Proceedings of the First Swiss Symposium on Pattern Recognition and Computer Vision, Lausanne, Switzerland, January 30-31, 1992

Rev fe! 92

Two vision-based behaviours for autonomous mobile robots

Claudio Facchinetti, Heinz Hügli.
Institut de Microtechnique
Rue A.L.-Breguet 2, CH-2000 Neuchâtel

Abstract

This paper is a contribution to the behavioural approach to navigation in robotics. The behavioural rather than functional decomposition of the robot task allows an easier development of separate modules because of their higher degree of mutual independence and therefore offers simpler system design. This paper presents two vision-based behaviours as examples of typical behaviours for the robot navigation task. Named *going toward* and *going to,* the two behaviours make use of vision to move the robot toward visible landmarks. Each of it represents a feedback control loop across actuators, the robot environment, the vision system and the controller. Each is described to some extent both in its principle and implementation.

1. Introduction

The behavioural approach for robot navigation is inspired by the animal world in which elementary behaviours can be observed. A behaviour is an action activated by a fixed stimulus and maintained as long as this one exists. A simple example is given by a bee: despite a poor visual system (the resolution is approximately 1/60 in comparison with the human one) it easily navigates to and from its hive. In fig. 1.1 the path of the bee could be, for example, to follow the fence and, when the hive is near enough, to go toward it. These two actions are basic behaviours which stimuli are the fence for the first one and the hive for the second one. Other behaviours based on vision can be inspired by this simple example: avoid obstacle, turn angle, ... [5][6].

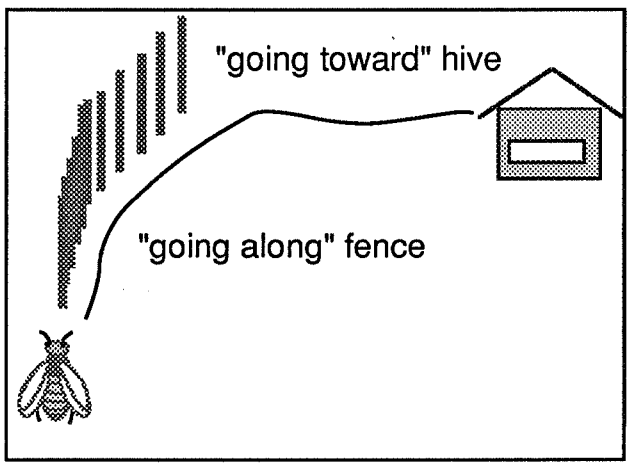

fig.1.1: bee navigation.

The behaviours form a set of partial navigation solutions, each one related to a specific input data.
In the reminder of this paper we present the design and implementation of two vision-based behaviours, namely *going toward* and *going to.*

2. Mobile robot and behavioural navigation

Behaviours as independent functions

Robot navigation is decomposed into behaviours, with interaction between them, rather than into functions. A direct advantage of such a solution is the development of the behaviours as independent units. Apart from some interaction constraints, the developer does not have to care about the nature of other behaviours. Each of these is a separate process which activation merely depends on the presence or absence of the stimulus. One can divide the system in hierarchical levels:

0: low-level functions: actuators (robot) and sensors.
1: modules: move, rotate (robot), landmark-following (vision).
2: behaviours: going toward, going along (vision).
3: behaviour management in terms of goals to be reached.

Note that inside level 2 itself, behaviours can be composed of other behaviours.
A behaviour uses all the inferior levels. The *going toward* behaviour, for example,

supposes a vision system capable of following a landmark in the scene. It will also use the low-level functions controlling the actuators, in order to regulate the robot movements with the observed position of the landmark.

Behaviours and vision

There is a distinction between an external and an internal behaviour. The first one (fig.2.1.i) performs an action within a feedback loop across the robot environment, while the second updates the robot knowledge from sensor data (fig.2.1.ii).

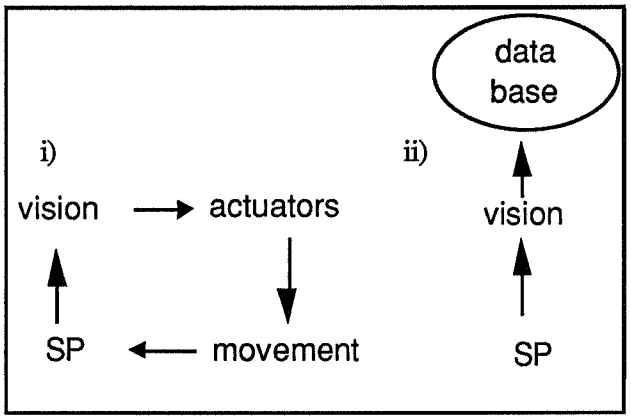

fig.2.1: a behaviour can be either i) external or ii) internal.

In our case the behaviour is based on vision. The stimulus is a 0-dimensional visual primitive representing a landmark in the scene. The scene is observed by a camera. The landmark becomes then a *particle* on the camera image, which mass center defines the sign pattern (SP).
The behaviour action consists of moving the robot by controlling its actuators. The vision system is used to regulate the robot movements with the particle position on the camera image.

Robot model and scene

The environment of the robot is a scene composed by a planar ground, walls and different obstacles (see fig.2.2). The robot is vertical and its movements are horizontal. The camera is fixed on the top of the robot and looking forward. Landmarks are spread in the scene.

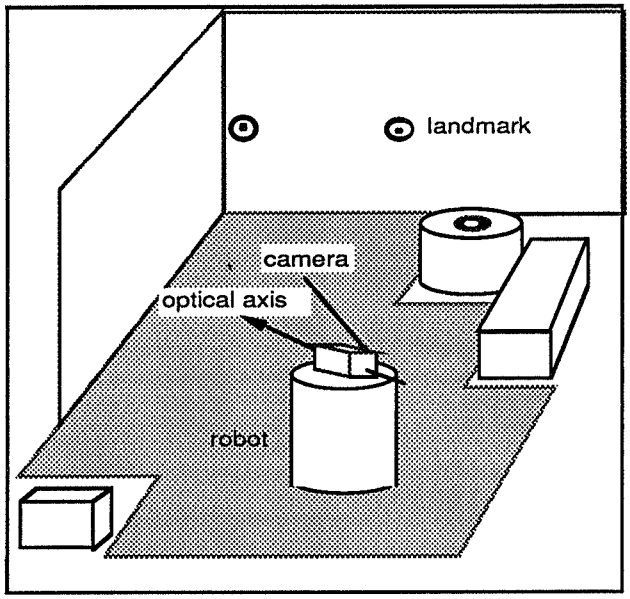

fig.2.2: robot environment.

The robot actuators execute the rotation and forward commands within error margins, that are estimated to be 1% of the command value.

3. Two vision-based behaviours

Going toward behaviour

Our first basic vision-based behaviour is external: it moves the robot in the direction of a landmark placed in front of it. The robot path will be a kind of zigzag line, centred on the axis going from start position (when the behaviour begins) to the landmark.

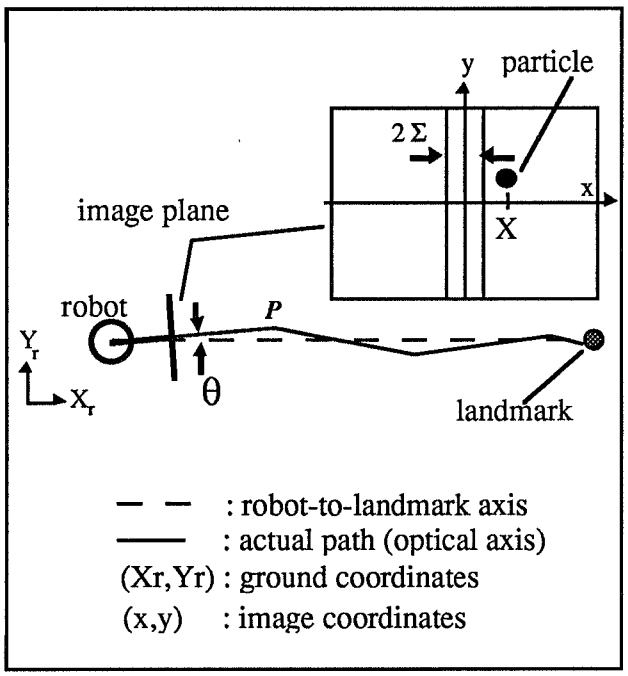

fig.3.1: Robot movements are performed within some error margins, resulting in a zigzag path.

Looking at fig.3.1 one can see that at point *P*, the particle is outside an Σ-margin defined around the image center. The step after *P* will be a rotation. Note that the x axis (image plane) is parallel to the ground.
The optical axis determines the direction in which the robot will move, whereas the robot-to-landmark axis determines the path the robot should follow. The angle between the two axis is θ. When θ is 0, these are superimposed and the robot is aligned on the landmark. In fact, the robot is supposed aligned when θ is smaller than (or equal to) a margin angle:

$$\text{aligned when } |\theta| \leq \theta_\Sigma$$

On the camera image, the robot-to-landmark axis intersects the image plane at the particle mass center, whereas the optical axis intersects the image plane at its origin.
At this points one can simplify the problem knowing that the robot is moving in a planar environment with a fixed camera, resulting in only horizontal movements of the particle on the image. We will further only consider the horizontal projection of the particle, described by *X*. The margin angle corresponds to Σ on the image. We have

$$\tan(\theta) = \alpha \cdot X$$

which gives a relation between the observed position of the particle and the angle between the optical axis and the robot-to target axis.
The aligned condition becomes then

$$\text{aligned when } |X| \leq \Sigma.$$

Thus, the *going toward* behaviour uses rotations and forward commands. These are performed by the module that controls the actuators, within some error margins. The optical axis will not coincide with the robot-landmark axis after the robot has advanced some distance, requiring a re-alignment on the landmark. This is done by rotating the robot, so that X falls again into the Σ-margins.
The alignment problem is solved by doing a recursive correction of the rotation angle commands sent to the robot (see fig.3.2). The result is a sequence of rotations more and more accurate.

$\theta_0 := \arctan(\alpha_0 \cdot X_0)$;
while X_i *outside Σ-margins* do
  if $(X_{i-1} - X_i) \neq 0$ then
    $\alpha_i := \tan(\theta_i)/(X_{i-1} - X_i)$;
  $\theta_i := \arctan(\alpha_i \cdot X_i)$;
  rotate(θ_i);

where: X_i : horizontal projection of the particle.
θ_i : angle for the robot to be rotated.
α_i : recursive coefficient.

fig.3.2: alignment algorithm.

The value α_0 is obtained by

$$\alpha_0 = \frac{1}{f}$$

where the focal length *f* is roughly approximating by some calibration measurements.

Supposing the landmark is visible (the stimulus is active), the *going toward* algorithm looks like:

❶ Align robot on particle;
❷ Loop.
  ① Loop while X is inside Σ-margins.
    ① Move forward;
(the robot moves in the landmark direction).
  ② Align robot on landmark;

Going to behaviour

Our second vision-based behaviour is an extension of the *going toward* behaviour, that uses triangulation to estimate the distance to the landmark and stops its movement when

$$d < d_0$$

where d : current robot to landmark distance.
d_0 : distance for stopping.

The distance is given by a triangulation-based method needing two observations of the same landmark. This can be done by performing a parallaxial movement with respect to the landmark (see fig.3.4) [1][3].

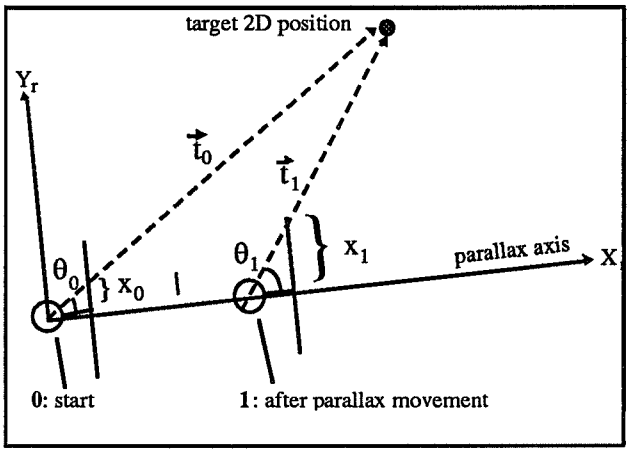

fig.3.4: parallax movement and distance estimation.

The parallax movement is performed along an axis, which describes an angle θ_0 with the robot-to-landmark axis. The travelled distance is l (measured by the robot odometers). At point *1* (after parallax movement) two image positions, X_0 and X_1, are known and determine the landmark position $\underline{t}_0$ in the ground plane:

$$\underline{t}_0 = l \cdot \begin{pmatrix} \frac{X_1}{X_1 - X_0} \\ \frac{X_1 \cdot \tan\theta_0}{X_1 - X_0} \end{pmatrix}$$

$$\underline{t}_1 = \underline{l} - \underline{t}_0 \; ; \; \underline{l} = \begin{pmatrix} l \\ 0 \end{pmatrix}$$

Once $d = \| \underline{t}_1 \|$ is known, the robot is in the same situation as the initialisation phase of the *going toward* behaviour. The continuation is identical, but with a supplementary criterion in the algorithm that will stop the robot. The criterion uses odometry to decrement the computed distance to the landmark till $d<d_0$.

Supposing the landmark is visible (the stimulus is active), the *going to* algorithm looks like:

❶ Align robot on landmark;
❷ Rotate θ_0;
❸ Move forward l (parallax movement);
❹ Compute d;
❺ Align robot on landmark.
❻ Loop <u>while d>do</u> .
① Loop while X is inside Σ-margins <u>and d>do</u>.
① Move forward;
② Get robot position and update d;
(the robot moves in the landmark direction).
② Align robot on landmark;

4. Vision system

Camera correction

The model usually used to describe the geometry of a camera is the pinhole model [1][3]. Unfortunately, this model does not take into account the lens distortion. The latter can be very significant, especially in the case of small f-number lenses [2].
We use here a fisheye lens in order to have a wide vision angle and to reduce the sensitivity to shakes of the robot.
The top of fig.3.3 shows the distorted calibration grid as it is produced by the fisheye lens. Camera correction transforms the observed position $\underline{u}_d$ into corrected position $\underline{u}_c$ that best fits the camera pinhole model.

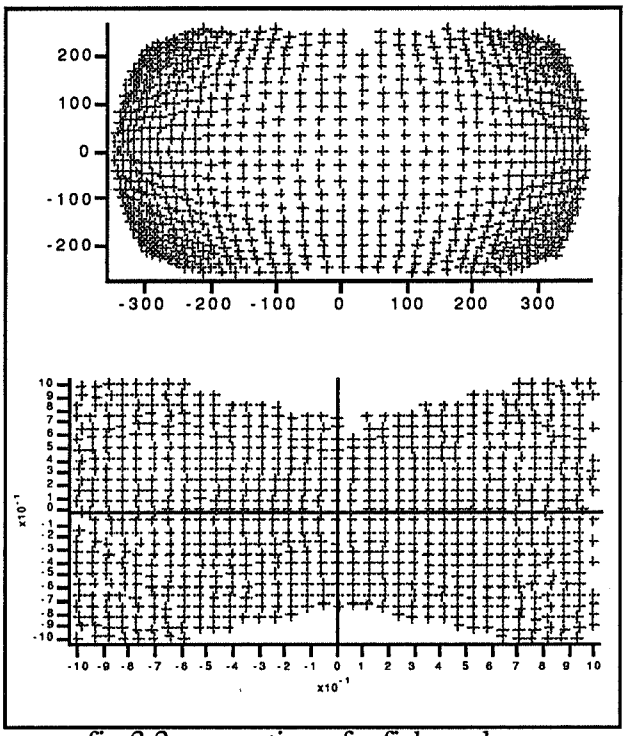

fig.3.3: correction of a fisheye lens: distorted (top) and corrected (bottom) image of a calibration grid.

The distortion characterization is inspired by the Brown-lines method [3], which principle is to use a grid composed of a mixture of straight lines and to correct the curved lines (issued from the camera image) in a least-square sense, so that they fit straight lines again. The grid is planar and parallel to the image plane.
We have restricted this model from general lines to horizontal lines (y=cst), so that we can easily establish a correspondence between the points on the grid lines and the points on the image (curved) line, within a scale factor.

The relation between a distorted point with

coordinates $\underline{u}_d$ and the corrected one $\underline{u}_u$ is:

$$\underline{u}_u = F(\underline{u}_d) + \underline{\varepsilon} \qquad ; F: \Re^2 \Rightarrow \Re^2$$
$$F = F_a * F_s$$

where F_a : asymmetrical correction
F_s : symmetrical correction
$\underline{\varepsilon}$: random error

The radial distortion can be represented by an odd polynomial (Franke [7]):

$$\Delta r = m_1 \cdot r^3 + m_2 \cdot r^5 + \ldots$$

By separating Δr on the u and v axes and keeping only the terms in O^2 et O^4, the symmetrical correction can be written as:

$$F_s(\underline{u}_d) = \underline{u} = \underline{q}.\underline{u}_d + \Delta\underline{u}_s$$

where $\Delta u_s = (u_d - u_o)\cdot(m_1 r^2 + m_2 r^4)$
$\Delta v_s = (v_d - v_o)\cdot(m_3 r^2 + m_4 r^4)$.
$\underline{q} = (q_u, q_v)^t$ (normalization vector)

The solution for the reduction of the asymmetrical distortion proposed by Kyle and Loser [8] handles many errors sources like anamorphism, Keystone effect, sensor rotation or magnification:

$$\Delta u_a = c_1 \cdot u + c_2 \cdot u^2 + c_3 \cdot u^3 + c_4 \cdot u^2 \cdot v + c_5 \cdot u \cdot v^2 + c_6 \cdot v^2 + c_7 \cdot v + c_8 \cdot u \cdot v$$
$$\Delta v_a = c_9 \cdot v + c_{10} \cdot v^2 + c_{11} \cdot v^3 + c_{12} \cdot v^2 \cdot u + c_{13} \cdot v \cdot u^2 + c_{14} \cdot u^2 + c_{15} \cdot u + c_{16} \cdot v \cdot u$$

The c_i coefficients contain the distortion parameters. The asymmetrical function is then

$$F_a(\underline{u}) = \Delta\underline{u}_a$$

The correction functions F_s and F_a are then calibrated by establishing a point to point correspondence between the distorted coordinates $<u_d, v_d>$ and the reference coordinates $<u_u, v_u>$. For each correction function we need a minimum of n/2 correspondences to resolve the linear equations system (where n is the number of coefficients to calibrate).
Generally the number of correspondences is greater than the minimum, producing an overdetermined system. Solution of such a system can be found in a least square sense by using the pseudo-inverse matrix method [1].

Image acquisition

The landmarks in the 3D scene are of highly retro-reflecting material and illuminated by a fluorescent tube mounted on the camera. This allows to easily separate the well-contrasted landmarks from the background by thresholding the image. This thresholding is done in an input look-up table in real-time. Then the image is stored in a frame-buffer that can be accessed by the host computer.
The available image is describe by a binary function g(x,y):

$$g \in \{0,1\} \; ; \; x = 1..N_x \; ; \; y = 1..N_y.$$

where the observed landmarks are bright spots (g=1) on a dark background (g=0).

Real-time constraint

The acquisition rate for a frame is 25 Hz. The time left for processing the image is about 1 ms at 25 Hz, 20 ms at 16 2/3 Hz, 40 ms at 12 1/2 Hz, ...
For our purpose we define the real-time constraint by

$$t_{exec} < 20 \text{ ms}$$

where t_{exec} is the execution time of the processing part, which allows an acquisition rate of 16 2/3 Hz.

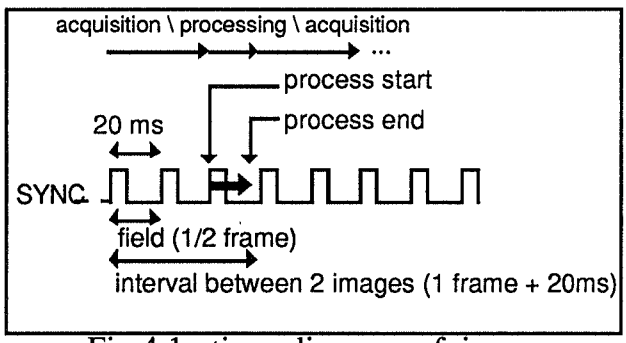

Fig.4.1: time diagram of image acquisition.

Finding the landmarks

The first step is to *detect* the landmarks as particles in the image, a particle being a connected group of bright pixels (g=1).
Once the particles are detected and their position determined, particle-following has to be performed for each of these [1][4].

Single landmark-following

The mass center of the particle is first issued from the entire image in a initialisation part.

$$\overline{X} = \frac{1}{N_x \cdot N_y} \sum_x \sum_y g(x,y) \cdot \underline{X}$$

where $x = k_x \cdot n$; $k = 0,1,2,..,\frac{N_x}{n}$

$y = k_y \cdot n$; $k = 0,1,2,..,\frac{N_y}{n}$

n : quantization step

$\overline{X}$: mass center.

N_x, N_y : image size

The *n* fixes the quantization step (resolution). It is dynamically adjusted in function of the particle size (i.e. the number of bright spots), so that the latter is smaller than a fixed value:

$$\text{largest n so that } \sum_x \sum_y g(x,y) < g_0$$

The next step is to define a square search window, of size N_w, that envelops a prediction of the particle position in the image that will be acquired just after the processing part. The prediction is based on the knowledge of the previouses particle positions. The mass center of the particle is then computed only in this window. This greatly decreases the computation time, since (N_x,N_y) becomes the size of the window.

This size is a critical parameter: the smaller it is, faster is the computation of the mass center. but higher is the risk that the particle "gets out" of the window (sudden rough movement of the camera).

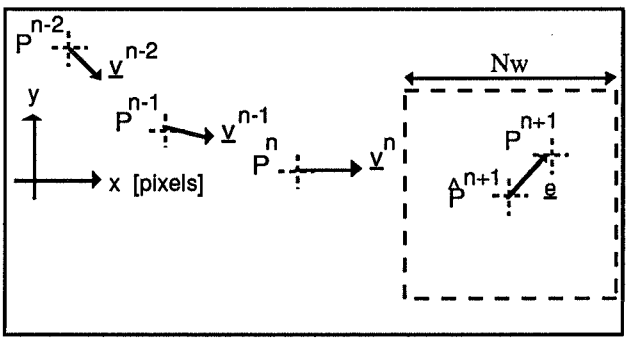

Fig.4.2: trajectory prediction in function of the previous positions. The search window is defined around the predicted position. e is the distance (error) between the estimated and the effective position

.

Approximation of the particle trajectory

The method is to predict the new location P^{n+1} of the mass center by extrapolating a trajectory estimated from the previous locations of the mass centers.

The image sequence is supposed to be acquired with a constant time interval Δt. The previous mass centers P^n, P^{n-1} and P^{n-2} are known. We have

$$da_x^{n+1} \rightarrow 0 \text{ for } \Delta t \rightarrow 0$$

where da_x^{n+1} is the difference of the acceleration projection between P_n and P_{n+1}.

The principle of the method is to consider

$$da_x^{n+1} = 0 \text{ with } \Delta t = \text{const.}$$

so that an approximation of P^{n+1} can be computed. This leads us to

$$\begin{pmatrix} \hat{X} \\ \hat{Y} \end{pmatrix} = \hat{P}^{n+1} = \begin{pmatrix} \frac{5}{2} \cdot x^n - 2 \cdot x^{n-1} + \frac{1}{2} x^{n-2} \\ \frac{5}{2} \cdot y^n - 2 \cdot y^{n-1} + \frac{1}{2} y^{n-2} \end{pmatrix}$$

Thus the predicted position $\hat{P}^{n+1}$ is simply a weighted sum of the 3 previous particle positions.

The error propagation of the fluctuation on the P^{n-i} (i=0,1,2), due to the resolution, leads us to:

$$|\Delta \hat{X}^{n+1}| = 5 \cdot \frac{n}{2}$$
$$|\Delta \hat{Y}^{n+1}| = 5 \cdot \frac{n}{2}$$

Landmark-following algorithm

Supposing the particle is present, the algorithm looks like:

❶ Compute the mass center on the entire image;

❷ Loop.

① Estimate the location of the search window;

② Acquire the image;

③ If the particle no more exists then go to ❸

④ Measure the effective mass center on the window.

⑤ Update the last 3 mass centers

⑥ Adapt the resolution n to the particle size within fixed limits.

❸ go to ❶

The method is fine for a sufficiently smooth trajectory (relative 3D landmark trajectory in comparison with the robot can be much more perturbated). The limit is fixed by the maximal difference of the acceleration between $\hat{p}_{n+1}$ and p_{n+1}:

$$\Delta a^{n+1} = \frac{5 \cdot n}{dt^2} \left[\frac{\text{pixels}}{s^2} \right]$$

Tests show that this limit is overrunned in cases like when the robot suddenly quakes due to an undetected obstacle.The search window then no more contains the effective particle.

5. Current implementation.

The algorithms presented here for the *go toward* and *go to* behaviours are designed for an implementation on a multiprocess system, since for each behaviour two processes must run simultaneously.
Our current implementation uses a single process system. A solution for the implementation of the parallel processes is to merge them into a single, sequential program.
For this purpose, the loop forming the slowest process is opened and pasted into the one forming the fastest process.
Our implementation is running on a Macintosh II and allows a real-time execution of the behaviours. Tests with a multi-landmark following algorithm show that the real-time constraint is also satisfied for up to 3 landmarks in the scene.

6. Conclusion

This paper presented the two vision-based behaviours named *going toward* and *going to* in the frame of the behavioural approach to robot navigation. *Going toward* moves the robot toward a landmark with visual feedback. *Going to* moves the robot up to a fixed distance to the landmark; to do so, it begins by a parallaxial movement with respect to the landmark in order to estimate its distance to it.
We showed that both these behaviours make use of a module named *landmark-following*. This one finds and tracks landmarks in the image sequence. It applies to either a simple or multiple landmarks. In the described implementation, *landmark-following* is performed in its single-landmark form by a simple (the landmark is treated as a point) and robust algorithm which has the advantage to be fast; it satisfies the real-time constraint.
The vision system considered uses retro-reflecting landmarks and a light source mounted on top of the robot, together with a fixed video camera. The intrinsic parameters of the camera are not used and thus no camera calibration is needed. However, in the case of a fisheye lens which is ours, lens distortion is very important and its correction is required.
The described behaviours have been implemented and tested successful on the robot. They represent example behaviours as well as building blocks for the more elaborate behavioural system under construction.

7. References

[1] D.H. Ballard, C.M. Brown
"Computer vision"
Prentice Hall, New Jersey, 1982.

[2] S.F. Ray
"Applied photographic optics"
Focal Press, London & Boston, 1988.

[3] G. Toscani
"Système de calibration et perception du mouvement en vision artificielle"
Diplôme de Docteur ès Sciences.
Université de Paris.

[4] T.S. Huang
"Image sequence analysis"
Springer-Verlag, New-York, 1981.

[5] R. Brooks
"A robust layered control system for a mobile robot"
Journal of Robotics and Automation, volume RA-2, number 1, 1986.

[6] H. Mori & al.
"A mobile robot strategy applied to Harunobu-4"
Conf. on Pattern Recognition, Rome, 1988.

[7] G. Franke
"Physical optics in photography"
The Focal Press, London & New-York. 1966.

[8] S. Kyle, R. Loser, J. Rogers
"Kern SPACE theodolite calibration"
SPIE Proceedings, vol. 1395 *Close-range photogrammetry meets machine vision*. 1990.

[9] C. Facchinetti
"Calibrage d'une caméra équipée d'un objectif grand-angle 'fisheye'".
Rapport interne, IMT Neuchâtel (CH). 1991.