Paper: jc9-6-2402 2005/7/29

Evolutionary Optimisation for Obstacle Detection and Avoidance in Mobile Robotics

Olivier Pauplin, Jean Louchet, Evelyne Lutton, and Arnaud de La Fortelle

INRIA, IMARA and COMPLEX Teams
BP 105, 78153 Le Chesnay Cedex, France
E-mail: {olivier.pauplin, jean.louchet, evelyne.lutton, arnaud.de_la_fortelle}@inria.fr
[Received February 9, 2005; accepted June 17, 2005]

This paper presents an artificial evolution-based method for stereo image analysis and its application to real-time obstacle detection and avoidance for a mobile robot. It uses the Parisian approach, which consists here in splitting the representation of the robot's environment into a large number of simple primitives, the "flies", which are evolved according to a biologically inspired scheme. Results obtained on real scene with different fitness functions are presented and discussed, and an exploitation for obstacle avoidance in mobile robotics is proposed.

Keywords: evolutionary algorithm, stereovision, vision systems for robotics, obstacle detection

1. Introduction

Artificial Vision, an important element in the design of autonomous robots, can be approached as the resolution of the inverse problem of reconstructing a probable model of the scene from the images. Although probabilistic optimisation methods like Evolutionary Algorithms [1–3] are in theory well adapted to the resolution of such inverse problems, their use in real applications has been relatively neglected because of their reputation of low speed and complexity. Indeed, evolving a population in which each single individual would be a complete 3-D representation of the environment should raise problems of code size and memory handling wildly out of the reach of current optimisation algorithms.

However, the technique of Parisian Evolution, introduced by Collet et al. [4] to solve the inverse problem for Iterated Function Systems, showed that in some cases, splitting the representation of the object to be optimised into a collection of smaller primitives and evolve them, then use them as a collective representation of the problem's optimal solution, may lead to fast and efficient optimisation algorithms. The Fly Algorithm [5, 6] has been developed along this line to solve Computer Vision problems, using a small grain decomposition of the scene representation and evolving its components following principles inspired from Darwin's principles of biological evolution.

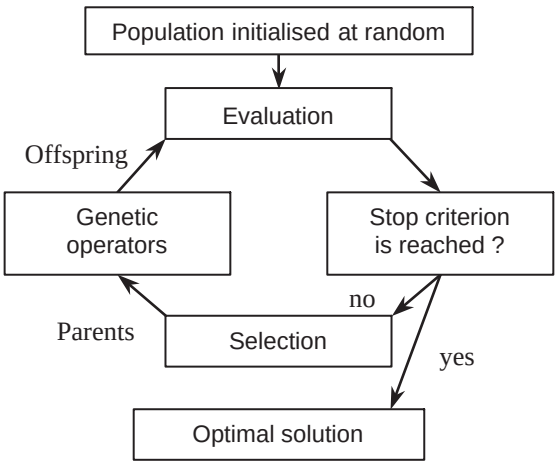

Fig. 1. General layout of genetic algorithms.

2. Evolutionary Algorithms

Darwin's theory assumes that a population of individuals, characterised by their genes, will evolve towards a better adaptation to its environment according to laws of natural selection. Genes mutations may occur and maintain diversity in the population.

Evolutionary algorithms manipulate individuals evaluated by a function, called fitness function, in a way similar to biological Evolution. The general diagram of such algorithms is presented in **Fig.1**, where:

- the population is a group of individuals,
- an individual is defined by his genes $X = (x_1, x_2, \ldots, x_n)$, usually coordinates in the search space,
- evaluation is the calculation of each individual's fitness value,
- selection eliminates part of the population, keeping preferably the best individuals,
- evolution applies genetic operators (crossover, mutations...), leading to new individuals in the population.

3. The Fly Algorithm

The Fly algorithm is a special case of Parisian evolution for which individuals (the "flies") are defined as 3-D

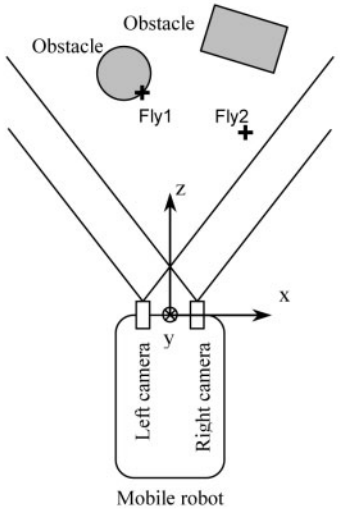

Fig. 2. Example of device using the Fly algorithm, showing two flies from the population (top view).

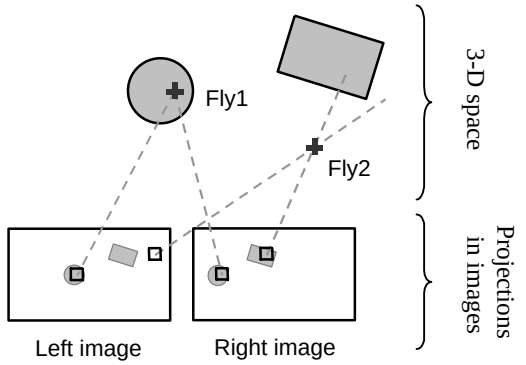

Fig. 3. Projections of two flies in left and right images.

points with coordinates (x,y,z). As far as we know, it is the only existing evolutionary algorithm used to detect obstacles by stereovision. The aim of the algorithm is to drive the whole population – or a significant part of it – into suitable areas of the search space, corresponding to the surfaces of visible objects in the scene.

The population of flies is initialised at random inside the intersection of two cameras' field of view. Flies then evolve following the steps of evolutionary algorithms. All cameras' calibration parameters are known.

3.1. Evaluation

The fitness function used to evaluate a fly compares its projections on the left and right images given by the cameras. If the fly is on an object's surface, the projections will have similar neighbourhoods on both images and hence this fly will be attributed a high fitness.

Figures 2 and **3** illustrate that principle. **Fig.3** shows neighbourhoods of two flies on left and right images. On that example, Fly1, being on an object's surface, will be given a better fitness than Fly2.

The general mathematical expression of the fitness function is [7, 8]:

$$F = \frac{|\nabla(M_L)| \cdot |\nabla(M_R)|}{\sum_{colors} \sum_{(i,j)\in N} \left[L_{(x_L+i,y_L+j)} - R_{(x_R+i,y_R+j)}\right]^2} \quad . \quad . \quad (1)$$

where:

- (x_L, y_L) and (x_R, y_R) are the coordinates of the left and right projections of the current individual
- $L_{(x_L+i,y_L+j)}$ is the grey value at the left image at pixel (x_L+i, y_L+j), similarly with R for the right image
- $(i,j) \in N$, N being a neighbourhood around the projection of each fly, introduced to obtain a more discriminating comparison of the flies
- $|\nabla(M_L)|$ and $|\nabla(M_R)|$ are Sobel gradient norms on left and right projections of the fly. That is intended to penalise flies which project onto uniform regions, i.e. less significant flies[1].

3.2. Selection

Selection is elitist and deterministic. It ranks flies according to their fitness values and retains the best individuals (around 40%).

A sharing operator [7, 8] reduces the fitness of flies packed together and forces them to explore other areas of the search space.

3.3. Genetic Operators

The following operators are applied to selected individuals.

- Barycentric cross-over: given two parents F_1 and F_2, the algorithm builds their offspring F such as:
$\vec{OF} = \lambda \vec{OF_1} + (1-\lambda)\vec{OF_2}$
with λ chosen at random in the interval $[0,1]$.
- Gaussian mutation adds a Gaussian noise to each one of the three coordinates of the mutated fly. The mutation rate is set to 40%, parisian algorithms normally using a higher mutation rate than conventional evolutionary algorithms.
- Another operator, "immigration", is used to improve exploration of the search space, creating new individuals at random. It ensures a constant exploration of the search space, whose high-fitness regions evolve as the scene in front of the cameras changes.

4. Improving the Algorithm

A large number of internal parameters can affect the behaviour of the Fly Algorithm. An inappropriate set of parameters can lead to a high convergence time and a low

1. A fly which project onto repetitional patterns may have a high fitness, whereas it may not be on the surface of an object of the scene.

Fig. 4. Front view of flies evaluated with a fitness function using Sobel gradient norms and a 5×5 correlation window.

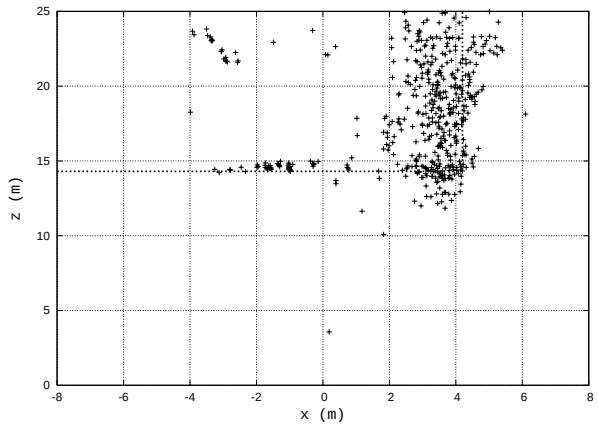

Fig. 5. Top view of flies evaluated with a fitness function using Sobel gradient norms and a 5×5 correlation window. The dotted lines represent the building.

Fig. 6. Front view of flies evaluated with a fitness function using x component of Sobel gradient norms and a 5×5 correlation window.

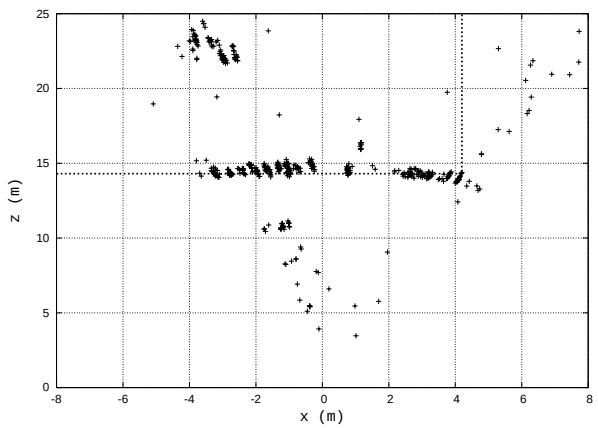

Fig. 7. Top view of flies evaluated with a fitness function using x component of Sobel gradient norms and a 5×5 correlation window. The dotted lines represent the building.

precision in detection. In particular, the choice of the fitness function plays a crucial role in the detection of obstacles.

In order to improve the algorithm efficiency, we carried out tests using several fitness functions. Those tests were performed on a pair of stereo images showing a building, whose distance to the cameras is known. The three coordinates of each fly being known, the population of flies gives a rough description of the real 3-D scene. In the following discussion, "F" designates the fitness function, experiments are carried out with a population of 5000 flies and 200 iterations.

Using Sobel gradient norms in the numerator of the fitness function, as shown in the general expression of F in Eq.(1), leads to the precision of the detection being entirely based on the correlation between neighbourhoods around left and right projections of a fly (denominator of F). That is insufficient in some situations, typically when the pictures contain lines separating two uniform regions and parallel to the line joining the two cameras (horizontal in our case). For instance, **Figs.4** and **5** show the population of flies resulting of the algorithm using a fitness function whose numerator is $|\nabla(M_L)|\cdot|\nabla(M_R)|$. The correlation window size used in the denominator of F is 5×5. The top view shows that many flies are not precisely located on the visible part of the building.

The next experiment is carried out using only the x component of Sobel gradient vectors in F. The numerator of F is: $\nabla_x(M_L)\cdot\nabla_x(M_R)$. Only the vertical features of the scene are detected (**Fig.6**), but the precision is greatly increased compared to the former experiment, as shown by **Fig.7**.

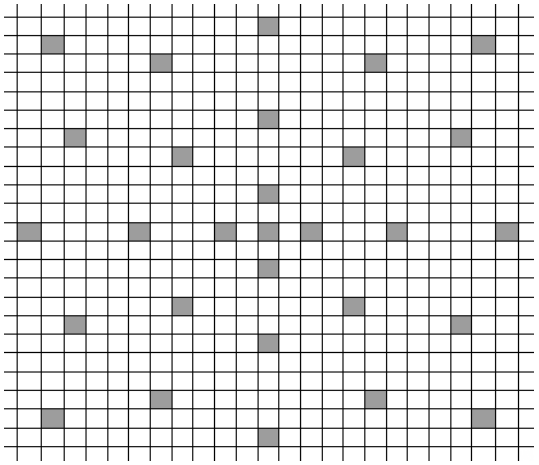

Fig. 8. Sampling of the 23×23 correlation window (29 pixels).

Fig. 9. Front view of flies evaluated with a fitness function using *x* component of Sobel gradient norms and a 23×23 correlation window.

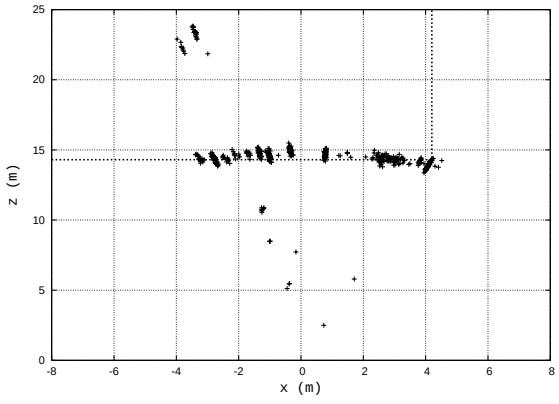

Fig. 10. Top view of flies evaluated with a fitness function using *x* component of Sobel gradient norms and a 23×23 correlation window. The dotted lines represent the building.

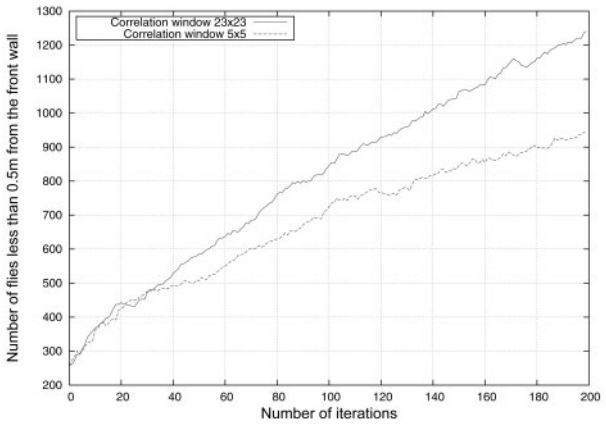

Fig. 11. Precision analysis of the flies obtained using different correlation window sizes (total population size: 5000 flies).

We notice some flies are still off the front wall of the building. Indeed, two different parts of the scene can have locally similar projections in left and right images. That problem can be reduced by using a larger correlation window to evaluate flies. We tested a 23×23 correlation window, in which we chose 29 pixels distributed as shown on **Fig.8** to calculate the correlation of left and right projections. We have also performed an experiment using all the pixels of this 23×23 correlation window, but it consistently doubled the calculation time compared to the 5×5 correlation window (25 pixels), for no obvious improvement of the precision compared to the 23×23 window.

Results obtained with the sampled 23×23 window are presented on **Figs.9** and **10**. Less flies appear to be off the building than in the former example. The capacity of the algorithm to precisely localise the building was measured by counting the number of flies less than half a metre from the front wall. **Fig.11** shows the evolution of the number of flies located on the front wall of the building ±0.5metres, and confirms that a 23×23 correlation window gives better results than a 5×5 one.

In the following obstacle detection experiments, we have chosen to use the parameters which gave the best results in the previous tests.

5. Application to Robot Obstacle Detection

The original way the scene is described by the population of flies led our team to adapt classical robot navigation methods in order to use the results of the Fly algorithm as input data. Boumaza [7, 9] developed a simulator of a robot moving in a simplified environment, to test several control methods using the output of the Fly algorithm.

The simulator showed the possibility to build guidance methods based on the output of the Fly algorithm. Our current work consists in transferring and extending these control methods to real life situations (vehicle guidance).

Fig. 12. A road with no immediate obstacle.

Fig. 13. A pedestrian at 4 metres from the cameras, on the middle of the road.

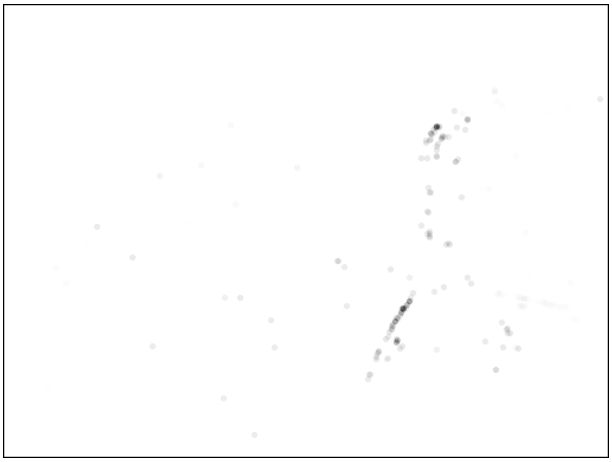

Fig. 14. Warning values of figure 13 flies.

5.1. Control

In the scope of using the Fly algorithm in the field of automatic driving – or at least assisted driving, we developed a strategy to make the program quantify the probability that an obstacle is in front of the vehicle. The aim is to deliver a slow down or stop order when an obstacle appears close enough in the field of vision, in order to avoid frontal collision.

The general idea to achieve this goal is to see each fly as the source of a "warning value", higher when:

- the fly is near the vehicle
- the fly is in front of the vehicle (i.e. close to the z axis)
- the fly has a good fitness.

Beforehand, flies useless for this specific application have there fitness value penalised, and thus have high probability to be eliminated by the evolutionary mechanisms. We considered such non desired flies are:

- flies more than 2 metres above the road surface
- flies with a height under 10 centimetres (detecting the ground)
- flies more than 16 metres ahead of the vehicle.

An experimental analysis led us to choose the simple following formula for the warning value of a fly:

$$warning(fly) = \frac{F}{x^2 \cdot z} \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (2)$$

where F is the fitness value of the fly, and z and x its coordinates as shown on **Fig.2**.

For $|x| < 0.5$m we consider $x = 0.5$m, and for $z < 1$m we consider $z = 1$m. This is to avoid giving excessive warning values to flies with a not necessarily good fitness but with a very small x or z coordinate. Moreover, obstacles within a range of half a metre to the left or to the right from the centre of the vehicle ($|x| < 0.5$m) are equally dangerous, and are consequently processed the same way.

The warning function was built in order to give high warning values to flies for which the three coefficients F, $1/x^2$ and $1/z$ are simultaneously high. Indeed a fly with a low fitness value (thus probably not on an obstacle), far from the vehicle or not in front of it, does not show evidence of an imminent collision. Experiments with a $1/x$ factor instead of $1/x^2$ did not give satisfactory results, as it tended to overestimate the importance of flies off the cameras axis.

5.2. Results

To validate the algorithm, we tested it on three stereo pairs of images: one representing a road with no immediate obstacle (**Fig.12**), one representing a pedestrian crossing the street in front of the vehicle (**Figs.13** and **14**), and one representing a tree in front of the vehicle (**Figs.15** and **16**). **Fig.12** does not show a case of emergency breaking, whereas **Fig.13** and **Fig.15** show situations closer to a collision.

Results are obtained using two commercial CCD cameras and a computer (Pentium 2GHz). The population of flies is 5000. Evaluating all the population (one generation) takes about 10 milliseconds. Population update and calculation of the warning values are done in a quasi-continuous way, and the system needs about 10 to 30 generations to react to a new event in the scene.

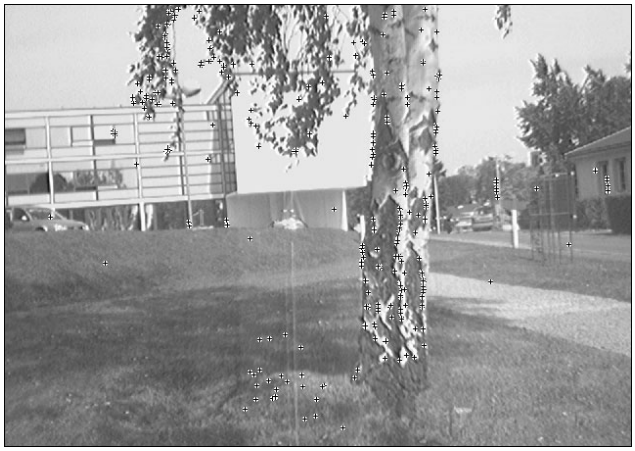

Fig. 15. A tree at 3.5 metres.

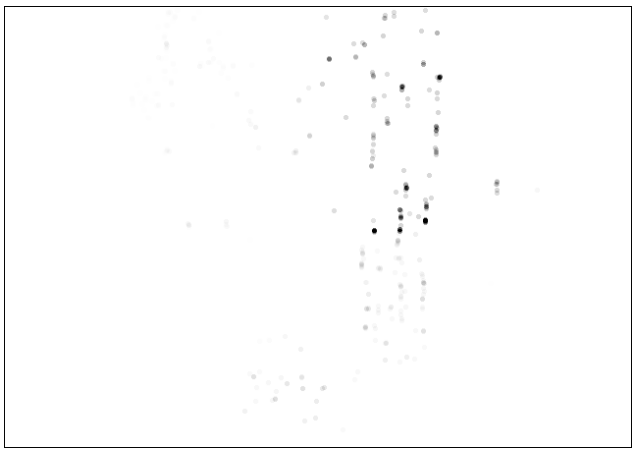

Fig. 16. Warning values of figure 15 flies.

Figures 12, **13** and **15** show the 250 best flies of the resulting population. Flies appear as black crosses. We note that flies gather on the visible objects of the scene (car, pedestrian, road sides, tree).

Figures 14 and **16** show the same (x, y) view as **Figs.13** and **15**, with only flies represented. Flies appear as spots as dark as their warning value is high. The warning values corresponding to **Fig.12** are close to zero and would result in almost a blank figure if shown in the same way. We note the algorithm detected the pedestrian and the tree as near obstacles.

A global warning value can be defined as the mean of the warning values of a population. In the first case (car scene), this mean is 0.09, whereas it is 0.85 in the second case (pedestrian) and 1.08 in the third case (tree). The high difference between these values suggests that they can be used to discriminate between the first situation in one hand, and the second and third in another hand. Further experiments will be performed to validate this procedure on a real moving car.

6. Conclusion

The Fly algorithm has proved a valid method for obstacle detection in outdoor environments. The simplicity of the fitness function used opens the way to real time applications. Real time vehicle control based on the information of flies (coordinates, fitness value) has been developed.

Classical image segmentation and stereo reconstruction methods are potentially able to give more complete and accurate results than the Fly algorithm, though requiring higher processing times. However, the Fly algorithm presents some features which are outstandingly interesting in real time vision applications: in particular its asynchronous properties and its principle of continuous refinement of previous results, giving reaction times to new events intrinsically faster than classical methods [8].

We are currently integrating the Fly algorithm into a vehicle of the IMARA project and test the validity of our hypothesis in real life situations. Integration can take place in both normal or automated cars, since both need help for features detection. This is obvious for automated cars, that need obstacle detection, but car manufacturers are very active to implement obstacle avoidance assistance in cars, in order to reduce fatalities, which is a major goal of many governments.

With respect to the theoretical problem, there are many critical points to overcome, but we believe this very innovative approach has a real application domain in Intelligent Transportation Systems (ITS).

The first experimental constraint is to implement a real-time version of the algorithm while keeping a sufficient output quality. If this is not technically feasible with our present hardware (i.e. a single laptop), we consider distributed processing.

The ability to easily distribute this algorithm over several processors or implement it into specific hardware is one interesting feature of the Fly Algorithm.

Another aspect is its ability to be used in conjunction with cheap mass-produced cameras and coarse calibration. Compared e.g. to active obstacle detectors based on laser rangefinder technology, stereovision would provide a cheap, efficient alternative if it was not usually requiring accurate camera calibration, which is not compatible with usual constraints on automotive design and maintenance. The Fly Algorithm is tolerant to coarse camera calibration, and its data fusion capabilities should enable easier interfacing with other embedded systems and offer better safety and reliability. This is to be test-validated in the near future.

Acknowledgements

We thank Dr Amine Boumaza for his important contribution to the development of the code used in our experiments.

This research was funded in part by the IST Programme of the European Commission in the CyberCars project: http://www.cybercars.org/

References:

[1] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, Reading, MA, 1989.

[2] I. Rechenberg, "Evolution strategy," J. M. Zurada, R. J. Marks II, C. J. Robinson, (Eds.), Computationnal Intelligence Imitating Life, IEEE Press, Piscataway, NJ, pp. 147-159, 1994.

[3] J.-P. Rennard, "Vie artificielle," Vuibert, ISBN:2-7117-8694-3, pp. 241-242, 2002.

[4] P. Collet, E. Lutton, F. Raynal, and M. Schoenauer, "Individual GP: an alternative viewpoint for the resolution of complex problems," Genetic and Evolutionary Computation Conference GECCO99, Morgan Kauffmann, San Francisco, CA, 1999.

[5] J. Louchet, "From Hough to Darwin: an individual evolutionary strategy applied to artificial vision," Artificial Evolution, European Conference, AE 99, Dunkerque, France, Selected papers, Springer Verlag, Lecture Notes in Computer Science, p. 1829, 1999.

[6] J. Louchet, "Stereo analysis using individual evolution strategy," Internat. Conf. on Pattern Recognition, ICPR2000, Barcelona, Spain, 2000.

[7] A. Boumaza, and J. Louchet, "Dynamic Flies: Using Real-Time Parisian Evolution in Robotics," EVOIASP 2001, Lake Como, Italy, 2001.

[8] J. Louchet, M. Guyon, M.-J. Lesot, and A. Boumaza, "Dynamic Fies: a new pattern recognition tool applied to stereo sequence processing," Pattern Recognition Letters, No.23, pp. 335-345, 2002.

[9] A. Boumaza, "Introduction de techniques d'évolution artificielle en vision tridimensionnelle et en robotique mobile," Thèse Université René Descartes, Paris, 2004.

[10] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," Springer Verlag, 1992.

Name:
Olivier Pauplin

Affiliation:
INRIA - IMARA team

Address:
B.P. 105, 78153 Le Chesnay Cedex, France

Brief Biographical History:
2003- Started its doctorate at INRIA, France

Main Works:
• "Obstacle detection by Evolutionary Algorithm: the Fly Algorithm," Proceedings of the Second International Conference on Autonomous Robots and Agents, pp. 136-140, Dec., 2004.

Name:
Jean Louchet

Affiliation:
INRIA - COMPLEX team

Address:
B.P. 105, 78153 Le Chesnay Cedex, France

Brief Biographical History:
1977- Joined French Ministry of Defence
1988- Professor, Ecole Nationale Superieure de Techniques Avancees, France
1991- Honorary Research Fellow, University of Exeter, UK.
2002- Visiting scientist, INRIA, France

Main Works:
• J. Louchet, "Using an Individual Evolution Strategy for Stereovision," Genetic Programming and Evolvable Machines, Vol.2, No.2, Kluwer Academic Publishers, pp. 101-109, March, 2001.
• J. Louchet, M. Guyon, M.-J. Lesot, and A. Boumaza, "Dynamic Flies: a new pattern recognition tool applied to stereo sequence processing," Pattern Recognition Letters, Elsevier Science B.V., March, 2001, revised June, 2001.
• J. Louchet, "An evolutionary algorithm for physical motion analysis," British Machine Vision Conference, York, UK, BMVA Press, pp. 701-710, 1994.

Membership in Learned Societies:
• Deputy secretary of AFRIF (French Association for Pattern Recognition)
• Member of the Galpin Society

Name:
Evelyne Lutton

Affiliation:
INRIA - COMPLEX team

Address:
B.P. 105, 78153 Le Chesnay Cedex, France

Brief Biographical History:
1990- Researcher at INRIA, France
2000- Head of the FRACTALES/COMPLEX team

Main Works:
- "Polar IFS + Parisian Genetic Programming = Efficient IFS Inverse Problem Solving", Genetic Programming and Evolvable Machines Journal, Vol.1, Issue 4, pp. 339-361, October, 2000.
- "Genetic Algorithms and Fractals," chapter in Evolutionary Algorithms in Engineering and Computer Science, 1999.
- "Fractals in engineering: New trends in Theory and Applications," Springer Verlag, 2005.

Membership in Learned Societies:
- IEEE member
- French association EA (Artificial Evolution)
- EuroGP steering committee

Name:
Arnaud de La Fortelle

Affiliation:
INRIA - IMARA team

Address:
B.P. 105, 78153 Le Chesnay Cedex, France

Brief Biographical History:
1997- Joined INRIA, France

Main Works:
- "Contribution to the theory of great deviations and applications," PhD thesis, Nov., 2000.
- "Large deviations problems for star networks: the min policy," The Annals of Applied Probability, Vol.14, No.2, pp. 1006-1028, 2004.