Simulating *Pursuit* with Machine Experiments with Robots and Artificial Vision

Jorge Dias, Carlos Paredes, Inácio Fonseca, Helder Araújo, Jorge Batista, and Anibal T. Almeida

***Abstract*— This article is concerned with the simulation of *pursuit*. The article describes one solution for the problem of *pursuit* of objects moving on a plane by using a mobile robot and an active vision system. The solution deals with the interaction of different control systems using visual feedback and it is accomplished by the implementation of a visual *gaze holding* process interacting cooperatively with the control of the trajectory of a mobile robot. These two systems are integrated to follow a moving object at constant distance and orientation with respect to the mobile robot. The orientation and the position of the active vision system running a *gaze holding* process give the feedback signals to the control used to *pursuit* the *target* in real-time. The paper addresses the problems of *visual fixation, visual smooth pursuit,* navigation using *visual feedback* and *compensation* for system's movements. The algorithms for visual processing and control are described in the article. The mechanisms of cooperation between the different control and visual algorithms are also described. The final solution is a system able to operate at approximately human walking rates as the experimental results show at the end of the article.**

***Index Terms*— Active vision, artificial vision, pursuit, navigation.**

I. Introduction

THE PURSUIT of moving objects with machines such as a mobile robot equipped with an active vision system deals with the problem of integration and cooperation between different systems. This integration has two distinct aspects: the interaction and cooperation between different control systems and the use of a common feedback information provided by the vision system. In this article a global solution is proposed based on the visual *gaze holding* process to establish a pursuit mechanism for *targets* moving in front of the mobile robot. The system is controlled to keep constant the distance and the orientation of the robot and the vision system. The solution for this problem deals with the interaction of different control systems using visual feedback. It also addresses the real-time tracking of objects by using a vision system. This problem has been addressed in different fields such as surveillance, automated guidance systems, and robotics in general. Several works addressed the problems of visual servoing but they are mainly concerned with object tracking by using vision and manipulators [1], [8], [9], [11] and only some address problems related with ours [13]–[15]. Coombs studied the real-time implementation of a *gaze holding* process using binocular vision [8]. The system used a binocular system mounted on a manipulator and Coombs studied different control schemes for visual *gaze holding* using the information of two images. These studies included predictive control schemes to avoid delays, [4], [8]. Papanikolopoulos also proposed a tracking process by using a camera mounted on a manipulator for tracking objects with a trajectory parallel to the image plane [11]. The information supplied by the vision system is processed by an optical flow based algorithm that is accurate enough for tracking the objects in real-time. The efficient use of windows in the image improves the performance of the method. A control process is also reported by Allen for tracking moving objects in three-dimensions (3-D) [1]. The system can be used for grasping the objects by using a manipulator. The vision system uses two cameras and the distance from the manipulator's tool to the object is computed in real-time by using stereo images. Once the tracking scheme is stable, the system controls the manipulator to intercept the moving object and pick it up. These studies have connection with the solution for pursuit proposed in this article, since they deal with the tracking problem by using visual information. However in our system we explore the concept of visual fixation to develop the application. The computational solution for visual fixation uses motion detection to initiate the *fixation process* and to define a pattern that will be tracked. During *pursuit* the system uses image correlation to continuously track the target in the images. The solution is a refinement of the Burt technique [6]. Burt reports a real-time feature detection algorithm using hierarchical scaling images for surveillance and robotics.

More recently several laboratories have been engaged in a large European project (the *Vision as Process* project) for the development of systems, based on active vision principles. During the project, studies have been made to integrate camera control, ocular reflexes, real-time image processing, image tracking, perceptual grouping, active 3-D modeling, and object recognition.

Some of the systems described above have similarities with ours but in our system we control the system to keep the distance and orientation of the mobile robot with respect to a *target*. The solution described in this article includes the control of the *gaze* of the active vision system. Furthermore,

Manuscript received March 9, 1995; revised July 18, 1996. This paper was recommended for publication by Associate Editor R. Chatila and Editor S. Salcudean upon evaluation of the reviewers' comments.

The authors are with the I.S.R.-Instituto de Sistemas e Robótica-Coimbra site, Departamento de Eugeuharia Electrotécnica, Polo II da Universidade de Coimbre, 3030 Coimbra, Portugal.

Publisher Item Identifier S 1042-296X(98)01546-8.

our hierarchical control scheme establishes a pursuit process using different degrees of freedom on the active vision system and the movement of the mobile robot. To simplify the solution several assumptions were made. These assumptions are based on the type of movements and targets that we designed the system to cope with, and system's physical constraints such as: maximum robot velocity, possibility of adjustment of the optical parameters for focusing, maximum computational power for image processing and, the nonholonomic structure of the mobile robot. We assume that:

1) target and the robot move on a plane (horizontal plane);
2) the difference between the velocities of the target and of the robot does not exceed 1.2 m/s;
3) the distance between the target and the mobile robot will be in the interval of [2.5 m, 5 m] and the focal length of both lenses is set to 12.5 mm;
4) the target is detected only when it appears inside the cameras' field of view.
5) the system is initialized by setting the vision system aligned with the vehicle (the cameras are oriented to see the vehicle's front).

These assumptions bound the problem and only two variables are used to control the system. One is the angle in the horizontal plane defined by the target position relative to the mobile robot referential. The other is the distance between the robot and the target.

In the next Section, an outline of the pursuit process and its solution is given. In Section III, the system architecture and all geometric relations used in the solution are described. Section IV describes the image processing methods used to obtain information about the target. Section V describes the control of the system and explains the integration of different control modules. Section VI describes and illustrates some system's parameters when the system is performing pursuit and the last Section summarizes and comments the solution proposed.

II. Pursuit of Moving Objects

The problem of pursuing a moving object is essentially a motion matching problem. The machine, the robot in our case, must be controlled to reach the same motion as the *target*. In practice this is equivalent to keep constant the distance and orientation from the robot to the *target*. However, the solution for this problem has some particular aspects that must be emphasized. If the target is a person walking, its trajectory can be suddenly modified and consequently its velocity. Any solution proposed must cope with these situations and perform the control of the system in *real-time*. Since the machines have physical limitations in their velocity and maneuvering capabilities, it is essential to classify the different sub-systems used according to their velocity characteristics. In our experiments we use a mobile robot and an active vision system, and these two systems have different movement characteristics. The active vision system presents greater velocity than the mobile robot and also has less mass. However, it is the mobile robot (the body of the system) that must follow the *target*—see Fig. 1.

Fig. 1. The information provided by the *active vision system* is used to control the mobile robot to *pursuit* a person in *real-time*.

To perform the pursuit of a moving *target* we use two basic control schemes: a visual *fixation* control of the active vision system and the trajectory control of the robot. The visual *fixation* control guarantees that the *target* is continuously tracked by the vision system, and gives information about its position to the robot control. The robot control uses that information as a feedback to maintain the distance and orientation to the *target*.

The visual *fixation* control must be one visual process that runs in the active vision system and has capabilities to define a *target*, to concentrate the vision system on the *target* and follow it. A process with these characteristics has similarities with the visual gaze-shifting mechanism in the humans [7]. The gaze-shifting mechanism generates movements in the vision system to put a new object of interest in the center of the image and hold it there. The movement used to put the object in the center is called *saccade*, it is fast and it is performed by the two eyes simultaneously. If the *target* of interest is moving relative to the world, the vision system must perform movements to hold the *target* in the image center. These movements are composed by two types of motions called *smooth pursuit* and *vergence*. These motions are the consequence of the control performed by the process that we designate as *fixation*.

The *fixation* process centers and holds the orientation of the vision system on a point in the environment. The principle is described graphically in Fig. 1 where the mobile robot with an active vision system is concentrated on a person. *Fixation* gives a useful mechanism to maintain the relative orientation and translation between the referential in the vehicle and the *target* that is followed. This results from the advantages of the *fixation* process, where the selected *target* is always in the image center (foveal region in the mammals). This avoids the segmentation of all the image to select the *target* and allows the

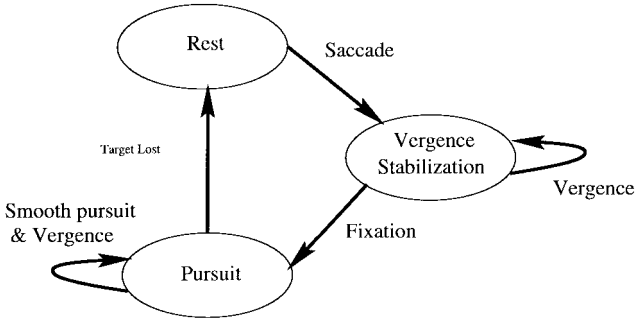


Fig. 2. State diagram of the *pursuit* process.

use of relative coordinate systems which simplifies the spatial description of the *target* (relationship between the observer reference system and the object reference system).

The pursuit process can be described graphically by the state diagram in Fig. 2. The process has three states: *Rest*, *Vergence Stabilization*, and *Pursuit*. The *pursuit* process must be initialized before starting. During this initialization, a *target* is chosen and several movements are performed by the active vision system: the gaze is shifted by a *saccade* movement and the vergence stabilized. In our system the *target* is chosen based on the visual motion stimulus. The selection corresponds to a region in the images that generates a large visual motion in the two images.

If a *target* is selected, a *saccade* movement is performed to put the *target* in the image center, and the system changes from the state *Rest* to *Vergence Stabilization*. During the *saccade* movement no visual information is used to feedback the movement. In the *Vergence Stabilization* state the system adjusts its *fixation* in the *target*. This is equivalent to establishing the correct correspondence between the centers of the two images, and defining a *fixation* point in the *target*. When the vergence is stabilized, the system is maintained in the *Pursuit* state.

III. Building a System to Simulate Pursuit

A. System Architecture

The main hardware components of the system are the mobile robot and the active vision system. These two basic units are interconnected by a computer designated *Master Processing Unit*. This unit controls the movements of the active vision system, communicates with the robot's on-board computer and is connected to two other computers designated *Slave Processing Units*. These units are responsible for processing the images provided by the active vision system. The connections between different processing units are represented in the diagram shown in Fig. 3 and a photograph of the system is presented in Fig. 4.

The *Right* and the *Left Slave Processing Units* are computers with i486DX2 CPU's running at 66 MHz. Each contains a DT-IRIS (50 Hz) frame grabber connected to each one of the cameras. The *Slave Processing Units* process the images and communicate their results to the *Master Processing Unit* (another computer with a i486DX2 CPU running at 66MHz). These communications use a 10 MBits connection provided by Ethernet boards (one board on each computer). The active

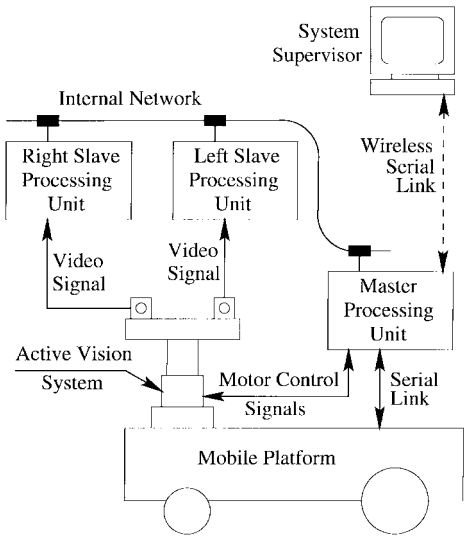


Fig. 3. System architecture.

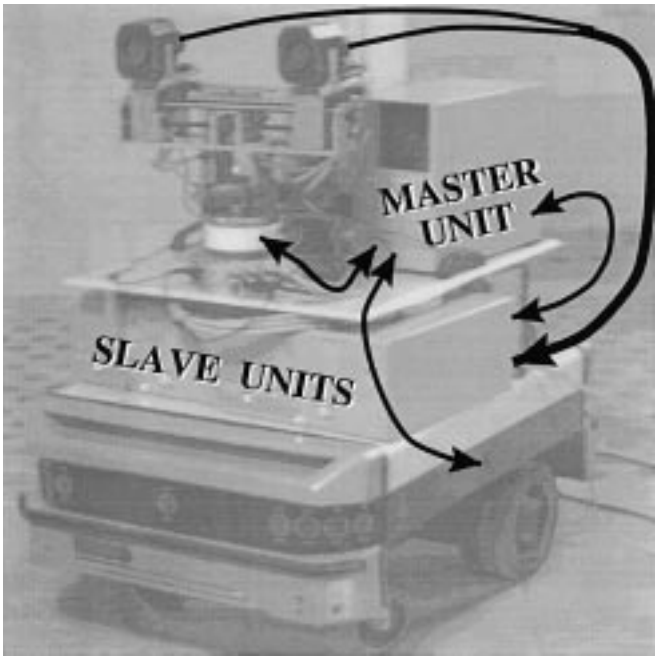


Fig. 4. The active vision system and the mobile robot.

vision system has two CDD monochromatic video cameras with motorized lenses (allowing for the control of the iris, focus and zoom) and five step motors that confer an equal number of degrees of freedom to the system (vergence of each camera, baseline shifting, head tilt and neck pan). The *Master Processing Unit* is responsible for the control of the degrees of freedom of the active vision system (using step motor controllers) and for the communication with the mobile platform (using a serial link).

The actual control of the mobile platform is done by a multiprocessor system based on a 68 020 CPU, installed on the platform. The management and the interface with the system is done by a computer, connected to the *Master Processing Unit* using the serial link and a wireless modem.

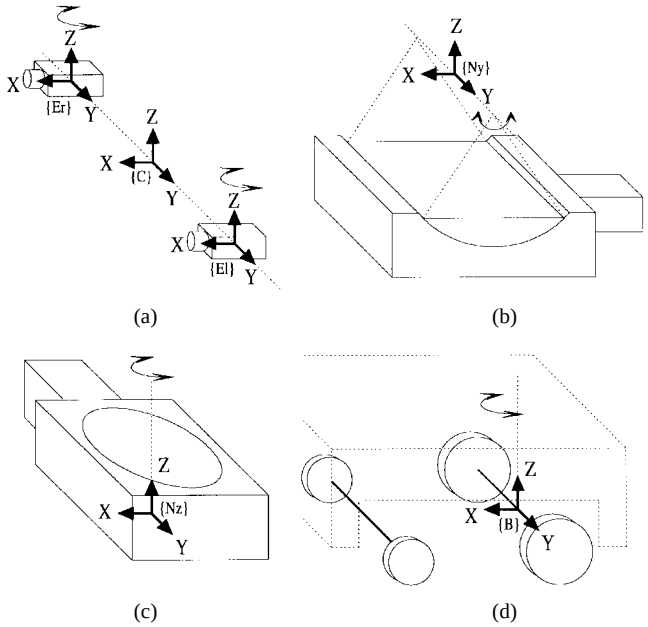


Fig. 5. Location of all the system's referentials: (a) *CYCLOP* and *EYES* referentials; (b) and (c) *NECK* referentials; (d) *BODY* referential.

B. System Geometry

There are six referentials associated with the system's mechanical structure, as illustrated in Fig. 5. The mechanical structure is divided into four groups: *BODY*, *NECK*, *CYCLOP* and *EYES*. The last three groups contain the referentials associated with the part of the structure referred to as head.

The upper part of the head structure is formed by the *EYES* and the *CYCLOP* groups [represented in Fig. 5(a)]. The *EYES* group contains two referentials designated $\{\boldsymbol{E}_r\}$ and $\{\boldsymbol{E}_l\}$. They are associated with the mechanical support for the two cameras, and each referential can be rotated around its z-axis, allowing for the cameras' vergence control. The cameras are supported by the *CYCLOP* mechanical part and between them is located the referential $\{\boldsymbol{C}\}$. This referential is located over the same vertical line as the referentials in the *NECK* group [described in Fig. 5(b) and (c)]. The distance between the referentials $\{\boldsymbol{E}_r\}$ and $]\{\boldsymbol{E}_l\}$ is known as the *baseline* and can be changed. For calibration purposes, both cameras can be adjusted along the x-axis. The appendix-A describes the parameters of the mobile robot and the vision system in detail [16], [18].

Located in the lower part of the head structure are the two referentials of the *NECK* group: $\{\boldsymbol{N}_y\}$ and $\{\boldsymbol{N}_z\}$ (represented in Fig. 5(b) and (c), respectively). Each can be rotated around a different axis (the index of each name indicates the rotation axis), allowing the upper part of the structure to be moved with two degrees of freedom (head tilt and neck pan).

The referential $\{\boldsymbol{N}_y\}$ his located directly above the $\{\boldsymbol{N}_z\}$ referential and they have coincident z-axis.

Finally the *BODY* group, that contains the referential $\{\boldsymbol{B}\}$, and is located in the mobile platform at the middle of the driving axle (i.e. the rear axle) and is represented in Fig. 5(d). The platform's driving wheels are independent, allowing the platform to rotate around the z-axis of the referential $\{\boldsymbol{B}\}$, and to move its origin.

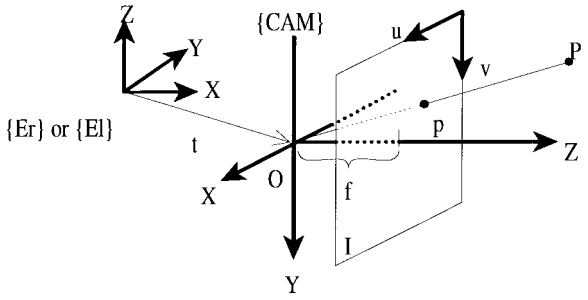


Fig. 6. Geometric model of each camera.

The head structure is placed on the mobile robot in such a way that the xz-planes of the referentials in the *CYCLOP*, *NECK*, and *BODY* groups are coincident.

C. Camera Model

To find the relation between a two-dimensional (2-D) point in one image obtained by either camera with its corresponding 3-D point in that camera's referential $\{\boldsymbol{CAM}\}$, we use the perspective model.

The projection of the 3-D point $\boldsymbol{P}$ in plane $\boldsymbol{I}$ is a point $\boldsymbol{p} = (u, v)$, that results from the intersection of the projective line of $\boldsymbol{P}$ with the plane $\boldsymbol{I}$. The perpendicular projection of the point $\boldsymbol{O}$ in the plane $\boldsymbol{I}$ is defined as the center of the image, with coordinates (u_o, v_o). The distance f between the point $\boldsymbol{O}$ and its projection is called the focal length.

If (x, y, z) are the 3-D coordinates of the point $\boldsymbol{P}$ in the $\{\boldsymbol{CAM}\}$ referential, the 2-D coordinates of the projection (x_u, y_v) of it on a continuous image plane is given by the perspective relationships

$$x_u = f\frac{x}{z}$$
$$y_v = f\frac{y}{z}. \tag{1}$$

Since the image for processing is a sampled version of the continuous image, the relation between the units (millimeters) used in the $\{\boldsymbol{CAM}\}$ referential and the image points (u, v) are related with (x_u, y_v) by

$$u = S_x x_u + u_0$$
$$v = S_y y_v + v_0. \tag{2}$$

That relation is obtained with a calibration process that gives the scale factors for both the x and the y-axis (S_x and S_y, respectively) [17]. The image center (u_o, v_o), the focal length f and the scale factors S_x and S_y are called the intrinsic parameters of the camera.

Three-dimensional point given in the $\{\boldsymbol{CAM}\}$ referential transforms into the 2-D image coordinates of that point's projection. To obtain this transformation in the $\{\boldsymbol{E}_r\}$ and $\{\boldsymbol{E}_l\}$ referentials we must take into consideration the orientation and location of $\{\boldsymbol{CAM}\}$ in those referentials. As can be seen in Fig. 6, the $\{\boldsymbol{CAM}\}$ referential is rotated with respect to

$\{\boldsymbol{E}_r\}$ and $\{\boldsymbol{E}_l\}$, and their origins are probably not coincident. However the camera's position can be adjusted, therefore permitting the vector $\boldsymbol{t}$ to become approximately equal to the null vector, making the origins almost coincident. This adjustment is realized during the cameras calibration phase. If the translation is null, the transformation of the $\{\boldsymbol{CAM}\}$ referential into the $\{\boldsymbol{E}_r\}$ or $\{\boldsymbol{E}_l\}$ referentials can be done with a simple transformation.

D. System Models and Geometric Relations

The information of the *target's* position in the images is used to control the position and orientation of the vision system and of the mobile robot in order to maintain the relative distance and orientation to the *target*. Essentially the system must control the position of each actuator to maintain this goal. This implies to control the actuators of the vision system and also of the mobile robot. In our system these actuators have their own *units* that control their position with accuracy. These units are designated low-level *control units*.

In the case of the vision system the actuators used are step motors. These motors are controlled by dedicated units supervised by the *Master Processing Unit* . These motors rotate a specific number of degrees for each pulse sent to their power driver unit. The pulses are generated by the dedicated control units. These units generate different profiles for the pulse rate curve which must be adjusted for each motor. This adjustment is equivalent to a step motor *identification* procedure. This procedure was performed for each motor used in the active vision system. With this procedure the correct curve profile was adapted for a precise position control.

The mobile robot has also its own on-board computer that controls the motors used to move it. The onboard computer is responsible for the correct execution of the movements, and it accepts commands for movements that can be modified during their execution. This possibility is explored in our system to correct the path during the movement execution. The commands sent to the mobile robot reflect the position that the robot must reach to maintain the distance to the *target*. If the commands sent do not exceed the possibilities of the system, the command will be sent to the robot to be executed with accuracy. This detail is verified before sending a command to the mobile robot. Appendix A gives some details about these mobile robot parameters. These low-level control units facilitate the global control of the system and its adjustment.

Since the *target* changes its position in space, in most of the time its image position will also change. The goal is to control the system in such a way that the object's image projects into the center of both images, maintaining at the same time the distance to the object. The control can be performed by controlling the robot position, the neck orientation and the vergence of both cameras. The control implies the use of these degrees of freedom to reach the goal of *pursuing* a target. It is possible to obtain expressions between the several degrees of freedom, useful for their control, based on the geometric relationships.

The goal is to change the cameras' angles θ_l and θ_r, by the amount necessary to keep the projection of the *target* in the center of the image (see Fig. 7). Since we assume that the target moves on the same plane as the mobile robot we will consider only the horizontal disparity $\Delta u = (u - u_o)$.

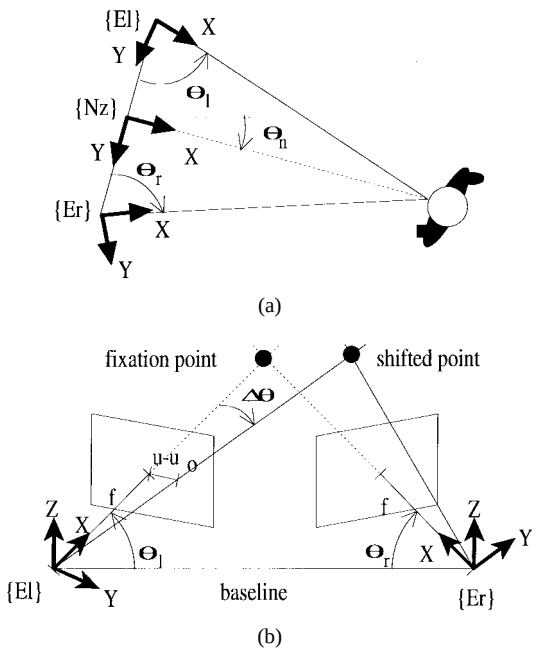


Fig. 7. Cameras' vergence angle control.

Let u be the coordinate in pixels of the reference point along the x-axis of either frame. The angle that each camera must turn is given by

$$\Delta\theta = \arctan\frac{u - u_o}{S_x f}. \tag{3}$$

This relation is easily derived from the (2) and from the representation in Fig. 7. To provide the system with the ability to react to the movements of the object's and with the ability to keep the distance and attitude between the two bodies, it is necessary to evaluate the distance of the object with respect to the robot. The position of the object to track is defined in terms of its distance D and the angle θ_n with respect to the $\{\boldsymbol{C}\}$ referential, and using the *fixation* point as reference (both parameters are represented in Fig. 8).

To obtain the equations that give the values of D and θ_n, we start by defining the following relations, taken directly from Fig. 9 (equivalent to Fig. 8, but with some auxiliary parameters)

$$\begin{aligned} h &= \tan(\theta_r)D_r \\ h &= \tan(\theta_l)D_l \\ B &= D_l + D_r \\ p &= D_l - \frac{B}{2}. \end{aligned} \tag{4}$$

The distance D and the angle θ_n of the *fixation* point with respect to the $\{\boldsymbol{C}\}$ referential can be obtained by the following equations (recall that the angle θ_n is positive clockwise—see Fig. 9):

$$\begin{aligned} \theta_n &= 90^\circ - \arctan\left(\frac{h}{p}\right) \\ D &= \sqrt{h^2 + p^2}. \end{aligned} \tag{5}$$

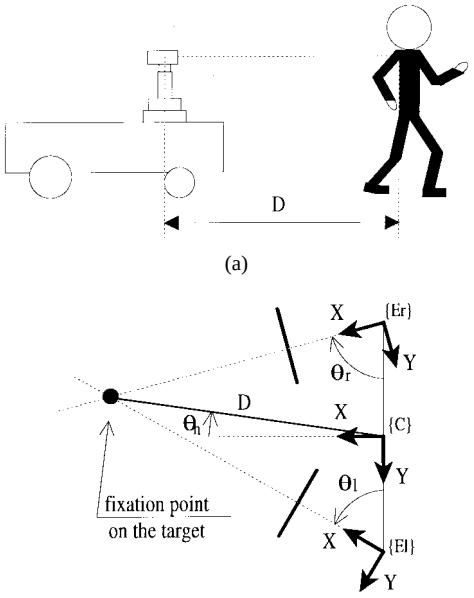


Fig. 8. Distance and angle to the object defined in the plane parallel to the xy-plane of the $\{\boldsymbol{C}\}$ referential.

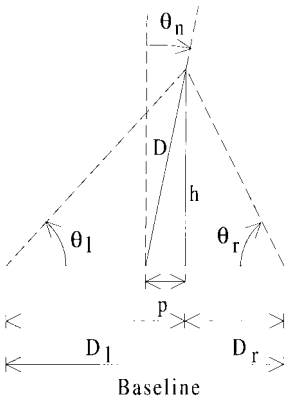


Fig. 9. Auxiliary parameters.

Note that, when θ_l equals θ_r, the above relations are not valid. In that case, the angle θ_n is zero, and the distance D is equal to

$$D = \frac{B}{2}\tan(\theta_l).$$

As described above, the motion and feature detection algorithms generate the position in both images of the object to follow. From that position, only the value along the x-axis will be used, since we assume that the object *moves in the horizontal plane*, and therefore *without significant vertical shifts*.

The trajectories of the moving platform are planned by the *Master Processing Unit* based on the values of D and θ_n given by (5). The values D and θ_n define a 2-D point in the $\{\boldsymbol{C}\}$ referential. These values can be related to the $\{\boldsymbol{B}\}$ referential since all the relationships between referentials are known. The result is a point $\boldsymbol{P}$ with coordinates x and y as shown in Fig. 10.

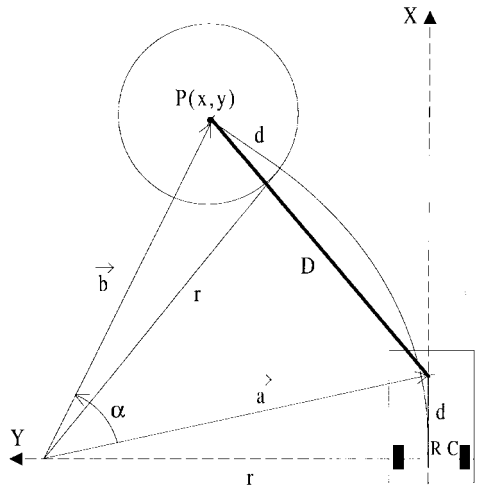


Fig. 10. Robot trajectory planning.

This figure is useful to establish the conditions for a mobile robot's trajectory when we want that the mobile robot reaches a point $\boldsymbol{P}(x, y)$. To clarify the situation we suppose that the object appeared in vehicle's front with an initial orientation $\theta = 0$. (the solution is be similar for an angle $\theta < 90°$). We know that several trajectories are possible to reach a specific point but, the trajectories' parameters are chosen according to the following.

1) The point $\boldsymbol{P}$ is assumed to be in front of the vehicle and the angle α is always greater than zero as show Fig. 10. This is a condition derived from the system initialization and the correct execution of the *pursuit* process (see Section I). Additionally, that condition helps to deal with the nonholonomic structure of the mobile robot.
2) The platform must stop at a given distance from the object. This condition is represented in Fig. 10 by the circle around the point $\boldsymbol{P}$ (the center of the platform's driving axle, point $\boldsymbol{RC}$, must stop somewhere over this circle).
3) The platform must be facing the object at the end of the trajectory. In other words, the object must be on the x-axis at a distance d from the origin of $\{\boldsymbol{B}\}$, when the platform stops.

The trajectory that results from the application of those conditions is a combination of translational and rotational movements that achieve the desired position. Notice that this type of mobile platform is subject to nonholonomic constraints and those constraints restricts the mobile platform motion. However since the position of the point $\boldsymbol{P}(x, y)$ that we want to control is not on the motor wheels, is possible to establish feedback control laws with exponential convergence around a given configuration [19], [20]. That is our case since the point $\boldsymbol{P}$, in the end of the trajectory, must be at a distance d from the center of the platform's driving axle, the point $\boldsymbol{RC}$. Note if the distance $d \to 0$ the system is no more controllable, since the mobile platform can not move in the direction of wheels axis.

Two parameters are needed to define the trajectory, as shown in Fig. 10: the radius r and the angle α. The analysis of Fig. 10

allows the derivation of the following relations:

$$\begin{aligned}\vec{\boldsymbol{b}} &= [x \quad (y-r)] \\ \vec{\boldsymbol{a}} &= [d \quad -r] \\ ||\vec{\boldsymbol{b}}|| &= ||\vec{\boldsymbol{a}}|| \\ r^2+d^2 &= x^2+(y-r)^2 \\ \vec{\boldsymbol{b}}\,\vec{\boldsymbol{a}} &= (\sqrt{x^2+(y-r)^2})(\sqrt{r^2+d^2})\cos(\alpha) \\ &= xd+r(r-y). \end{aligned} \tag{6}$$

After simplification we get

$$\begin{aligned} r &= \frac{x^2+y^2-d^2}{2y} \\ \alpha &= \arccos\left(\frac{xd+r(r-y)}{r^2+d^2}\right). \end{aligned} \tag{7}$$

The equations (7) are not defined when the y coordinate is equal to zero. In that case, the trajectory is linear, and the distance r that the mobile platform must cover is given by

$$r = x - d. \tag{8}$$

E. Compensating for System's Movements

Two aspects must be pointed out during the trajectory generation.

1) The robot's velocity is limited to a maximum value, and therefore it will probably only cover partially the planned trajectory. This problem is due to the system being controlled discretely, and given only a slice of time (sampling period) to perform the trajectory. During this period the active vision system must not only locate the object, but also keep track of it, until the robot is able to reach the goal condition.
2) The movements executed by the robot and/or by the neck will of course have influence in the position of the object's projection originating a reaction of the system to a movement that the object wasn't responsible for.

Supposing that the robot will have an uniform movement, it's possible to infer the amount of the trajectory arc that will be completed in one sampling period, based on the linear and angular velocities of the robot. This assumption gives the solution for the problem described on the items introduced in the beginning of the section and is equivalent to a *compensation* for the robot's movement, and results on the control of the neck based on the estimated position of the object at the end of the sampling period. The same process can be used to control the camera's vergence, *compensating* for movement of the neck.

Fig. 11 illustrates the problem if only a part of the robot's movement is performed. In that case only a part of the planned arc movement is considered. The letters $(D, \theta_n$ and D', θ'_n define the object as seen by the active vision system, respectively, before and after a sampling time $t(k)$ and the

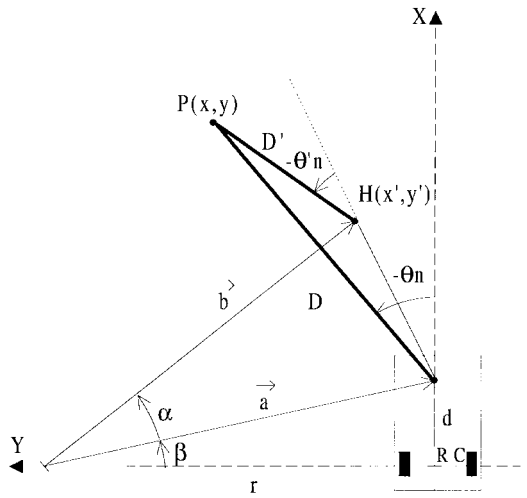


Fig. 11. Compensation for the robot's movement. The letters D, θ_n and D', θ'_n define the object before and after the movement.

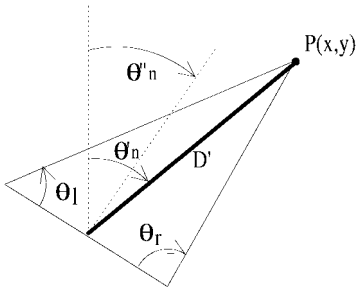


Fig. 12. Compensation for the *neck* movement.

minus signs in the angles are due to the counter clockwise representation). From direct analysis of the figure, we obtain the following relations:

$$\begin{aligned} |\vec{\boldsymbol{b}}| &= |\vec{\boldsymbol{a}}| = \sqrt{d^2+r^2} = R \\ \beta &= \arctan\left(\frac{d}{r}\right) \\ \boldsymbol{H}(x',y') &= (R\sin(\alpha+\beta), r - R\cos(\alpha+\beta)). \end{aligned} \tag{9}$$

The point $\boldsymbol{H}(x',y')$ represents the new position of the active vision system at the end of the sampling period and due to robot's movement. The values of D' and θ'_n for this situation are given by

$$\begin{aligned} D' &= \sqrt{(x-x')^2+(y-y')^2} \\ -\theta'_n &= \arctan\left(\frac{y-y'}{x-x'}\right) - \arctan\left(\frac{y'}{x'-d}\right). \end{aligned} \tag{10}$$

This new position (D', θ'_n) is equivalent to the new position of the object at the end of the sampling period $t(k)$, assuming that the object stands still.

If this is not the case and the object is moving, we must remember that its position is obtained using images acquired in the beginning of the sampling period $t(k)$. To obtain the real object position we will need to take into account the robot's movement and that is equivalent to using (D', θ'_n) to infer the new object's position. In that sense, this is equivalent to a robot's movement compensation.

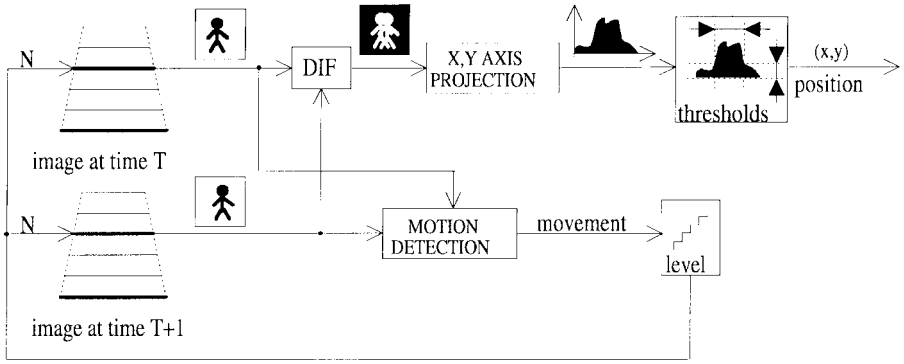


Fig. 13. Illustration of the image processing used for saccade. The *saccade* is preceded by searching for a large movement in the images. The searching phase is concerned with the detection of image movements, within certain limits (the pixel displacement detection can be adjusted by switching between pyramid levels).

Note that the above relations are not valid when the trajectory is linear. In that case, the neck angle is zero, and the new distance will be $D' = D - r$ with the value of r given by (8).

The compensation for the vision system movements is performed by neck movements. Since this degree of freedom induces a new object's position when it moves, the compensation is similar to the robot's compensation. The angle θ'_n calculated by (5) represents the absolute angle that the neck should reach at the end of the sampling period. Since the system has physical limitations, very often the angle is executed only partially. In that case, as illustrated in Fig. 12, the angle of the neck at the end of the sampling period will be θ''_n. In that case the angles θ_r and θ_l can be controlled to compensate for this limitation. The values of θ_r and θ_l can be computed with the help of the relations 4. Since the values of D' and θ'_n are known, the values of h and p can be obtained by using the relations 4 and from them the values for D_l and D_r. The values for the angles θ_r and θ_l are given by the relations

$$\theta_l = \arctan\left(\frac{h}{D_l}\right)$$
$$\theta_r = \arctan\left(\frac{h}{D_r}\right). \tag{11}$$

Note that the above relations are not defined when either D_l or D_r are zero. Those cases are solved assuming the value $90°$ for the angle.

This section described the relations used to compensate for the physical limitations of the system. The approach suggests a hierarchical mechanism for movement generation, starting by the mobile robot and finishing in the vergence of the cameras. That approach will be described in the Section V.

IV. Vision Processing and State Estimation

A. Image Processing

The *Slave Processing Units* analyze, independently, the images captured by the frame grabbers connected to the cameras. Therefore, the motion and feature detection algorithms described here are intended to work with the sequence of images obtained by each camera.

The *Slave Units* are responsible by processing the sequence of images during all states illustrated in Fig. 2. When the system is initialized, the *Rest* phase starts and the *Master Processing Unit* commands the *Slave Units* to begin a searching phase. This phase implies the detection of any movement that satisfies a set of constraints described below. At a certain point during this phase, and based on the evolution of the process in both *Slave Units*, the *Master Unit* decides if there is a *target* to follow. After this decision the *Master Unit* sends a *saccade* command to the *Slave Units* to begin the vergence stabilization phase. During this phase, the system will only follow a specific pattern corresponding to the *target* previously defined and ignoring any other movements that may appear. This phase proceeds until the *vergence* is considered stable and after that it changes to the *pursuit* state. The system remains in this state until the pattern can no longer be found in the images.

B. Gaussian Pyramid

In order to speedup the computing process, the algorithms are based on the construction of a Gaussian pyramid [3], [5]. The images are captured with 512×512 pixels but are reduced by using this technique. Generally speaking, using a pyramid allows us to work with smaller scale images without loosing significant information. Climbing one level on the pyramid results in an image with half the dimensions and one quarter of the size. Level 0 corresponds to 512×512 pixels and level 2 to 128×128. Each pixel in one level is obtained by applying a mask to the group of pixels of the image directly bellow it. The applied mask is basically a low pass filter, that helps in reducing the noise and smoothing the images [10].

C. Image Processing for Saccade

The *saccade* is preceded by searching for a large movement in the images. As described above, the searching phase is concerned with the detection of any type of movements, within certain limits. For that purpose, two consecutive images $t(k)$ and $t(k+1)$ separated by a few milliseconds are captured. These images are analyzed at the pyramid level 2.

The analysis consists in two steps described graphically by the blocks diagram shown in Fig. 13.

1) Computation of the area of motion using the images acquired at time $t(k)$ and $t(k+1)$. This calculation measures the amount of shift that occurred from one

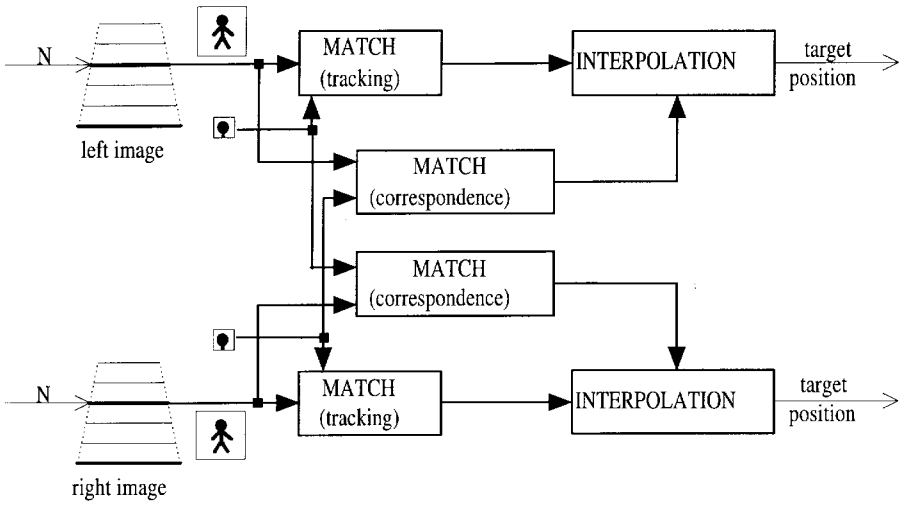


Fig. 14. Illustration of the image processing used for *Fixation*. To maintain the *fixation* dynamically, the process uses two phases: pattern's tracking and correspondence phase. The pattern is defined as an area around the center of the image acquired after the execution of the *saccade* movement. The tracking phase locates the *target* and measures the variables necessary to control the system and keep the object centered in the image. The correspondence phase confirms the results of the searching phase and estimates the images' disparity. This disparity allows the evaluation of object's distance.

frame to the other, and is used to decide when to climb or to descend levels on the pyramid.

2) Absolute value subtraction of both images, pixel by pixel, generating an image of differences, followed by the computation of the projections of the image in the x and y-axis. Since we assume that the target will have a negligible vertical motion component, only the image projection on the horizontal axis is considered for the *saccade* movement. Two thresholds are then imposed: one defining the lowest value of the projection that can be considered to be a movement, and the other limiting the minimum size of the image shift that will be assumed as a valid moving *target*. If both thresholds are validated, the object is assumed to be in the center of the moving area. If the movement is sensed by both cameras and it satisfies these two thresholds, a *saccade* movement will be generated.

D. Image Processing for Fixation

The goal of *fixation* is to keep the visual *target's* image steady and centered. This presumes that the *target* is the same for the two cameras. Vergence is dependent on this assumption and, in this work, it is assumed that the vergence is driven by the position of the 3-D *fixation* point. This point corresponds to the 3-D position of the *target* that must be followed. This is equivalent to the problem of finding the correspondence between *target's* zones in the two images. In this work this process is called correspondence. Since the system is continuously controlled to keep the images centered on the fixated *target*, the correspondence zone is defined around the image center and the search process becomes easy. The correspondence used for *fixation* starts by receiving the pattern from the other *Slave Unit*. The pattern that is needed to follow in one image (left/right) is passed to the other (right/left) to find the position of the correspondent pattern. The search starts around the image center and tries to find an image that matches the pattern received. The test uses the

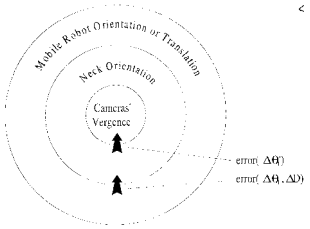


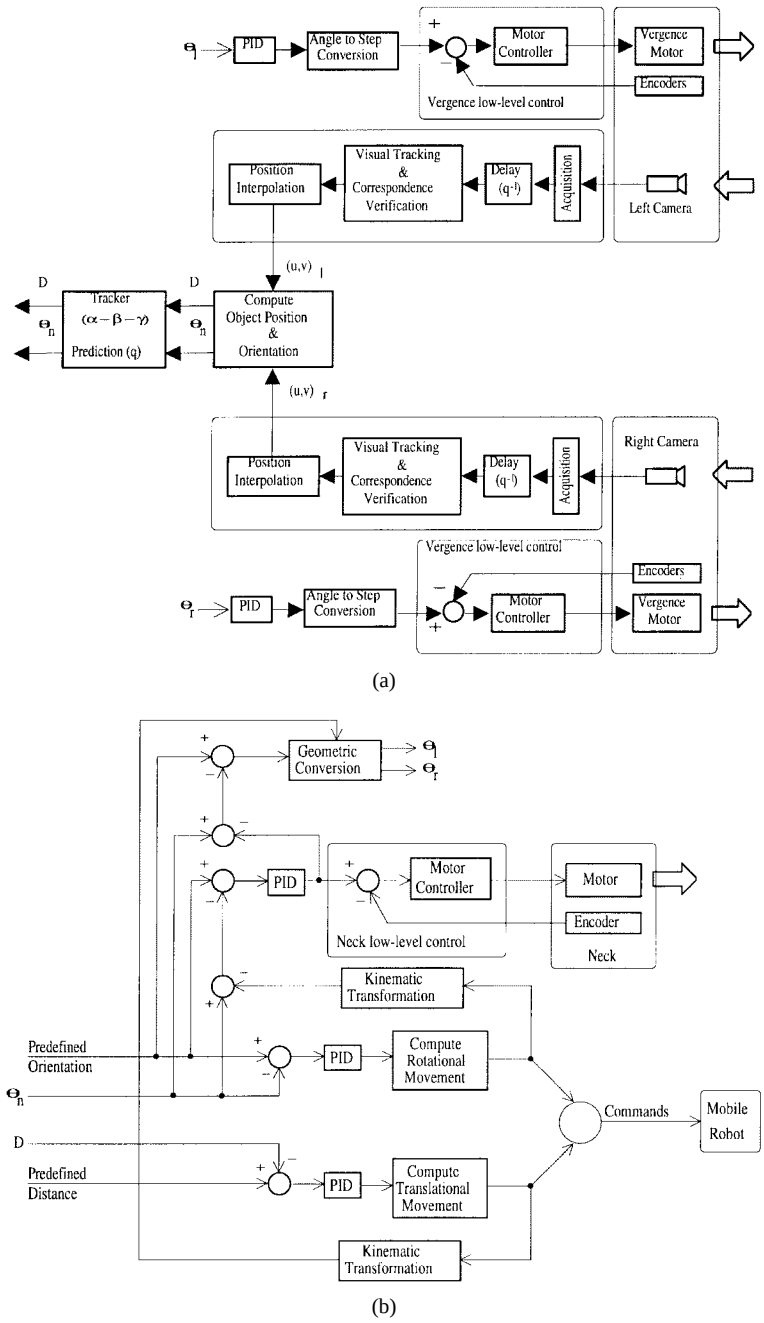
Fig. 15. Graphic scheme of the principle used for the control of the system. Since the mobile robot has more inertial mass than the active vision system, it will receive commands to cover the error. Very often the robot does not have possibility to eliminate all the errors. In that case it is the other fastest degree of freedom that tries to do it. If in the inmost level the object's movement can not be compensated for then it will be assumed that the *target* was lost and the pursuit process will re-start again.

operator described by (12) and the image area with lowest difference will be considered if the value of $\delta(x,y)$ is less than a maximum threshold. The similarity operator applied to the position (x,y) is defined as follows:

$$\delta(x,y) = \sum_{i=-n}^{n} \sum_{j=-m}^{m} (P(i,j) - I(x-i, y-j))^2 \qquad (12)$$

where $\boldsymbol{P}$ is the pattern to search and $\boldsymbol{I}$ is the image. In the definition, $2n+1$ and $2m+1$ represent the width and height of the pattern in pixels.

The process to keep the *fixation* dynamically is equivalent to a process of *gaze holding*. The *gaze holding* uses similar image processing as used for *fixation*. The *gaze holding* is implemented through *smooth pursuit* and *vergence* movements of the active vision system. These movements are based on the tracking of a pattern in each image. The pattern is defined as an area around the center of the image acquired after the execution of the *saccade* movement. The *gaze holding* uses two image processing phases: the tracking of the patterns in the left and right images, and the correspondence of the left and right patterns. The tracking phase uses the pattern defined at the end of the *saccade* movement. That pattern



(a)

(b)

Fig. 16. Global control scheme for vergence control and object's position calculation *(top)* and mobile and neck control *(bottom)*.

will be searched in each image captured, at a certain pyramid level, as described by the blocks diagram shown in Fig. 14. In our system, for a significant percentage of the time the level selected is level 2, corresponding to images of 128 × 128 pixels. The search consists in the computation of a similarity measure between the pattern and the areas with equivalent size taken from the image—see (12). The image area that resulted in the lowest difference is considered to be a match, but only if the value of the difference is less than a maximum threshold, above which no matches are accepted. This process is finished with the verification of correspondence between *left* and *right* patterns in similar manner as the *fixation* phase.

Since the images used during *fixation* and *gaze holding* are sub-sampled, the target's position is defined with more precision with the interpolation process described in the appendix B. The final position is combined with the position obtained by the *fixation* and *gaze holding* phases. Actually the final position corresponds to the middle point between these two positions.

E. System State

The system state used for control is described by two variables: the distance D from the system to the *target* and the angle θ_n of the neck pan, defined when the vision system is fixated on the *target*. Both variables have values that can be obtained by the geometry of the system and are described by (5).

These two quantities are continuously estimated by using a fixed coefficient Kalman filter—see Appendix C. The fixed co-

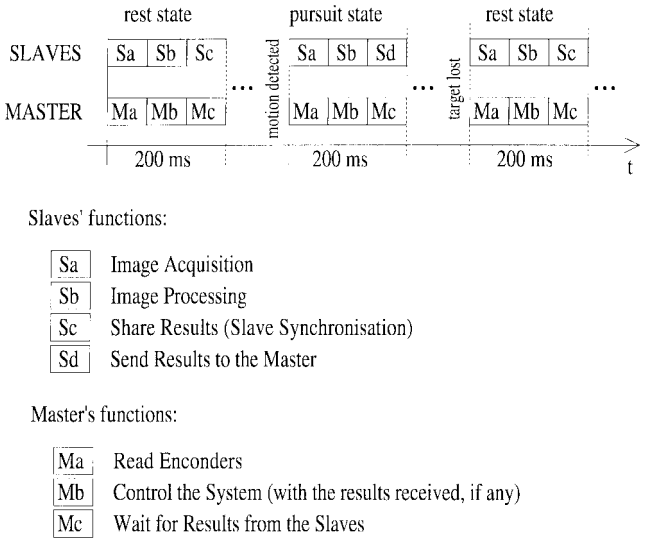


Fig. 17. Timing diagram for the different states in the system.

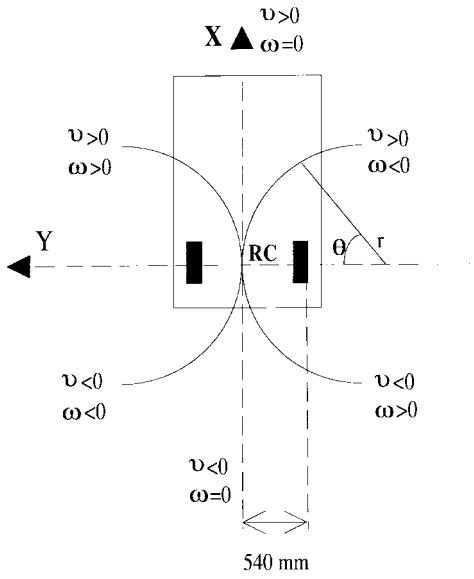


Fig. 18. Movements of the mobile robot: ω represents the angular velocity, v the linear velocity, r is the radius and θ is the orientation angle.

efficients filters have the advantage of simple implementation and the most extensively applied of these filters is the $(\alpha$-β-$\gamma)$ tracker. The values estimated by this filter are described by

$$\hat{\boldsymbol{x}}_D = \begin{bmatrix} D \\ \dot{D} \\ \ddot{D} \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{x}}_\theta = \begin{bmatrix} \theta \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix}. \tag{13}$$

This filter is specifically useful for estimating the state variables assuming that the *target* has uniform accelerations. It tends to be able to predict smoothly varying velocities but also filtering abrupt changes on the *target's* position. This technique can also be used to predict future *target* kinematics quantities. This enables the compensation for delays in the system as will be described below.

The values described by (13) are computed by the *Master Processing Unit* and are used to control the degrees of freedom of the active vision system and the mobile robot, as will be described in the next section.

V. System Control

A. Introduction

A complete implementation of the methods described in this paper requires fast processing capabilities for vision processing and control. The implementation proposed in this work is based on control loops working in parallel and based on the visual *pursuit* process. Since the inertia of the neck and the mobile robot are greater than the vergence mechanism inertia, this type of control simulates a control system with different levels—see Fig. 15. The inmost level comprises cameras and the vergence motors of the active vision system, responsible for tracking the *target* in real-time. This sub-system controls the cameras' position to maintain the visual system fixed in the *target*. At the intermediate level there is the neck sub-system, that provides the control of the orientation of the vision system and compensates for the cameras' vergence movements. At the outmost level is the mobile robot sub-system that provides the compensation for the orientation of the active vision system and also controls the orientation and distance to the *target*.

Conceptually, the error between the actual distance and orientation of the *target* and the system is propagated from the outmost level to the inmost level. This concept is graphically described in Fig. 15. In each level the error is compensated for by the sub-system associated to each level. This error must be such that the maximum characteristic values of each sub-system are not exceeded. In the cases where the error exceeds these maximum values, the difference of error that can not be compensated for in that level is passed to the next inmost level. This scheme establishes a mechanism to propagate the error through the different control systems, giving more priority to the mobile robot, followed by the neck and eyes at the end. This gives the effect of compensation for the *target's* movements, simulating its *pursuit*. The control scheme used is illustrated by the global block diagram in Fig. 16. If in the inmost level the object's movement can not be compensated for then it will be assumed that the *target* was lost and the pursuit process will re-start again. That is equivalent to the transition between the *Pursuit* and *Rest* state described in Fig. 2.

B. Timing Considerations

The pursuit control loop consists basically of three stages: image acquisition, error estimation (the orientation and distance) and error correction. These steps are realized by the *Slave* and *Master Processing Units* at cycles synchronized by a general clock in the system. In the experimental site used to develop the system the clock has a 200 msec cycle and the system's parameters are adjusted for that cycle. During this cycle the system performs different computations, depending on the state of the system. The different states of the system are illustrated by Fig. 2 and the timing for the cycles is illustrated in Fig. 17.

The images generated by each camera are acquired and analyzed by the *Slave Processing Units*. These units analyze the images and give the position of the *target* in each image. That position gives the necessary information to compute the

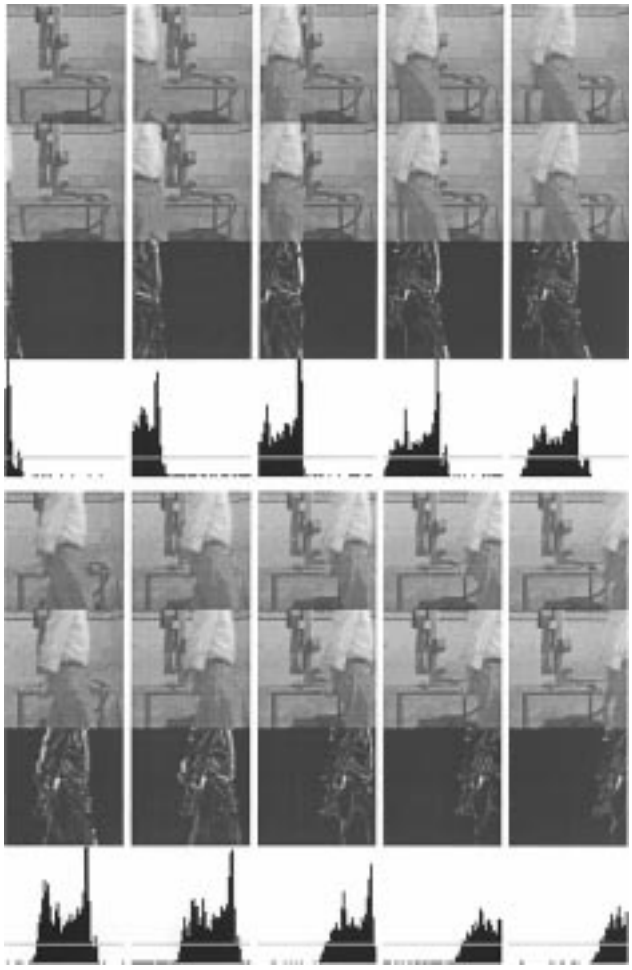


Fig. 19. Evolution of the searching algorithm when a target is moving. This image searching algorithm is responsible by the trigger of the saccade movement.

system state D and θ_n. This state is passed to the $(\alpha$-β-$\gamma)$ tracker. The information provided by the *Slave Units* is delayed by one cycle of 200 msec. To avoid the lateral effects of this delay we use the prediction capabilities of the $(\alpha$-β-$\gamma)$ filter to estimate a value for the system state (see Appendix C) (25).

The *Master Processing Unit* repeatedly performs the control algorithm by using the error between the predicted system state and the desired system state. This error is passed to the different sub-systems according to the illustration in Fig. 16. This error is passed to the different PID discrete time algorithms implemented in each subsystem. The results will be changes in the positions of the step motors associated with the vision system and the commands for the mobile robot to maintain the desired system state D and θ_n.

The movements executed by the mobile platform are based on two motors associated with each of the driving wheels (rear axle) and are essential to make the compensation for the error in the distance D. The movements permitted with this type of configuration are represented in Fig. 18. Pure rotations are around the center of the driving axle represented in Fig. 18 by RC.

The values of the velocities are dependent on the type of movement and the duration time. In our experiments the period

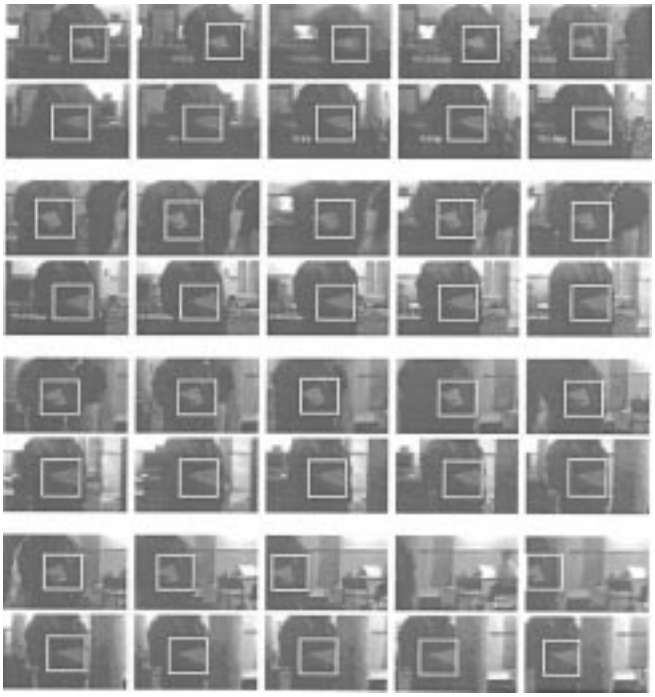


Fig. 20. Evolution of the tracking algorithm. This image tracking algorithm is used for keep the visual system *fixated* on the person that is moving.

of time is a multiple integer of the system cycle (200 msec). For movements composed by linear and angular velocities, the radius of the trajectory and the trajectory arc can be obtained by the relations (14) and (15) presented in Section B of Appendix A.

VI. Experimental Results

This section gives examples of the system performances. The first example is related with the evolution of the searching algorithm used for saccade movement. Fig. 19 represents ten cycles of the process with the entire system stopped. From top to bottom we can see the image taken at time $t(k)$, the image taken at time $t(k+1)$, the differences image and its projection on the x-axis. The analysis of the sequence allows us to conclude that we can detect the motion by efficiently thresholding the differences obtained due to the noise. It also shows that the values above the threshold, although unstable, appear in the area where the movement occurred. Notice that the images are pre-filtered and smoothed as explained in Section V.

The second example concerns the evolution of the algorithm used during pursuit. Fig. 20 shows a sequence of pairs of images (right image on top of the left image) obtained with the system running and the *target* moving. As it can be se en, apart from some minor deviations, the selected area of the *target* is matched correctly. Also important is the obvious tendency to keep the *target* in the middle of the images, as should be expected from the type of control implemented.

It is interesting to know the system response to a input signal like the the step signal. The step-response reveals the general form of the system model. The response of the neck's angle to a simulation of the step signal is shown in Fig. 21. The

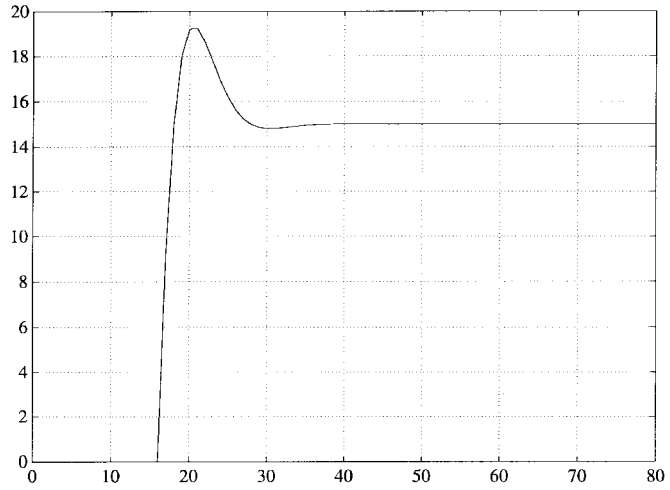


Fig. 21. The response of the neck's angle to a simulation of the step signal when the filter prediction is used. The vertical axis represents θ_n (degrees) for diferent time clocks (k) represented on the horizontal axis.

signal used is equivalent to the signal resulting from a person standing in front of the system, and then quickly taking a step aside. The distance is constant and equal to 2 meters. The use of the filter prediction, described in appendix C by (25), results in the signal shown in Fig. 21.

The next three graphics (Fig. 22) show the angle shifts of the neck motor, right camera motor and left camera motor respectively, when filtering and prediction are used. The curves shows traces for the proportional, integrative and differential gains (PID) of 0.6 and 0.1.

The same experiment was repeated with a proportional gain of 0.1 for the neck motor and the results are depicted in Fig. 23. Note that, since the system is now slower, the use of compensation and prediction methods assumes a greater importance.

The final examples (Figs. 24 and 25) show the importance of filtering the distance D and the angle θ. The $(\alpha$-β-$\gamma)$ filtering is used mostly to predict the position of the *target* one cycle in advance. This will compensate for the delay between the acquisition of the images and the reaction of the system (due to the computational burden).

VII. Conclusion

This article reports the integration of an active vision system in a mobile platform. It describes a control scheme used for real-time *pursuit* of objects moving in front of the vehicle. The *pursuit* process controls the system to maintain the initial orientation and distance to the object. The control is based on multiple independent processes, controlling different degrees of freedom of the vision system and the mobile robot's position and orientation. The system is able to operate at approximately human walking rates, and experimental results were presented with the robot following a person. The system has limitations and some of them were already discussed and established as assumptions in Section I.

Future developments will address the problem of different objects moving in front of the mobile robot. These extensions also include the use of more elaborated control schemes and the improvement of the image processing hardware and routines.

TABLE I
Parameters of the Step Motors used in the Active Vision System

Movement Type	Maximum Offset	Resolution	Steps/ Shifting Units
Vergence	$\pm 34.2°$	$\pm 0.18°$	$\approx 6 steps/°$
Baseline	$2 \times 138 mm$	$0.15 mm/100 steps$	$\approx 6 steps/mm$
Head tilt	$\pm 25°$	$\pm 0.01°$	$\approx 100 steps/°$
Neck pan	$\pm 100°$	$\pm 0.01°$	$\approx 100 steps/°$

Appendix A
Vision System and Robot Parameters

A. Active Vision System

The vision system can be divided into three major areas: the platform that supports the cameras, the mechanical system that permits the orientation of that platform and the motorized lenses. The platform allows for the cameras to execute the movement *A* shown in Fig. 26 (the vergence) and also the control of the distance between them (the baseline shifting). The mechanical system that supports the platform is responsible for the execution of the movements *B* and *C* shown in Fig. 26 (the head tilt and the neck pan respectively). The cameras' lenses are motorized, using direct current motors to control the degrees of freedom corresponding to the iris, focus and zoom. The lenses used can have a focal length between 12.5 and 75 mm, focusing between 1.3 m and the ∞ depending on the focal length.

The offsets permitted to the various degrees of freedom associated with the active vision system are listed in Table I, resulting from the limits imposed to the system by the electrical and mechanical limits, and due to the mechanical structure of the system itself.

The velocities reached by the step motors depend on the values of the parameters measured experimentally. The values obtained do not permit movements from one end-position to the other in acceptable periods of time (once a moving command is issued to the controllers, the motors cannot be interrupted until they stop). Therefore, any movement requested must be divided into smaller movements that can be completed by the motors during one control period.

Typically, the control period for the vergence motors is 0.2 s with a maximum of 40 steps, and for the neck pan motor is 0.2 s with a maximum of 100 steps. The choice for these periods resulted from a compromise between the reaction speed of the system and the amplitude of the movements executed.

B. Mobile Robot System

The movements executed by the mobile robot are based on two direct current motors associated with each of the driving wheels (rear axle). The control of the motors is done by a multi-processing system based on the 68 020 CPU, installed

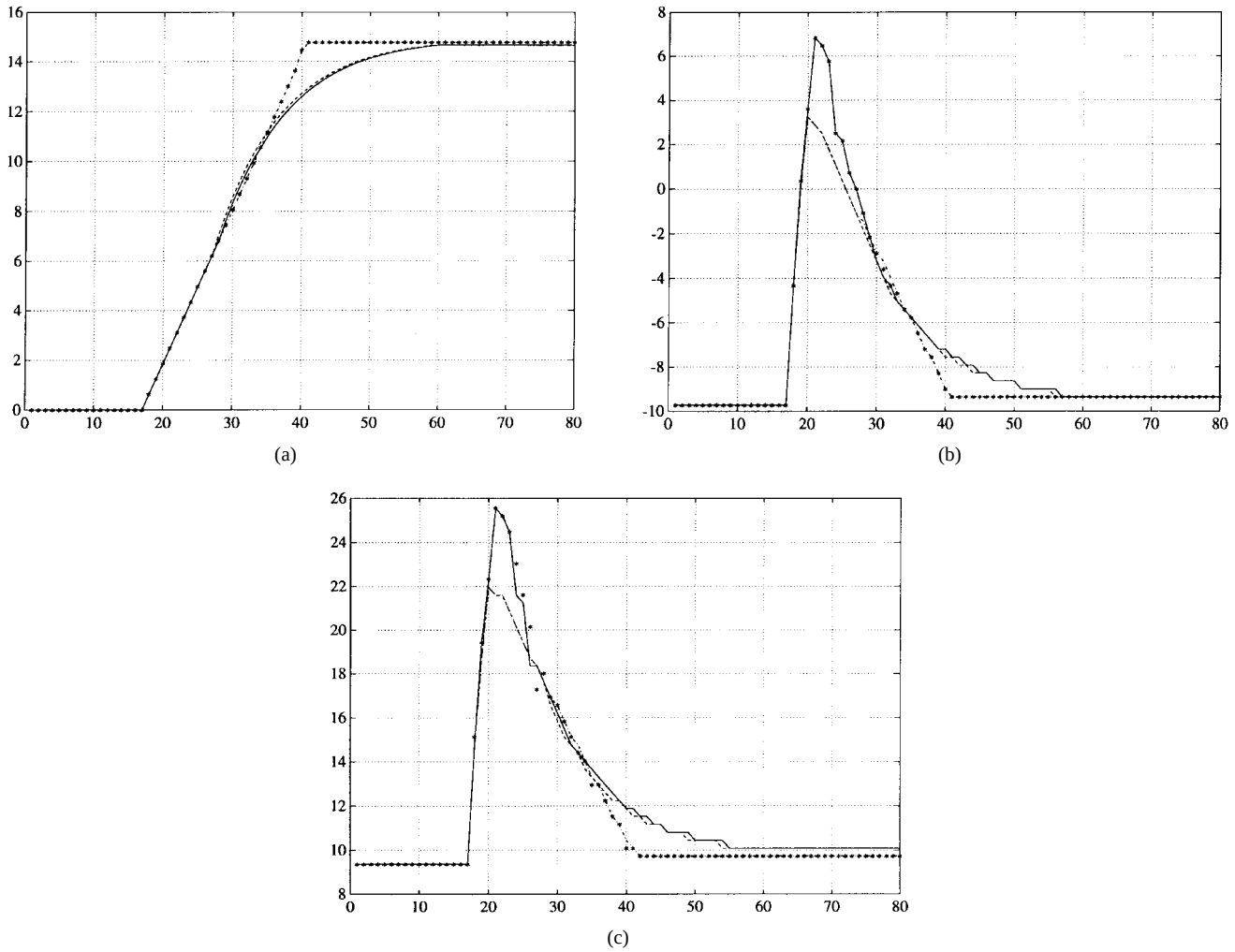


Fig. 22. Tracing of some of the system's variables with movements with compensation and prediction. (a) Angle of the neck motor. The response is slower when the gain is 0.1. (b) Angle of the right camera motor. The prediction results in a stronger signal. (c) Angle of the left camera motor. The prediction results in a stronger signal. The type of filtering for each curve is: *continuous line*—filtering with prediction and a proportional gain of 0.1; *dashed line*—filtering without prediction and a proportional gain of 0.1; *star line*—filtering with prediction and a proportional gain of 0.6; *Dash-dot* line—filtering without prediction and a proportional gain of 0.6.

on the mobile platform. Therefore, the only responsibility of the *Master Processing Unit* is to send to the platform the parameters of the required movement.

The robot's movements can be divided into three groups: translational (no angular velocity), rotation around the center of the driving axle and compositions of both movements. To define each one of these three movements, it is necessary to supply not only the values for the linear and angular velocities (v, ω), but also the duration time of the movement (T).

The valid values for the velocities and the duration time depend on the type of movement, and are represented in Table II. If the movement is of the composed type, the radius of the trajectory and the trajectory arc can be obtained with the following relations (*T* designates the time available to execute the movement)

$$r[\text{mm}] = \frac{v[\text{mm/s}]}{\omega[\text{mrad/s}]} 10^3 \tag{14}$$

TABLE II
PARAMETERS FOR THE CONTROL OF MOBILE PLATFORM'S VELOCITY

Movement Type	Translational	Rotational	Composed
Linear Velocity (mm/s)	$\pm 1000, ..., \pm 50, 0$	0	$\pm 1000, ..., \pm 50, 0$
Angular Velocity $(mrad/s)$	0	$\pm 1000, ..., \pm 100, 0$	$\pm 1000, ..., \pm 100, 0$
Trajectory Radius	0	0	$> 400mm$
Duration Time (40ms **units**)	> 0	> 0	> 0

$$\theta[\text{mrad}] = \frac{\omega[\text{mm/s}]}{T[\text{s}]}. \tag{15}$$

The movements described are based on the control of the velocity of the platform, and can be stopped or replaced at

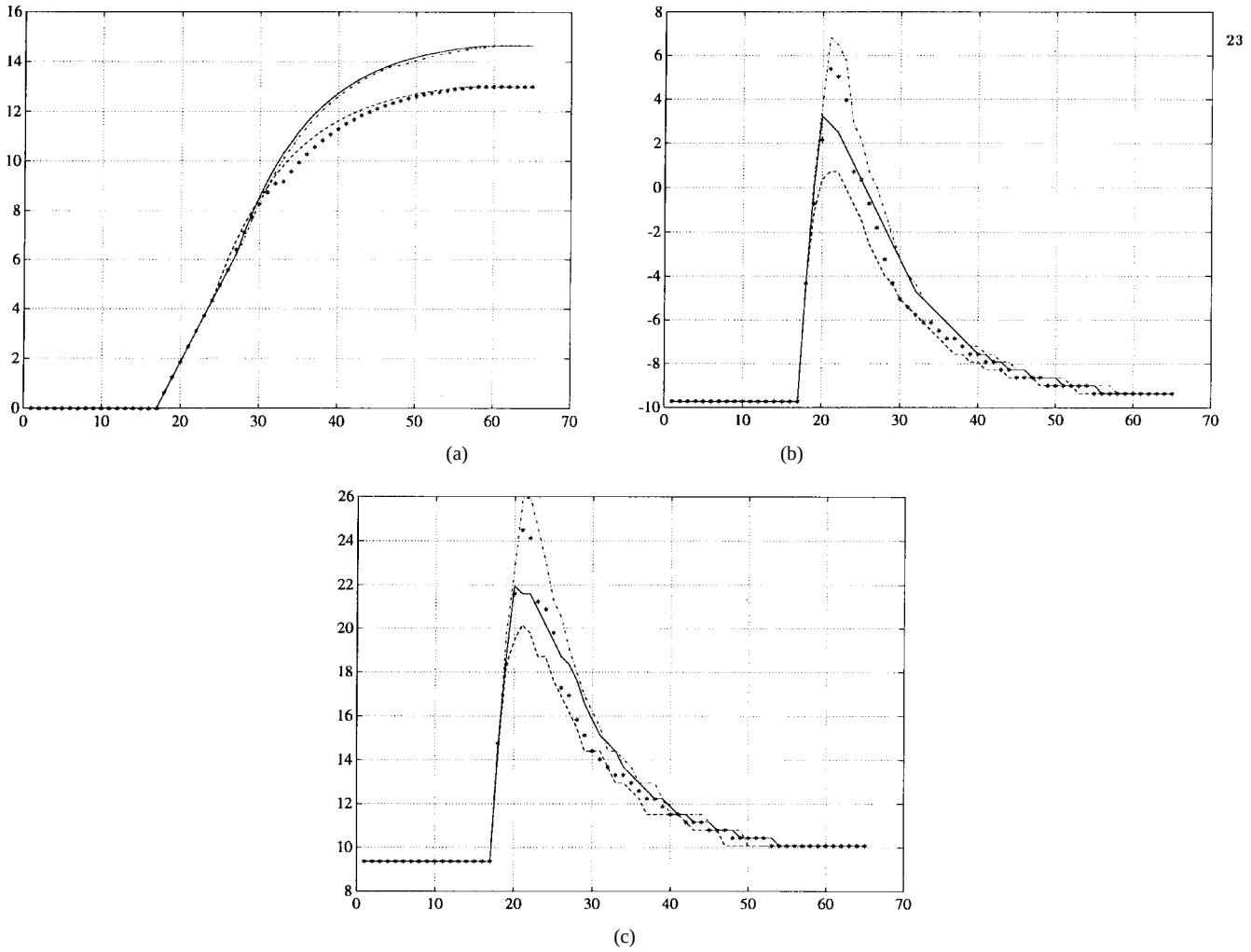


Fig. 23. Movements with compensation and prediction with smaller proportional gains (0.1). (a) Angle of the neck motor. The vertical axis represents θ_n (degrees) for diferent time clocks (k) represented on the horizontal axis. (b) Angle of the right camera motor. The vertical axis represents θ_r (degrees) for diferent time clocks (k) represented on the horizontal axis. (c) Angle of the left camera motor. The vertical axis represents θ_l (degrees) for diferent time clocks (k) represented on the horizontal axis. *Continuous line*—filtering without prediction or compensation; *dashed line*—filtering without prediction and with compensation; *star line*—filtering with prediction and compensation; *dash-dot line*—filtering with prediction and without compensation.

any time by another movement with other parameters. The transition between movements is controlled by the multiprocessing system, assuring the smoothness of the global movement of the robot.

The following lines are examples of commands that can be issued to the platform to launch velocity controlled movements issuing a:

1) composed movement: "MOTV LA V = 100 W = 100 T = 50";
2) pure rotation movement: "MOTV LA V = 0 W = 100 T = 50";
3) a linear movement: "MOTV LA V = 100 T = 50".

APPENDIX B
QUADRATIC SUB PIXEL INTERPOLATION

The objective of this appendix is to explain the mechanism used to find the maximum of a function of which only discrete samples are known. Since we are using images, the samples are discrete values of the function and we use interpolation to find the function's maximum with sub pixel precision (Fig. 27).

Suppose that the function is parameterized by the polynomial

$$y = ax^2 + bx + c \tag{16}$$

with the extreme point at $x_{\max} = -(b/2a)$. If the discrete points are at ${}^{-}x = x - d$ and ${}^{+}x = x + d$ with d the distance between samples, their values for the function $f(x)$ are given by

$$\begin{aligned} f({}^{-}x) &= ax^2 - 2adx + ad^2 + bx - db + c \\ f({}^{+}x) &= ax^2 + 2adx + ad^2 + bx + db + c \\ f(x) &= ax^2 + bx + c. \end{aligned} \tag{17}$$

Combining these equations we obtain the $x_{\max}$ position is given by

$$x_{\max} = x - \frac{d}{2}\frac{[f({}^{+}x) + f({}^{-}x)]}{[f({}^{-}x) - 2f(x) + f({}^{+}x)]}. \tag{18}$$

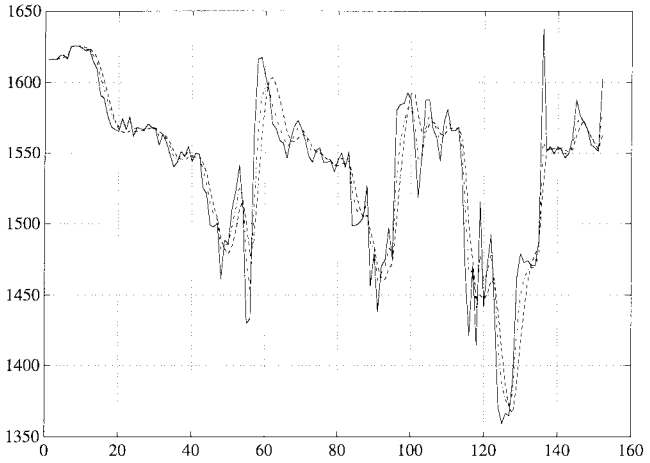


Fig. 24. Trace of the state variable D during a execution of *pursuit*. The distance is measured in meters, for a continuous set of time samples (k) represented on the horizontal axis. The curves have the following meaning: *continuous line*—The distance measured by the system; *dash-dot line*—the low-pass filtering of the distance; *dashed line*-$(\alpha$-β-$\gamma)$ filtering after the low-pass filter as been applied.

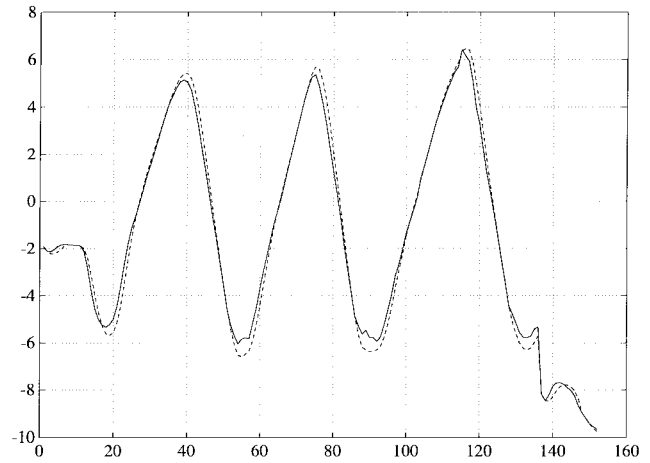


Fig. 25. Trace of the state variable θ_n during a execution of *pursuit*. The angle is measured in degrees, for a continuous set of time samples (k) represented on the horizontal axis. The angle θ. The curves have the following meaning: *continuous line*—the angle measured by the system; *dashed line*-$(\alpha$-β-$\gamma)$ filtering of the angle.

APPENDIX C
FILTERING AND PREDICTION

Filtering and prediction methods can be used to estimate the present and the future of the *target* kinematics quantities such as position, velocity, and acceleration. This is specially useful for tracking and also for estimating the state variables of a system from measurements made on it. In the current work we use the prediction capabilities of the filters to compensate for the delays in the system.

There are two common approaches to filtering and prediction. The first is to use fixed coefficients (α-β and α-β-γ filters), and the second, Kalman filtering, which generates time-variable coefficients that are determined by an a priori model for the statistics of the measured noise and *target* dynamics. The first approach has computational advantages, but Kalman filtering performs a high-accuracy tracking. The

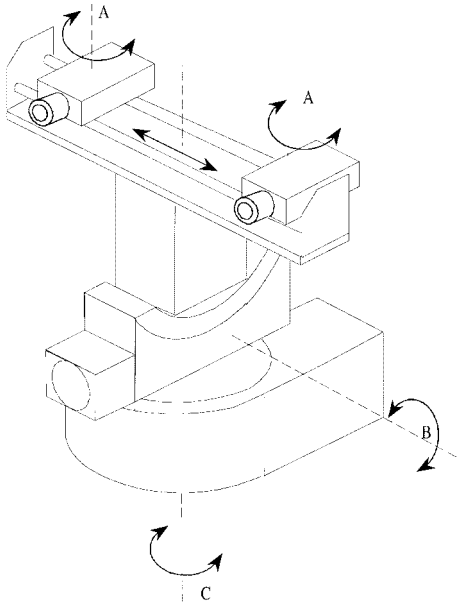


Fig. 26. Mechanical degrees of freedom of the active vision system.

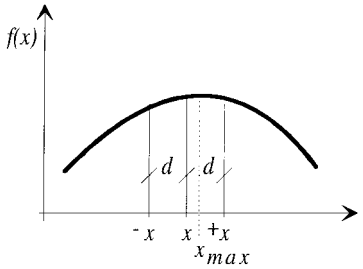


Fig. 27. Polynomial representation.

filter used in this system is of the first type, and can be used in linear dynamic systems with time-invariant coefficients in their state transition matrix and measurement equations. For these systems the filter gain achieves steady-state values that can often be computed in advance. This advantage is important to save some computational time and in practice both approaches are valid in our system, since we do not expect sudden changes in our model's parameters.

Both of these filters can be implemented recursively and the data received in the past is included in the present estimates. Therefore, all data is used but forgotten at an exponential rate. The estimate of the variable value at time k is $\hat{x}(k|k)$ and will be denoted by the smoothed estimate x_s. With these filters we can predict the values for the next step. The one-step prediction is denoted by $x_p = x(k+1|k)$, and signifies the estimate at time $k+1$ given data through time k.

Fixed coefficients filters have the advantage of simple implementation using fixed parameters for the filter's gains. The most extensively applied of these filters is the α-β tracker. This filter is used with constant velocity models when only position measurements are available. The α-β tracker is defined by the following equations:

$$x_s(k) = \hat{x}(k|k) = x_p(k) + \alpha[x_0(k) - x_p(k)] \quad (19)$$

$$v_s(k) = \hat{\dot{x}}(k|k) = v_s(k-1) + \frac{\beta}{qT}[x_0(k) - x_p(k)] \quad (20)$$

$$x_p(k+1) = \hat{x}(k+1|k) = x_s(k) + Tv_s(k). \quad (21)$$

The variable T is the sampling interval, $x_0(k)$ is the measurement at time k and the α and β are the fixed filter gain coefficients. The quantity q is normally defined as one, but in the case where missing observations occur its value may be taken as the number of scan steps (interactions) since the last measurement. The initialization process can be defined by

$$\begin{aligned} x_s(1) &= x_p(2) = x_0(1) \\ v_s(1) &= 0 \\ v_s(2) &= \frac{[x_0(2) - x_0(1)]}{T}. \end{aligned}$$

The (19) is used directly when an observation is received at time k. The optimal values for α and β are derived in [2] and depend only on the ratio of the process noise standard deviation and the measurement noise standard deviation.

The logical extension of the α-β filter is the α-β-γ filter, which includes an estimate for the acceleration and can be used with the assumption of uniform acceleration. This filter makes a quadratic prediction instead of a linear one, and tends to be more sensitive to noise but better able to predict smoothly varying velocities. The equations for this filter are defined as

$$x_s(k) = \hat{x}(k|k) = x_p(k) + \alpha[x_0(k) - x_p(k)] \quad (22)$$

$$v_s(k) = \hat{\dot{x}}(k|k) = v_s(k-1) + Ta_s(k-1) + \frac{\beta}{qT}[x_0(k) - x_p(k)] \quad (23)$$

$$a_s(k) = \hat{\ddot{x}}(k|k) = a_s(k-1) + \frac{\gamma}{(qT)^2}[x_0(k) - x_p(k)] \quad (24)$$

$$x_p(k+1) = \hat{x}(k+1|k) = x_s(k) + Tv_s(k) + \frac{T^2}{2}a_s(k). \quad (25)$$

The usual initialization is

$$\begin{aligned} x_s(1) &= x_p(2) = x_0(1) \\ v_s(1) &= a_s(1) = a_s(2) = 0 \\ v_s(2) &= \frac{[x_0(2) - x_0(1)]}{T} \\ a_3(3) &= \frac{[x_0(3) - 2x_0(2) + x_0(1)]}{T^2}. \end{aligned}$$

The optimal values for α and β are defined as [2] and the optimal value for γ is given by

$$\gamma = \frac{\beta^2}{\alpha}. \quad (26)$$

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the anonymous referees for their careful review and constructive comments on the earlier versions of this article.

REFERENCES

[1] P. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, "Automated tracking and grasping of a moving object with a robotic hand-eye system," *IEEE Trans. Robot. Automat.*, vol. 9, Apr. 1993.
[2] S. Blackman, *Multiple-Target Tracking with Radar Applications*. Boston, MA: Artech House, 1986, pp. 19–25.
[3] J. Bergen, P. Burt, R. Hingorani, and S. Peleg, "Computing two motions from three frames," David Sarnoff Research Center, Subsidiary of SRI Int., Princeton, NJ 08543-5300, Apr. 1990.
[4] C. Brown, "Gaze controls with interaction and delays," *IEEE Trans. Syst., Man., Cybern.*, vol. 20, pp. 518–527, May, 1990.
[5] P. Burt, C. Anderson, J. Sinniger, and G. van der Wal, "A pipelined pyramid machine," RCA David Sarnoff Research Center, NATO ASI Series, vol. F25, 1986.
[6] P. Burt, J. Bergen, R. Hingorani, R. Kolczynski, W. Lee, A. Leung, J. Lubin, and H. Shvaytser, "Object tracking with a moving camera," in *Proc. IEEE Workshop Visual Motion*, Irvine, CA, 1989.
[7] H. Carpenter, *Movements of the Eyes*. London, U.K.: Pion Ltd., 2nd ed.
[8] D. Coombs, "Real-time gaze holding in binocular robot vision," Ph.D. dissertation, Univ. of Rochester, Rochester, NY, June 1992.
[9] E. Grosso and D. Ballard, "Head-Centered Orientation Strategies in Animate Vaision," Tech. Rep. 442, Comput. Sci. Dept., Univ. of Rochester, Rochester, NY, Oct. 1992.
[10] P. Meer, E. Baugher, and A. Rosenfeld, "Frequency domain analysis and synthesis of image pyramid generating kernels," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 512–522, July 1987.
[11] N. Papanikopoulos, P. Khosla, and T. Kanade, "Visual tracking of a moving *target* by a camera mounted on a robot: a combination of control and vision," *IEEE Trans. Robot. Automat.*, vol. 9, pp. 14–34, Feb. 1993.
[12] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge, MA: MIT Press, 1981, pp. 9–11.
[13] C. M. Brown and D. Terzopoulos, Eds., *Real-time Computer Vision*. Cambridge, MA: Cambridge Univ. Press, 1994.
[14] J. L. Crowley and H. I. Christensen, Eds., *Vision as Process*. New York: Springer-Verlag, 1995.
[15] A. Blake and A. Yuille, Eds., *Active Vision*. Cambridge, MA: MIT Press, 1992.
[16] J. Dias, H. de Araújo, J. Batista, A. de Almeida, and C. Simplício,s "Implementation of an active vision system," in *Proc. Int. Workshop Mechatron. Comput. Syst. Perception Action*, June 1–3, 1993, Halmstad University, Sweden.
[17] J. Batista, J. Dias, H. de Araújo, and A. de Almeida, "Monoplanar camera calibration-iteractive multi-step approach," in *Proc. British Machine Vision Conf. BMVA'93*, Surrey, U.K., July 1993.
[18] H. Araújo, J. Dias, J. Batista, and P. Peixoto, "Autonomous robots and active vision systems: Issues on architectures and integration," in *Proc. Workshop Appl. Artif. Intell. Robot. Vision*, Funchal, Portugal, 1995.
[19] C. Samson and K. Ait-Abderrahim, "Mobile-robot control. Part 1: Feedback control of a nonholonomic wheeled cart in Cartesian space," Rapport de Recherche 1288, INRIA, Sophia Antipolis, France, Oct. 1990; *Active Vision*, A. Blake and A. Yuille, Eds. Cambridge, MA: MIT Press, 1992.
[20] *Theory of Robot Control*, C. Canudas de Wit, B. Siciliano, and G. Bastin, Eds. New York: Springer-Verlag, 1996.

Jorge Dias was born in Coimbra, Portugal, in 1960. He received the B.S. and Ph.D. degrees in electrical engineering from the University of Coimbra, Portugal, in 1984.

He is an Assistant Professor in the Department of Electrical Engineering, University of Coimbra and is a Researcher at the Institute of Systems and Robotics, Coimbra Pole. His primary research interest is in the area of computer vision, with an emphasis on the active vision and vision based navigation.

Carlos Paredes was born in Aveiro, Portugal, in 1971. He received the B.S. degree in electrical engineering from the University of Coimbra, Portugal, in 1994 and is currently pursuing the M.Sc. degree.

His research interests are in the area of computer vision and robotics.

Mr. Paredes received the PRAXIS XXI scholarship granted by Junta Nacional de Investigao Cientfica.

Inácio Fonseca was born in Coimbra, Portugal, in 1971. He received the B.S. degree in electrical engineering from the University of Coimbra, Portugal, in 1994 and is currently pursuing the M.Sc. degree.

He works as a Teaching Assistant at the Polytechnical Institute of Coimbra. His research interest is in the area of computer vision and robotics.

Helder Araújo is currently an Associate Professor in the Department of Electrical Engineering, University of Coimbra, and a Researcher in the Institute of Systems and Robotics, Coimbra Pole. His primary research interests are in computer vision and mobile robotics.

Jorge Batista received the B.S. and M.Sc. degrees both from the University of Coimbra, Portugal, in 1986 and 1992, respectively, and is currently pursuing the Ph.D. degree in active vision.

Since 1986, he joined the Electrical Engineering Department, Faculty of Science and Technology, University of Coimbra, as a Research Assistant, working on camera calibration, computer vision, and active vision. He is also a Researcher at the Institute of System and Robotics, Coimbra Pole.

Anibal T. Almeida is currently Professor of the Electrical Engineering Department, University of Coimbra, Portugal. His research interests include industrial automation and robotics.

Dr. Almeida is also Director of the Institute of Systems and Robotics, Coimbra Pole.