Dynamic Flies: a new pattern recognition tool applied to stereo sequence processing

Jean LOUCHET*† Maud GUYON* Marie-Jeanne LESOT* Amine BOUMAZA†

*Ecole Nationale Supérieure de Techniques Avancées
32 boulevard Victor, 75739 Paris cedex15, France
e-mail:[louchet|guyon|lesot]@ensta.fr
URL: http://www.ensta.fr/~louchet

†INRIA Rocquencourt
BP75, 78153 Le Chesnay cedex, France
e-mail: boumaza@inria.fr
URL: http://www-rocq.inria.fr/~boumaza

Abstract.

The "fly algorithm" is a fast artificial evolution-based technique devised for the exploration of parameter space in pattern recognition applications. In the application described, we evolve a population which constitutes a particle-based three-dimensional representation of the scene. Each individual represents a three-dimensional point in the scene and 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 and low complexity fitness function allow fast processing. Test results and an application to mobile robotics are presented.

Keywords

Artificial evolution, pattern recognition, computer vision, image processing, parameter space exploration.

1 Introduction

Parameter-space vote approaches to computer vision and pattern recognition like the generalised Hough transform [8] are limited by the number of parameters and the size of the search space. Artificial Evolution [5][7][16] seen as a class of parameter space exploration techniques, provides successful solutions to some image processing problems [13][17][19] but often suffer from the costly calculations needed if they have to manipulate populations where each individual is in itself a complex description of the scene. The "Parisian" approach [2][11][12] to artificial evolution introduces each individual as a part of the solution of the problem. Thus, the solution is represented by the total population or by a large part of it. The algorithm presented in this paper uses 3-D points[1] ('flies') as the population's individuals, and evolves the population of flies using a pixel-based fitness function such that the flies concentrate onto the objects to be detected in the scene[2] (Section 2). Section 3 shows an extension to stereo image sequences through the introduction of velocity parameters. An application to mobile robot obstacle avoidance is shown in Section 4.

2 Evolving Flies

2.1 Geometry and Fitness Function

A fly is defined as a 3-D point with coordinates (x, y, z). The coordinates of the fly's projections are (x_L, y_L) in the image given by the left camera and (x_R, y_R) for the right camera. The cameras' calibration parameters

[1] Particle swarms [15][4] also use 3-D points as primitives. They were designed as optimisation tools using a metaphor of particles moving towards a target, using collective behaviours inspired by particle systems used in computer graphics. The essential difference is that our approach is based on evolutionary operators (selection, mutation, crossover) which are not used in particle swarm techniques. However one can outline some similarities between these two approaches: e.g. random speeds vs. mutations, or mutual avoidance vs. sharing.

[2] The tensor voting approach of Tang and Medioni [20] also uses a random distribution of 3-D points which do not evolve but are only used to initialise the calculation of a dense tensor field.

are known and so x_L, y_L, x_R, y_R may be easily calculated from x, y, z using projective geometry [6][9]. If the fly is on the surface of an opaque object, then the corresponding pixels in the two images will normally have the same grey level[3]. Conversely, if the fly is not on the surface of an object, thanks to the non-uniformity of objects and illumination, the grey levels of its projections and their immediate neighbourhoods will be probably not identical. This is expressed as a fitness function used to control the evolution of the flies' population from random initial positions to converge onto the surfaces of the apparent objects.

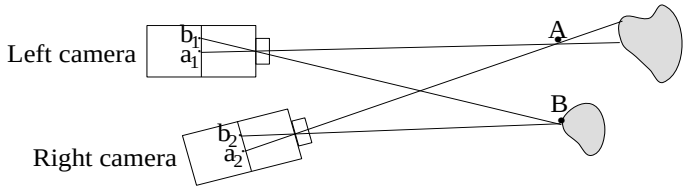


Fig. 1. Pixels *b*1 and *b*2, projections of fly B, have identical grey levels, while pixels *a*1 and *a*2, projections of fly A, which receive their illumination from two different physical points on the object's surface, have different grey levels.

The fitness function evaluates the degree of similarity of the pixel neighbourhoods of the projections of the fly onto each image. This ensures highest fitness values for the flies lying on the surface of an object :

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

- (x_L, y_L) and (x_R, y_R) are the coordinates of the left and right projections of the current individual (see Fig. 1),
- $L(x_L + i, y_L + j)$ is the grey value of the left image at pixel $(x_L + i, y_L + j)$, similarly with R for the right image.
- N is a neighbourhood introduced to obtain a more discriminant comparison of the fly's projections.

On colour images, square differences are calculated on each colour channel.

The normalizing factor is a local contrast measurement based on an image gradient calculation, giving higher fitness values to flies which project onto contrasted or textured areas of the images, and reducing the fitness of those over uniform regions. Experiments showed that best results are obtained when G is defined as the square root of an image gradient norm such as Sobel's [9]. Thus, highest fitness values are obtained for flies whose projections have similar *and significant* pixel surroundings. Additionally, we modified the denominator of the fitness function to suppress the continuous component and reduce its sensitivity to lower spatial frequencies (e.g. due to different camera responses).

2.2 Artificial Evolution Operators

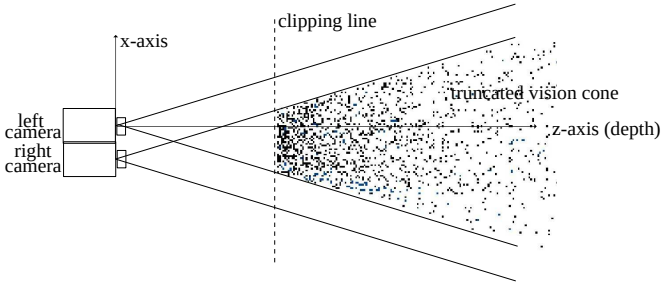


Fig. 2. The fly population is initialised inside the intersection of the cameras 3-D fields of view.

[3] Or the same colour components if colour images are used. This is essentially true with matt (Lambertian) surfaces where scattering of incident light is isotropic. Usually, surfaces differ slightly from the Lambertian model, but this may be at least partly taken into account with the fitness function (see the remark on low spatial frequencies, Section 2.1). Reflections from glossy surfaces may give rise to virtual objects and wrong 3-D interpretation, as with other stereo analysis algorithms.

An individual's chromosome is the triple (x, y, z) which contains the fly's coordinates. The population is initialised randomly inside the intersection of the cameras fields of view, from a given distance (clipping line) to infinity (Fig. 2). The statistical distribution is chosen in order to obtain a uniformly distributed projections in the left image. Their depth is chosen by uniformly distributing the values of z^{-1} between zero and $1 / d_{min}$.

Selection is elitist and deterministic. It ranks the flies according to their fitness values and retains the best individuals (typically 50%).

2-D *sharing* [7] reduces the fitness values of flies located in crowded areas to prevent them from getting concentrated into a small number of maxima. It reduces each fly's fitness by $K \times N$, where K is a "sharing coefficient" and N the number of flies which project into the left image within a distance R ("sharing radius") from the given fly. We determined an empirical rule to choose a satisfactory sharing radius R, considering that if the image is divided into squares of width $2R + 1$ pixels, there should ideally be one fly per square on average. Therefore $(2R + 1)^2 \times N_{flies} \approx N_{pixels}$ and:

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

For example, on a 500×500 pixel image with 5000 flies, this gives $R = 3$.

Mutation allows an extensive exploration of the search space. It uses an approximation of a Gaussian random noise added to the flies' chromosome parameters (x, y, z). We chose standard deviations σ_x, σ_y, σ_z equal to R, so that they are the same order of magnitude as the mean distance between neighbouring flies.

In real-world scenes, many objects contain straight lines or planar surfaces. We translated this geometrical property into a *barycentric crossover operator* which builds an offspring randomly located on the line segment between its parents: the offspring of two flies $F_1\ (x_1, y_1, z_1)$ and $F_2\ (x_2, y_2, z_2)$ is the fly $F_3\ (x_3, y_3, z_3)$ defined by $\overrightarrow{OF_3} = \lambda \overrightarrow{OF_1} + (1 - \lambda)\overrightarrow{OF_2}$. The weight λ is chosen using a uniform random law in $[0,1]$ [4].

2.3 Results

2.3.1 Synthetic Images

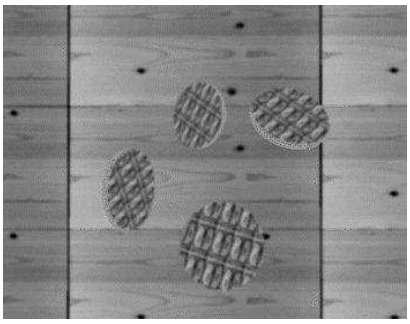


Fig. 3. "Money" image, left

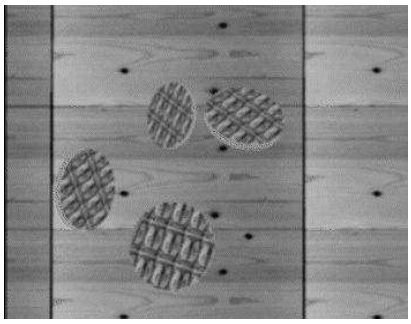


Fig. 4. "Money" image, right.

Results are shown with the synthetic "Money" stereo image pair[5] (Figs. 3 and 4) which allows easy readability of results, as most objects are vertical. Fig. 5 shows convergence results using a 5000 individual population, 50% mutation[6] and 10% crossover rates[7] at different stages of convergence. The scene is seen from

[4] It is generally accepted that such a crossover operator has contractive properties which may be avoided by using a larger interval. However the goal of this crossover operator is to fill in surfaces whose contours are easier to detect, rather than to extend them. It is therefore not desirable to use coefficients allowing the centre of gravity to lie outside of the object's boundary. This is confirmed by most of our experiments showing that there is no benefit using a wider interval, unlike in other applications of barycentric crossover.

[5] "Money" colour image pair, 384×288 pixels, ©INRIA - Mirages project.

[6] Unusually high mutation rates give best results in Parisian evolution.

[7] Experiments showed that best results are obtained with elitist selection, keeping the total mutation plus crossover rate

above , showing the axes *x* (right) and *z* (camera axis, pointing towards the top of the page).

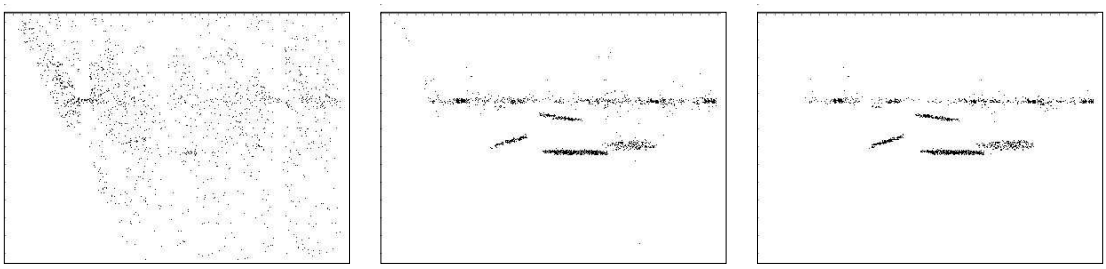


Fig. 5. Convergence results after 10 (left), 50 (centre) and 100 (right) generations. The line of flies which appears at the background corresponds to the wooden panel in the image background, the four fly clusters in the foreground correspond to the 4 coins. Note that the coin on the right is not vertical and gives a less distinct cloud.

2.3.2 Fine tuning the parameters

2.3.2.1 Sharing

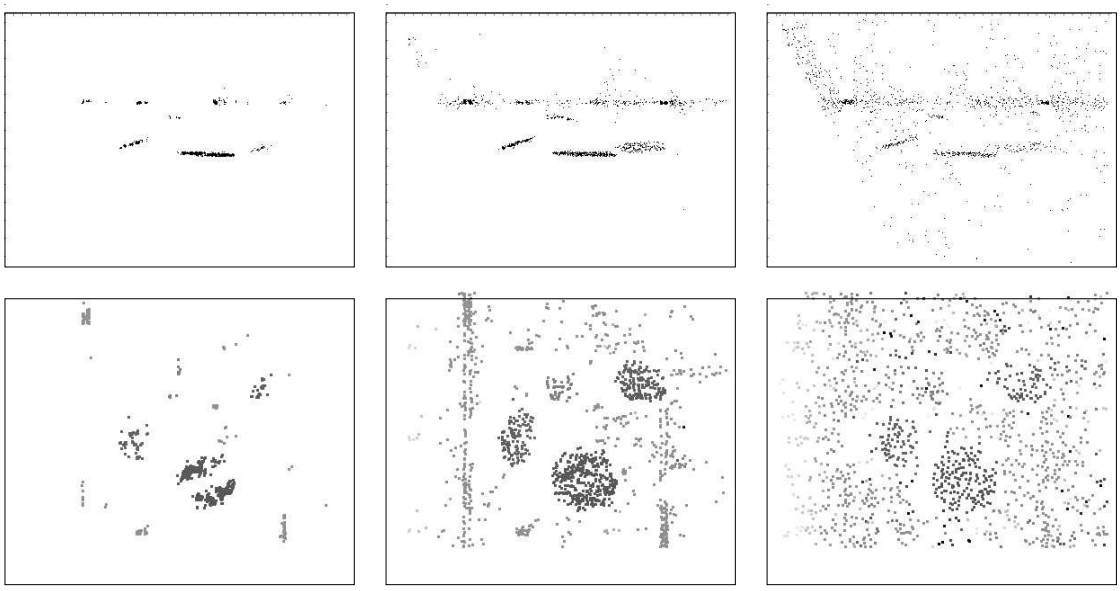


Fig. 6. No sharing

Fig. 7. Medium sharing (radius 2, coefficient 0.2)

Fig. 8. High sharing (radius 3, coefficient 0.5)

Figures 6 - 8 show how sharing affects the results on the same image data, showing the 1500 best individuals (30% of 5000) after 50 generations, with a mutation rate of 60%, no crossover and different sharing coefficients. Top images show the fly distributions seen from above (plan view), bottom images are front views (as seen from the camera) where darker pixels represents flies at shorter distances. Low sharing gives unwanted fly concentrations into high gradient regions; high sharing coefficients tend to degrade the results.

2.3.2.2 Mutation and Crossover Rates

Fig. 9 shows typical average fitness evolution curves with different combinations of mutation and crossover rates. Best results are obtained with mutation and crossover rates around 50% and 10% on interior scenes. The bottom curve corresponds to a mutation-only evolution.

around 60%.

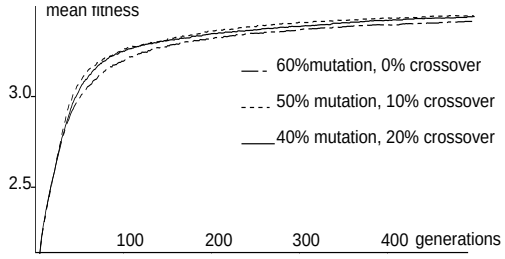
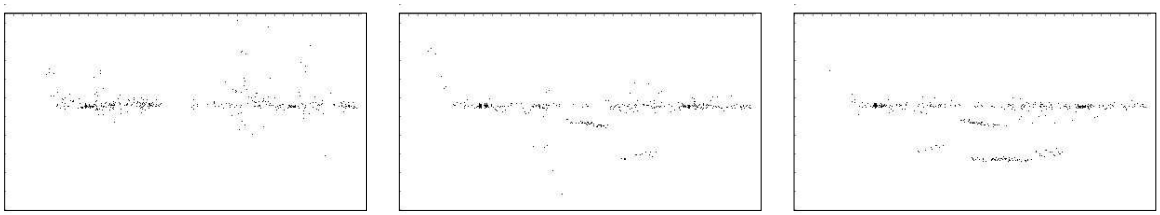


Fig. 9. Evolution of average fitness values for a 5000 individual population using three different combinations of mutation and crossover rates.

2.3.2.3 Small Populations

When smaller populations are used (e.g. to reduce computation time) the sharing coefficient must be increased to ensure a fair repartition of the population, as seen in Section 2.2. In addition, higher crossover rates then allow more reliable object detection (see typical example on Fig. 10). A fair all-purpose compromise is around 10-20%, depending on the importance of planar objects in the scene.



mutation 60%, no crossover | mutation 40%, crossover 20% | mutation 20%, crossover 40%

Fig. 10. Using different crossover rates with a small population:1000 flies, 100 generations

Experiments on real data show that the choice of population size depends on processing time constraints and on the level of detail required in the application (size of objects to be detected (see details below).

2.3.3 Results on real images

We tested the algorithm on monochrome stereo 768 × 576 image pairs of interior scenes (Figs.11 - 12). Typical results are shown on Fig.13, using similar genetic parameters (5000 flies, 100 generations, 40% mutation, 20% crossover, sharing radius 2, sharing coefficient 0.3). One can see the two sides of the cabinet, the beginning of the wall on the right and the front half-circle of the stool in spite of its small number of pixels.

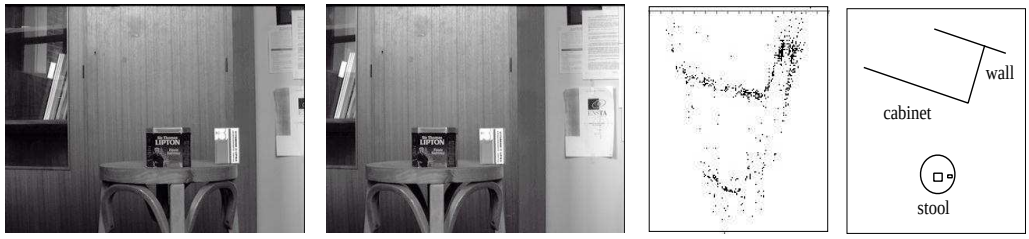


Fig. 11. Left image *Fig. 12.* Right image *Fig. 13.* Results and plan view

3 Detecting mobile obstacles: real time and dynamic flies

We have extended the algorithm to process stereo pair sequences taken by moving cameras.

3.1 Random method

The random approach consists in keeping the same population evolving through frame changes. Thus, only the images (and therefore the parameters used by the fitness function) are modified while the fly population evolves. When motion is slow enough, using the results of the last step speeds up convergence significantly

compared to using a totally new random population. This method may be seen as the introduction of a collective memory of space, exploiting the similarity between consecutive scenes. It does not alter much the simplicity of the static method and requires minimal algorithm changes. An extra mutation is applied to flies' positions at each frame change. To improve detection of new objects appearing in the field of view, we introduced an extra mutation operator, *immigration*, which creates new flies randomly in a way similar to the procedure already used to first initialise the population: used with a low probability (1% to 5%) this favours a convenient exploration of the whole search space.

3.2 Forward method

In order to process faster motion, we introduced an extended chromosome, which contains both position and speed coordinates in the cameras' mobile coordinate system, and modified the fitness and genetic operators accordingly. To exploit the redundancy of data and the likely continuity of velocities in most applications, the most handy approach ("*forward approach*") is the following. Each fly's genome is the 6-uple $\left(x, y, z, \dot{x}, \dot{y}, \dot{z}\right)$, At the first frame of the sequence, the positions x, y, z are evolved in the usual way, and the velocities $\dot{x}$, $\dot{y}$, $\dot{z}$ are initialised randomly. Then for the following frames in the sequence, knowing the fly's genome $\left(x, y, z, \dot{x}, \dot{y}, \dot{z}\right)$ at instant t, we are able to extrapolate its genome at instant $t + \Delta t$ as $\left(x + \dot{x}\Delta t, y + \dot{y}\Delta t, z + \dot{z}\Delta t, \dot{x}, \dot{y}, \dot{z}\right)$, and now evolve the speed components $\dot{x}$, $\dot{y}$, $\dot{z}$. The two possible fitness functions we use are described in Section 3.5. At the end of this step, the flies' coordinates will be updated to $x + \dot{x}\Delta t$, $y + \dot{y}\Delta t$, $z + \dot{z}\Delta t$, so that the population is ready for the next step.

This method may be considered as the introduction of a speed memory in flies. It does not involve important changes in the program, except that evolution is applied to velocities rather than positions.

3.3 Backward method

To better exploit time redundancy, we have introduced the "*backward*" approach: to evaluate a fly with genome $\left(x, y, z, \dot{x}, \dot{y}, \dot{z}\right)$, containing position and speed components at instant t, we calculate both the coherence of (x, y, z) with the current image pair and the coherence of $\left(x - \dot{x}\Delta t, y - \dot{y}\Delta t, z - \dot{z}\Delta t\right)$ with the previous images. This can be done two different ways, as detailed in Section 3.5.

This approach means that the set of pairs (old position, current position) and therefore all six parameters of the genome $\left(x, y, z, \dot{x}, \dot{y}, \dot{z}\right)$ are explored more systematically than with the forward method. It implies that the algorithm has been thoroughly rewritten. The results given in Section 3.8 allow a qualitative comparison of these approaches.

3.4 Dynamic fitness functions: fitness consensus or grey level comparison?

The static fitness function described in Section 2.1 only involves the position parameters (x, y, z) of a fly. It is now necessary to define a "dynamic fitness function" to take into account two consecutive stereo pairs and evaluate a 6-component fly genome.

The first method ("*fitness consensus*") consists in calculating the static fitness of the fly's old position $\left(x - \dot{x}\Delta t, y - \dot{y}\Delta t, z - \dot{z}\Delta t\right)$ on the old images, and calculating the static fitness of its new position (x, y, z) on the new images; then a fly will be given a high dynamic fitness only if both static fitness values are high. For the results not to be biassed by possible different sharing values, only the new position of the fly is used to calculate both fitness values, and the dynamic fitness is defined as the product of the new static fitness with a Gaussian function of the difference between old and new fitness:

$$dynamic\ fitness\ =\ fitness_1\ exp\left(-\frac{(fitness_1\ -\ fitness_2)^2}{2s^2}\right)$$

where the parameter *s* allows adjustment of tolerance on fitness dispersion. The flies' fitness values are easily stored from the preceding generation.

The second method consists in comparing the grey levels of all 4 images[8] ("*grey level comparison*") from two consecutive stereo pairs, and only give high fitness values to flies whose 6 coordinates give similar grey levels on all 4 images. We built the dynamic fitness function as the inverse of the sum of the four grey level differences: old left vs. new left, old right vs. new right, old left vs. old right, new left vs. new right.

The first two terms will give a penalty to flies with wrong speeds, the last ones to flies with wrong positions: we had to introduce and adjust weight coefficients experimentally.

3.5 Genetic operators

In order to use the static version of the algorithm with image sequences, we introduced the "immigration" operator, which helps the algorithm take into account new objects that would appear in the scene but would not be useful in processing static scenes.

In the dynamic version, the genetic operators: immigration, mutation and barycentric crossover, are similar to those used in the static version. Immigration again creates totally new individuals located in the intersection of the two cameras' fields of view, and speeds randomly chosen in a given domain. Mutation now affects the norm and direction of the speed vector.

3.6 Initialising the population

The algorithm is initialised using its static version on the first frame. At the second frame it is necessary to initialise the velocities as well, which is left to the user. In the examples below, the simulated robot's speed was known and used to help initialise the flies' velocities. Accurate *a priori* knowledge on speed allows reducing the variance of the initial distribution, privilege local research and help convergence.

3.7 Results

Results in Figures 14 - 16 have been obtained with the same sequence of 6 image pairs of an interior scene. The order of magnitude V of the robot's speed was known, and used to initialise the flies' depth derivatives $\dot{z}$ using a uniform distribution in the interval $\left[-\frac{3V}{2},\ \frac{3V}{2}\right]$. The other components $\dot{x}$, $\dot{y}$ have been initialised with a Gaussian distribution around 0. Unless otherwise stated, figures show the results obtained at the 6th image, projected on the horizontal plane (x, z), the camera pointing towards the top of the page. Velocity images (Figures 15 - 16) show the projections of the velocity vectors on the horizontal plane $(\dot{x}, \dot{z})$.

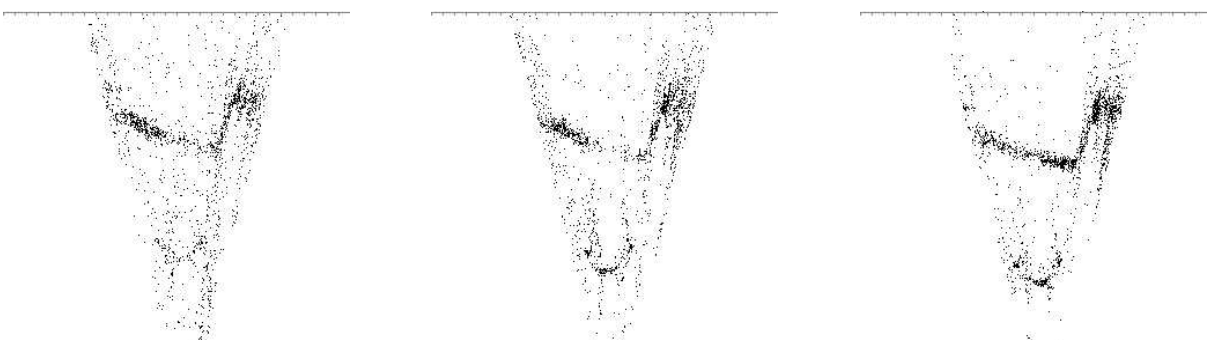


Fig. 14 : Random method: results on images 2, 4, 6 of the sequence.

[8] It is also possible to apply this method using a single moving camera, but its accuracy may then be limited around the focus of expansion if the direction of motion is close to the camera axis.

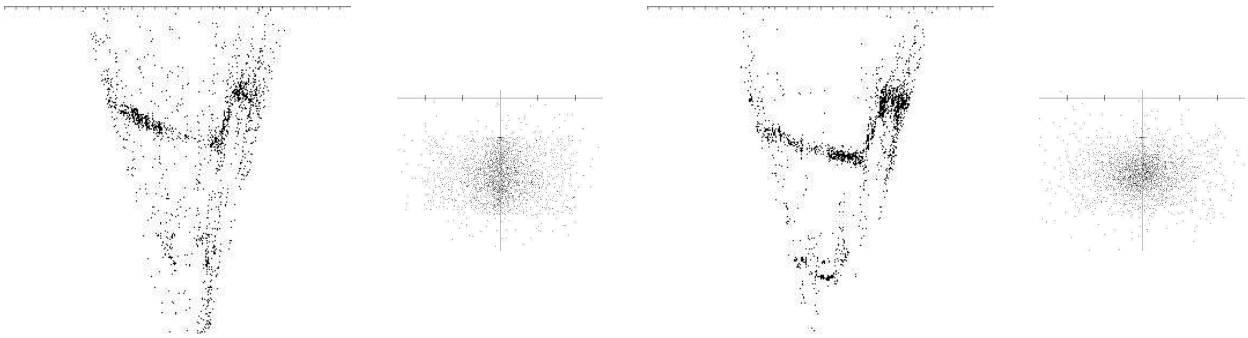


Fig. 15 : Forward approach, evaluation by fitness comparison: results on frames 2 (left) and 6 (right). Second and fourth images show velocities (magnified).

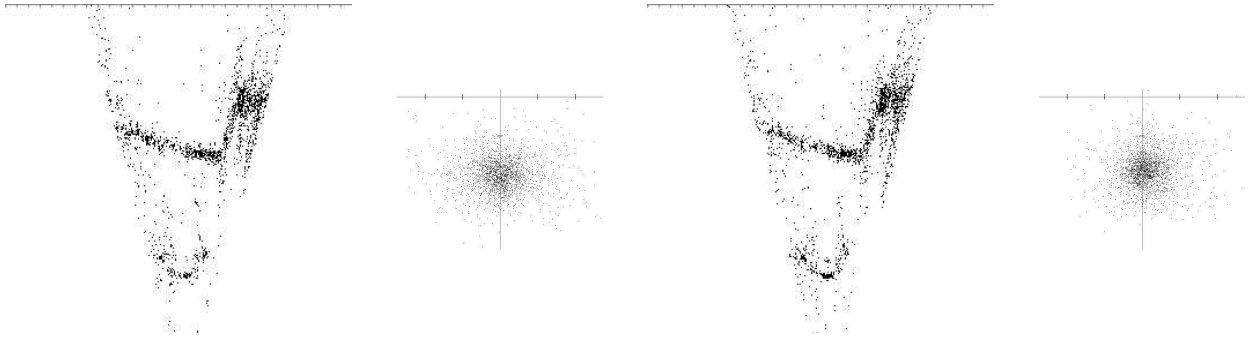


Fig. 16 : Grey level comparison on 4 images: forward approach (left) and backward approach (right). With both methods, improvement on positions is obvious. The backward approach gives better velocities but needs longer calculations.

Concerning positions, results are more convincing when the fitness function is based on the comparison of grey levels on 4 images. The backward method gives a better convergence of speeds, and therefore possibly better results on long image sequences when motion is smooth enough, but uses more computing resources.

4 Application to robot obstacle avoidance

The unusual way the scene is described using the fly algorithm (compared to more classical stereovision algorithm used in mobile robotics), led us to adapt a classical robot navigation method in order to use the results of the fly algorithm as input data, and build a simulator.

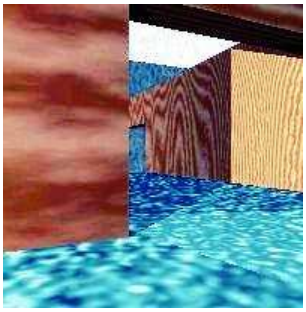


Fig. 17 : the scene seen by the robot.

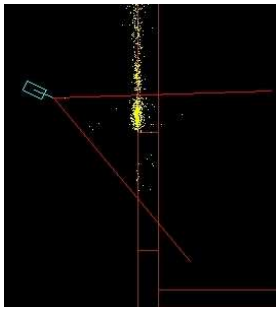


Fig. 18 : the robot facing a wall and a door aperture. White dots represent the flies memorised from the preceding positions.

To keep the potential speed benefit of the fly algorithm, we chose to implement a fast potential-based method [10][21]. We define a force derived from the sum of an attractive (the target) and a repulsive (the flies)

potential $\vec{F} = \vec{F_a} + \vec{F_r}$, acting as a steering command. The attractive force $\vec{F_a}$ points towards the target and has a constant amplitude except in the close neighbourhood of the target. The repulsive force $\vec{F_r}$ takes into account all the flies. Our internal 2-D representation of the robot's environment is inspired by Martin & Moravec's *robot evidence grids* [14]: each cell concentrates the information contained in the flies it contains and generates its contribution to the repulsive force, proportionally to the sum of the flies' fitness values. The complete simulator includes: a stereo camera pair simulator (using PovRay)(Figure 17), the fly algorithm, the potential-based planner described above, and a simple kinetic robot simulator. Blockage situations corresponding to local potential minima, are solved using two heuristics [1] which create a new force attracting the robot out of the potential minimum (random walk and wall following methods) (Figures 18 - 22).

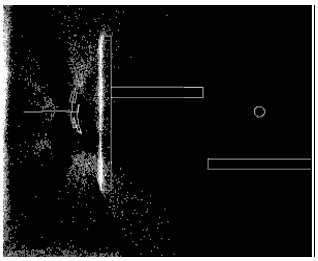


Fig. 19 : blocked situation

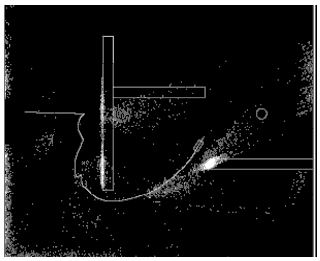


Fig. 20 : resolution using secondary targets

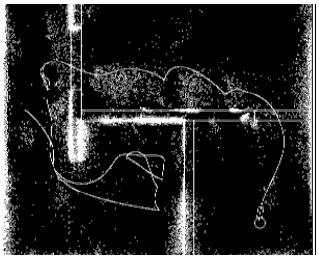


Fig. 21 : another trajectory generated using secondary targets

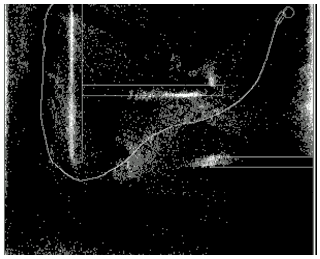


Fig. 22 : a trajectory generated by wall following

5 Conclusion

It is often considered that artificial evolution is slow and not suited to real-time applications. However,

- Evolution strategies are generally capable of adaptation, i.e. to cope with modifications of the fitness during the algorithm's execution [18]. Real time refers to the ability of the system to exploit input data as soon as they become available, and provide the user with suitable updated results at any time.
- With conventional image processing algorithms [9][3], a frame must have been fully scanned before the algorithm can calculate and deliver the result. The segmentation-free fly algorithm tolerates data updating during its execution and is always using freshly updated data. This feature can be exploited using CMOS camera technogy allowing asynchronous data reading.
- The fly algorithm's results are updated continuously (at the rate of generations), which enables the user (e.g. a robot path planifier) to react faster to external events.
- The main time-consuming task in artificial evolution is usually calculating the fitness function. Here, the "Parisian approach" allows to spread the scene representation over a large number of extremely simple primitives, hence a simple fitness function and a fast algorithm.

Fairly unusual in image processing, the program's structure is largely application-independent, as most of

the problem-specific knowledge is expressed in the fitness function, enabling easy transfer to other applications. We are currently implementing the method described in this paper into ENSTA's mobile robot project, and are developing an application of the same pattern recognition technique to medical image reconstruction.

Average processing time is from 7 to 25 milliseconds per generation for the random method, on a 800MHz PentiumIII PC, Linux, gcc, and may go up to about 50 milliseconds with the backward method, depending on population size and genetic parameters. Detailed results may be found on the site:
`http://www.ensta.fr/~louchet/FlyBenchmark.html`

References

1. Amine Boumaza, Jean Louchet, *Using Real-time evolution in Robotics*, EVOIASP2001, Artificial Evolution in Image Analysis and Signal Processing, avril 2001, Como , Italy.
2. P. Collet, E. Lutton, F. Raynal, M. Schoenauer, *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.
3. CUMULI project: *Computational Understanding of Multiple Images*, `http://www.inrialpes/fr/CUMULI`
4. R.C. Eberhart, J.A. Kennedy, *New Optimizer Using Particle Swarm Theory*, Proc. Sixth Int. Symposium on Micro Machine and Human Science, Nagoya, IEEE Service Center: Piscataway, NJ, pp. 39-43, 1995.
5. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley: Reading, MA, 1989.
6. R. M. Haralick, *Using Perspective Transformations in Scene Analysis*, Computer Graphics and Image Processing 13, pp. 191-221, 1980.
7. J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press: Ann Arbor, 1975.
8. P. V. C. Hough, *Method and Means of Recognizing Complex Patterns*, U.S. Patent n°3, 069 654, 18 December 1962.
9. R.C. Jain, R. Kasturi, B.G. Schunck, *Machine Vision*, McGraw'-Hill: New York, 1994.
10. Y. Koren and J. Borenstein, Vfh+: *Potential field methods and their inherent limitations for mobile robot navigation*, Proceedings of the IEEE conference on Robotics and Automation, ICRA'91, pp. 1398-1404, Sacramento, California, April 7-12 1991.
11. J. Louchet, *Stereo Analysis Using Individual Evolution Strategy*, International Conference on Pattern Recognition ICPR2000, Barcelona (Spain), September 2000.
12. J. Louchet, *Using an individual evolution strategy for stereovision*, Genetic Programming and Evolvable Machines, Kluwer, to appear, 2001.
13. E. Lutton, P. Martinez, *A Genetic Algorithm for the Detection of 2D Geometric Primitives in Images*, 12th ICPR, Jerusalem, Israel, S. Ulman, S. Peleg (gen. co-chairs), October 9-13, 1994, IEEE Computer Society: Los Alamitos, CA, pp. 526-528, 1994.
14. MC Martins, HP Moravec, *Robot evidence grid*, technical report, the robotics institute, Carnegie Mellon University, March 1996
15. M. Millonas, *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.
16. I. Rechenberg, *Evolution Strategy*, in J.M. Zurada, R.J. Marks II, C.J. Robinson, Computational Intelligence imitating life, IEEE Press: Piscataway, NJ, 147-159, 1994.
17. G. Roth, M. D. Levine, *Geometric Primitive Extraction using a Genetic Algorithm*, IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition 1992, IEEE Press: Piscataway, NJ, pp. 640-644, 1992.
18. R. Salomon, P. Eggenberger, *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 no. 1363, Springer-Verlag: Berlin, pp. 251-262, 1997.

19. P.K. Ser, S. Clifford, T. Choy, W.C. Siu, *Genetic Algorithm for the Extraction of Nonanalytic Objects from Multiple Dimensional Parameter Space*, Computer Vision and Image Understanding, vol. 73 no. 1, Academic Press: Orlando, FL, pp. 1-13, 1999.
20. C. K. Tang, G. Medioni, *Integrated Surface, Curve and Junction Inference from Sparse 3-D Data Sets*, Proc. ICCV98, Bombay, IEEE Computer Society Press: Piscataway, NJ., pp. 818-823, 1998
21. John S. Zelek, *Complete real-time path planning during sensor-based discovery*, IEEE RSJ int. conf. on intelligent robots and systems, 1998.