View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

154
Countries delivered to

Our authors are among the
TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

21

Application of Artificial Evolution to Obstacle Detection and Mobile Robot Control

Olivier Pauplin and Arnaud de La Fortelle
National Institute for Research in Computer Science and Control (INRIA), IMARA Team
France

1. Introduction

The Fly Algorithm is an evolutionary algorithm used for stereoscopic reconstruction. In the classical approach, a pair of stereo images is processed in order to extract 3-D information and to infer a representation of the scene. Conversely, the Fly Algorithm builds potential 3-D models of the scene and tests their consistency with the two stereo images. This chapter will present some recent improvements of the algorithm, and its concrete application to obstacle detection and avoidance in mobile robotics.

Section 2 presents the notion of individual approach in evolutionary algorithms and the principle of the Fly Algorithm. New genetic operators are introduced. Several internal parameters can drastically change the algorithm behaviour: this issue is the topic of section 3, where a Pareto multi-objective optimisation leads to the obtention of an efficient set of parameters.

Section 4 describes a real time application to automatic driving on an electrical vehicle of the IMARA Team (INRIA). We focus on stop/go control and on direction control in order to avoid obstacles in front of the vehicle. The methods used are explained and results are shown.

Finally, section 5 concludes the chapter and gives some ideas of future work to further improve the algorithm.

2. The Fly Algorithm

2.1 Presentation

Evolutionary algorithms (Holland, 1975; Goldberg, 1989; Rechenberg, 1994), inspired by the evolution of species by natural selection, reproduction and mutation, are now common and widely used optimisation methods. The usual goal of these optimisation methods is to search the extremum of a function – called fitness function, or objective function – that is to say the best individual of the population after a certain number of iterations. The so called individual approach (or Parisian approach) (Collet et al., 1999) considers the solution to a problem to be given not only by the best individual, but by the whole population, or a significant part of the population. That is made possible by an appropriate formulation of the problem, splitting the representation of the object to be optimised into smaller primitives evolved separetly.

Open Access Database www.intehweb.com

Source: Frontiers in Evolutionary Robotics, Book edited by: Hitoshi Iba, ISBN 978-3-902613-19-6, pp. 596, April 2008, I-Tech Education and Publishing, Vienna, Austria

The Fly Algorithm (Louchet, 2000; Boumaza & Louchet, 2003; Pauplin et al., 2005) is an evolutionary algorithm based on the individual approach, and used in the domain of computer vision (Jähne, 1999). The aim of the algorithm is to drive the population of individuals, defined as 3-D points (the "flies") in front of a pair of cameras, into suitable areas of the search space, corresponding to the surfaces of objects present in the scene. The search space where the flies evolve is the intersection of the two cameras' field of view, as shown on figure 1.

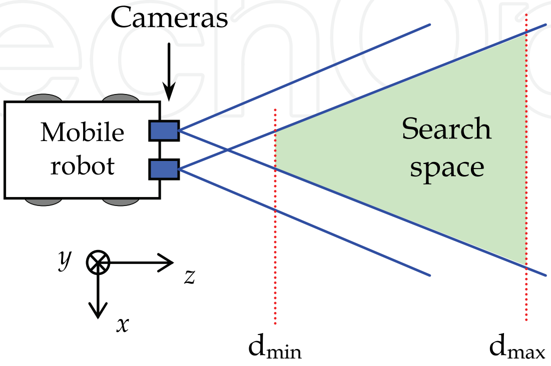


Figure 1. Example of device using the Fly Algorithm (top view)

The population of flies is initialised at random in the search space, and then evolves following the steps of an evolutionary algorithm.

2.2 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. The mathematical expression of the fitness function is:

$$F(fly) = \frac{|\nabla x(M_L)| . |\nabla x(M_R)|}{\sum_{colours} \sum_{(i,j) \in N} [L(x_L + i, y_L + j) - R(x_R + i, y_R + j)]^2} \qquad (1)$$

where:

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

More details about the fitness function used can be found in (Pauplin et al., 2005; Pauplin, 2007).

2.3 Sharing and selection

If nothing is done to prevent it, flies tend to gather around a unique point of the search space. Such a population does not give a 3-D description of the scene. In order to force flies to explore the whole search space, a sharing (Boumaza, 2001) reduces the fitness of flies packed together.

The sharing is applied to the projections of the flies in the left image. A grid divides the left image into squares (figure 2), and the new fitness is given by:

$$F_{sharing}(fly) = F(fly) - n \times C_{sharing} \tag{2}$$

where n is the number of flies projecting into the same square as the considered fly, and $C_{sharing}$ (sharing coefficient) is a coefficient whose value can be tuned experimentally. A high sharing coefficient prevents areas in the search space to be unexplored, but also results in a lower average fitness.

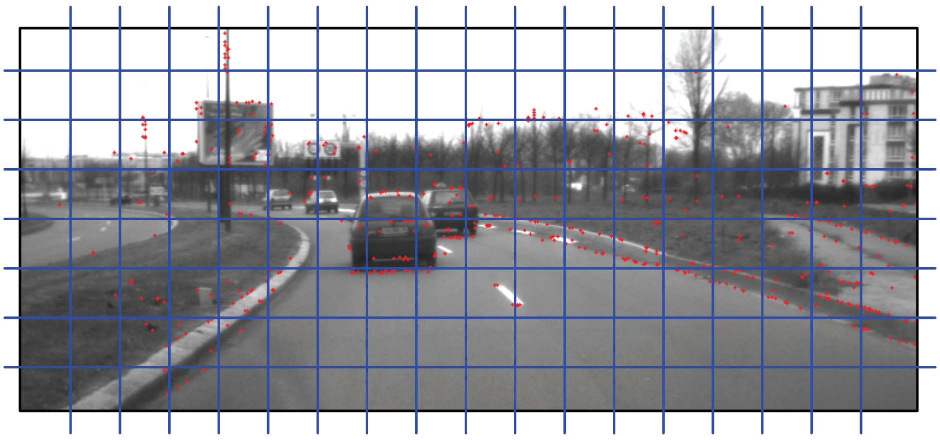


Figure 2. The grid used to apply the sharing. Flies appear as red dots

The selection is based on the values of $F_{sharing}$. The population is ranked according to $F_{sharing}$ and the best half is kept and forms the population of parents ($S/2$ individuals, S being the size of the population). $S/2$ children must be created.

2.4 Genetic operators

The population of offspring is created by applying one of five genetic operators $S/2$ times – each operator creates one child at a time. For each child, which genetic operator will be used is determined according to probabilities (p_i, $i = 1 \ldots 5$) assigned to each operator.

The five genetic operators are described hereafter.

- Initialisation or "immigration". An individual is created at random. That ensures a constant exploration of the search space.
- Mutation. A new individual is created by adding random perturbations to a parent's coordinates. The random perturbations belong to the interval $[-i_m, i_m]$.
- Rank-scaled mutation. An individual is picked at random among the parents. The interval of mutation is modified as follows, in function of the rank r of the parent:

$$i_m \to i_m' = i_m \times \left(\frac{2 \cdot r}{S} + 0.5 \right) \tag{3}$$

The mutation operator is then applied with i_m' instead of i_m. That multiplies by two the interval of mutation for the parent with the lowest fitness, and divides it by two for the best parent. It can be indeed interesting to create new individuals close to the better parents. Conversely, a parent with a low rank is probably not on an object's surface, and it is probably not efficient to create an individual in this area.

- Elitist mutation. Two new individuals are created by mutation. Only the best of the two children is kept. Unlike the usual mutation, this elitist mutation is not a simple exploration of the search space but effectively tends to increase the average fitness of the population. However, it requires to evaluate two individuals, which may be a drawback for a real-time application.
- Crossover. The crossover can be useful to detect flat surfaces or oblong objects. Three individuals are picked at random in the best half of the parents, and the two closest of these three individuals are combined to produce one child. That aims to rarefy crossovers between far away parents, which are probably not on the same object, or between parents which are not on an object at all. The child is obtained as a classical linear combination of the coordinates of the two parents:

$$fly_1\begin{pmatrix}x_1\\y_1\\z_1\end{pmatrix}\times fly_2\begin{pmatrix}x_2\\y_2\\z_2\end{pmatrix}\rightarrow fly_3\begin{pmatrix}\lambda\cdot x_1+(1-\lambda)\cdot x_2\\\lambda\cdot y_1+(1-\lambda)\cdot y_2\\\lambda\cdot z_1+(1-\lambda)\cdot z_2\end{pmatrix} \tag{4}$$

where λ is a random number belonging to [-0.1 , 1.1]. This interval is a bit larger than usual [0 , 1] weighting so as to avoid contraction: the potential underlying object can be larger than the segment linking the two parents.

3. Parameters adjustment

As in most evolutionary algorithms, the convergence and robustness of the algorithm depend on numerous internal parameters. Inappropriate values of parameters may for instance lead to a very slow convergence or to a premature convergence to a local optimum. In this section we will try to find a set of parameters allowing the fastest and most efficient convergence of the algorithm, in order to enable its use in real time obstacle detection and robot control.

The Fly Algorithm pursues two antagonist objectives:

- to increase the average fitness of the population
- to exlore the whole search space.

Genetic operators tend to realise the first objective, whereas the sharing tends to realise the second one. However, the number of parameters makes it uncertain, if not impossible, to tune the parameters manually. A solution consists in batch-testing a great amount of sets of parameters and evaluating the results according to given objectives.

3.1 Objectives and parameters

The two objectives above-mentioned can be computed as follows.

- Average fitness of the best third of the population.
- Diversity of the population: we chose to measure the diversity using a grid similar to (but not identical to) the one used in the sharing, on the left image (cf. figure 2). The

number of horizontal divisions of the diversity grid has been set to 30, which enables an accuracy high enough for our applications (the view angle of our cameras is 42.5 degrees, hence the average angle corresponding to one division of the grid is 1.4 degrees). The diversity can be estimated as:

$$div(P) = \frac{\sum_{squares} \min(Nf_{square}, Nf_{average})}{N_{squares}} \tag{5}$$

where:

- P is the population
- Nf_{square} is the number of flies in the considered square
- $N_{squares}$ is the total number of squares
- $Nf_{average} = S / N_{squares}$, S being the population size.

We chose eight parameters:

- Sharing coefficient $C_{sharing}$, varying in [0 , 0.65]
- Number of horizontal divisions of the sharing grid, varying in [5 , 75]
- Interval of mutation, varying in [0 , 2] (metres)
- Probabilities of the five genetic operators, p_i , i = 1 … 5, varying in [0 , 1] with $\Sigma p_i = 1$.

3.2 Pareto front

Generally, the optima of the different objectives are not reached for the same set of parameters. The aim of Pareto's method is to display a collection of optimal compromises between the objectives. These optimal solutions form the Pareto front.

In the case of a maximisation problem with l objectives, an individual X^* belongs to the Pareto front if there is no other solution X which verifies the condition:

$$\forall k \in [1,l], \quad f_k(X) > f_k(X^*) \tag{6}$$

In other words, no other solution is better than X^* for all the objectives.

3.3 Procedure and results

200000 sets of parameters have been picked at random. The algorithm runs 0.25 seconds for one set of parameters, and then computes the two objectives corresponding to the final population; that is repeated nine times, so that the algorithm runs ten times for each set of parameters. The average of the ten values of fitness and diversity obtained gives the values of the two objectives for that set of parameters.

It is important to run the algorithm for a given duration, not a given number of iterations, as the length of iterations depends on the parameters. The size of the population is fixed to a standard value: S = 3000.

The results appear on figure 3: the performances of the 200000 sets of parameters are represented in the space of objectives. The Pareto front (green) is made of 237 sets of parameters, which are the best compromises found for these two objectives.

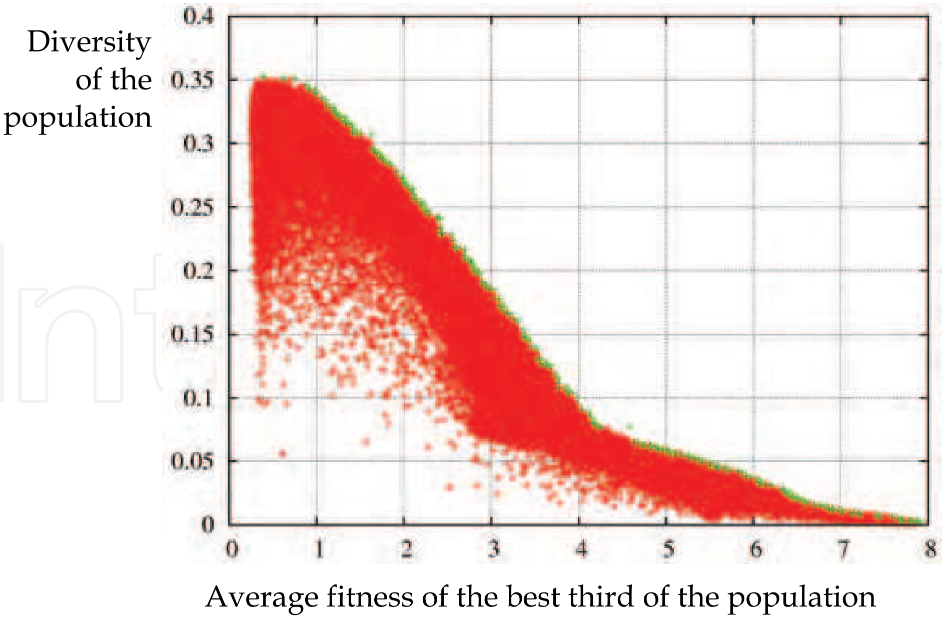


Figure 3. 200000 sets of parameters in the space of objectives. 237 sets of parameters form the Pareto front

It is difficult to justify the choice of one set of parameters belonging to the Pareto front instead of another. In the context of obstacle detection and automatic driving, diversity is mainly a way not to miss a new object entering the search space. Moreover, the density of flies must be significantly different in the areas where an object is present and in the empty areas. Hence, after several tests, it appeared that sets of parameters located on the "high fitness" side of the Pareto front were far better adapted to our applications than sets of parameters on the "high diversity" side of the Pareto front. We have finally chosen the compromise presented in table 1, which is located on the Pareto front with an average fitness around 5.

These parameters have proved experimentally to be much more efficient than other parameters manually tuned. They have been used to realise the experiments presented in section 4.

Sharing coefficient:	0.15
Number of horizontal divisions of the sharing grid:	30
Interval of mutation:	0.35 m
Probabilities to apply genetic operators:	
Initialisation	0.15
Crossover	0.27
Elitist mutation	0.23
Mutation	0.15
Rank-scaled mutation	0.2

Table 1. Chosen set of parameters

4. Application to mobile robotics

4.1 Material

The IMARA team (INRIA, France) owns electrical vehicles – CyCabs – which can be driven manually or automatically. A pair of cameras has been installed in front of a CyCab and connected to an embedded PC on which runs the Fly Algorithm. That PC is connected to CyCab controlers and can send them speed or direction instructions.

The two cameras provide rectified stereo images so that the three axes of each camera are parallel two by two, and their optical centers belong to the same horizontal axis (x). As a result, the computation time to calculate the projections of flies in left and right images is minimal.

The computer used is a standard PC under Windows XP, 2 GHz, 1 GB RAM.

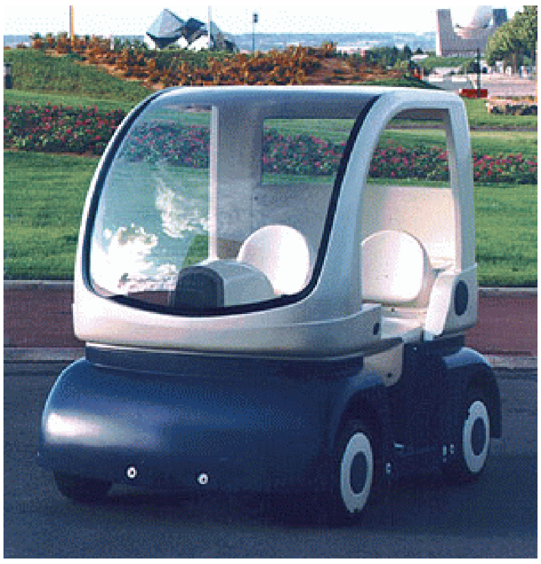


Figure 4. The CyCab

4.2 Collision avoidance

The aim is to deliver a stop order when an obstacle appears close enough in the field of vision, in order to avoid frontal collision. The general idea is to see each fly as the source of a "warning value", higher when:

- the fly is in front of the vehicle ($|x|$ small)
- the fly is near the vehicle (z small)
- the fly has a high fitness value.

The average of the warning values of the best part of the population gives a global indication of the probability that an obstacle is in front of the vehicle and that a collision could happen. A threshold can be experimentally determined, beyond which a stop signal has to be sent.

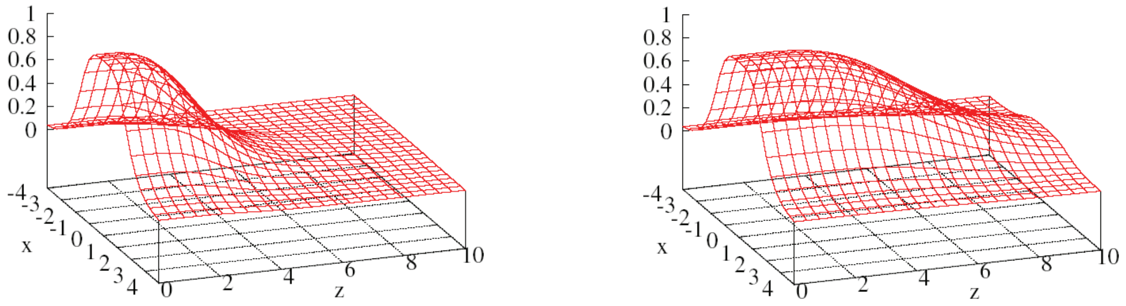
The function used to assign a warning value to each fly is made of three factors corresponding to the three conditions above:

$$w(fly) = w_{space}(x,z) \times F^{\gamma} \tag{7}$$

$$w_{space}(x,z) = \frac{10}{10+x^4} \times \frac{100}{100+\left(\frac{z}{\beta}\right)^4} \tag{8}$$

where x, z and F are the coordinates and the fitness value of the fly.

$w_{space}(x,z)$ gives the weight of the fly in the global warning value according to its position in the 3-D space. It does not have to depend on y as the search space has been vertically limited between the ground and the height of the CyCab. The parameter β allows to set the distance (on z) at which an object is considered as a potential risk (figure 5).



$$w_{space}(x,z) = \frac{10}{10+x^4} \times \frac{100}{100+z^4}$$

$$w_{space}(x,z) = \frac{10}{10+x^4} \times \frac{100}{100+\left(\frac{z}{2}\right)^4}$$

Figure 5. Influence of the parameter β (left: $\beta = 1$, right: $\beta = 2$)

Given the relatively small speed of a CyCab, around 2 m/s, we wish to distinguish between obstacles closer than $z = 5$ m (for $x = 0$) and those beyond $z = 5$ m. That is achieved when the value of β is such that the function

$$w_{space}(z)_{x=0} = \frac{100}{100+\left(\frac{z}{\beta}\right)^4} \tag{9}$$

has a slope which is maximal (in absolute value) when $z = 5$ m. The maximal slope in absolute value of $w_{space}(z)_{x=0}$ is in $z = z_0$ given by

$$\beta = \frac{z_0}{\sqrt{10}} \tag{10}$$

For our application, $z_0 = 5$ m so we can take $\beta \approx 1.58$.

Another parameter from equation 7 is γ, the exponent of the fitness value. A small value for γ would favour flies with a small fitness, which are not very reliable. On the other hand, a high value for γ would favour better flies but could make the global warning value depend on a few individuals with a high fitness, which would considerably increase the noise of the warning, as a small variation in a fitness value would result in a large variation in the warning value.

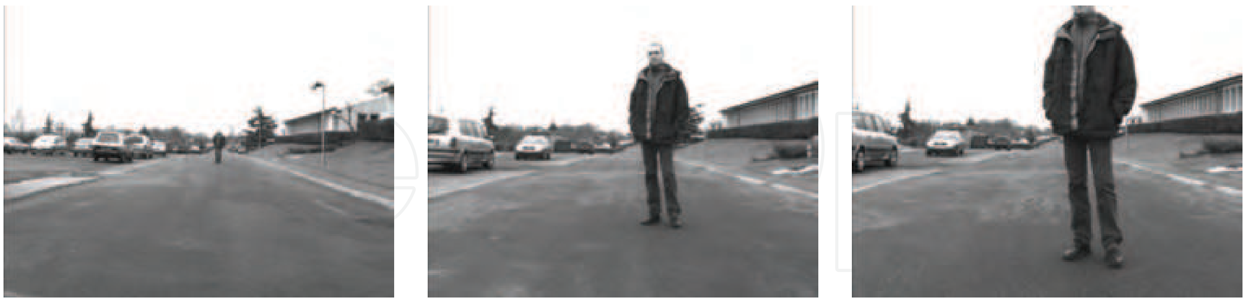

No obstacle Pedestrian at 6 m Pedestrian at 4 m

Figure 6. Scenes used to determine γ

The value of γ has been determined experimentally, considering the ratio between the average warning value when an obstacle is at 4 m and the average warning value when an obstacle is at 6 m, on the one hand, and the ratio between these values when an obstacle is at 4 m and when no immediate obstacle is present. Figure 7 and 8 show these ratios in function of γ. The scenes used for the experiment are shown on figure 6.

The curves on figure 7 and 8 have a different aspect, because when a pedestrian is present at 4 or 6 m the flies gather on him (with high fitness values), whereas when no obstacle is present the flies have a roughly uniform density in the space (with very low fitness values). It is then possible that, when $\gamma = 0$, the average value of $w_{space}(x,z)$ in the population is higher in the case of an empty scene than when an obstacle is present at 6 m. That shows that figure 7 is not enough to determine γ.

The value of γ is a compromise fulfilling the following conditions:

- The average warning value of a scene showing no obstacle must be lower than the one of a scene showing an obstacle at 6 m.
- The average warning value of a scene showing an obstacle at 6 m must be lower than the one of a scene showing an obstacle at 4 m.
- The warning values obtained on the scenes of figure 6 must be significantly different, in particular the "min" value on figure 7 should not be too close to 1.

According to figure 7 and figure 8, γ can be chosen between 1 and 2. That has been confirmed experimentally on the CyCab. We also checked that the value $\gamma = 4$ makes the CyCab stop inopportunely.

Results of individual warning values are shown on figure 9. Figure 10 shows the evolution of the global warning value when the CyCab moves towards a pedestrian; during the first 12.3 seconds, no obstacle is detected (the pedestrian is too far), and the global warning value is close to 0. The decision to stop the CyCab is given by a comparison of the global warning value with a threshold, for example 0.4 or 0.5.

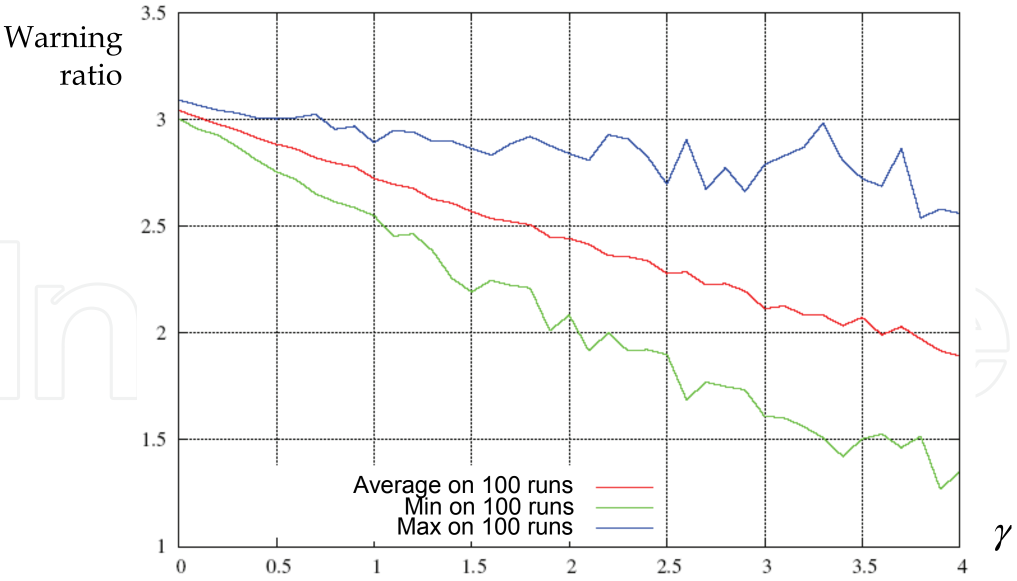


Figure 7. Average warning value obtained with an obstacle at 4 m divided by the average warning value obtained with an obstacle at 6 m

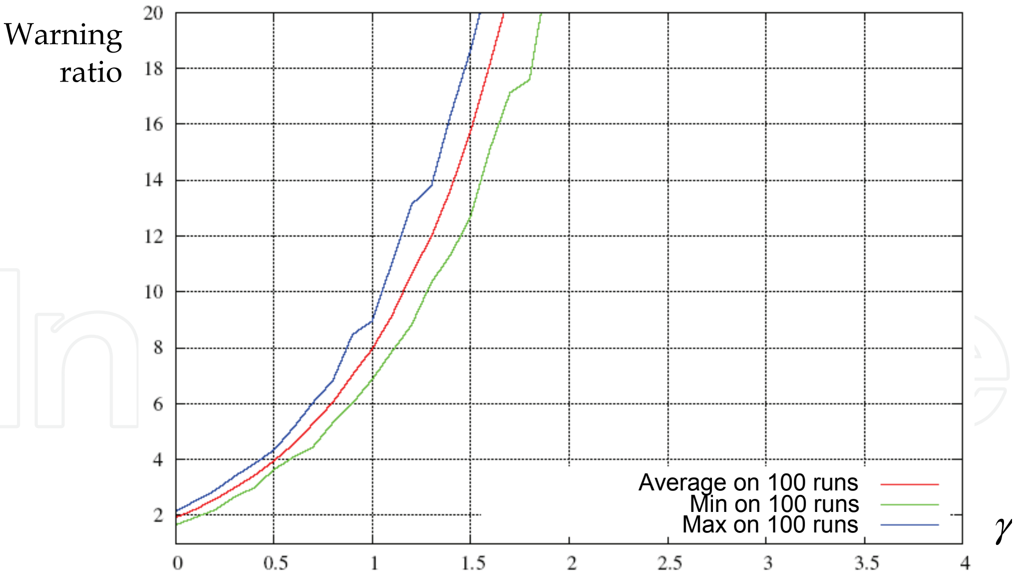


Figure 8. Average warning value obtained with an obstacle at 4 m divided by the average warning value obtained without any obstacle

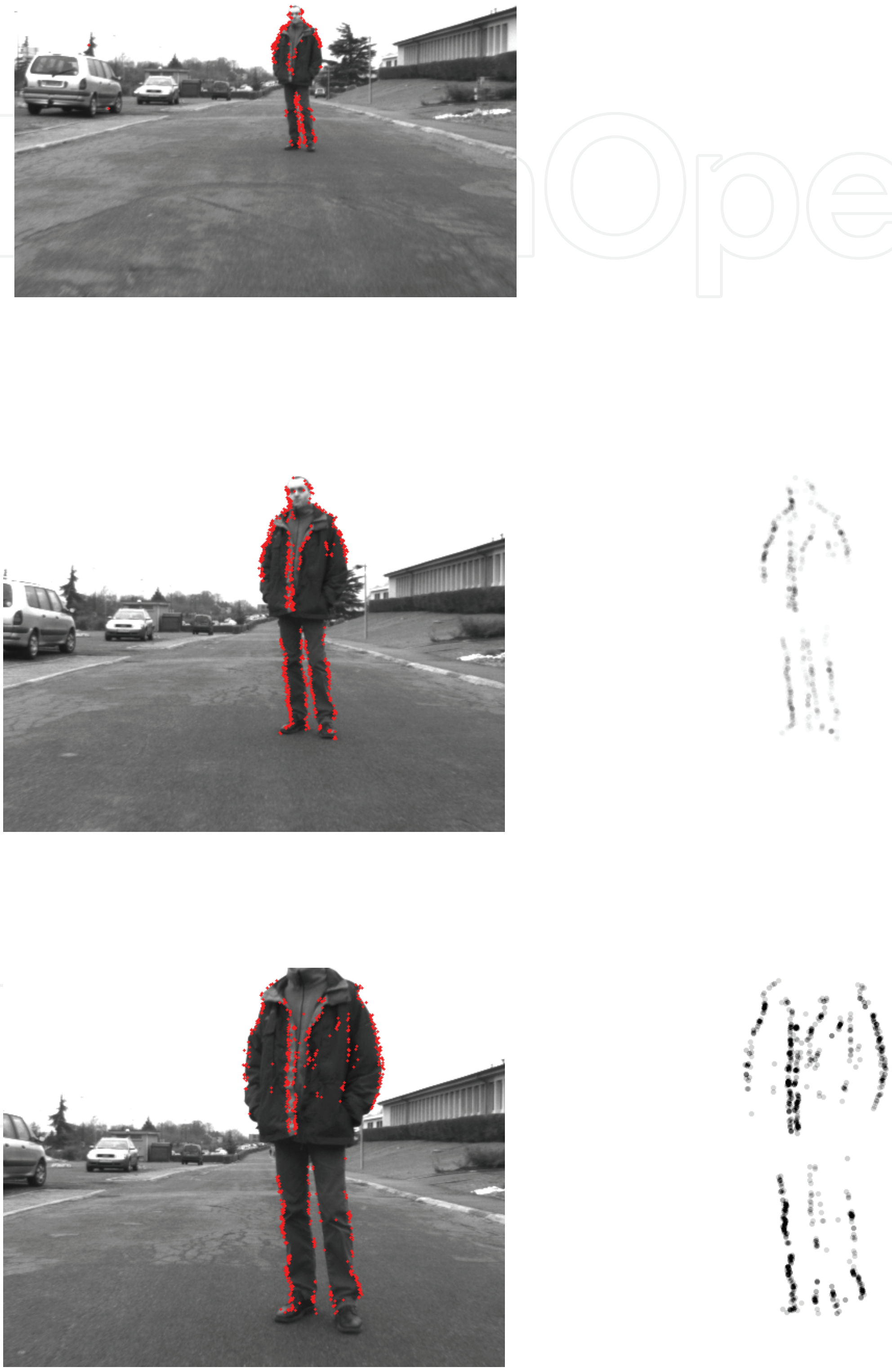


Figure 9. Warning values in different situations. Left column: flies are represented as dots. Right column: flies appear as spots as dark as their warning value is high

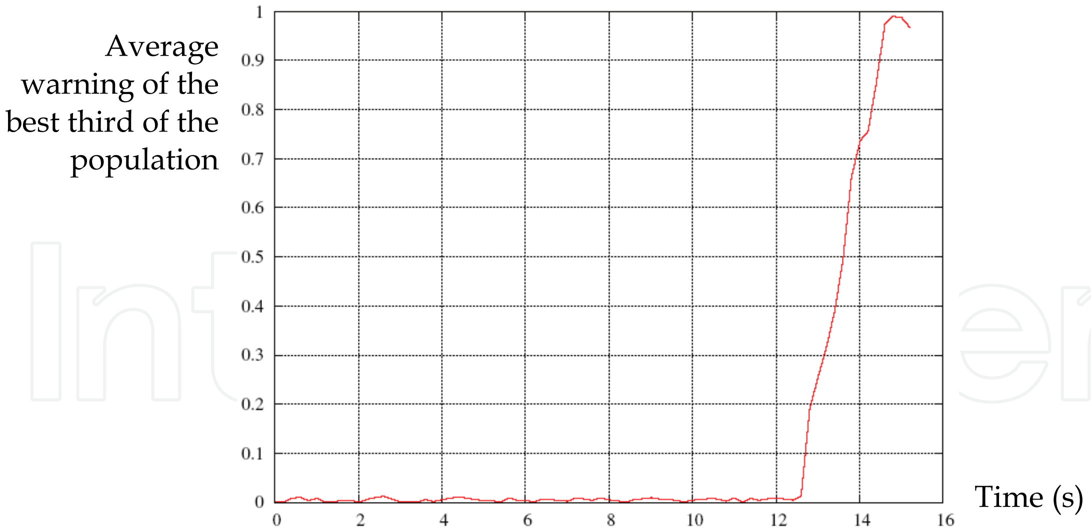


Figure 10. Evolution of the global warning value when the CyCab moves towards a pedestrian

4.3 Direction control

In some cases, objects are present in the scene but there is no immediate risk of collision. We can consider modifying the trajectory of the robot so that it will move forward staying as far as possible of the objects. The CyCab being a non holonom robot, it can not change its orientation without moving forward (or backward). The first signal to be sent to the CyCab controlers, after processing the actual population of flies, is the stop/go signal based on the global warning value (cf. previous section). If that signal is a "go", a direction control is computed and sent to the wheels' controlers.

The method we use is based on the warning values. The projection on the horizontal plan of the search space is divided into $N_{sectors}$ sectors centered on the intersection of the two cameras' field of view, as described by figure 11.

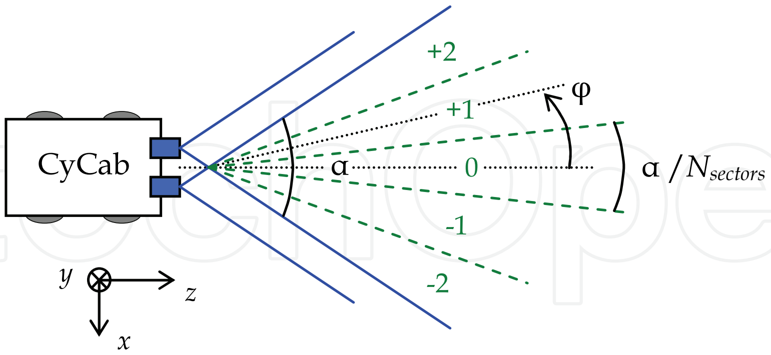


Figure 11. Example of division of the space in five sectors ($N_{sectors}$ = 5)

The average warning value is computed for each sector, which gives a measure of the obstruction in each corresponding direction. Those values are compared to a threshold, and those under the threshold are considered equal to zero. This threshold, relatively low, is necessary for the case where no obstacle would be present. The number $N_{sectors}$ is a compromise: too small a number of sectors makes the representation of the obstruction imprecise, and too high a number makes it less reliable (the number of flies in each sector must be high enough to give a reliable information). We have obtained good experimental results with five or seven sectors. It is appropriate to take an odd number of sectors, so it is possible to go straight ahead.

Let φ be the angle of the direction where to go, $\varphi = 0$ corresponding to the "straight" direction, as shown on figure 11. We tested two strategies.

- Strategy 1: the CyCab goes in the direction corresponding to the sector with the smallest warning value. If several sectors have a warning value equal to zero (which happens frequently due to the threshold), the direction chosen among these sectors is the closest to $\varphi = 0$.
- Strategy 2: the CyCab goes in the direction "opposed" (see below) to the sector corresponding to the highest warning value. If the warning values of each sector are null, the CyCab goes straight. If the highest warning value is in the direction $\varphi = 0$, the CyCab goes on the very left or the very right depending on the warning values in these directions.

  Let Φ be the direction of the highest warning value. The opposed direction is defined by:

$$\varphi_{opposed} = \Phi + \alpha \cdot \frac{N_{sectors} + 1}{2 \cdot N_{sectors}} \qquad \text{if} \quad \Phi < 0 \tag{11}$$

$$\varphi_{opposed} = \Phi - \alpha \cdot \frac{N_{sectors} + 1}{2 \cdot N_{sectors}} \qquad \text{if} \quad \Phi > 0$$

  α being the horizontal view angle. For instance, on figure 11, the direction opposed to "+2" is the direction "-1", the direction opposed to "+1" is the direction "-2" (and vice versa).

Both strategies have been tested on a CyCab. The strategy 2 has proved more satisfying. The strategy 1 gets easily trapped, specially if $N_{sectors}$ is high (7 or more). For example, if the highest warning value is in direction "+1", the CyCab will go straight until it is stopped by the anti-collision signal. The strategy 2 is more active (the CyCab goes straight only if no obstacle is detected), which results in a greater efficiency for obstacle avoidance.

Three steps of the trajectory of a CyCab controlled by strategy 2 are shown on figure 12.

- Step (a): a pedestrian is detected on the right. The CyCab starts to turn left.
- Step (b): the border of the road is now detected on the left. The CyCab starts to turn right.
- Step (c): no more obstacle. The CyCab goes straight.

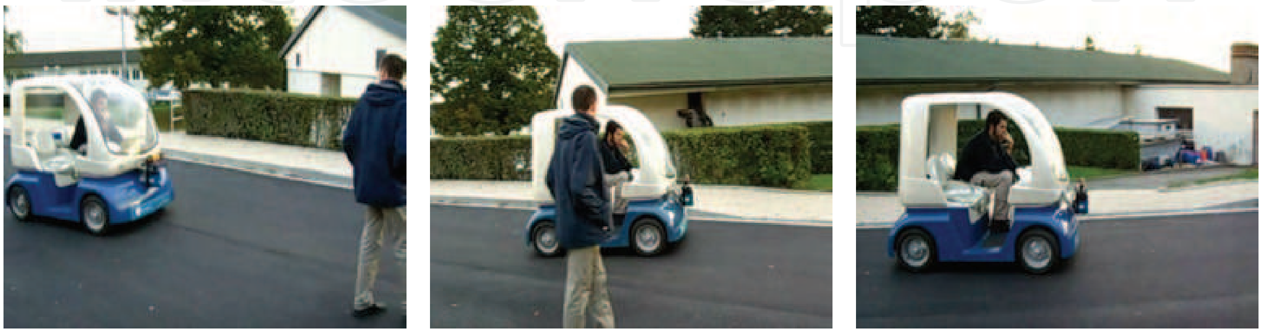

(a) (b) (c)

Figure 12. CyCab controlled by strategy 2

5. Conclusion

The Fly Algorithm embedded in a CyCab is able to detect obstacles, and to compute stop/go and direction controls accordingly, in real time. That is largely due to the optimisation of the efficiency conducted in section 3. It is also due to the fact that we have voluntarily stayed close to the natural output of the algorithm – a cloud of 3-D points – and have used it directly, without any prior processing. The control strategies tested are very simple and may be improved.

Future work includes speeding up the frame processing using CMOS sensors – which may be well adapted to the computation of the fitness of the flies - instead of CCD, and to increase the speed using FPGA in the evaluation part of the evolutionary algorithm. Concerning the algorithmic part, we could consider adapting dynamically the search space according to the application or the conditions (e.g. the speed of the robot). Other ways to enhance the algorithm could be to change the set of parameters during the convergence period (a bit like Simulated Annealing), and to change the paradigm (at the moment: use a lot of very simple features, here 3D-points) and to use more complex features with dynamics adapted to the use. This is then closer to swarm work. But it could also offer a better interaction with more classical obstacle detection/classification: use the Fly Algorithm to detect region of interest within which dedicated algorithm would refine the detection. An open problem is then: can we also use this detection to enhance the Fly Algorithm runs?

6. References

Boumaza, A. & Louchet, J. (2001). Dynamic Flies: Using Real-Time Parisian Evolution in Robotics, *Proceedings of EVOIASP '01*, Lake Como, Italy, April 2001

Boumaza, A. & Louchet, J. (2003). Robot Perception Using Flies. *Proceedings of SETIT'03*, Sousse, Tunisia, March 2003

Collet, P.; Lutton, E.; Raynal, F. & Schoenauer, M. (1999). Individual GP: an Alternative Viewpoint for the Resolution of Complex Problems. *Proceedings of GECCO'99,* Orlando, USA, July 1999, Morgan Kauffmann, San Francisco

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 0201157675, Boston, MA, USA

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 0262581116, Ann Arbor, MI, USA

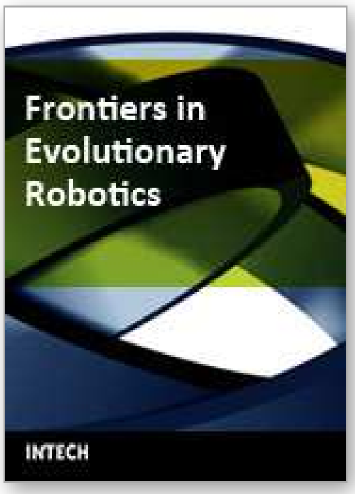
Jähne, B.; Haussecker, H. & Geissler, P. (1999). *Handbook of Computer Vision and Applications*. Academic Press, 0123797705, San Diego, CA, USA

Louchet, J. (2000). Stereo Analysis Using Individual Evolution Strategy. *Proceedings of ICPR'00*, Barcelona, Spain, September 2000

Pauplin, O.; Louchet, J.; Lutton, E. & de La Fortelle, A. (2005). Evolutionary Optimisation for Obstacle Detection and Avoidance in Mobile Robotics. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 9, 6, (November 2005), pp 622-629

Pauplin, O. (2007). Application de techniques d'évolution artificielle à la stéréovision en robotique mobile autonome. PhD thesis, Université Paris Descartes, Paris, 2007

Rechenberg, I. (1994). *Computational Intelligence: Imitating Life*. IEEE Press, 0780311043, Piscataway, NJ, USA



Frontiers in Evolutionary Robotics
Edited by Hitoshi Iba

ISBN 978-3-902613-19-6
Hard cover, 596 pages
Publisher I-Tech Education and Publishing
Published online 01, April, 2008
Published in print edition April, 2008

This book presented techniques and experimental results which have been pursued for the purpose of evolutionary robotics. Evolutionary robotics is a new method for the automatic creation of autonomous robots. When executing tasks by autonomous robots, we can make the robot learn what to do so as to complete the task from interactions with its environment, but not manually pre-program for all situations. Many researchers have been studying the techniques for evolutionary robotics by using Evolutionary Computation (EC), such as Genetic Algorithms (GA) or Genetic Programming (GP). Their goal is to clarify the applicability of the evolutionary approach to the real-robot learning, especially, in view of the adaptive robot behavior as well as the robustness to noisy and dynamic environments. For this purpose, authors in this book explain a variety of real robots in different fields. For instance, in a multi-robot system, several robots simultaneously work to achieve a common goal via interaction; their behaviors can only emerge as a result of evolution and interaction. How to learn such behaviors is a central issue of Distributed Artificial Intelligence (DAI), which has recently attracted much attention. This book addresses the issue in the context of a multi-robot system, in which multiple robots are evolved using EC to solve a cooperative task. Since directly using EC to generate a program of complex behaviors is often very difficult, a number of extensions to basic EC are proposed in this book so as to solve these control problems of the robot.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Olivier Pauplin and Arnaud de La Fortelle (2008). Application of Artificial Evolution to Obstacle Detection and Mobile Robot Control, Frontiers in Evolutionary Robotics, Hitoshi Iba (Ed.), ISBN: 978-3-902613-19-6, InTech, Available from:
http://www.intechopen.com/books/frontiers_in_evolutionary_robotics/application_of_artificial_evolution_to_obstacle_detection_and_mobile_robot_control

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820

www.intechopen.com

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.