# Mobile Robot Sensor Fusion Using Flies

A. M. Boumaza[1], J. Louchet[1,2]

[1] INRIA, projet FRACTALES, Domaine Voluceau BP 105,
78153 Le Chesnay cedex, France
[2] ENSTA, 32 Boulevard Victor, 75739 Paris cedex 15, France
{Amine.Boumaza|Jean.Louchet}@inria.fr

**Abstract.** The "Fly algorithm" is a fast artificial evolution-based image processing technique. Previous work has shown how to process stereo image sequences and use the evolving population of "flies" as a continuously updated representation of the scene for obstacle avoidance in a mobile robot. In this paper, we show that it is possible to use several sensors providing independent information sources on the surrounding scene and the robot's position, and fuse them through the introduction of corresponding additional terms into the fitness function. This sensor fusion technique keeps the main properties of the fly algorithm: asynchronous processing. no low-level image pre-processing or costly image segmentation, fast reaction to new events in the scene. Simulation test results are presented.

## 1 Introduction.

The Fly Algorithm [17] is an image processing technique based on evolving a population of points in space (the "flies") using a fitness function designed in such a way that the flies converge onto the physical objects in the scene. The fitness of a fly is calculated using image grey levels. According to the general scheme of "Parisian Evolution" [3], the problem's solution is represented by the whole population rather than by the only best individual. Contrary to the general belief that artificial evolution is slow and not suited for real-time applications, we show that the Fly algorithm is a way to exploit asynchronous data delivery made possible by CMOS camera technology. A matching asynchronous robot path planner using the fly algorithm's output has been proposed in [2]. The evolutionary program's structure is widely application-independent, using problem-specific knowledge expressed in the fitness function, which is not an usual feature in image processing. The aim of this paper is to exploit this property a bit further and show how, without any major alteration of the algorithm's architecture and genetic operators, it is possible to integrate exteroceptive and proprioceptive sensors and use the Fly algorithm as a real-time sensor fusion method.

## 2 Stereovision using Flies.

### 2.1 Processing Stereo Pairs.

**Geometry and Fitness Function.** A fly is defined as a 3-D point with coordinates $(x, y, z)$. The coordinates of a fly's projections are $(x_L, y_L)$ in the image

given by the left camera and $(x_R, y_R)$ in the right camera. The cameras' calibration parameters are known, therefore $x_R, y_R, x_L, y_L$ maybe readily calculated from $x, y, z$ using projective geometry [15] [10]. If the fly is on the surface of an opaque object, then the corresponding pixels in the two images will normally have highly similar neighbourhoods (Fig. 1)[3]. Conversely, if the fly is not on the surface of an object, their close neighbourhoods will be usually poorly correlated. The fitness function exploits this property and evaluates the degree of similarity

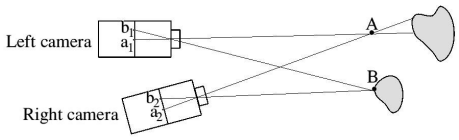


**Fig. 1.** Pixels $b_1$ and $b_2$ , projections of fly $B$, have identical grey levels, while pixels $a_1$ and $a_2$, projections of fly $A$, which receive their illumination from two different physical points on the object's surface, have different grey levels.

of the pixel neighbourhoods of the projections of the fly onto each image, giving highest fitness values for the flies lying on objects surfaces:

$$fitness(indiv) = \frac{G}{\sum_{colours} \sum_{(i,\,j)\in N} \left(L\left(x_L + i,\, y_L + j\right) - R\left(x_R + i,\, y_R + j\right)\right)^2} \quad (1)$$

- $(x_L, y_L)$ and $(x_R, y_R)$ are the coordinates of the left and right projections of the current individual (see Fig. 1)
- $L\left(x_L + i,\, y_L + j\right)$ 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 numerator $G$ is a normalizing factor designed to reduce the fitness of the flies which project onto uniform regions. It is based on an image gradient norm calculation. The best experimental results were obtained when $G$ is defined as the square root of Sobel's gradient norm [9] [15]: 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 reduce its sensitivity to the constant component of the image, which strongly depends on camera sensitivity adjustments.

**Artificial Evolution Operators.** An individual's chromosome is the triplet $(x, y, z)$ which contains the fly's coordinates. The population is initialised randomly inside the intersection of the cameras' fields of view (Fig. 2). The statistical

[3] This may not be completely true if the surfaces differ from Lambert's law, which assumes that for a given illumination, the object's radiance does not depend on the observer's position. Most surface-based stereovision methods are also sensitive to this property.

distribution is chosen in order to obtain uniformly distributed projections in the left image. The values of $z^{-1}$ are uniformly distributed between zero (or $1/d_{max}$ ) and $1/d_{min}$.

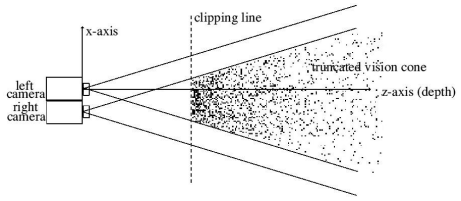


**Fig. 2.** The fly population is initialised inside the intersection of the cameras 3-D fields of view.

*Selection* is elitist and deterministic, ranking the flies according to their fitness values and retaining the best individuals (typically 50%).
*2-D sharing* [11], 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 $K * N$ , 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 current fly, given by the following formula [19] :

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

*Mutation* allows 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 $(\sigma_x, \sigma_y, \sigma_z)$ equal to $R$, so that they are the same order of magnitude as the mean distance between neighbouring flies.
In order to take into account the frequent straight lines or planar surfaces existing in real-world scenes, we translated this into two *barycentric crossover operators.* The first one 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 $F2\,(x_2, y_2, z_2)$ is the fly $F\,(x, y, z)$ defined by $\overrightarrow{OF} = \lambda\overrightarrow{OF_1} + (1-\lambda)\,\overrightarrow{OF_2}$. The weight $\lambda$ is chosen using a uniform random law in $[0, 1]$[4]. Similarly, the second operator uses three parents and determines the offspring $F$ such that $\overrightarrow{OF} = \lambda\overrightarrow{OF_1} + \mu\overrightarrow{OF_2} + (1-\lambda-\mu)\,\overrightarrow{OF_3}$ in the parents' plane, using two random weights $\lambda$ and $\mu$.

### 2.2 Processing Stereo Sequences.

We have developed several methods to process stereo image sequences. They are described in detail in [19]. The simplest one is the *random approach*, which

[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 have already been detected, 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.

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 frames. It does not alter the simplicity of the static method and does not require significant algorithm changes. 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%) it favours a convenient exploration of the whole search space.
*Dynamical approaches* [19] introduce explicit velocity components into each fly's genome. The advantage is a better precision for a given number of generations or evaluations, but this goes with a significantly higher calculation cost at each generation, and therefore a smaller number of generations in a given time interval: the best trade-off will depend on the scene style and complexity of motion.
Unlike in image segmentation-based algorithms, image pixels only need to be read when the fitness of a fly has to be evaluated: therefore the random pixel access allowed by CMOS imagers can be fully exploited and new events in the scene can be processed without usual frame refreshing delays. In the following Sections, as we consider a vision system embedded into a mobile robot, we will use the simple random approach described above, but update the flies' positions at each generation, using the information available about the robot's motion in order to give better initial conditions and allow faster convergence of the fly population. This will be the basis of the proprioceptive fusion described in Section 3.4.

### 2.3 Path Planning with Obstacle Avoidance.

The unusual way the scene is described using the fly algorithm (compared to more classical stereovision algorithms 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. The simulator is described in Section 3. To keep the speed advantage of the fly algorithm, we chose to adapt fast potential-based methods from the literature [16]. In former work [2] we defined a force derived from the addition of an attractive (target) and a repulsive (flies) potential acting as a steering command: the blockage situations [28] were resolved using two heuristics creating a new force attracting the robot out of the potential minimum (random walk and wall following methods). In this paper we use a different potential field-based path planner based on harmonic functions [5]. A harmonic function is a function which satisfies Laplace's equation:

$$\Delta U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0 \tag{3}$$

One of the interesting properties of harmonic functions is the absence of local minima, which in our case eliminates the blockage situations.
The robot's environment memory consists of a sampled grid which represents a harmonic function. Values of the function at the target and obstacle positions are modelled as Dirichlet boundaries: 1 for obstacles, 0 for the target position [4]

[5]. The harmonic function is built iteratively using a finite difference operator, such as the Gauss-Seidel operator which consists in replacing the value of a point with the average value of its four neighbours. After convergence, we end up with a smooth function that has a single global minimum at the target position.
The steering command of the robot is the gradient of the harmonic function. Linear interpolation is used when the robot position falls between grid points. During the movement toward the target, new obstacles detected by the fly algorithm are introduced into the harmonic function according to the local fly density, as high potential Dirichlet boundaries. Since the harmonic function is constantly iterated, there will be a constantly updated obstacle avoiding path for the robot to follow (Fig. 3). The Dirichlet boundaries are set at the positions of the best flies in the robot's field of view.

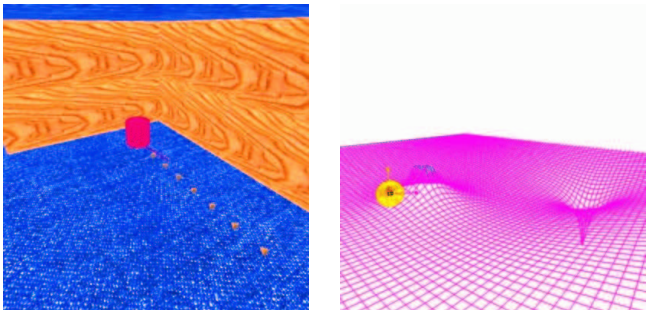

**Fig. 3.** The robot facing an obstacle (left) and the corresponding harmonic function (right).

### 2.4 Real-time compliance.

With the Fly algorithm, each individual evaluation only needs pixel values on a small neighbourhood. CMOS camera technology is well adapted to this requirement as it is able to deliver these values at any time, while conventional image processing techniques do not exploit this feature. Similarly, the Fly algorithm delivers updated scene information at any time, giving the planner faster reaction to scene events (Table 1).

| | CCD sensor + standard approach | CMOS + flies |
|---|---|---|
| image sensor | delay between capture and restitution | asynchronous data reading |
| image processing (input) | image segmentation must wait end of capture cycle | random access to pixels needed by current fitness calculation |
| image processing (output) | not available until segmentation process has ended | current state of representation always available |
| trajectory planner | must wait for image processing output | saves 2 acquisition cycles |

**Table 1.** Comparison of the Fly algorithm with conventional methods.

## 3 Data Fusion using extra sensors.

Sensor fusion [13] plays an important role in a robot's perception system. It synthesizes the information given by exteroceptive and proprioceptive sensors. Most classical approaches are based on Bayes' theorem. In this Section, we show that in spite of the absence of any formal link between a fly's fitness value and the probability that it lies on an obstacle, similar sensor fusion may be achieved using the fly technique with an extended fitness function.

### 3.1 Simulated ultrasonic obstacle detectors.

Usual ultrasonic obstacle detectors (sonar) are able to detect the presence or absence of an obstacle in their field of view. The view angle depends on the transducer size and acoustic wavelength: we used an array of six simulated sonars with (adjustable) 15° angle and a given maximum detection distance, shorter than the vision system range. In the real world, the sonar's detection range may depend on obstacle's size and sound dampening properties. To simulate such a detector, we first use the Zbuffer map from the image synthesis inside the sonar's field of view and calculate the obstacle-robot distances. Their smallest value is returned as the simulated ultrasonic sensor's output.

### 3.2 Fusion of exteroceptive sensors: the multi-sensor fitness function.

The new fitness function should integrate information issued by all the exteroceptive sensors. As stated above, it is difficult to give a formal mathematical justification of how to extend the expression of the fitness function to integrate several sensors. In qualitative terms, if a fly's position is in accordance with several independent sensors, its fitness must be increased accordingly. Conversely, a fly confirmed by the visual sensor should still be considered seriously as a real obstacle even if unnoticed by another sensor (e.g. a visible obstacle covered with sound damping material). We defined each sensor's contribution to the fitness function as follows: if a fly is located inside the vision angle of a triggered detector, and its distance to the detector matches the obstacle distance given by the detector accurately enough, then the fly's fitness is increased by a given percentage $B$ :

$$fitness_{new} = (1 + B)\, fitness_{old} \tag{4}$$

If the fly's position does not match any range finder information, then the fitness remains unchanged. We chose a multiplicative rather than additive contribution to the fitness because of the low angular resolution of the ultrasonic sensors: this prevents all flies at the correct distance inside the field of an ultrasonic sensor from getting high fitness values even if they are inconsistent with image data. In addition to the integration of additional sensors into the fitness function, we introduced an additional *immigration* operator: new flies are introduced into the population with a bias in favour of 3-D positions given by the sonars. This helps detecting close obstacles which might otherwise have been overlooked by the system. This method allows the fusion of an arbitrary number of sonars (or any similar exteroceptive sensors) at the fitness level, but it is not appropriate to the integration of proprioceptive sensor information, which we examine in the following Sections.

### 3.3 Simulated Odometry.

On a real robot, odometric data given by wheel rotation coders give an estimation of the robot's position. In most applications, wheel rotation is under control of the trajectory planner. The actual robot trajectory does not exactly match the target trajectory due to "trajectory noise" caused by factors such as wheel slipping, tyre wear or rough ground surface. In the simulator, the robot's actual position is simulated by adding a Gaussian noise to the planner's command, but the sensor fusion algorithm only gets the odometric estimation which is identical to raw trajectory planner data.

### 3.4 Fusion of proprioceptive sensors: proprioception as an operator.

At each time step, the initial population of flies is based on the preceding generation's results, and positions are updated using the best available knowledge about robot's motion. On the simulator, this information is given by odometric data (Section 3.3), but there could be any other source, e.g. an inertial sensor. The information received by the fly algorithm on the robot's position may not be accurate but remains useful in updating the flies' positions. We noticed that throughout our experiments, after a small number of generations (2 or 3) the flies converge again to the obstacles surfaces.
Figures 4 and 5 show images $N$ and $N + 5$ from a sequence, with a 1 degree right turn of the robot at each frame, and one generation of the fly evolution per frame. On Figure 4, the flies' positions have not been updated using odometric information: the result is a delay in the evolution of flies (the black dots on the images). Figure 5 shows the results on the same sequence but integrating noisy odometric proprioceptive information: even if proprioceptive information is not required for the algorithm to work, it allows better fly stability and more accurate obstacle detection.

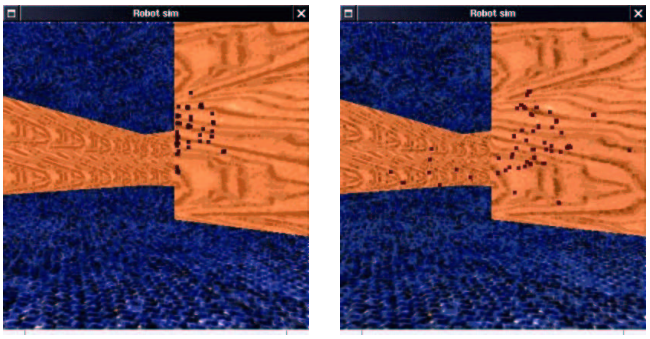


**Fig. 4.** frames $N$ and $N + 5$, without proprioception.

## 4 Simulation Results.

Figure 6 shows the functional diagram of a mobile robot simulator using the fly algorithm, including cameras and other sensors (e.g. ultrasonic obstacle detec-

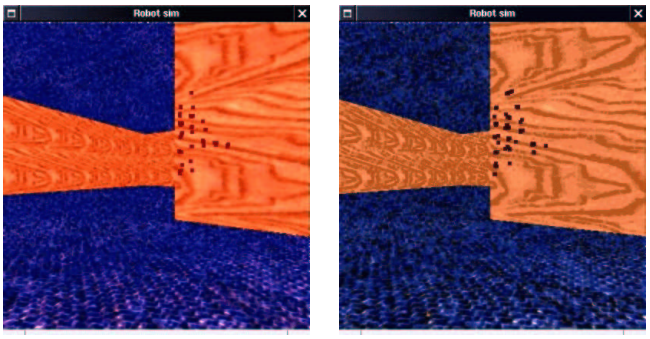


**Fig. 5.** the same two frames, with proprioception: better obstacle following.

tors). The software implementation uses a shared memory containing the images, sonar data, the fly population and the harmonic function. This memory can be accessed at any time by any process. Figure 7 shows typical examples of robot trajectories produced by the simulator. On the right we may compare the actual robot trajectory (green line) with the robot's internal trajectory representation (yellow line). The purple lines show the obstacles (in perspective).

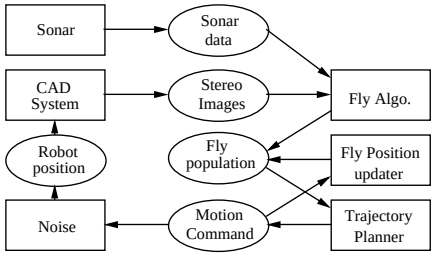


**Fig. 6.** The complete robot simulator.

Figure 8 shows an example of how the addition of sonar information into artificial evolution improves the detection of a short-range obstacle (on the right side of the robot axis, right image). On the left image, the sonar information has not been used: the flies concentrated onto further obstacles and neglected the closer one.

## 5 Conclusion.

We showed in this paper that the evolutionary strategy underlying the fly algorithm may be extended to perform embedded robot sensor fusion, including proprioceptive and exteroceptive information. It retains the whole structure, original spirit and properties of the original fly algorithm: low calculation cost and fast convergence, noise robustness, asynchronous properties well adapted to state-of-the-art CMOS imagers. The robot simulator presented also includes

a non-evolutionary trajectory planner adapted to use the fly population as an input, with matching asynchronous properties.

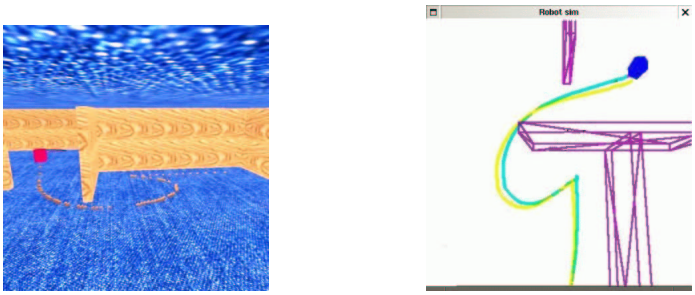


**Fig. 7.** On the left, an obstacle avoidance trajectory. On the right, the actual robot trajectory (green) and the trajectory given by odometric data (yellow)

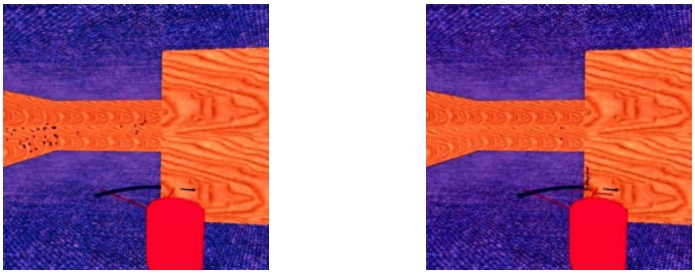

**Fig. 8.** Obstacle detection without (left) and with (right) fusion of ultrasonic data. Red lines show the limits of the camera's field of view.

## References


1. D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, 1982.
2. Amine M. Boumaza and Jean Louchet. Dynamic flies: Using real-time parisian evolution in robotics. In Egbert J. W. Boers, Stefano Cagnoni, Jens Gottlieb, Emma Hart, Pier Luca Lanzi, Gunther R. Raidl, Robert E. Smith, and Harald Tijink, editors, *Applications of Evolutionary Computing*, volume 2037 of *LNCS*, pages 288–297, Lake Como, Italy, 18 April 2001. Springer-Verlag.
3. P. Collet, E. Lutton, F. Raynal, and M. Schoenauer. Individual gp: an alternative viewpoint for the resolution of complex problems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honovar, M. Jakiela, and R. E. Smith, editors, *Genetic and Evolutionary Computation Conference GECCO99*. Morgan Kaufmann,San Francisco, CA, 1999.
4. C. I. Connolly, J. B. Burns, and R. Weiss. Path planning using laplace's equation. In *The Proceedings of IEEE Internetioal Conference on Robotics and Automation, ICRA'90*, pages 2102–2106, May 1990.
5. C. I. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotic and Systems*, 10(7):931–946, October 1993.
6. Cumuli project: Computational understanding of multiple images, http://www.inrialpes.fr/cumuli.
7. R. C. Eberhart and J. A. Kennedy. New optimizer using particle swarm. In *Proc. Sixth Int. Symposium on Micro Machine and Human Science, Nagoya*, pages 39–43, Piscataway, NJ, 1995. IEEE Service Center.

8. D. E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addison Wesley, Reading, MA, 1989.
9. R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Wiley, 1992.
10. R. M. Haralick. Using perspective transformations in scene analysis. *Computer Graphics and Image Processing*, (13):191–221, 1980.
11. J. Holland. *Adaptation in Natural and Artificial Systems.* Univ. of Michigan Press Press, 1975.
12. J. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 1992.
13. B. H. Horn. *Robot Vision.* McGraw Hill, 1986.
14. P. V. C. Hough. Method and means of recognizing complex patterns. U.S. Patent no3, 069 654, 18 December 1962.
15. R.C. Jain, R. Kasturi, and B.G. Schunck. *Machine Vision.* McGraw-Hill, New York, 1994.
16. O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–99, Spring 1986.
17. J. Louchet. From Hough to Darwin: an individual evolutionary strategy applied to artificial vision. In *Proceedings of Artificial Evolution 99*, Dunkerque, France, November 1999.
18. J. Louchet. Using an individual evolution strategy for stereovision. *Genetic Programming and Evolvable Machines. Kluwer*, to appear, 2001.
19. Jean Louchet, Maud Guyon, Marie-Jeanne Lesot, and Amine Boumaza. Dynamic flies: a new pattern recognition tool applied to stereo sequence processing. *Pattern recognition letters*, 23:335–345, 2002.
20. E. Lutton and P. Martinez. A genetic algorithm for the detection of d geometric primitives in images. In *The Proceedings of the International Conference on Pattern Recognition, ICPR'94*, pages 526–528, Los Alamitos, CA, October 9-13 1994. IEEE Computer Society.
21. M. C. Martins and H. P. Moravec. Robot evidence grid. Technical report, The Robotics Institute, Carnegie Mellon University, March 1996.
22. M. Millonas. Swarms, phase transitions and collective intelligence, artificial life iii. In C.G Langton, editor, *Santa Fe Institute Studies in the Sciences of Complexity*, volume XVII. Addison Wesley, Reading, MA, 1994.
23. I. Rechenberg. Evolution strategy. In J.M. Zurada, R.J. MarksII, and C.J. Robinson, editors, *Computational Intelligence imitating life*, pages 147–159. IEEE Press, Piscataway, NJ, 1994.
24. G. Roth and M. D. Levine. Geometric primitive extraction using genetic algorithm. In *Proceedings of the IEEE Conference on Computer vision ans Pattern Recognition, CVPR'92*, Piscataway, NJ, 1992. IEEE Press.
25. R. Salomon and P. Eggenberger. Adaptation on the evolutionary time scale: a working hypothesis and basic experiments. In *Springer Lecture Notes on Computer Science*, number 1363, pages 251–262. Springer-Verlag, Berlin, 1997.
26. P.K. Ser, S. Clifford, T. Choy, and 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*, pages 1–13, 1999.
27. C. K. Tang and G. Medioni. Integrated surface, curve and junction inference from sparse 3-d data sets. In *ICCV98*, pages 818–823, Piscataway, NJ, 1998. IEEE Computer Society Press.
28. Y.Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Procedings of the IEEE Conference On Robotics and Automation, ICRA'91*, pages 1398–1404, Sacramento, California, April 7-12 1991.
29. John S. Zelek. Complete real-time path planning during sensor-based discovery. In *IEEE/RSJ International Conference on Intelligent Robots and systems*, 1998.