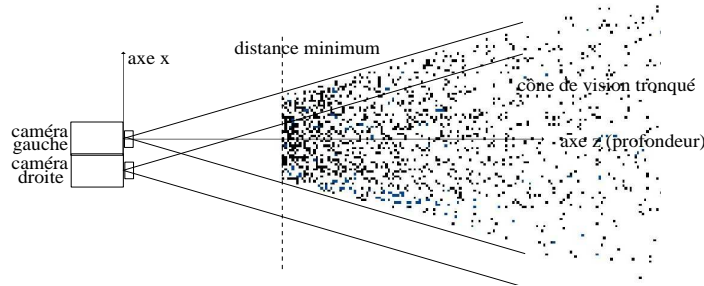
## 2.2. Opérateurs évolutionnaires

Le chromosome d'un individu est le triplet  $(x, y, z)$  qui contient les coordonnées de la mouche. La population est *initialisée* aléatoirement dans le champ de la caméra de référence (gauche), à partir d'une distance minimum (figure 2). La distribution est uniforme en projection dans le champ de la caméra de gauche, et les profondeurs sont réparties en distribuant  $z^{-1}$  uniformément entre zéro et  $1/d_{min}$ .

La *sélection* est élitiste et déterministe (taux de l'ordre de 50 %)

Le *partage* (sharing) [HOL 75] bidimensionnel pénalise les mouches qui se projettent dans des zones surpeuplées de l'image en réduisant leur fonction d'objectif de  $K \times N$ , où  $K$  est le "coefficient de partage" et  $N$  le nombre de mouches qui se projettent dans un rayon  $R$  ("rayon de partage") autour de la mouche courante.

La *mutation*, qui permet l'exploration extensive de l'espace de recherche, utilise une approximation d'un bruit gaussien s'ajoutant aux paramètres  $(x, y, z)$ .



**Figure 2.** La population des mouches est initialisée dans le champ de la caméra de référence

La mutation de  $z$  est ici en réalité l'ajout d'un bruit gaussien à l'inverse de  $z$ , de façon que la variance des mutations de  $z$  soit cohérente avec la densité des mouches qui décroît avec la distance. Ainsi, nous avons choisi les variances de mutations  $\sigma_x, \sigma_y, \sigma_z$  égales à  $R$ , c'est-à-dire du même ordre que la distance moyenne entre mouches voisines.

$$z_{\text{nouveau}} = \frac{d_{\text{min}}}{\text{bruit} + \frac{d_{\text{min}}}{z}}$$

La variance  $\sigma_z$  du bruit est l'un des paramètres de l'algorithme. La mutation de  $x$  et de  $y$  s'effectue de sorte que la variance du déplacement de l'image de la mouche soit indépendante de la profondeur; d'où :

$$x_{\text{nouveau}} = \frac{z_{\text{ancien}}}{focale} \times \left( \frac{focale \times x}{z_{\text{ancien}}} + \text{bruit} \right)$$

Le bruit choisi pour  $x$  et  $y$  est gaussien, sa variance est expérimentalement du même ordre de grandeur que la distance de la projection d'une mouche à sa plus proche voisine: nous avons choisi pour cela

$$\sigma_x = \sigma_y = \sqrt{\frac{N_{\text{pixels}}}{\text{population}}}$$

Les objets du monde réel contiennent souvent des arêtes rectilignes ou des surfaces planes. Nous avons traduit cette propriété en un opérateur de *croisement barycentrique* qui crée une nouvelle mouche située sur le segment de droite reliant

ses deux parents : la mouche  $F_3(x_3, y_3, z_3)$  résultat du croisement des parents  $F_1(x_1, y_1, z_1)$  et  $F_2(x_2, y_2, z_2)$  est définie par  $\overrightarrow{OF_3} = \lambda \overrightarrow{OF_1} + (1 - \lambda) \overrightarrow{OF_2}$ . Le poids  $\lambda$  est choisi selon une loi uniforme<sup>4</sup> dans  $[0,1]$ .

### 2.3. Résultats

#### 2.3.1. Images de synthèse

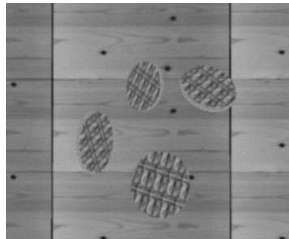


Figure 3. image "Money" gauche

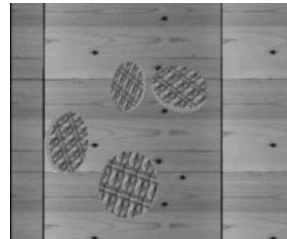


Figure 4. image "Money" droite

L'image-test de synthèse "Money"<sup>5</sup> (figures 3 et 4) donne des résultats facilement lisibles car les principaux objets sont verticaux. La figure 5 montre des résultats de convergence avec une population de 5 000 individus, un taux de mutation<sup>6</sup> de 50 % et de croisement de 10 %, à différents stades de la convergence<sup>7</sup>. La scène est vue de dessus: l'axe  $z$  de la caméra est orienté vers le haut de la page.

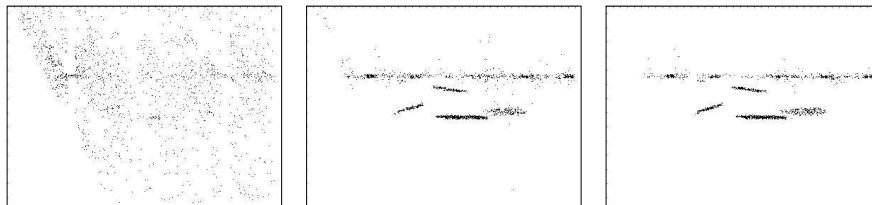


Figure 5. Convergence après 10 (gauche), 50 (centre) et 100 (droite) générations. La ligne de mouches en haut correspond au panneau de bois au fond de la scène, les 4 nuages plus bas aux 4 pièces. On remarque que la pièce de droite n'est pas verticale et que sa représentation en projection paraît de ce fait moins nette

4. On considère généralement que le croisement barycentrique possède des propriétés contractantes pour la population si les poids sont dans l'intervalle  $[0,1]$ . Cependant nous visons ici plutôt à favoriser le remplissage de surfaces dont les points de contours sont habituellement plus aisés à détecter grâce à leur contraste souvent plus élevé. Il n'est donc pas souhaitable de choisir des poids tels que le barycentre puisse être situé à l'extérieur de la surface considérée. Cette hypothèse a été confirmée expérimentalement dans notre application.

5. Couple stéréo "Money", 384x288 pixels, ©INRIA - Mirages.

6. L'évolution parisienne utilise habituellement des taux de mutation beaucoup plus élevés que les méthodes standard.

7. L'expérience montre que l'on obtient les meilleurs résultats avec une sélection élitiste et un taux total de mutation et croisement autour de 60 %.

## 2.3.2. Réglage fin des paramètres

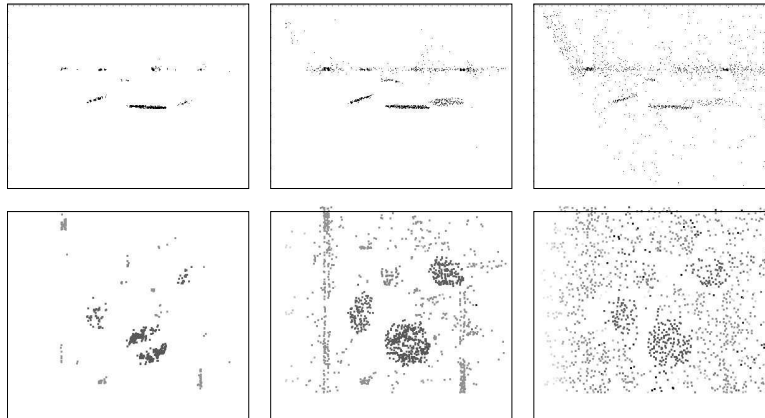
## 2.3.2.1. Partage

Les figures 6 à 8 montrent l'effet du partage sur les mêmes données d'images. Elles représentent les 1 500 meilleurs individus (30 % de 5 000) après 5 générations, avec un taux de mutation de 60 %, sans croisement et avec différents coefficients de partage. Les images en haut montrent une vue de dessus de la distribution de mouches, celles d'en bas une vue de face (carte de profondeur) où les pixels plus foncés correspondent aux points plus proches. En l'absence de partage ou avec un partage trop faible, les mouches se concentrent sur quelques points à gradient élevé et au bout d'un certain nombre de générations finissent même par se concentrer sur un seul point. Un partage trop fort dégrade les résultats en forçant les mouches dans des régions non significatives.

Nous avons déterminé une règle empirique de choix du rayon de partage  $R$ , considérant que si l'image est subdivisée en carrés de côté  $2R + 1$ , il doit idéalement y avoir une mouche par carré en moyenne : d'où  $(2R + 1)^2 \times N_{flies} \approx N_{pixels}$ , et :

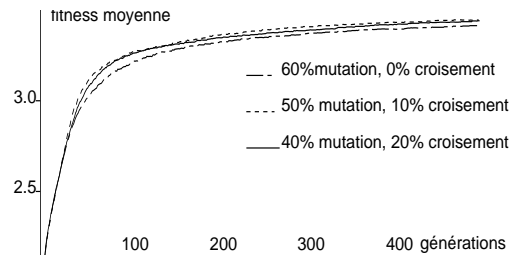
$$R \approx \frac{1}{2} \left( \sqrt{\frac{N_{pixels}}{N_{flies}}} - 1 \right)$$

Par exemple, avec 5 000 mouches et une image  $500 \times 500$  on obtient  $R = 3$ .



**Figure 6.** Sans partage    **Figure 7.** Partage moyen (rayon 2, coefficient 0.2)    **Figure 8.** Partage élevé (rayon 3, coefficient 0.5)

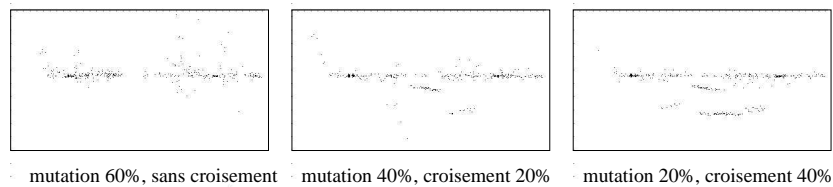
### 2.3.2.2. Taux de mutation et de croisement



**Figure 9.** Evolution des valeurs moyennes de la fonction d'objectif pour 5000 individus, avec trois combinaisons de taux de mutation et de croisement

La figure 9 montre des courbes typiques d'évolution de la fonction d'objectif avec différents taux de mutation et de croisement. On obtient les meilleurs résultats sur des scènes d'intérieur avec un taux de mutation de l'ordre de 50% et un taux de croisement de l'ordre de 10%. La courbe du bas correspond à une évolution sans croisements. Pour une population de 5000 individus, les temps CPU sont d'environ 15 ms par génération sur un PC i686 366MHz, Linux, ANSI C, gcc, indépendamment de la taille d'image.

### 2.3.2.3. Petites populations



**Figure 10.** Différents taux avec une population de 1 000 mouches et 100 générations

Il peut être intéressant d'utiliser de petites populations, lorsque le temps d'exécution est critique ou lorsque la taille de l'image est réduite. A taille d'image égale, il convient d'augmenter le rayon de partage pour assurer une bonne répartition de la population (cf. § 2.3.2.1) mais aussi il est souvent bénéfique d'augmenter les taux de mutation pour améliorer la détection (figure 10). Un bon compromis est entre 10 % et 20 %, selon l'importance des objets planaires dans la scène.

### 2.3.3. Résultats sur images réelles

Nous avons testé l'algorithme sur des images monochromes stéréo  $768 \times 576$  de scènes d'intérieur (Fig 11 - 12). La figure 13 montre des résultats typiques d'une scène d'intérieur, obtenus avec des paramètres génétiques similaires.

population	générations	taux de mutation	taux de croisement	rayon de partage	coefficient de partage
5000	100	40	10	2	0.3

On peut voir les deux faces de l'armoire, le début du mur à droite et le demi-cercle avant du tabouret.



**Figure 11.** Image gauche    **Figure 12.** Image droite    **Figure 13.** Résultat et carte

### 3. Obstacles mobiles : mouches dynamiques et temps réel

Nous avons étendu l'algorithme décrit plus haut, aux séquences d'images stéréo (caméras en déplacement).

#### 3.1. Méthode aléatoire

L'approche aléatoire consiste à faire évoluer la population en continu à travers les changements d'images. Cela signifie qu'au cours de l'évolution, la fonction d'objectif n'est plus statique mais dépend des images courantes. Si le mouvement est suffisamment lent, le fait d'initialiser l'algorithme par les résultats (nuage de mouches) obtenus sur les images précédentes en accélère la convergence, par rapport à ce qui serait le cas en utilisant une initialisation totalement aléatoire. On peut voir ceci comme une forme de mémorisation de la représentation de l'espace *via* l'exploitation de la similarité des points de vue successifs. Cette méthode ne nécessite pas d'importants changements algorithmiques, sinon l'introduction d'un opérateur supplémentaire de mutation, l'immigration, qui crée de nouveaux individus *ex nihilo* de manière semblable à celle utilisée pour l'initialisation de la population. Utilisé avec une faible probabilité (1 % à 5 %) cet opérateur assure une forme de surveillance des zones de la scène jusque là vides ou invisibles, afin de réagir correctement à l'apparition éventuelle d'objets nouveaux dans le champ des caméras.

#### 3.2. Méthode futuriste

Pour traiter les mouvements plus rapides, nous avons introduit un chromosome étendu, qui contient désormais les composantes de position et de vitesse de la mouche dans le repère mobile lié à la caméra de référence. Il faut alors adapter la fonction d'objectif et les opérateurs génétiques. Pour cela, la méthode la plus simple

(“approche futuriste”) est la suivante. Le chromosome d’une mouche est le 6-uple  $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ . Au premier instant de la séquence, les positions  $x, y, z$  sont traitées comme avec l’algorithme statique et les vitesses  $\dot{x}, \dot{y}, \dot{z}$  sont initialisées aléatoirement. Ensuite, à partir de la deuxième paire d’images, considérant comme acquis le chromosome  $(x, y, z, \dot{x}, \dot{y}, \dot{z})$  à l’instant  $t$ , on peut extrapoler ses valeurs à l’instant  $t + \Delta t$  comme  $(x + \dot{x}\Delta t, y + \dot{y}\Delta t, z + \dot{z}\Delta t, \dot{x}, \dot{y}, \dot{z})$ , puis faire évoluer  $\dot{x}, \dot{y}, \dot{z}$  (les deux fonctions d’objectif que nous avons utilisées sont décrites au § 3.4). A la fin de cette étape on met à jour les coordonnées de position à  $x + \dot{x}\Delta t, y + \dot{y}\Delta t, z + \dot{z}\Delta t$ , et la population est prête pour le pas suivant. On peut considérer cette méthode comme l’introduction d’une mémoire de vitesse des mouches. Les changements par rapport à l’algorithme statique sont modérés, et concernent surtout l’application de l’évolution aux vitesses.

### 3.3. Méthode passéiste

Pour mieux exploiter la redondance temporelle, nous avons introduit l’approche “passéiste” : pour évaluer la mouche dont le chromosome vaut  $(x, y, z, \dot{x}, \dot{y}, \dot{z})$  à l’instant  $t$ , on évalue à la fois la cohérence de  $(x, y, z)$  avec le couple stéréo courant et la cohérence de  $(x - \dot{x}\Delta t, y - \dot{y}\Delta t, z - \dot{z}\Delta t)$  avec le couple précédent. Cette évaluation peut s’effectuer de deux manières (§3.4). Ici, les 6 paramètres du chromosome sont exploités plus systématiquement qu’avec la méthode futuriste. Cette méthode nécessite la réécriture d’une partie importante de l’algorithme. Une comparaison qualitative de ces méthodes est donnée au § 3.7.

### 3.4. Fonction d’objectif dynamique: consensus ou comparaison de niveaux de gris?

La fonction d’objectif statique décrite au § 2.1 n’utilise que les paramètres de position  $(x, y, z)$ . Avec l’introduction des composantes de vitesses il devient nécessaire de redéfinir une “fonction d’évaluation dynamique” qui prenne en compte les 6 composantes du nouveau chromosome de la mouche.

La première méthode (“consensus”) consiste à calculer la fonction d’objectif statique sur la position de la mouche à l’instant précédent  $(x - \dot{x}\Delta t, y - \dot{y}\Delta t, z - \dot{z}\Delta t)$ , et celle sur la position à l’instant courant  $(x, y, z)$  : on ne donne à la mouche une fonction d’objectif dynamique élevée que si ces deux valeurs statiques sont élevées. Pour éviter de biaiser le critère de ressemblance par une éventuelle différence du partage, on ne fait pas intervenir le partage dans la comparaison de l’ancienne et de la nouvelle fonction d’objectif :

$$fitness\ dynamique = fitness_t \exp\left(-\frac{(fitness_t - fitness_{t-\Delta t})^2}{2s^2}\right)$$

où le paramètre  $s$  permet de régler la sensibilité à la dispersion de la fonction d'objectif.

La deuxième méthode (“*comparaison de niveaux de gris*”) compare les niveaux de gris des 4 images<sup>8</sup> provenant des deux couples stéréo consécutifs. On ne donne des valeurs élevées qu'aux mouches dont les projections sur les 4 images ont des niveaux de gris proches : pour une mouche de chromosome  $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ , ses coordonnées courantes sont  $(x, y, z)$  et ses coordonnées passées  $(x - \dot{x}\Delta t, y - \dot{y}\Delta t, z - \dot{z}\Delta t)$ : on calcule les quatre imageries (resp.  $a, b, c, d$ ) des voisinages de ses projections sur l'image courante gauche, l'image courante droite, l'image passée gauche, l'image passée droite, et la fonction d'objectif est définie comme

$$fitness = \frac{1}{R(a, b) + R(c, d) + K(R(a, c) + R(b, d))}$$

où  $R(a, b)$  est la somme des carrés des différences pixel à pixel des imageries  $a$  et  $b$ , et  $K$  un coefficient de pondération. Les deux premiers termes pénalisent les mouches qui ont de mauvaises positions, les deux termes suivants pénalisent celles qui ont de mauvaises vitesses.

### 3.5. Opérateurs génétiques

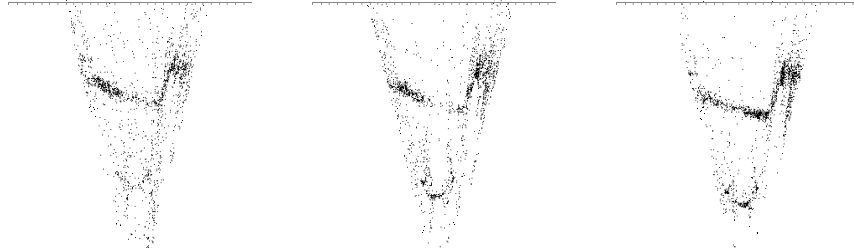
Pour adapter la version statique de l'algorithme au traitement des séquences, nous avons vu qu'il suffit d'introduire l'opérateur d'immigration qui permet à l'algorithme d'assurer une fonction de veille vis-à-vis des événements nouveaux. Dans la version dynamique, il faut étendre les opérateurs génétiques aux composantes de vitesse. L'immigration crée des individus nouveaux dans le champ des caméras, et leur donne des vitesses tirées aléatoirement dans un domaine donné. La mutation affecte le vecteur vitesse en perturbant son module et sa direction. Le croisement barycentrique crée une mouche dont la vitesse est cohérente avec l'hypothèse de l'appartenance de ses deux parents à un même objet solide.

### 3.6. Initialisation

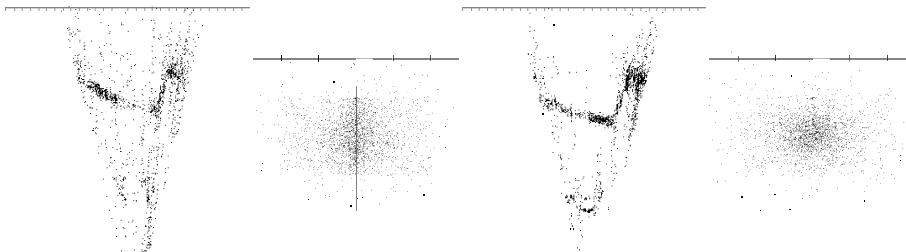
L'algorithme dynamique est initialisé comme la version statique. Pour le deuxième couple stéréo il faut initialiser les vitesses. Dans les exemples ci-dessous, on connaissait la vitesse a priori du robot par rapport à la scène et on a pu initialiser les vitesses des mouches autour de l'opposé de cette vitesse. La connaissance *a priori* sur la vitesse permet de réduire la variance de la distribution originale de vitesses et d'accélérer la convergence dans la majorité des cas.

8. Nous avons aussi appliqué cette méthode au cas d'une caméra unique en mouvement, mais la précision est alors médiocre au voisinage du foyer d'expansion.

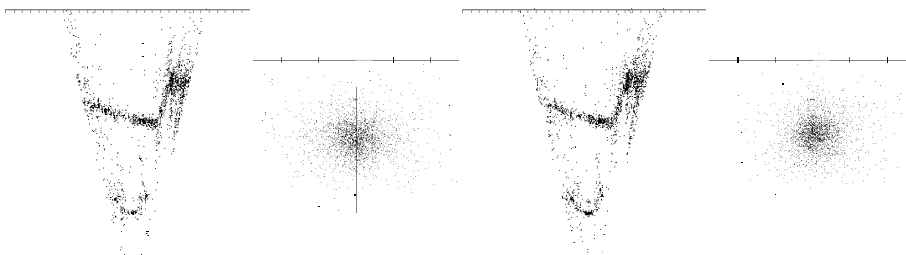
### 3.7. Résultats



**Figure 14.** Méthode aléatoire : résultats sur les images 2,4,6 de la séquence de test.



**Figure 15.** Approche futuriste, évaluation par comparaison de fonction d'objectif: résultats (positions, vitesses) sur les images 2 (à gauche) et 6 (à droite). Les images des vitesses sont agrandies



**Figure 16.** Comparaison de niveaux de gris sur 4 images : approche futuriste (gauche) et passéiste (droite). Dans les deux cas les positions sont mieux détectées. L'approche passéiste, plus coûteuse, converge mieux sur les vitesses

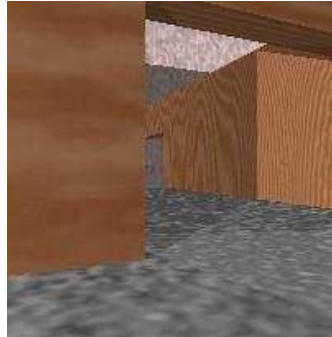
Les figures 14 à 16 montrent les résultats obtenus avec la même séquence de 6 couples stéréo d'une scène d'intérieur. L'ordre de grandeur connu de la vitesse d'avancement  $V$  du robot a permis d'initialiser les dérivées des vitesses axiales  $\dot{z}$  des

mouches en utilisant une distribution uniforme dans  $[-\frac{3V}{2}, \frac{3V}{2}]$ . Les autres composantes de vitesse  $\dot{x}$ ,  $\dot{y}$  sont initialisées avec une distribution gaussienne autour de 0. Sauf mention du contraire, les figures montrent les résultats à la 6ème image de la séquence, projetées sur le plan horizontal  $(x, z)$ , la caméra étant orientée vers le haut de la page. Les images de vitesse (Figures 15 - 16) représentent les projections des vecteurs vitesse dans le plan horizontal. Concernant les positions, les résultats sont plus convaincants lorsque la fonction d'objectif est basée sur la comparaison des niveaux de gris sur 4 images. La méthode passéiste permet une convergence plus rapide des vitesses, et donc des résultats potentiellement meilleurs sur les longues séquences si le mouvement est suffisamment régulier, mais utilise plus de puissance de calcul que la méthode futuriste.

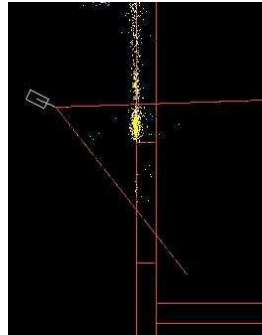
#### 4. Application à l'évitement d'obstacles pour un robot

L'algorithme des mouches fournit une description de la scène très différente de celle fournie par les algorithmes conventionnels de stéréovision, ce qui rend difficile la comparaison de la qualité des résultats. La traduction entre le langage "population de mouches" et le langage "polyèdres" semble inconfortable, et d'un point de vue applicatif sa complexité ferait perdre son intérêt à notre approche. Il n'est possible de les comparer qu'à travers les résultats d'une chaîne de traitement complète. Pour cela, nous avons choisi le problème de l'évitement d'obstacles pour un robot mobile, en adaptant aux nouvelles données un algorithme classique de navigation, et nous avons construit un simulateur simple de robot afin de valider la chaîne de traitement.

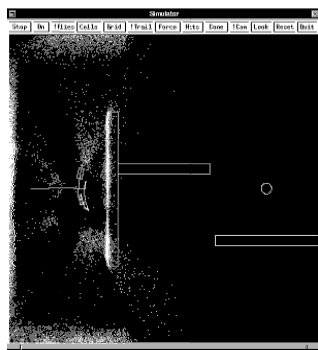
Pour ne pas perdre l'avantage que procure la rapidité potentielle de l'algorithme des mouches, nous avons choisi d'implanter une méthode simple et rapide de planification à base de potentiels [KOR 91, ZEL 98]. La force  $\vec{F} = \vec{F}_a + \vec{F}_r$ , dérivant de la somme d'un potentiel attractif (la cible) et d'un potentiel répulsif (les mouches), agit comme commande de direction. La force attractive  $\vec{F}_a$ , dirigée vers la cible, a une amplitude constante sauf au voisinage immédiat de la cible. La force répulsive  $\vec{F}_r$  tient compte de toutes les mouches présentes dans la mémoire locale du robot. La représentation 2-D de l'environnement, interne au robot, s'inspire des *robot evidence grids* de Martins & Moravec [MAR 96] : ici, chaque cellule concentre l'information contenue dans les mouches qu'elle contient pour doser sa contribution au potentiel répulsif, proportionnellement à la somme des fonctions d'objectif des mouches concernées. Le simulateur complet inclut : un simulateur du couple de caméras stéréo (utilisant PovRay), l'algorithme des mouches, le planificateur par potentiels décrit ci-dessus, et un simulateur cinétique simple du robot. Les situations de blocage du mouvement, correspondant à des minima locaux de potentiel, sont détectées et résolues à l'aide de deux heuristiques [BOU 01] qui créent une nouvelle force qui attire le robot hors du minimum local: méthodes par marche aléatoire et par suivi de murs (figures 19 - 22).



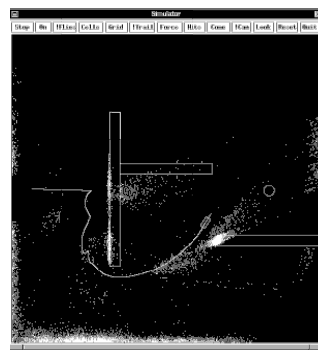
**Figure 17.** La scène vue par le robot



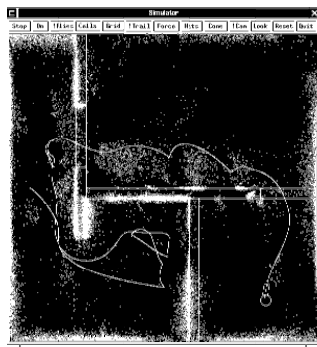
**Figure 18.** Le robot face à un mur et à une ouverture de porte. Les points blancs représentent les mouches mémorisées depuis les positions précédentes



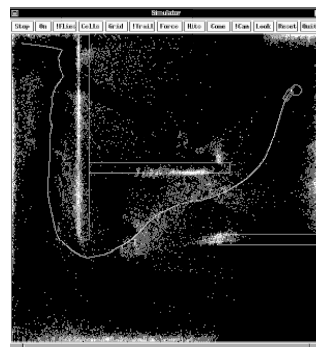
**Figure 19.** Situation de blocage



**Figure 20.** Résolution : cibles secondaires



**Figure 21.** Une trajectoire engendrée par la méthode des cibles secondaires (avec franchissement d'une porte)



**Figure 22.** Une trajectoire engendrée par suivi de murs

## 5. Conclusion

On considère souvent que l'évolution artificielle est lente et inadaptée aux applications en temps réel. Pourtant (voir tableau 1) on peut remarquer plusieurs particularités de l'application décrite dans cet article :

- les stratégies d'évolution sont normalement capables d'adaptation, c'est-à-dire d'accepter les modifications de la fonction d'objectif pendant l'exécution de l'algorithme [SAL 97]. Le "temps réel" se réfère précisément à la capacité d'un système à exploiter les données d'entrée dès qu'elles sont disponibles, et de tenir à la disposition de l'utilisateur des résultats constamment remis à jour ;
- les algorithmes conventionnels de traitement d'images [JAI 94] et de stéréovision [CUM 99], basés sur la segmentation, nécessitent un balayage complet de l'image avant que l'algorithme puisse calculer et délivrer le résultat. Ici, l'algorithme de traitement d'images sans segmentation tolère la mise à jour des données d'entrée pendant son exécution et utilise donc des données toujours à jour (tableau 1). Cette caractéristique est particulièrement intéressante avec les caméras CMOS qui permettent une lecture asynchrone des pixels ;
- les résultats de l'algorithme des mouches sont mis à jour continûment (au rythme, rapide, des générations), ce qui permet à l'utilisateur (qui peut être un planificateur de trajectoire de robot, par exemple) de réagir plus vite aux événements nouveaux ;
- la tâche habituellement la plus lourde en évolution artificielle est le calcul de la fonction d'objectif. Ici, l'"approche parisienne" permet de partager la représentation de la scène entre de très nombreuses primitives très simples, d'où une fonction d'objectif simple et un algorithme rapide.

	technique classique et CCD	CMOS + évolution parisienne
capteur d'image	<i>retard entre acquisition et restitution</i>	<i>lecture asynchrone possible</i>
traitement d'images (entrée)	<i>la segmentation d'images nécessite une image entière donc doit attendre la fin d'un cycle d'acquisition</i>	<i>interroge le capteur à chaque évaluation</i>
traitement d'images (sortie)	<i>doit avoir fini pour délivrer le résultat</i>	<i>résultats progressifs mais toujours disponibles</i>
planificateur	<i>doit attendre la fin du traitement d'images en cours</i>	<i>gagne 2 cycles d'acquisition</i>

**Tableau 1.** Comparaison de la vision classique et de la méthode proposée

De plus, les applications de traitement d'images agencent en général un séquençement d'opérateurs standard de manière spécifique à l'application : ici, la structure de l'application est largement indépendante de l'application, car la plus grande partie de la connaissance spécifique est concentrée dans la fonction d'objectif, et en dehors de celle-ci peu de choses doivent être modifiées si l'on veut transférer le programme vers une autre application en traitement d'images. Les recherches actuelles portent sur l'implantation de la méthode dans un projet de robotique mobile et sur le transfert des principes de la méthode en reconstruction d'images médicales.

## 6. Bibliographie

- [BOU 01] BOUMAZA A., LOUCHET J., «Using Real-time evolution in Robotics», EVOIASP2001, *Artificial Evolution in Image Analysis and Signal Processing*, avril 2001, Como, Italie.
- [COL 99] COLLET P., LUTTON E., RAYNAL F., SCHOENAUER M., «Individual GP: an Alternative Viewpoint for the Resolution of Complex Problems», *Genetic and Evolutionary Computation Conference GECCO99*, W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honovar, M. Jakiela, R. E. Smith (Eds.) Morgan Kaufmann: San Francisco, CA, 1999.
- [CUM 99] CUMULI project: «Computational Understanding of Multiple Images», <http://www.inrialpes.fr/CUMULI>
- [EBE 95] EBERHART R. C., KENNEDY J. A., «New Optimizer Using Particle Swarm Theory» *Proc. Sixth Int. Symposium on Micro Machine and Human Science*, Nagoya, IEEE Service Center: Piscataway, NJ, p. 39-43, 1995.
- [FON 00] FONLUPT C., «Applications des algorithmes évolutifs au traitement d'images», <http://www-lil.univ-littoral.fr/~fonlupt/Recherche/analyse/sld001.htm>
- [GOL 89] GOLDBERG D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley: Reading, MA, 1989.
- [HAR 80] HARALICK R. M., «Using Perspective Transformations in Scene Analysis», *Computer Graphics and Image Processing 13*, p. 191-221, 1980.
- [HOL 75] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press: Ann Arbor, 1975.
- [HOU 62] HOUGH P. V. C., *Method and Means of Recognizing Complex Patterns*, U.S. Patent n° 3, 069 654, 18 December 1962.
- [JAI 94] JAIN R. C., KASTURI R., SCHUNCK B. G., *Machine Vision*, McGraw-Hill: New York, 1994.
- [KOR 91] KOREN Y. and BORENSTEIN J., «Vfh+: Potential field methods and their inherent limitations for mobile robot navigation», *Proceedings of the IEEE conference on Robotics and Automation*, ICRA'91, p. 1398-1404, Sacramento, California, April 7-12 1991.
- [LOU 00] LOUCHET J., «Stereo Analysis Using Individual Evolution Strategy», *International Conference on Pattern Recognition*, ICPR2000, Barcelona (Espagne), September 2000.
- [LOU 01] LOUCHET J., «Using an individual evolution strategy for stereovision», *Genetic Programming and Evolvable Machines*, Vol. 2, n° 2, Kluwer Academic Publishers, p. 101-109, mars 2001.

- [LUT 94] LUTTON E., MARTINEZ P., «A Genetic Algorithm for the Detection of 2D Geometric Primitives in Images», *12th International Conference on Pattern Recognition*, ICPR, Jerusalem, Israel, S. Ulman, S. Peleg (gen. co-chairs), October 9-13, 1994, IEEE Computer Society: Los Alamitos, CA, p. 526-528, 1994.
- [MAR 96] MARTINS M. C., MORAVEC H. P., «Robot evidence grid», technical report, the Robotics institute, Carnegie Mellon University, March 1996.
- [MIL 94] MILLONAS M., «Swarms, phase transitions and collective intelligence», *Artificial Life III* (ed. C.G Langton), Santa Fe Institute Studies in the Sciences of Complexity, Vol. XVII, Addison Wesley: Reading, MA, 1994.
- [REC 94] RECHENBERG I., «Evolution Strategy», in J.M. Zurada, R.J. Marks II, C.J. Robinson, *Computational Intelligence imitating life*, IEEE Press: Piscataway, NJ, 147-159, 1994.
- [ROT 92] ROTH G., LEVINE M. D., «Geometric Primitive Extraction using a Genetic Algorithm», *IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition 1992*, IEEE Press: Piscataway, NJ, p. 640-644, 1992.
- [SAL 97] SALOMON R., EGGENBERGER P., «Adaptation on the Evolutionary Time Scale: a Working Hypothesis and Basic Experiments», *Artificial Evolution '97*, J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), Springer Lecture Notes on Computer Science n° 1363, Springer-Verlag: Berlin, p. 251-262, 1997.
- [SER 99] SER P.K., CLIFFORD S., CHOY T., SIU W. C., «Genetic Algorithm for the Extraction of Nonanalytic Objects from Multiple Dimensional Parameter Space», *Computer Vision and Image Understanding*, vol. 73 n° 1, Academic Press: Orlando, FL, p. 1-13, 1999.
- [TAN 98] TANG C. K., MEDIONI G., «Integrated Surface, Curve and Junction Inference from Sparse 3-D Data Sets», *International Conference on Computer Vision ICCV98*, Bombay, IEEE Computer Society Press: Piscataway, NJ., pp. 818-823, 1998.
- [ZEL 98] ZELEK J. S., «Complete real-time path planning during sensor-based discovery», *IEEE RSJ International Conference on Intelligent Robots and Systems*, IEEE Computer Society Press: Piscataway, NJ 1998.