Parametric and Evolutionary Methods in Image Processing

Jean Louchet, Evelyne Lutton

COMPLEX team
INRIA
BP71, F-78153 Le Chesnay cedex
France
jean.louchet@gmail.com, evelyne.lutton@inria.fr

Summary

1 Image segmentation as a heuristic to build a model of the scene

1.1 A (highly conjectural?) historical view of Image Processing

Role of image synthesis:

- Assessment and validation of image analysis algorithms
- Getting around live data capture, field truth etc.

Tracking algorithms...

Development of image synthesis techniques:

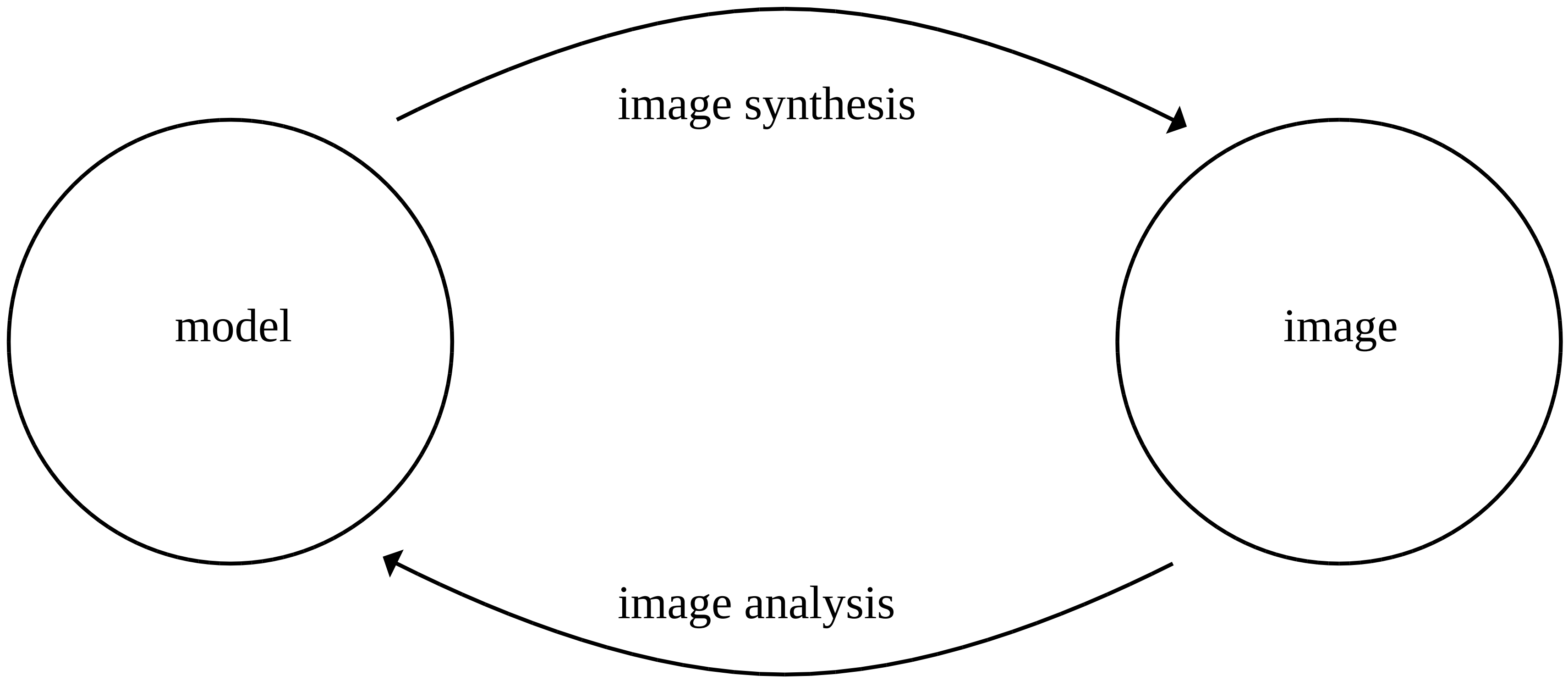
- photometry
- probabilistic texture theory
- solid and fluid mechanics
- robotics
- high visual quality as dictated by the image and film industry
- make these techniques more affordable

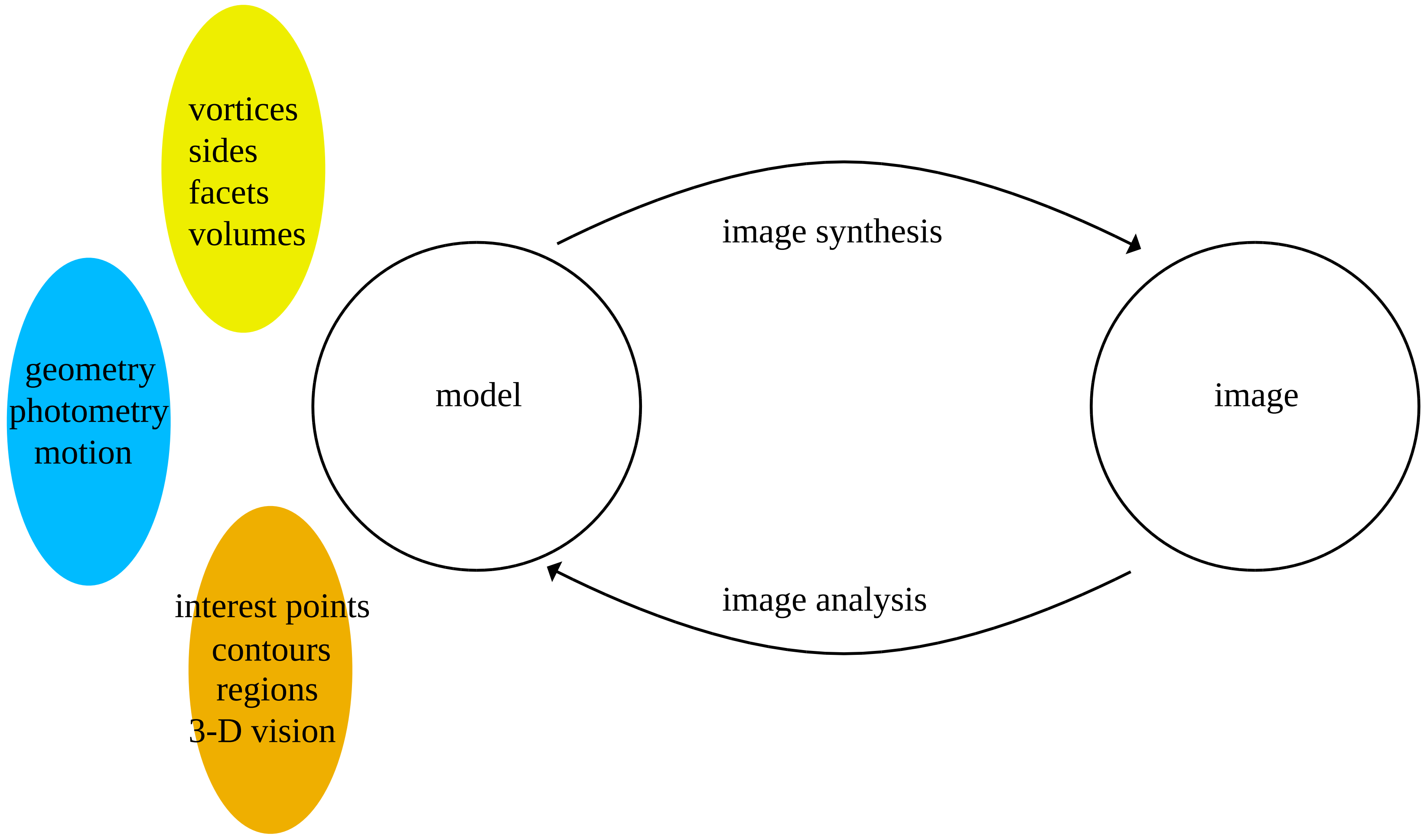
Influence of the “artificial intelligence” and “knowledge-based systems” philosophy:

rather than merely use image synthesis as a tool to build *a posteriori* evaluation systems,
use it to express and manipulate *a priori* knowledge on the scene to be analysed.

Computer vision may be considered as an engine aimed at identifying the parameters of a scene model, using pixel-level calculations.

1.2 Heuristics and segmentation





3-D model and segmentation

dimension	segmentation type	synthesis equivalent	technique	property
0	interest point	vortex	local extremum	concentration
1	contour	side	gradient	disparity
2	region	facet	colour, texture	uniformity
3	dynamics	motion, stereo	correspondence	similarity, disparity length

Correspondence between image analysis and synthesis primitives

Can we explain the “historical” success story of differential operators and contour detection?

An alternative to contours: region extraction.

rely on discontinuities (which may have a psychophysic flavour) $\rightarrow$ development of differential methods to detect contours.

Signal Analysis inspiration $\rightarrow$ similar differential operators.

relevance of **contours** as probable projections of 3-D edges.

A contour is a line in the image, that has a high probability to be the projection of an edge of an object in the scene.

Region algorithms $\rightarrow$ detection of probable projections of homogeneous facets from the 3-D scene
rely more explicitly on physics: geometric (connectivity) and photometric (material homogeneity) properties

A region is an area in the image, that has a high probability to be the projection of a facet of an object in the scene.

THE FUNDAMENTAL AMBIGUITY OF SCENE ANALYSIS:

IMAGE SYNTHESIS IS NOT AN INJECTIVE OPERATOR
$\Rightarrow$ SCENE ANALYSIS IS AMBIGUOUS

several different scene models can give the same image

Role of stereovision...

Use specific and non-specific knowledge on the scene:

- specific:
  - geometric description,
  - texture and photometric attributes: albedo, diffusion diagrams, light sources
- non-specific (general knowledge on physics):
  - optics, light propagation
  - rules about objects collisions (if applicable), etc.
- heuristics: shape from shading, etc.

2 Direct exploration of parameter space: the Hough transform

2.1 The original Hough transform

The well-known equation of a straight line is

$$y = ax + b$$

In the co-ordinate system (a, b), the set of lines containing a given pixel (x, y) is given by:

$$b = (-x)a + y$$

The original image (x, y) co-ordinate system is often called the *plane*,
and the (a, b) co-ordinate system the *dual plane.*

each point (pixel) in the plane $\leftrightarrow$ one line in the dual plane.
Each point (a, b) in the dual plane $\leftrightarrow$ one line in the plane.
$\Rightarrow$ a straightforward computer implementation: given an original binary image,

- create an (a, b) buffer (the dual plane, or “Hough” image)
- for each interesting pixel (x, y) in the plane, increment each pixel (a, b) of the Hough image such that $y = ax + b$
- important alignments in the original image $\rightarrow$ high peaks in the dual image.
- The highest peaks give the main alignments.

ISOTROPY?

- near-vertical lines need extreme values of a and correspondingly large buffers,
- accumulation in the Hough space does not give appropriate peaks for extreme values of a

The solution proposed by Hough is to use isotropic line equations:

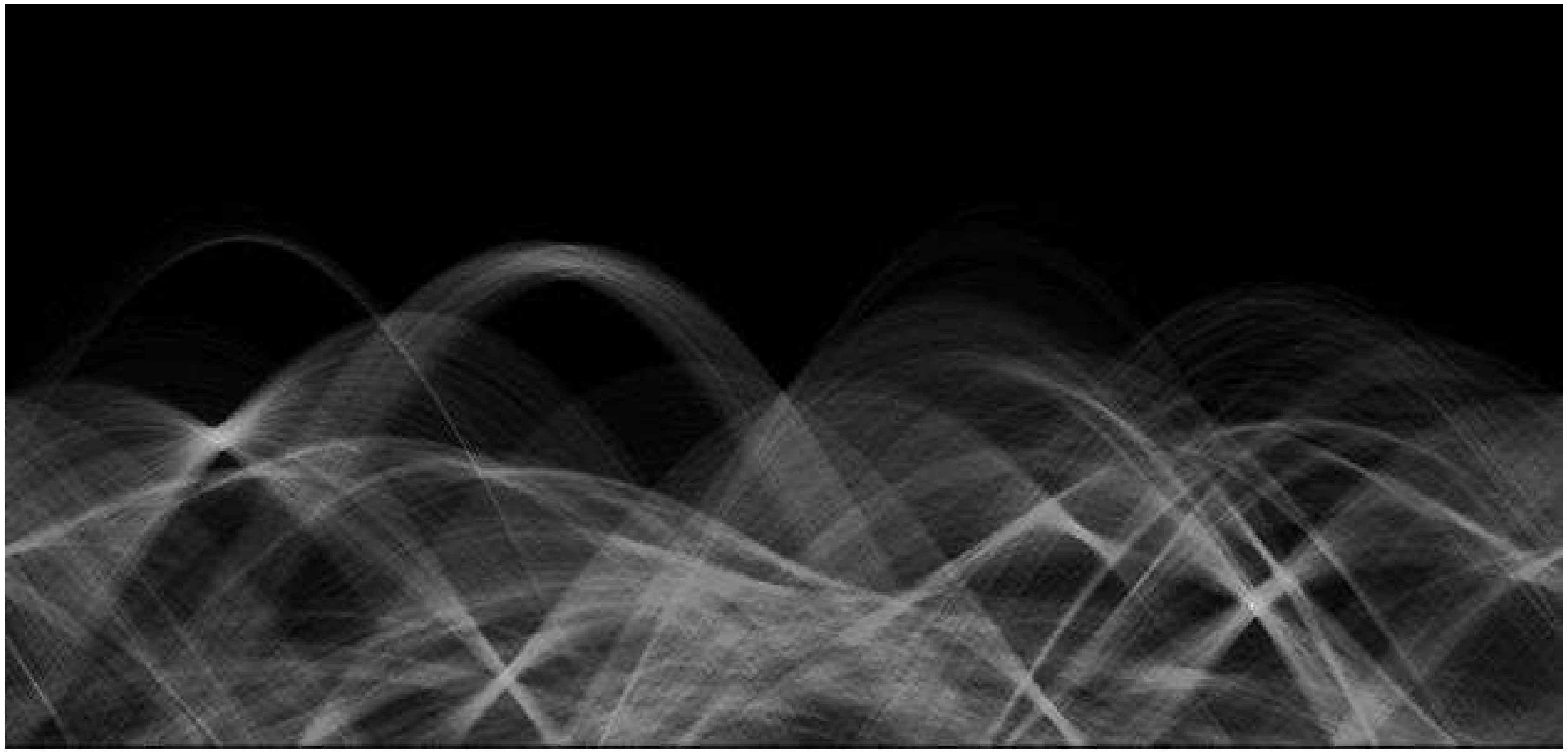
$$\rho = x\,cos\theta + y\,sin\theta$$

in the (ρ, θ) Hough space.
This time, alignments in the original image yield intersecting sinusoid curves in the Hough space, which will be detected as peaks.

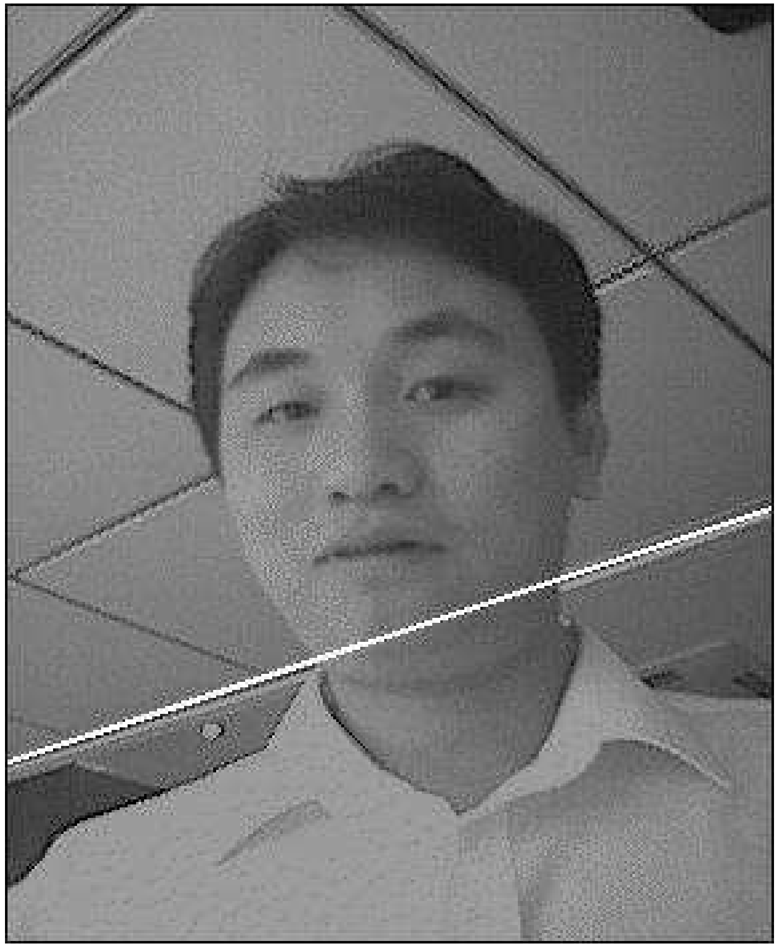
- This method gives no preference to any direction.
- Moreover, the Hough buffer is limited to the intervals $\theta \in [0,2\pi[$ and $\rho \leqslant (half\ image\ diagonal)$.

In practice, this algorithm is rather slow and benefits from tabulating trigonometric functions.

It is not absolutely necessary to use binary images as an input.



Parameter space (θ, ρ) containing the Hough accumulator of the following image (image 256×300).



Result of the classical Hough transform (image 288×352)

2.2 Multivariable Hough

The same ‘vote’ method $\rightarrow$ detection of any curve given its implicit equation.
However, computation time and buffer size increase exponentially with the number of parameters.

An example is the **detection of circles** with a given radius r: the equation is

$$(x - a)^2 + (y - b)^2 = r^2$$

Motion detection - matching interest points etc.

Quickly becoming hard... depending on the number of parameters!

3 Artificial Evolution and the art of searching parameter spaces

3.1 Basics of Artificial Evolution

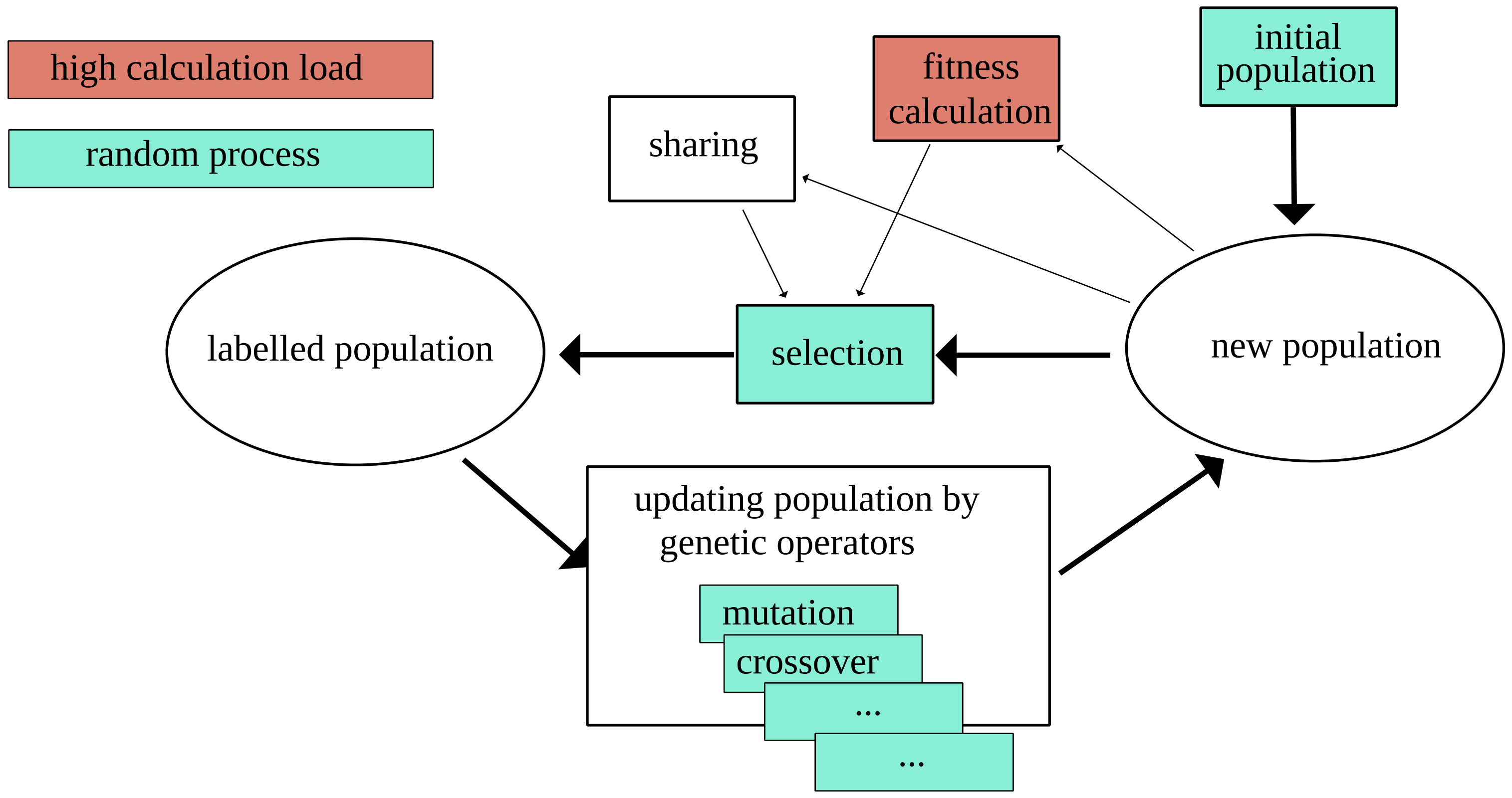
This is *not* a complete tutorial on evolutionary algorithms!

parallel operators looking for a global maximum

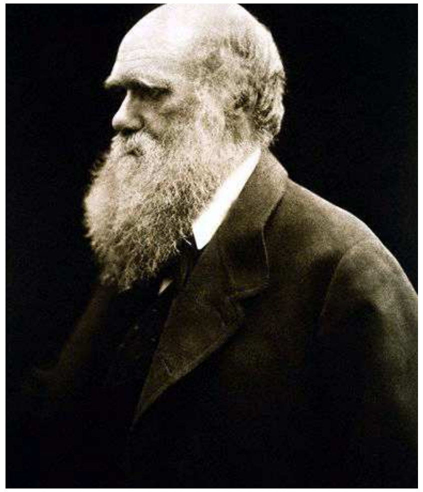
Darwin
Lamarck

$$f \; : \; R^n \rightarrow R$$

$$(x_1, x_2, \ldots x_n) = Argmax\, f$$



One of the possible architectures of an AE algorithm


Charles Darwin (1809-1882)

Hans-Paul Schwefel ("Evolutionsstrategie")

Riccardo Poli, David Goldberg (“Genetic algorithms”)

Other pionneers of Artificial Evolution: Wolfgang Banzhaf, William Langdon, Michèle Sebag.

3.2 Evolutionary Algorithm classes

Coding:

- boolean: genetic algorithm
- float (analog): evolution strategies
- rules: classifier systems
- code: genetic programming

Phenotype vs. genotype

3.3 Genetic Operators

3.3.1 Selection

The fittest will survive

Metaphor: give a better survival rate and more descendance to individuals with high fitness values.

- Ranking
- Tournament
- Elitism
- Constraints: best translated into smooth penalties into the fitness function
- sharing: allow better exploration of space
  - not miss global optima by over-concentrating on local maxima
  - search multiple solutions

3.3.2 Mutation

Random exploration...

Role of temperature in simulated annealing

Semantically weak, but...

3.3.3 Crossover

Recombine genotype elements

- cut and paste
- barycentric (only in evolution strategies)

semantics and dependence on coding?

A useless but typical example of crossover.

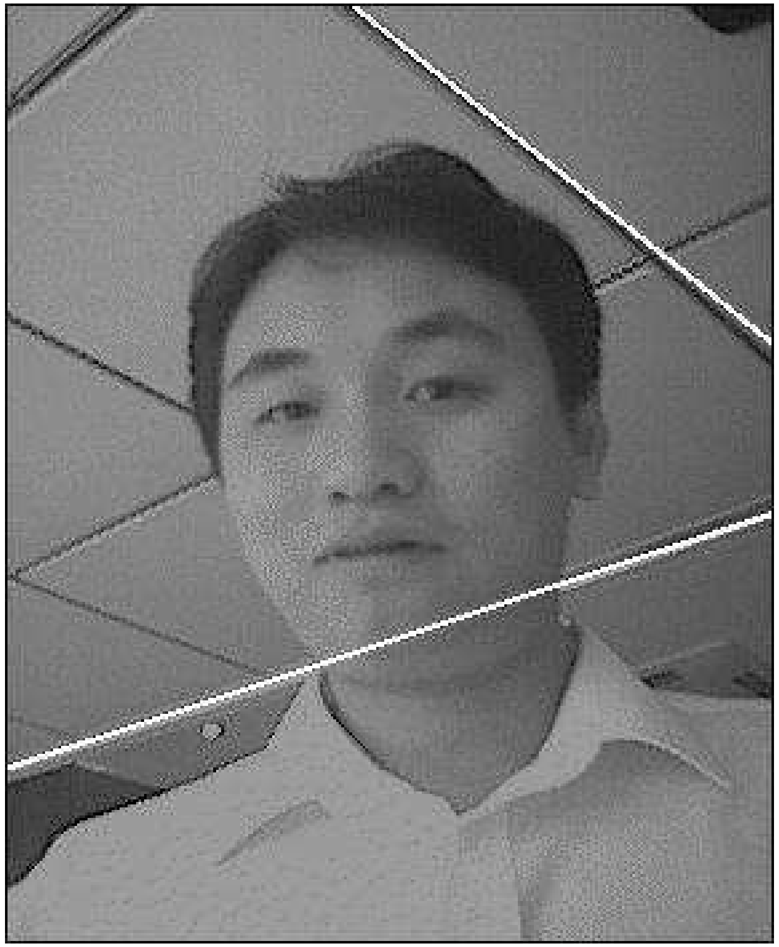
3.3.4 Refinements

- sub-population (Cohoon), co-evolution, niching
- individual approach (“parisian”)
- from AE to multi-agent systems... artificial life, autonomous agents
- coding refinement: diploidy, multiploidy, recessive vs. dominant characters, etc.
- unique property: it is possible to edit the fitness function during algorithm execution. Internal and external time.
- Towards a “real-time” version of optimisation techniques?

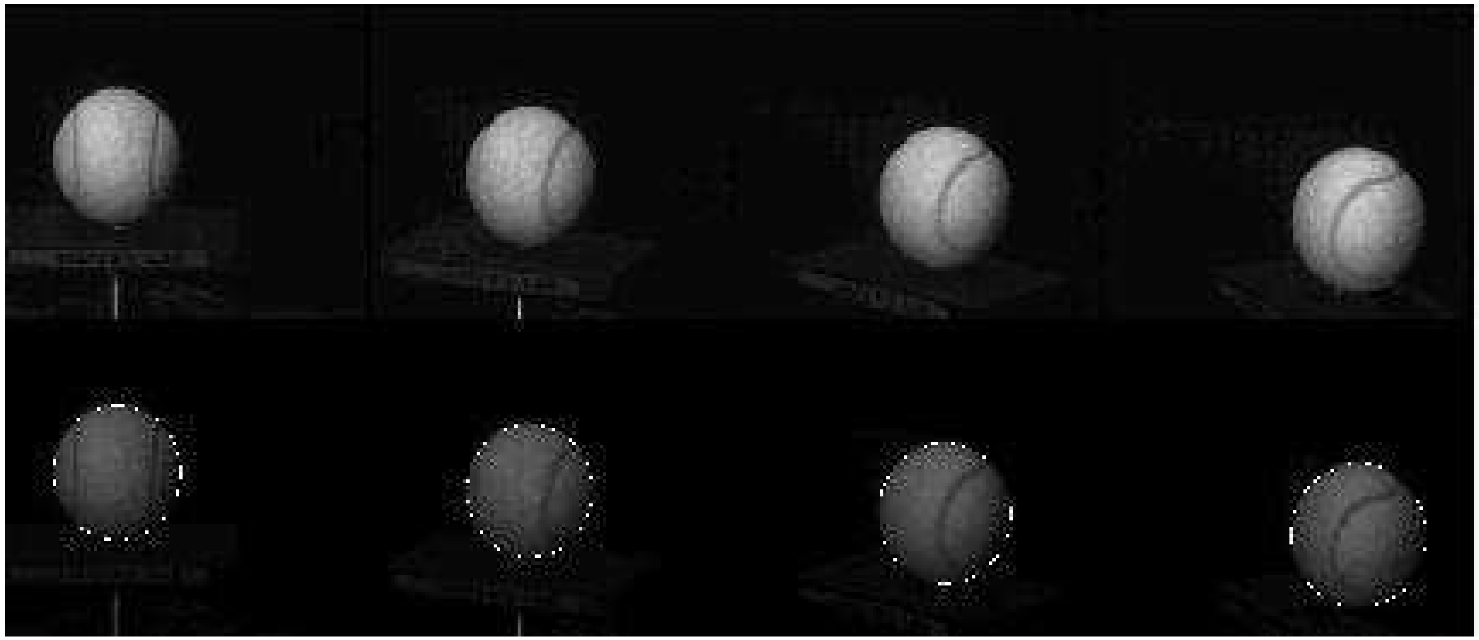
4 Evolutionary Hough extensions

Directly explore the parameter space

How to calculate faster the fitness of a straight line?



Result of the evolutionary Hough transform.



Four original images from the “tennis ball” sequence (top) and the results of circle tracking using an evolutionary Hough transform (bottom).

Population size	**100**
Selection	2 - tournament
Mutation rate (%)	15
Mutation amplitude *r*	10
Mutation amplitude *a*, *b*	40
Crossover rate (barycentric) (%)	5
Generations per frame	800 to initialise. then 240 per frame

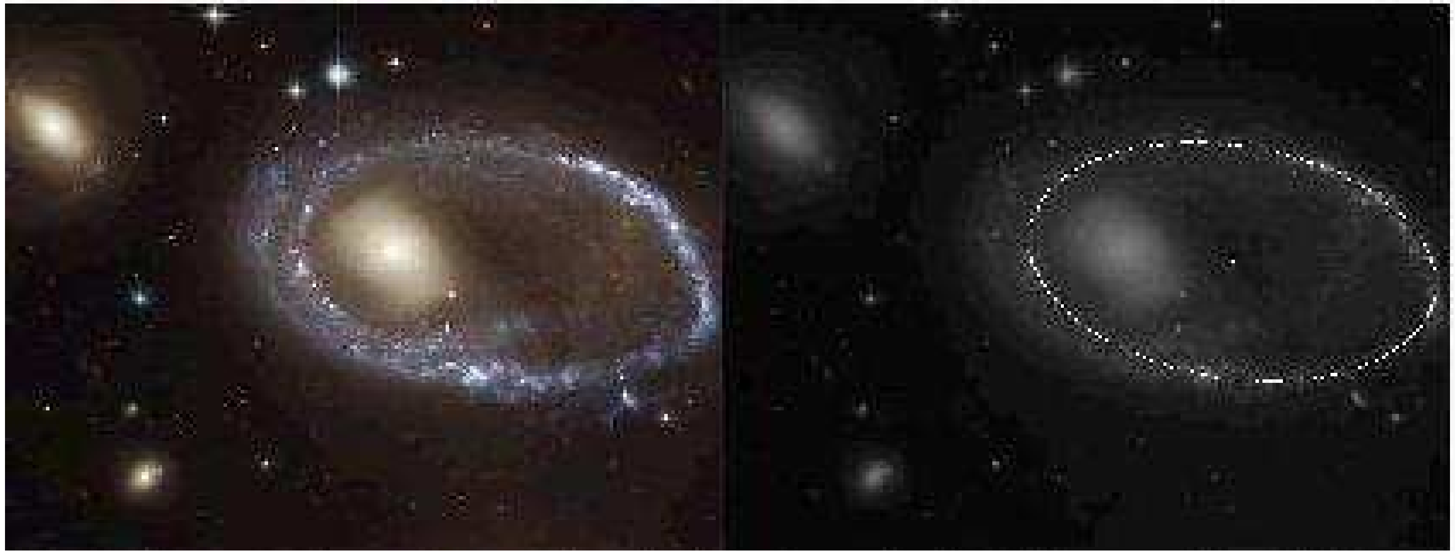
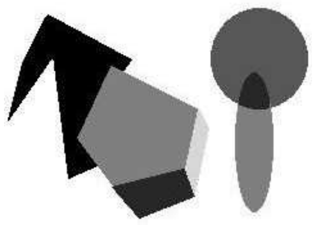


Image of galaxy AM 0644-741 taken by the Hubble telescope (left) and result of the evolutionary Hough ellipse detector (right).

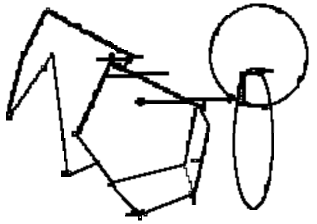
5 Genetic programming in image processing

John Koza and the artificial ant in a binary world.

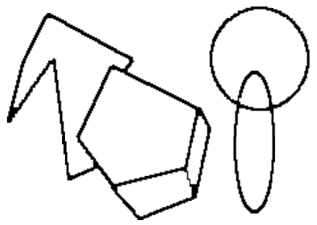
- one sensor
- pheromone (not in this example)
- autonomy: number of steps before starving
- penalise rotations
- $fitness = (quantity\ of\ food) - A \times (\#\, steps) - B \times (code\ size)$



test image



animat's trajectory



contour detected

Synthetic test image and results

```
if (contour ahead)
else turn right;
move;
move;
turn left;
if (contour ahead) {
  move;
  move;
}
else {
  if (already visited) turn right;
  else {
    turn left;
    move;
    turn right;
  }
}
```

first part

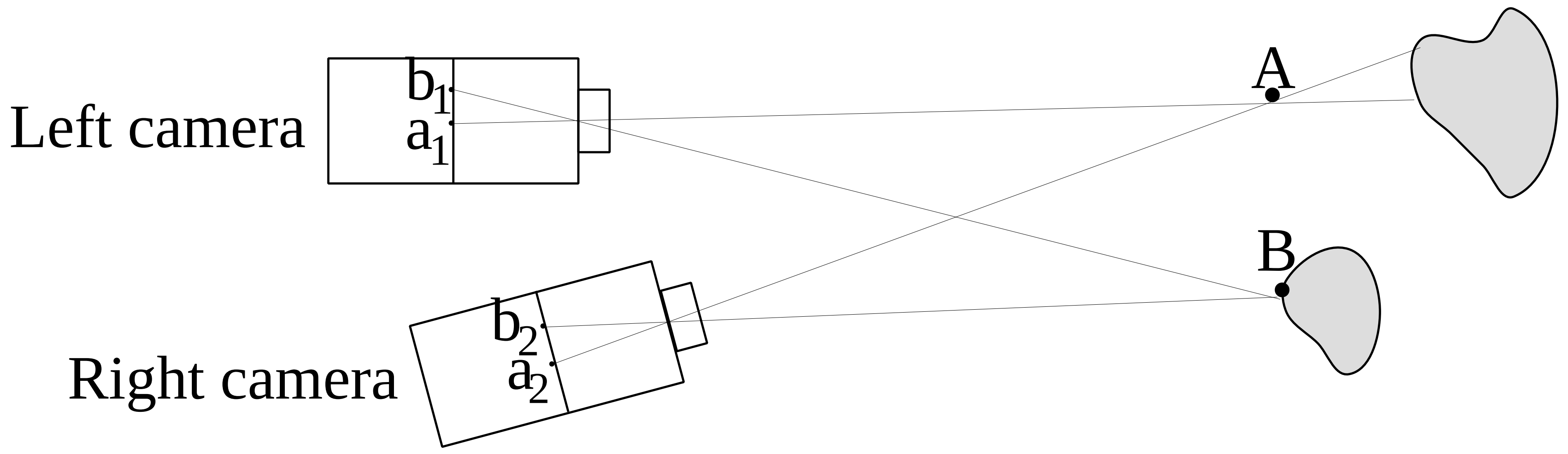
second part

Code given by the evolutionary algorithm - redundent instructions have been removed by hand.

6 The Parisian Approach: the Fly algorithm

6.1 The basics of the Fly algorithm

$$\begin{pmatrix} x_L \\ y_L \\ 1 \end{pmatrix} \equiv F_L \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} , \quad \begin{pmatrix} x_R \\ y_R \\ 1 \end{pmatrix} \equiv F_R \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



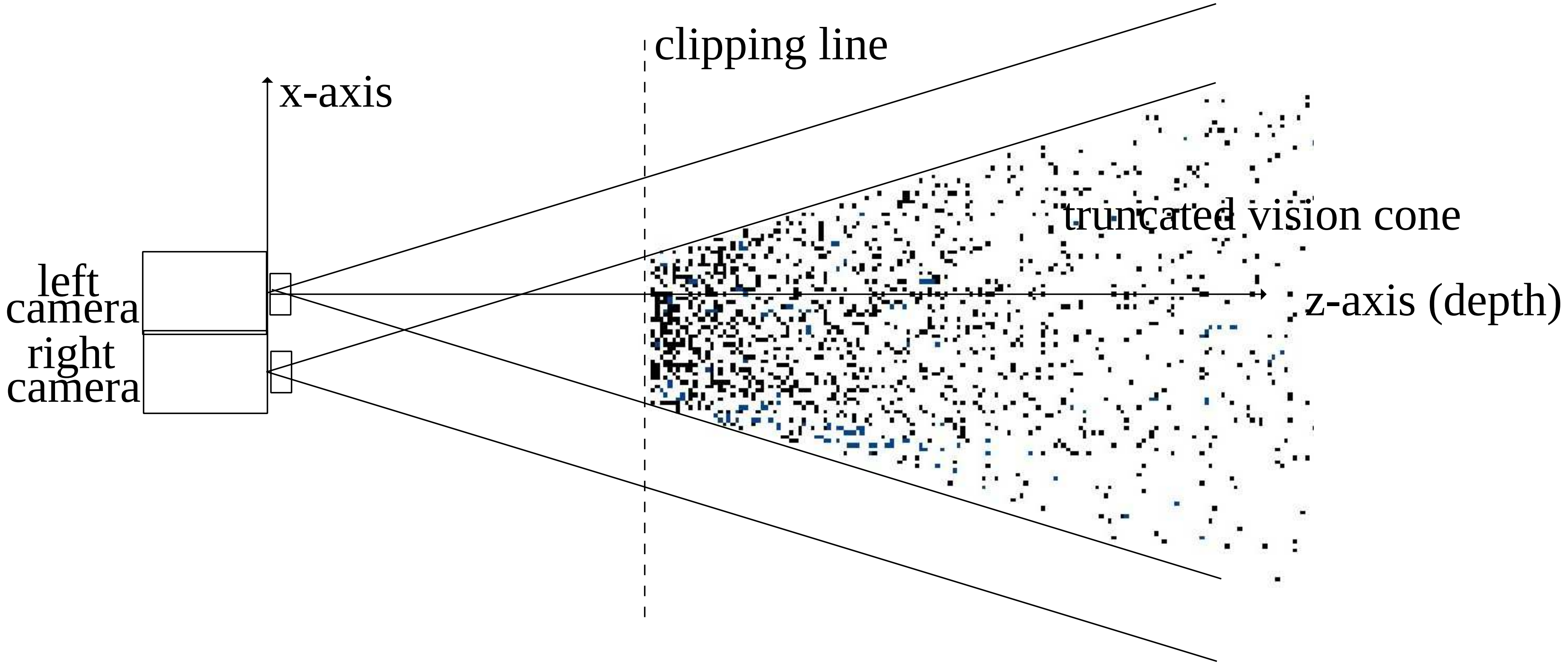
Pixels b_1 and b_2, projections of fly B, have identical grey levels, while pixels a_1 and a_2, projections of fly A, which receive their illumination from two different physical points on the object's surface, have different grey levels.

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

- (x_L, y_L) and (x_R, y_R) are the co-ordinates 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.

$$G = \sqrt{\sum_{(i,j) \in N} \left(L(x_L + i, y_L + j) - L(x_L, y_L)\right)^2}$$



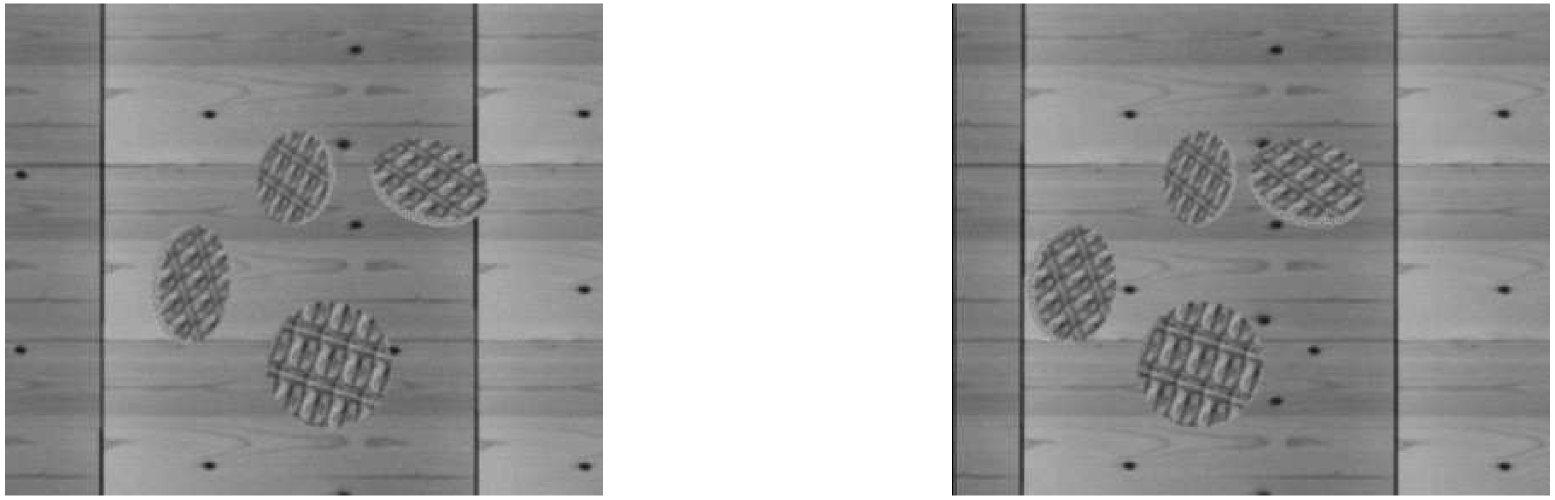
The fly population is initialised inside the intersection of the cameras 3-D fields of view.

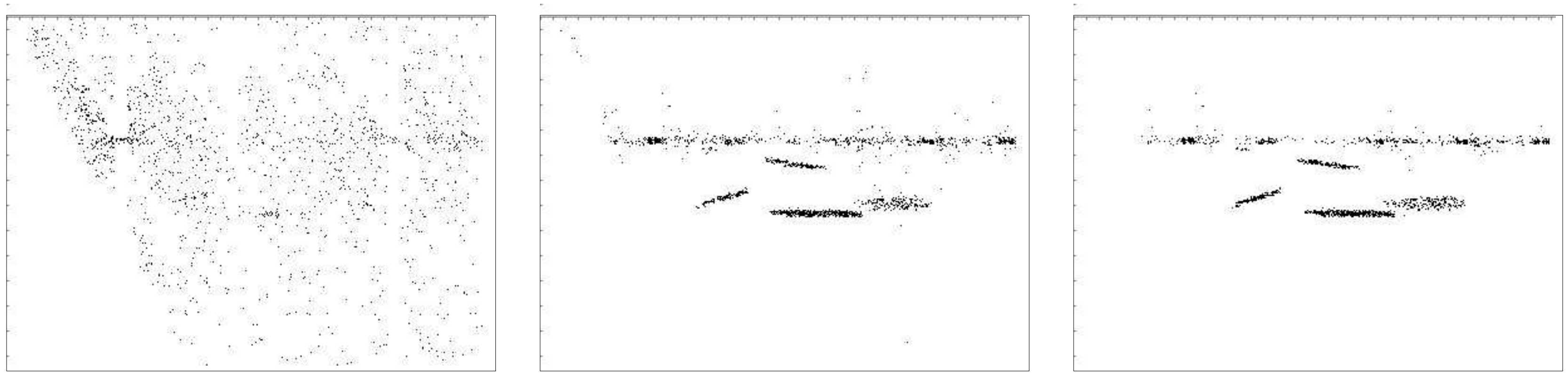
Sharing radius:

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

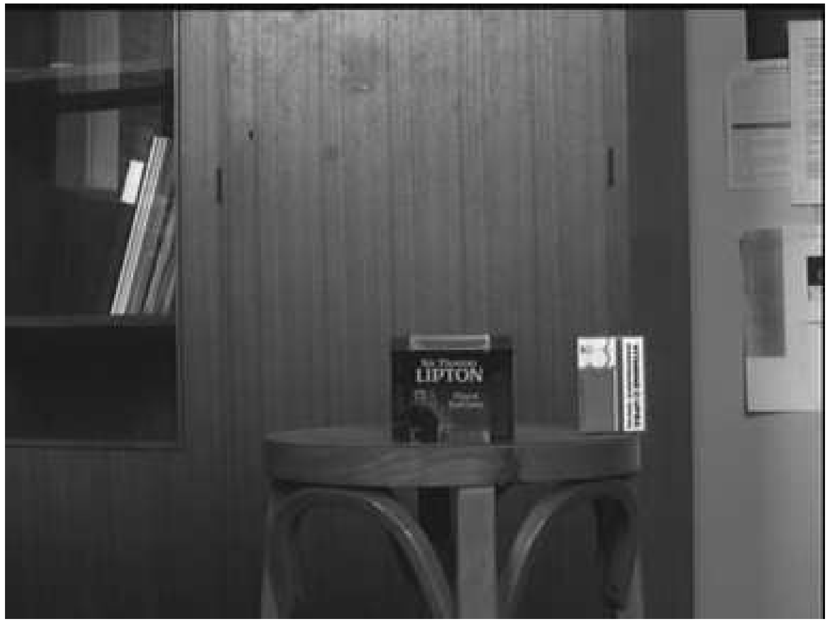
Mutation allows extensive exploration of the search space. It uses an approximation of a Gaussian random noise added to the flies' chromosome parameters (x, y, z).
Standard deviations $\sigma_x, \sigma_y, \sigma_z$ equal to R, so that they are the same order of magnitude as the mean distance between neighbouring flies.

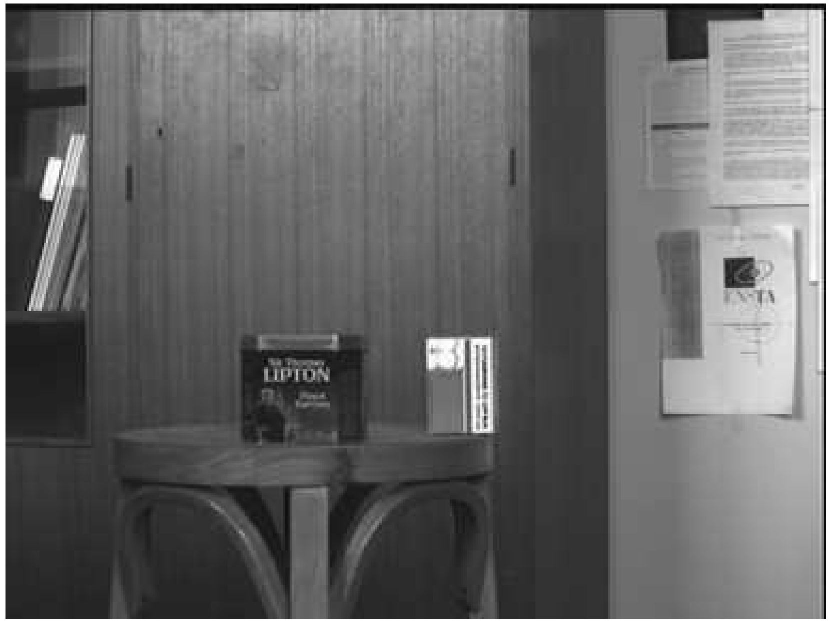
Two *barycentric crossover operators*, in order to take into account the frequent straight lines and planar surfaces existing in real-world scenes.
The first one builds an offspring randomly located on the line segment between its parents: the offspring of two flies $F_1\,(x_1, y_1, z_1)$ and $F_2\,(x_2, y_2, z_2)$ is the fly $F(x, y, z)$ defined by $\overrightarrow{OF} = \lambda\,\overrightarrow{OF_1} + (1 - \lambda)\overrightarrow{OF_2}$. The weight λ is chosen using a uniform random law in the interval $[0,1]$ or $[-0.5,1.5]$.
Similarly, the second crossover 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 λ and μ.

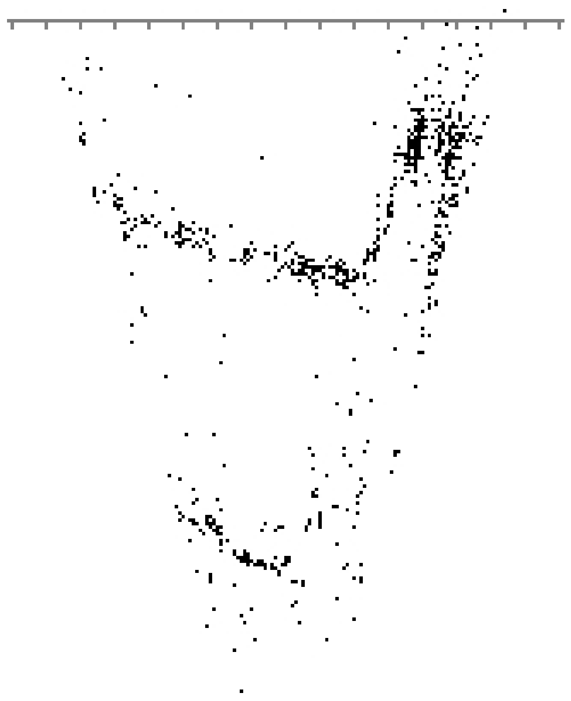




Convergence after 10 , 50 and 100 generations.

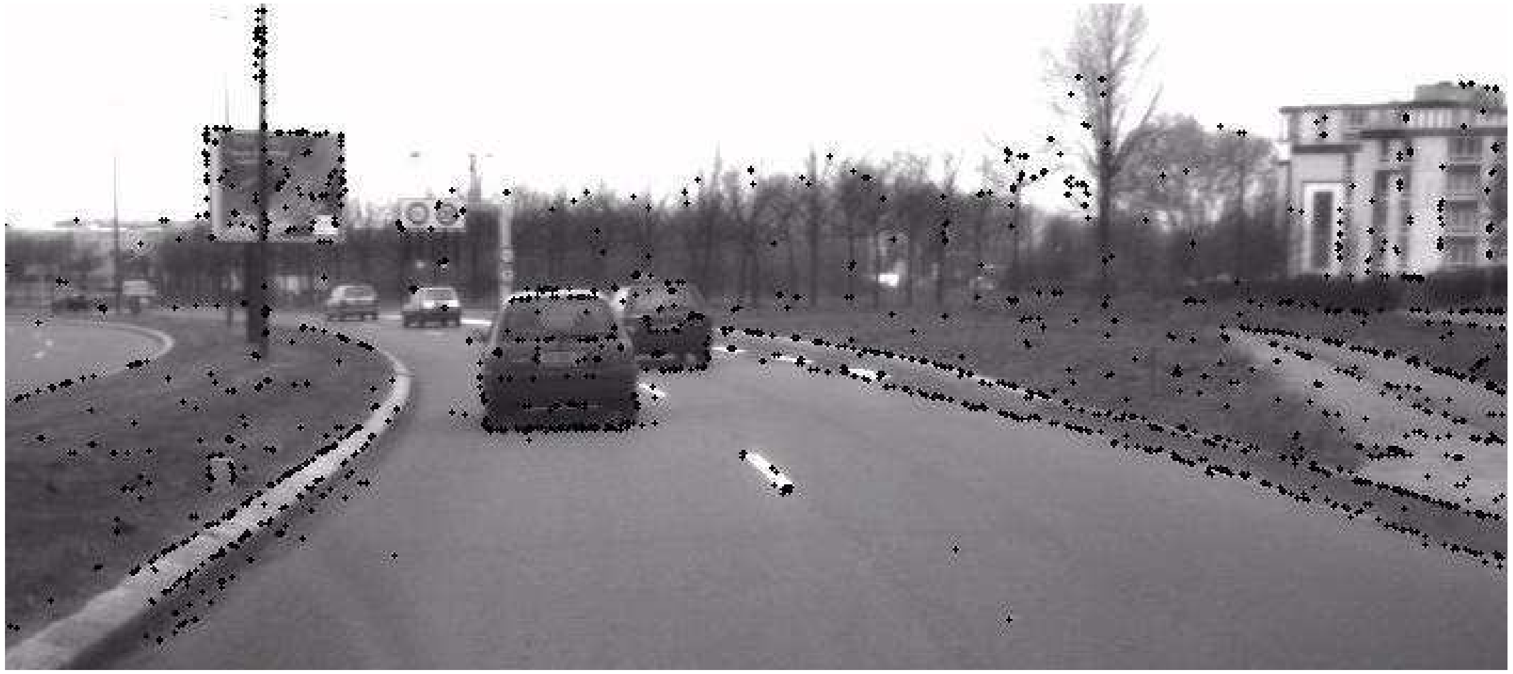

Left image.


Right image.

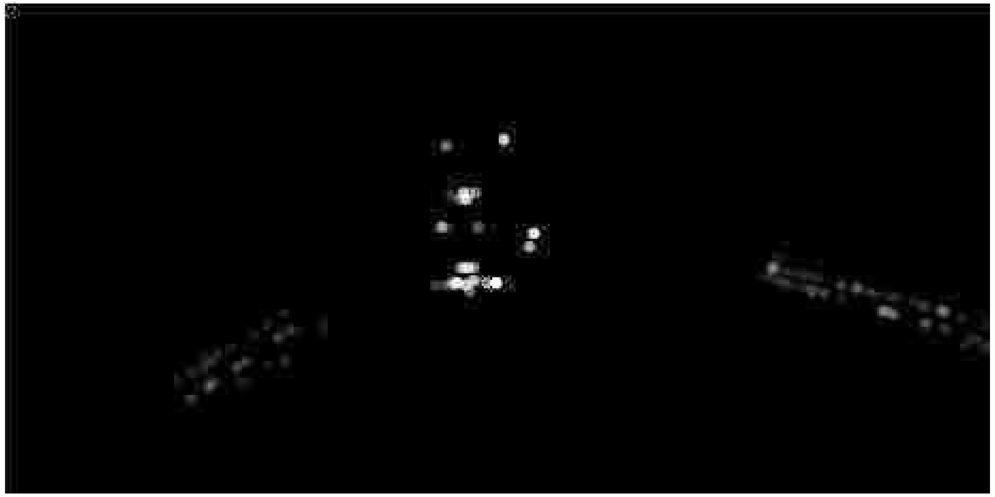


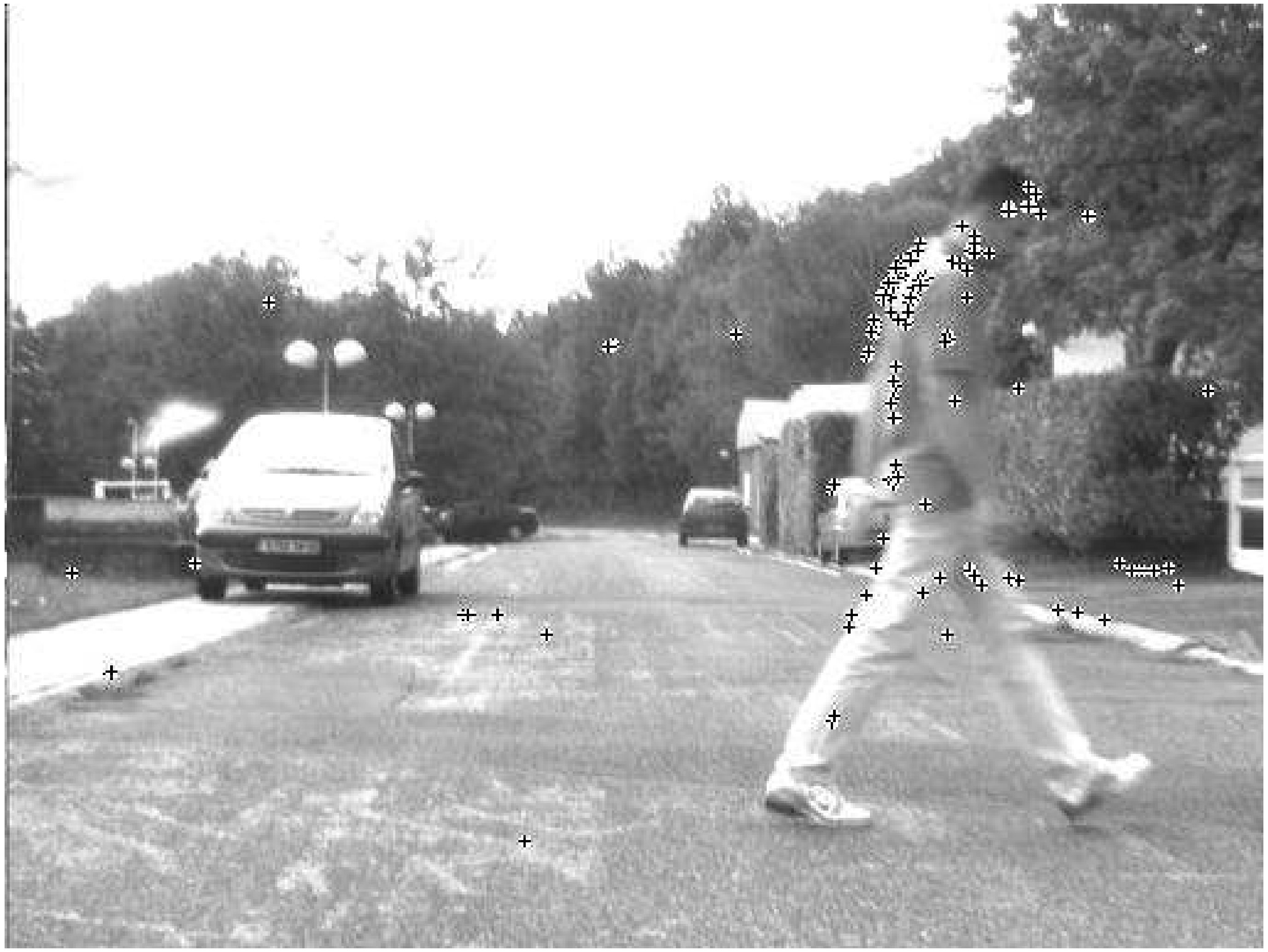
Result, seen from above the scene (image size 384 × 288)

6.2 Introducing time: stereo sequences

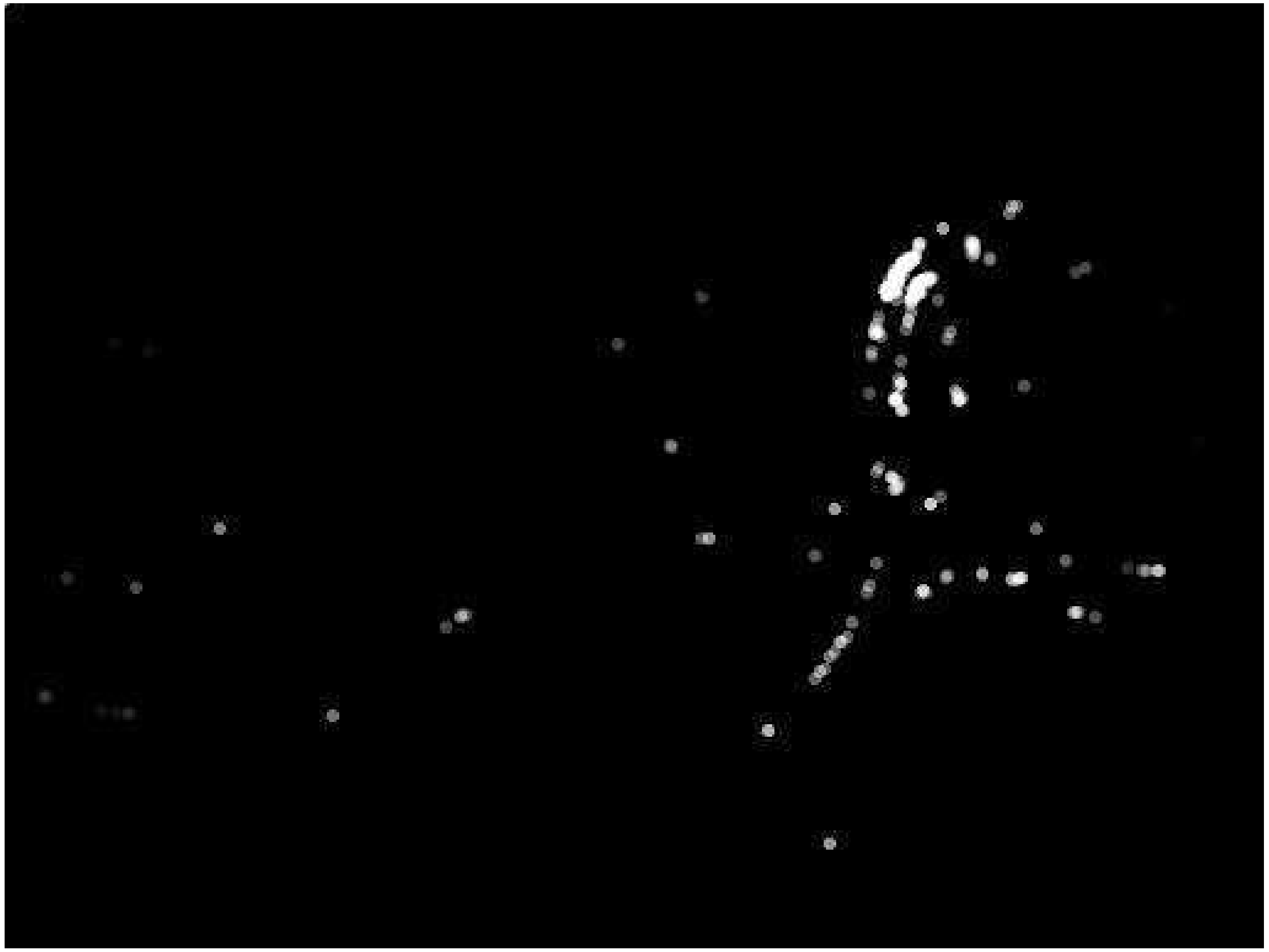


Real-time processing on a highway. The flies (black dots) concentrate on contrasted road edges and other cars.


Alarm values



Pedestrian.

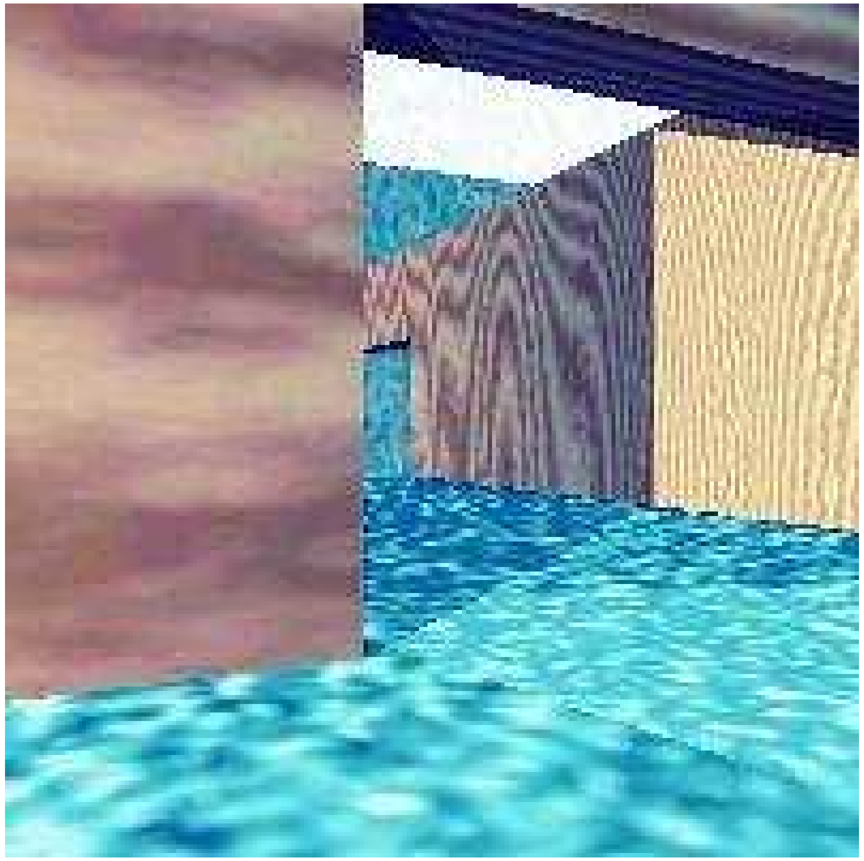


Alarm values (pedestrian)

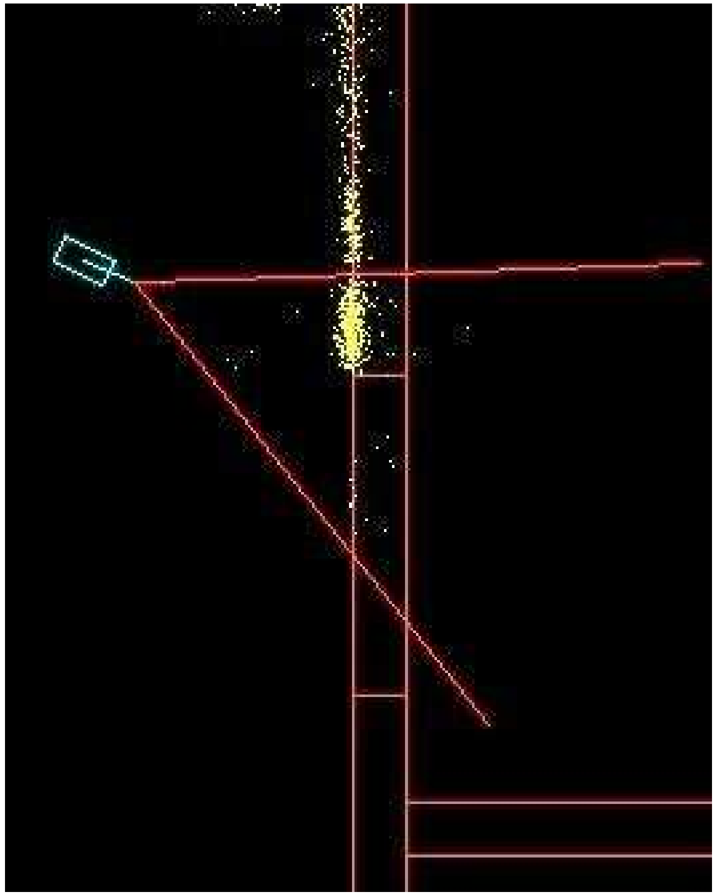
6.3 Flies in Robots: fly-based control

The results shown in this Section have been obtained using the simulator written by A. Boumaza, which simulates all the perception-action loop:

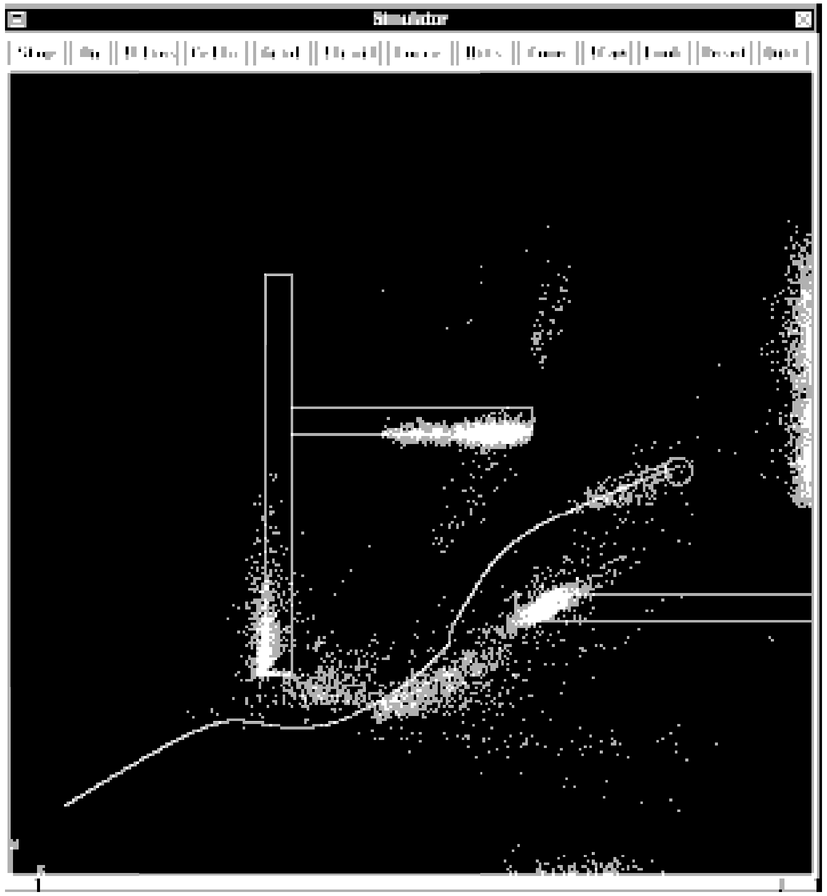
- a stereo camera pair simulator (image synthesis),
- the fly algorithm,
- the trajectory planning algorithm,
- a kinematic simulator of the robot’s motion.



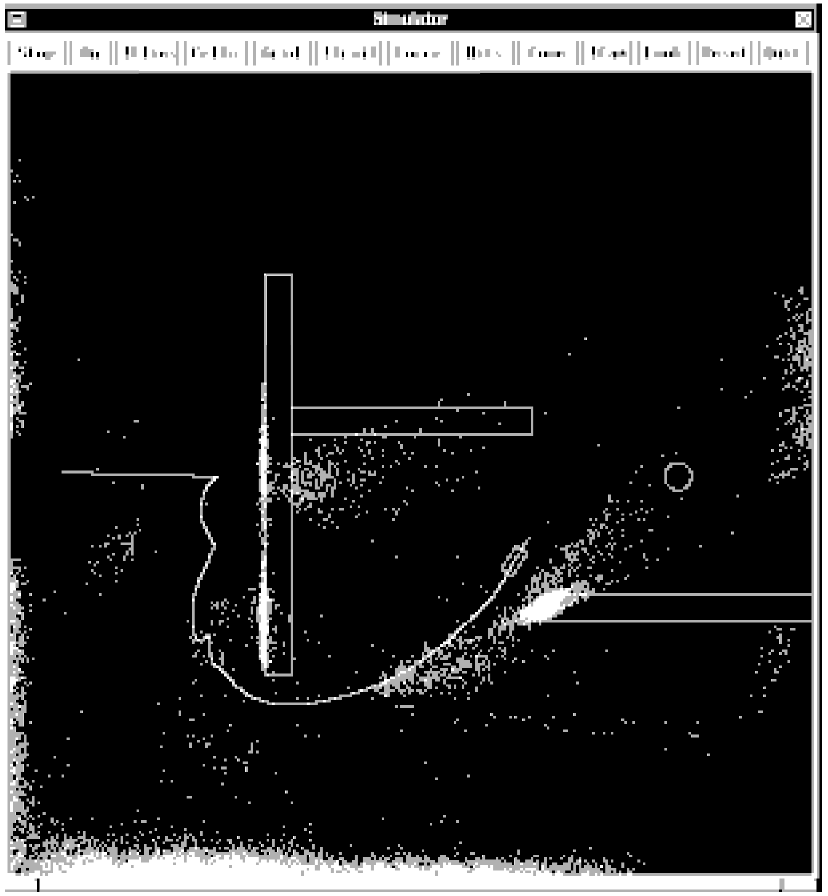
A synthetic image of the scene as seen by one of the robot’s cameras.



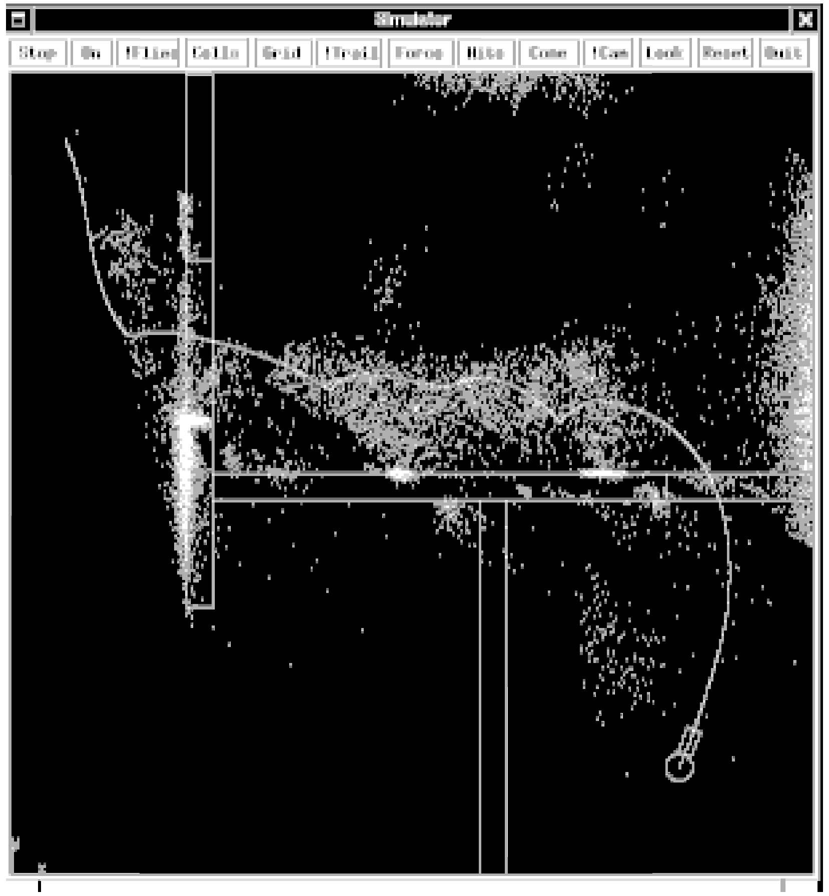
The robot facing a door and a wall. The bright dots represent the flies memorised from the preceding positions.



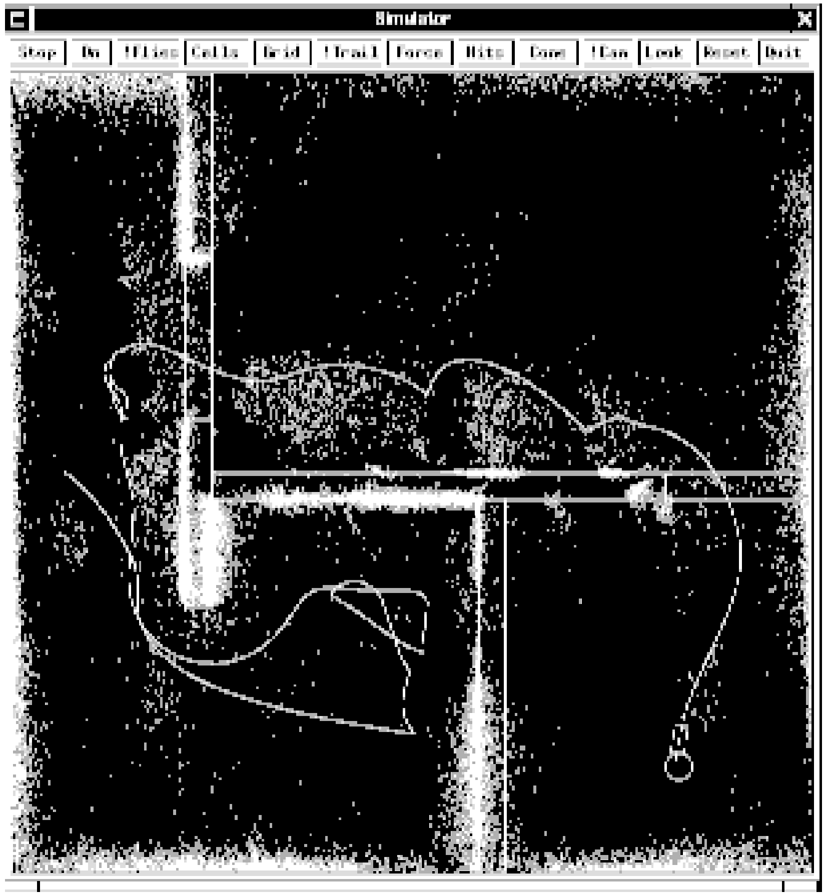
A direct trajectory, without blockage situations.



Obstacle avoidance using secondary targets. The circle represents the main target.



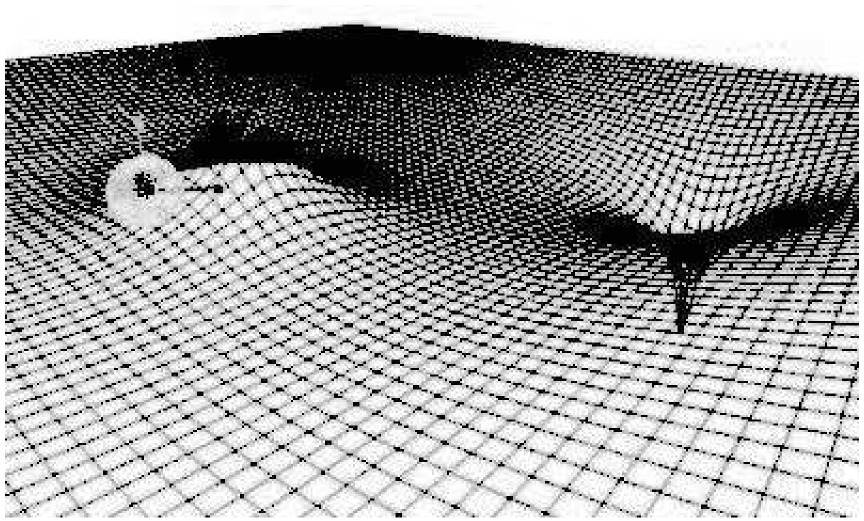
Another example of secondary targets.



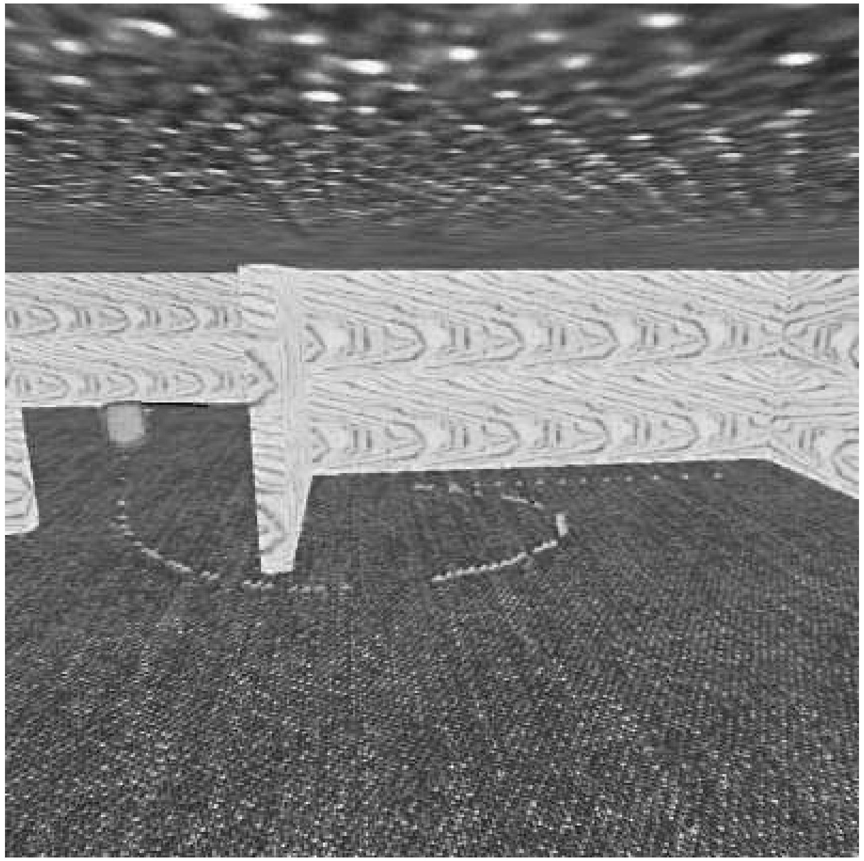
Another example of secondary targets after a difficult blockage.

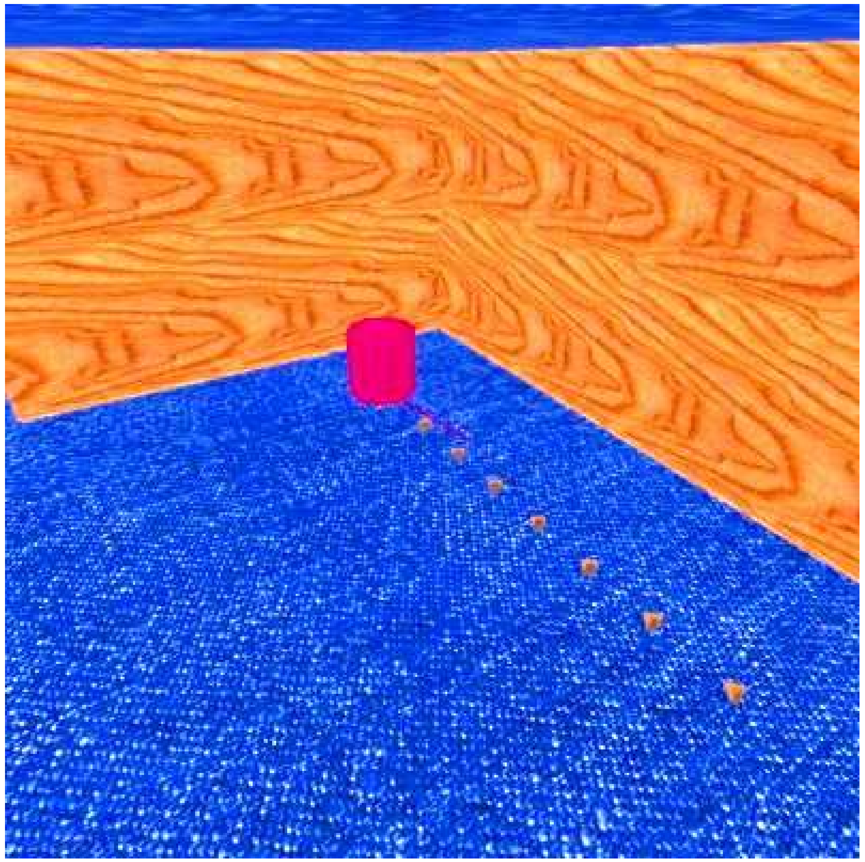
$$\Delta U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0$$

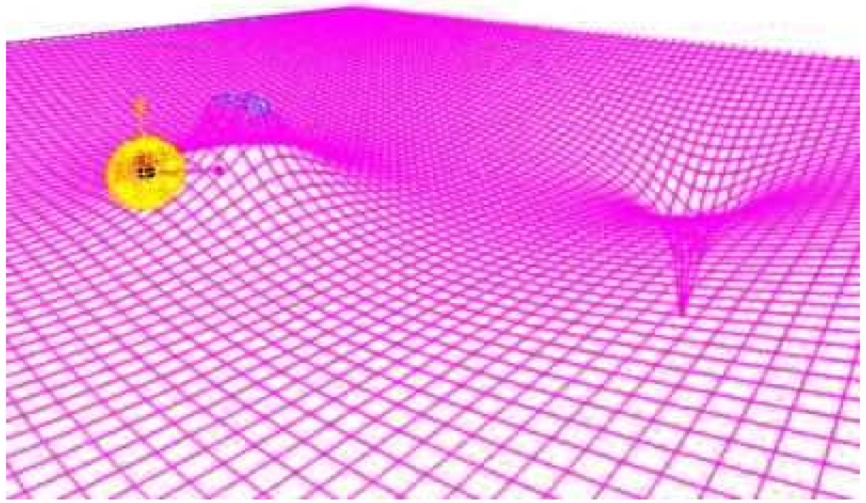
One of the interesting properties of harmonic functions is the absence of local minima, which in our case eliminates the blockage situations found in the first simulator.



A harmonic function used for obstacle avoidance.


Obstacle avoidance using a harmonic function.





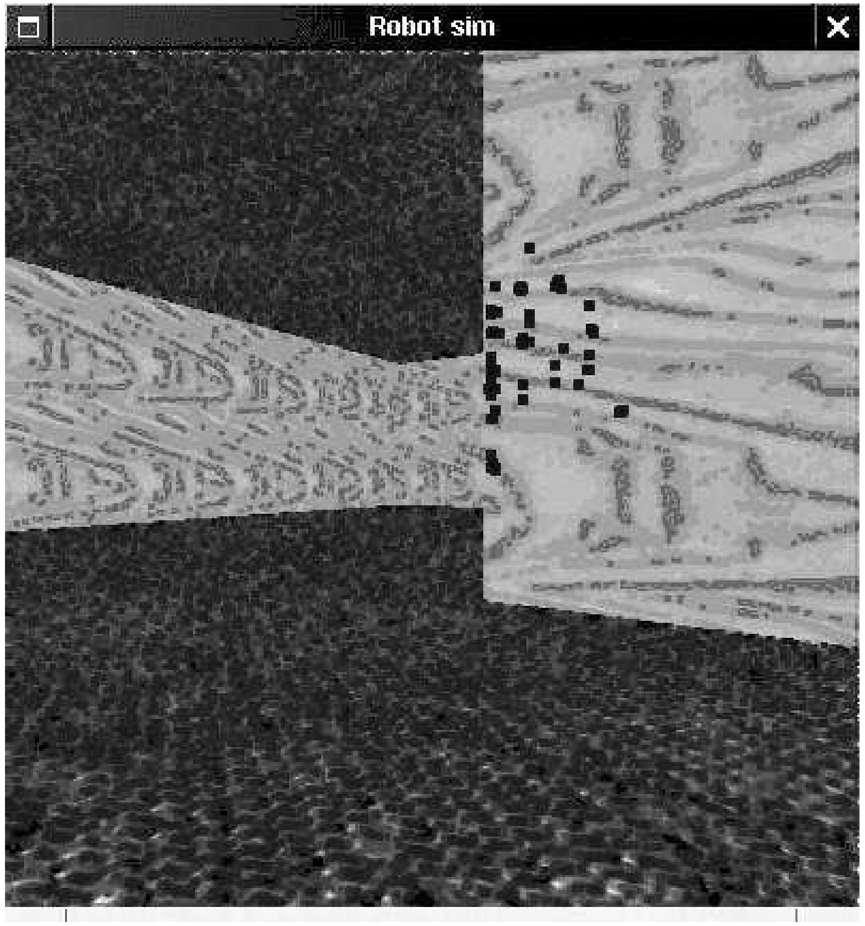
The robot facing an obstacle (left) and the corresponding harmonic function (right).

6.4 Sensor Fusion

Exteroceptive:

$$newfitness = oldfitness(1 + B)$$

Proprioceptive: the proprioception genetic operator
updating flies...



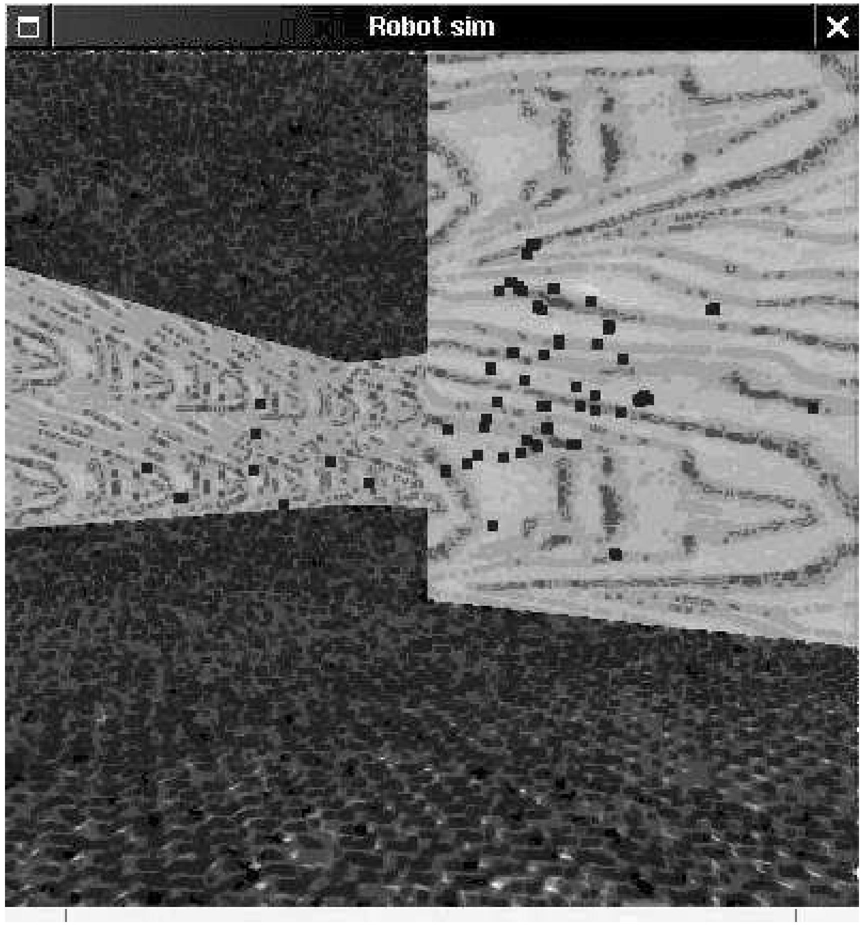


Figure 27. *Frames N and N + 5, without proprioception.*

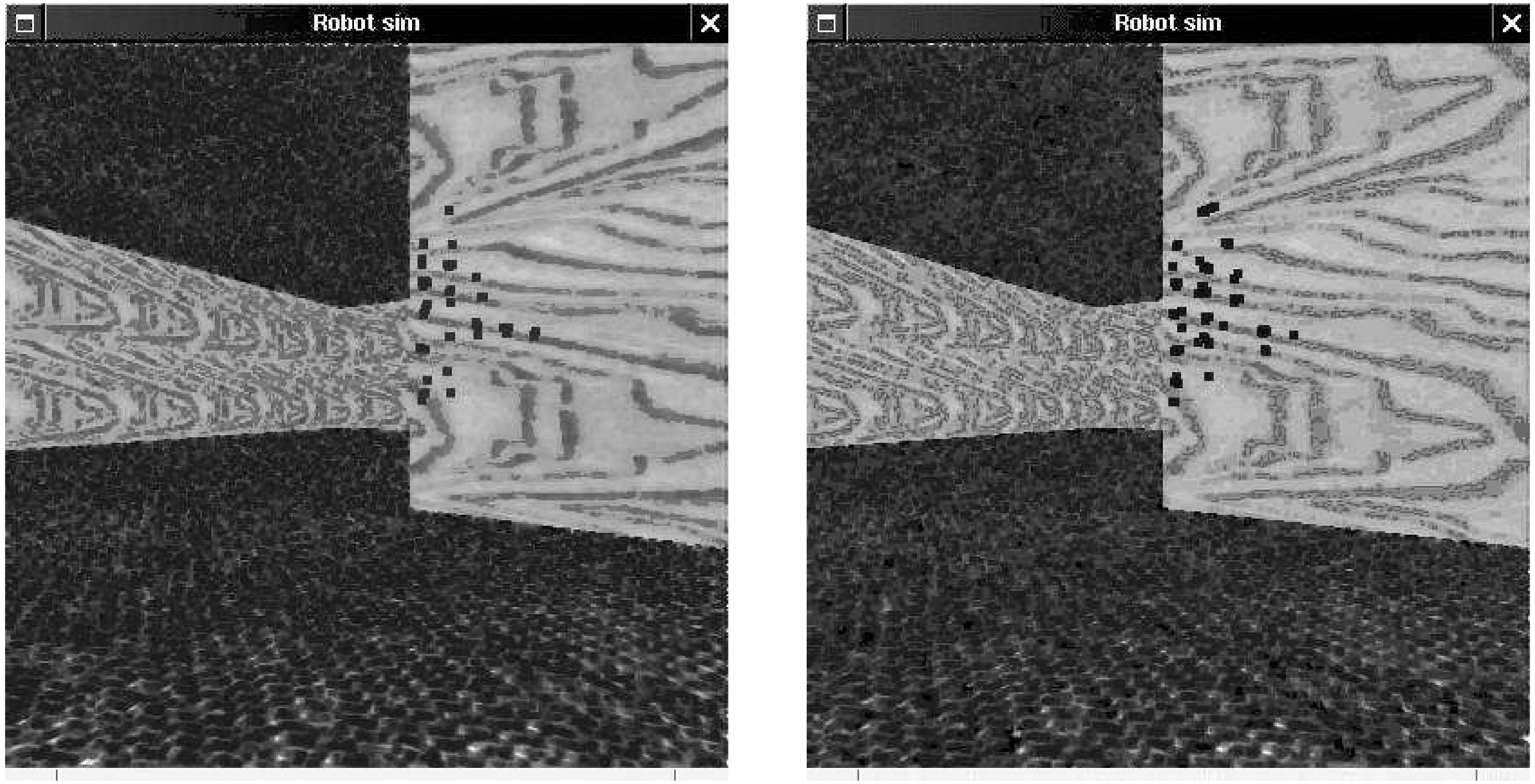


Figure 28. *The same images, with proprioception: better detection of obstacles.*

7 Perspectives

7.1 Real-time

	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

- fast execution, free from the classical delays of frame delivery found in synchronous vision systems,
- easy programming (essentially editing the fitness function) inside a stable algorithmic architecture,
- the process will self-adapt to the apparent velocities of objects in the images,
- ability to exploit and fuse data from other proprioceptive (odometry, inertial) and exteroceptive (radar, acoustic) sensors,
- optimal exploitation of the asynchronous and local properties of state-of-the-art CMOS

imagers.

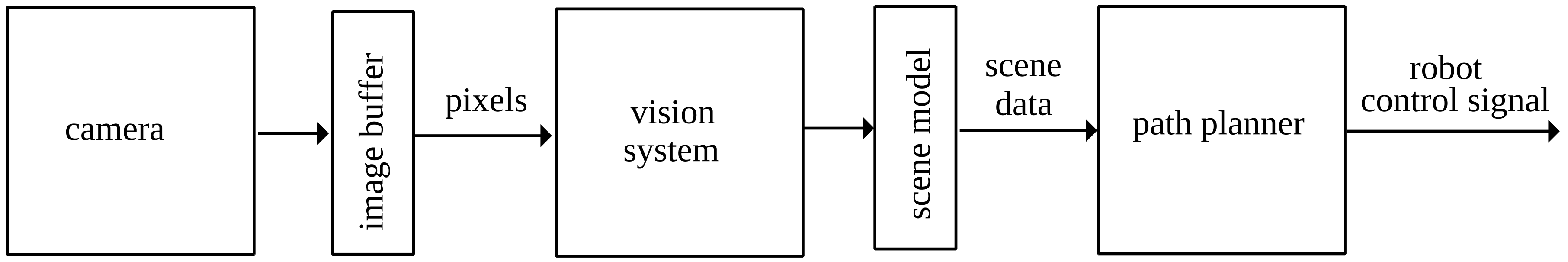
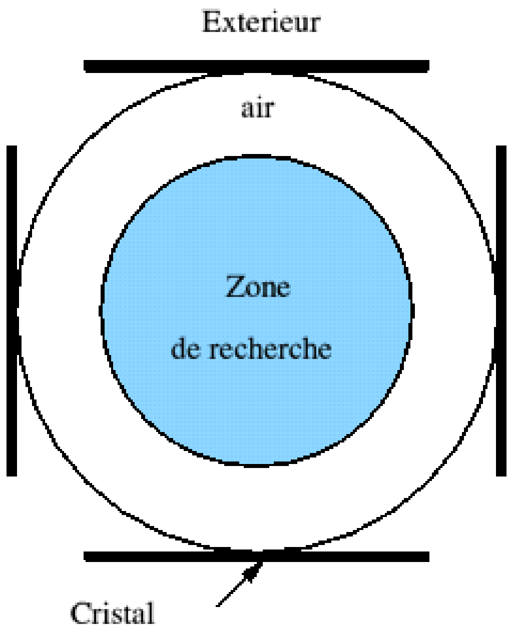
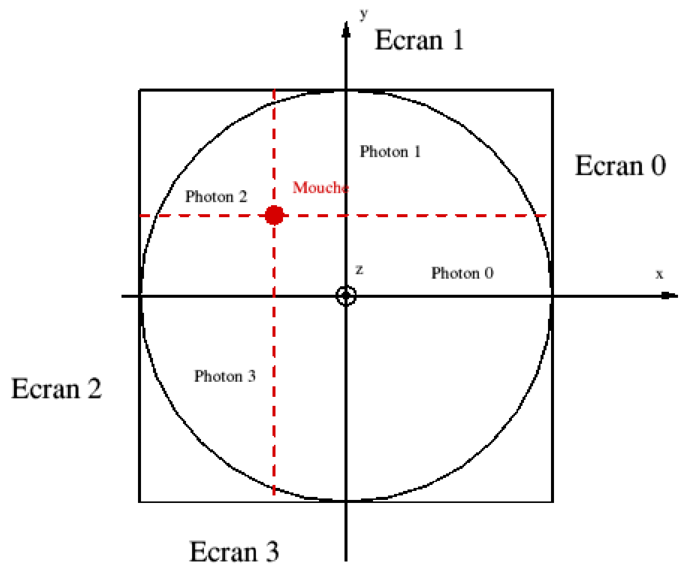


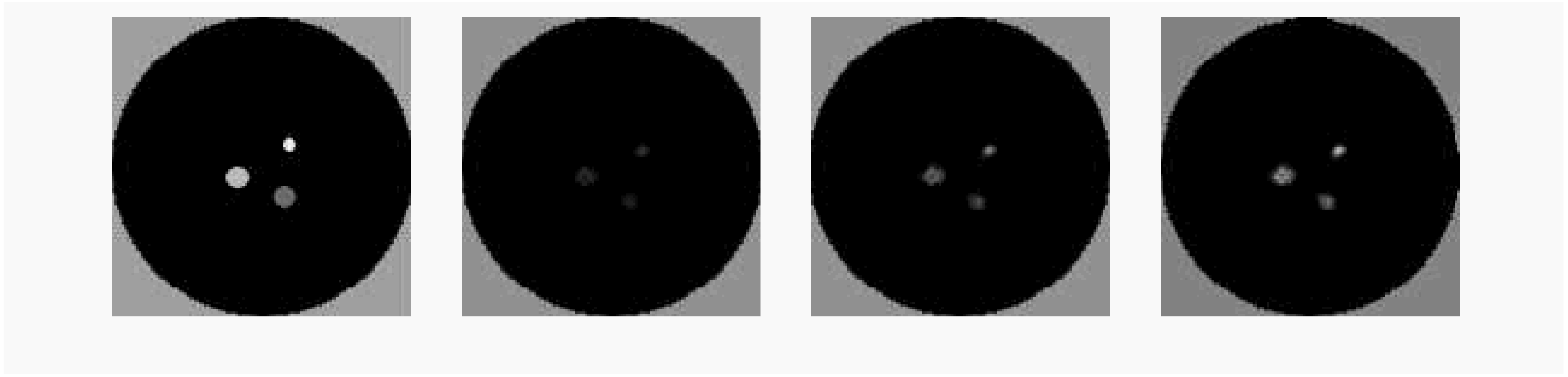
Figure 29. *A classical (synchronous) robot vision-planning system. Defining T as the frame rate period, a classical vision system will use input data which may react to real-world events with a delay up to 2T, and a synchronous planner must wait until complete update of the vision's output to begin its own work. This leads to a minimal delay of 3T in addition to processing time.*

7.2 Medical imaging

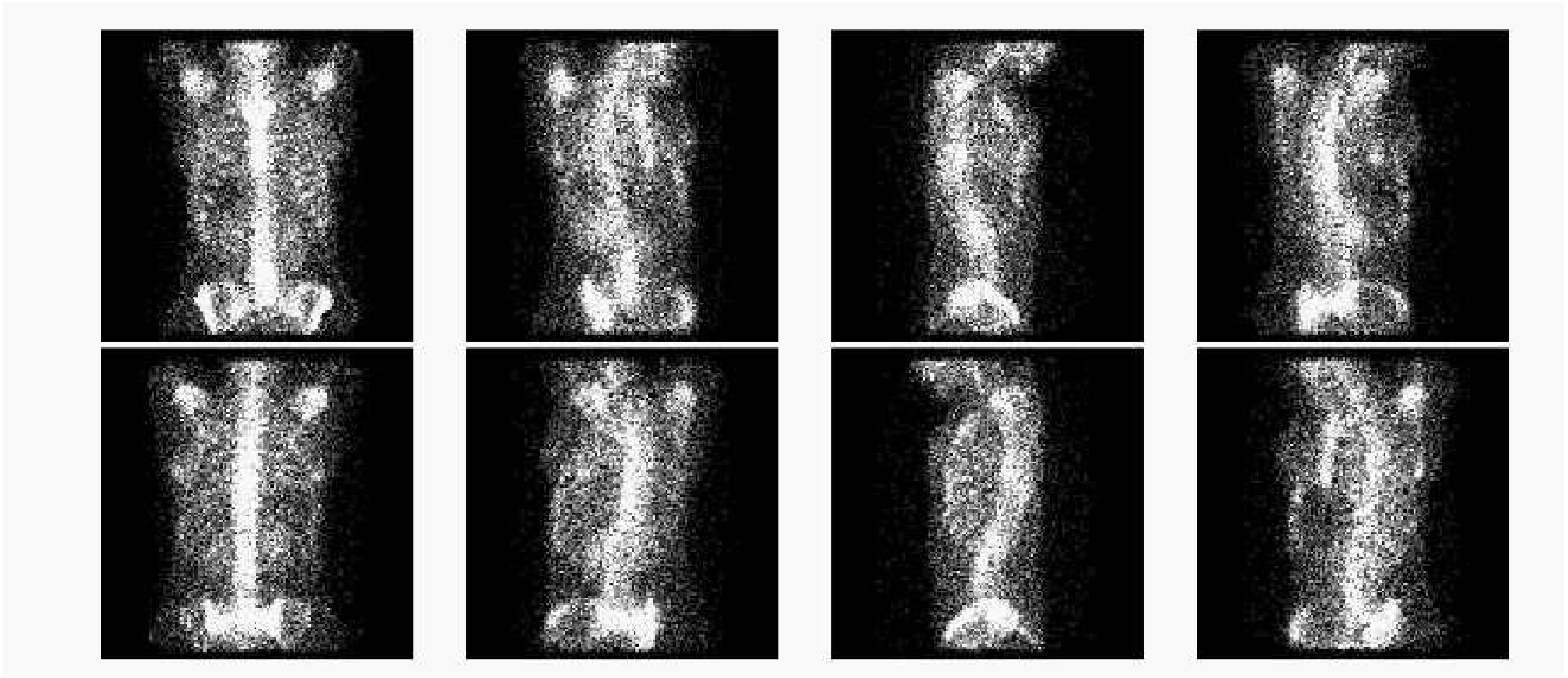
- gamma-camera rotating around the patient
- fly = simulated photon transmitter
- each photon gives a contribution to the fitness of its fly mother
- precalculate random photon trajectories? lose dependence on neighbourhood :-(
- bonus/malus - based fitness $\rightarrow$ loosing the weakest sources :-(
- marginal fitness: $note(i) = note(pop - \{i\}) - note(pop)$:-)


Exterieur
air
Zone
de recherche
Cristal


y
Ecran 1
Photon 1
Ecran 0
Photon 2
Mouche
z
Photon 0
x
Ecran 2
Photon 3
Ecran 3



Reference (left) and recalculated images



Reconstructed images at angles 0, 45 deg, 90 deg., etc.

7.3 Vision prosthesis for the blind

use a tactile graphical display (tactile or vibrating pin array onthe patient's chest
state of the art: 20 pins per inch
Optacon

7.4 Musical analysis: wave2midi

- polyphonic version, single instrument
- time-frequency diagram
- parisian evolution
- one individual = one note
- chromosome: (pitch, time, duration, intensity)
- marginal fitness

laws of Physics / Acoustics $\rightarrow$ mutation and same-time crossover
laws of Harmony $\rightarrow$ sequential crossover

success rates of operators $\rightarrow$ composer’s signature?

8 Conclusion and aknowledgements

We give special thanks to

- Dr. Amine Boumaza (INRIA Lorraine)
- Dr. Pierre Collet (université du Littoral)
- Anders Ekman (Royal Technical Institute, Stockholm)
- Dr. Philippe Guermeur (Ecole Nationale Supérieure de Techniques Avancées)
- Cpt. Baudoin Coppieters de Gibson (Belgian Army)
- Dr. Marie-Jeanne Lesot (Université Paris-6)
- Dr. Bogdan Stanciulescu (Ecole des Mines de Paris)
- Enzo Bolis, Christian Zerbi (Politecnico de Milano)
- Dr. Michel Parent, Olivier Pauplin (PhD student) and the IMARA project
- Dr. Jean-Marie Rocchisani (Hopital Avicenne) and Ms. Aurelie Bousquet (INSA Rouen)

who contributed to the methods, examples and results given in this tutorial.

9 Source code examples

```
/* evolutionary Hough transform */

#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#define WIDTH  1024                    /* max image width*/
#define HEIGHT 1024                    /* max image height */
#define RESERV WIDTH*HEIGHT
#define PI    3.1415926535
#define RADIUS 800
#define MAXPOPULATION 5000
typedef struct {int theta;int ro;int life;float fitness;} fly;
int i,j,k,ic,jc;
int width, height,radius;
float a,b,c,d;
unsigned char entree[15+RESERV],sortie[RESERV],hough[628*
RADIUS],grad[RESERV],contour[RESERV];
int ientree[WIDTH][HEIGHT];
float isortie[WIDTH][HEIGHT];
          /* functions */
void input(void);
void initialise(void);
void process(void);
void output(void);
float randfloat0(void);
float randfloat1(void);
float fitness(int);
          /* application-specific declarations */
int theta,ro;
float SIN[628],COS[628];
float igrad[WIDTH][HEIGHT]; /* gradient norm image */
int THRESHOLD;
int POPULATION,GENERATIONS,PMUT,PCROSS,SAMPLES;
float MUTRO,MUTTHETA;
fly indiv[MAXPOPULATION];
int generation;
int nb_evaluations=0, nb_gradients=0, nb_solutions=0;
float lambda;
float maxfitness;
float exigence;
/************************************************************/
void input()
{
int truc;
FILE *fp;
/* read parameter file under NFS */

  if (fp = fopen("setup","r"))
    {fscanf(fp,"threshold %d\n",&THRESHOLD);
     fscanf(fp,"population %d\n",&POPULATION);
     fscanf(fp,"generations %d\n",&GENERATIONS);
     fscanf(fp,"pmutation %d\n",&PMUT);
     fscanf(fp,"pcrossover %d\n",&PCROSS);
     fscanf(fp,"mutability ro %f\n",&MUTRO);
     fscanf(fp,"mutability theta %f\n",&MUTTHETA);
     fscanf(fp,"samples %d\n",&SAMPLES);
     fscanf(fp,"exigence %f\n",&exigence);
     fclose(fp);
     printf ("threshold = %f\n",THRESHOLD);
     printf ("population = %d\n",POPULATION);
     if (POPULATION > MAXPOPULATION)
       {printf("\n debordement, population trop grande");
        POPULATION =MAXPOPULATION;
        printf ("population = %d\n",POPULATION);
       }
     printf ("number of generations = %d\n",GENERATIONS);
     if (PMUT<0) PMUT=0;  else if(PMUT>100) PMUT=100;
     if (PCROSS<0)PCROSS=0;   else if(PCROSS>100)PCROSS=100;
     printf("prob.mut.=%d\tprob.cross.=%d\n",PMUT,PCROSS);
     printf("mutabilities: ro=%f\t theta=%f\n",MUTRO,MUTTHETA);
     printf("number of samples = %d\n",SAMPLES);
     printf("coeff of exigence = %f\n",exigence);
    }
  else {printf("setup file not found\n"); exit(1);}
/* read image file under NFS */
  if (fp = fopen("entree.pbm","r"))
    {fscanf(fp,"P%d\n",&truc); /* truc = 4, 5 ou 6 */
     fscanf(fp,"%d %d\n",&width,&height);
     fclose (fp);
     if(width*height > WIDTH*HEIGHT) {printf("format image trop grand\n");exit(1);}
     printf("width = %d  height = %d \n", width, height);
    }
  else {printf("file entree.pbm not found\n"); exit(1);}
  radius = height*height+width*width;
  radius = sqrt(radius)/2;
  fp = fopen("entree.pbm","r");
  fread (entree,1,RESERV+15,fp);
  fclose(fp);
  if (truc != 5)
    {printf("\n attention this program needs monochrome P5 pbm images as an input -
abandon\n");
     exit(1);
    }
/* reading and coding parameters */
 for (i=0;i<width;i++)
   for (j=0;j<height;j++)
      ientree[i][j] = (int) entree[15+(i+width*j)];
}
/*********************************************/
```

```
void output()
{
FILE *fp;
/* transform tables into .pbm files */
for (i=0;i<width;i++)  for (j=0;j<height;j++)
   grad[i+width*j] = (unsigned char) (igrad[i][j]);
for (i=0;i<width;i++)  for (j=0;j<height;j++)
   sortie[i+width*j] = (unsigned char) (isortie[i][j]);


/* write output files under NFS */
fp=fopen("grad.pbm","w");
fprintf(fp,"P5\n%d %d\n255\n",width,height);
fwrite(grad,1,width*height,fp);
fclose(fp);
fp=fopen("sortie.pbm","w");
fprintf(fp,"P5\n%d %d\n255\n",width,height);
fwrite(sortie,1,width*height,fp);
fclose(fp);
}
/*******************************************/
```

```
void initialise()
{
/* tabulate trigonometric functions */
for(theta=0;theta<628;theta++)
 {
  COS[theta]=cos(theta/100.);
  SIN[theta]=sin(theta/100.);
 }
/* initialise evolution strategy */
for(i=0;i<POPULATION;i++)
  {
   indiv[i].theta=628*randfloat0();
   indiv[i].ro=radius*randfloat0();
   indiv[i].fitness=0;
   indiv[i].life=1;
                      /* life = 0 alive evaluated
                                1 alive not evaluated
                                2 dead to replace by mutation
                                3 dead to replace by crossover
                      */
  }
maxfitness=0.;
}
/********************************************/
```

```
void process()
{
 for (i=1;i<width-1;i++) for (j=1;j<height-1;j++)
   {
    a = abs(ientree[i][j+1] - ientree[i][j-1]);
    c = abs(ientree[i+1][j] - ientree[i-1][j]);
    igrad[i][j] = 2*(a+c);
   }
printf("\ngradient calculated");
maxfitness=0;
for(generation=0;generation<GENERATIONS;generation++)
  {for(i=0;i<POPULATION;i++)                              /* updating fitness */
      if(indiv[i].life == 1) {indiv[i].fitness=fitness(i);indiv[i].life=0;}
   printf("\nmaxfitness %5.0f",maxfitness);
                                                      /* selection tournaments */
   for(k=0;k<(POPULATION*PMUT)/100;k++)
     {
      i=POPULATION*randfloat0(); j=POPULATION*randfloat0();
      if(indiv[i].fitness<indiv[j].fitness) indiv[i].life=2;else indiv[j].life=2;
                                                        /* kill the worst of two */
     }
   for(k=0;k<(POPULATION*PCROSS)/100;k++)
     {
      i=POPULATION*randfloat0(); j=POPULATION*randfloat0();
      if(indiv[i].fitness<indiv[j].fitness) indiv[i].life=3;else indiv[j].life=3;
     }
```

```
          /* activation of genetic operators */
   for(i=0;i<POPULATION;i++)
    {if(indiv[i].life==2)        /* individual to be replaced through mutation */
      {
       j=POPULATION*randfloat0();                          /* choice of parent j */
       a=abs(indiv[j].ro+MUTRO*randfloat1());
       if(a<1)a=1;else if(a>radius)a=radius-1;
       indiv[i].ro=a;
       a=indiv[j].theta+MUTTHETA*randfloat1();
       if(a<0) a+=628; else if (a>628) a -= 628;
       indiv[i].theta=a;
       indiv[i].life=1;
      }
     else if (indiv[i].life==3)        /*individual to be replace by crossover */
      {
       j=POPULATION*randfloat0();
       k=POPULATION*randfloat0();              /* choice of 2 parents j and k */
       lambda=-.5+2*randfloat0();
       a=abs(lambda*indiv[j].ro+(1-lambda)*indiv[k].ro);
       if(a<1)a=1;else if(a>radius)a=radius-1;
       indiv[i].ro=a;
       a=lambda*indiv[j].theta+(1-lambda)*indiv[k].theta;
       if(a<0) a+=628; else if (a>628) a -= 628;
       indiv[i].theta=a;
       indiv[i].life=1;
      }
    }
 }
```

```
/* now a population, what to do with it?  */
for (i=1;i<width-1;i++) for (j=1;j<height-1;j++) isortie[i][j]=ientree[i][j]/2;
for(k=0;k<POPULATION;k++)
 if(indiv[k].fitness > exigence * maxfitness )
  {
   nb_solutions ++;
printf("\ntheta=%d,ro=%d,quality=%f",indiv[k].theta,indiv[k].ro,indiv[k].fitness/
maxfitness);
   a=COS[indiv[k].theta];b=SIN[indiv[k].theta];
   for (i=1;i<width-1;i++) for (j=1;j<height-1;j++)
    {ic=i-width/2;jc=j-height/2;    /* centred coordinates */
     if (abs(indiv[k].ro-ic*a-jc*b)<.5) isortie[i][j]=255;
    }
  }
 if(nb_solutions<2)
for(k=0;k<POPULATION;k++)
 if(indiv[k].fitness > exigence * exigence*maxfitness )
  {printf("\n one solution, theta=%d,ro=%d,quality=
%f",indiv[k].theta,indiv[k].ro,indiv[k].fitness/maxfitness);
   a=COS[indiv[k].theta];b=SIN[indiv[k].theta];
   for (i=1;i<width-1;i++) for (j=1;j<height-1;j++)
    {ic=i-width/2;jc=j-height/2;    /* centred coordinates */
     if (abs(indiv[k].ro-ic*a-jc*b)<.5) isortie[i][j]=255;
    }
  }
   printf("\nEnded: %d solutions, %d evaluations, %d gradients calculated
\n\n",nb_solutions,nb_evaluations,nb_gradients);
}
```

```
/*****************************************/
float fitness(int i)
{
 float mu;int x,y,xc,yc;
 c=0;
 theta=indiv[i].theta;
 a=COS[theta];b=SIN[theta];
 ro=indiv[i].ro;
 for(j=0;j<SAMPLES;j++)
  {
   mu=radius*randfloat1();

   xc=ro*a+mu*b;
   yc=ro*b-mu*a;
   x=xc+width/2;
   y=yc+height/2;
   if((x>0)&&(y>0)&&(x<width)&&(y<height))
     {nb_gradients++;
      if(igrad[x][y]>THRESHOLD) c += igrad[x][y];
     }
  }
if(c>maxfitness)maxfitness=c;
nb_evaluations ++;
return c;
}
/*****************************************/
```

```
float randfloat0(void)
{
  float alea, numerator, denominator;
  numerator = rand() & 0x7fff;
  denominator = 0x7fff;
  alea = numerator/denominator;
  return alea;  /* equireparti sur [0,1] */
}
/*******************************************/
float randfloat1(void)
{
  float alea, numerator, denominator;
  numerator = rand() & 0x7fff;
  denominator = 0x7fff;
  alea = 2. *(numerator/denominator) - 1.;
  alea = alea * (.2 + .8 * alea * alea ) ;
  return alea;  /* reparti sur [-1,1], concentre pres de 0 */
}
/*******************************************/
main()
{
input();
initialise();
process();
output();
}
```