Multi-Layer Perceptrons for Task Visual Servoing in Robotics

Nadine Rondel, Gilles Burel

International Workshop on Fuzzy Logic and Neural Networks in Engineering, co-located with EPIA'95, October 3-6th , 1995, Funchal, Madeira Island, Portugal.

Multi-Layer Perceptrons for Task Visual Servoing in Robotics

Nadine RONDEL [†‡] & Gilles BUREL [†]

[†] L.E.S.T., Faculté des Sciences, 6 Av Le Gorgeu, 29200 Brest, France

[‡] SEFT, 18 rue du Dr. Zamenhof, 92130 Issy-Les-Moulineaux, France

Abstract. A six-degree-of-freedom robot with a CCD-camera at the extremity of its end-effector, is associated to a closed loop control system. An object moves freely in the 3D space (translation + rotation). The aim is to servo the camera on the object, so that the image of the object on the camera screen is always the "reference image", defined by an arbitrary reference position of the object versus the camera.
Classical mechanics equations are first studied, in order to determine the significant parameters of the problem. Whereupon, a control system based on the use of a MultiLayer Perceptron (MLP) is built. Results and comparisons detailing the behavior of the system in different cases (uniform and non-uniform translations and rotations of the object) are presented, and show the interest of the approach vs classical methods.

1 Introduction

This paper is interested in the following visual task servoing problem: a 6 degree-of-freedom robot with a camera-in-the-hand is assigned to follow the movement of an object in the 3D space, as shown in figure (1). An arbitrary picture of the object is first taken by the camera and is considered henceforth as the "reference image" associated to this "reference position" of the camera relatively to the object. As the object moves, it is desired that the control system should send commands to the robot, so that the relative position between the camera and the object should be as close as possible to its "reference position". This task is said to be servoed visually, in the sense that the error criterion is the discrepancy between the current and the reference image.
In order to find a neural solution to this control problem, classical equations are first derived (section 2). Then, in section 3, the significant parameters of the problem are detailed, wherefrom a neural solution is investigated (section 4). Finally, section 5 provides experimental results and comparisons with previous work. Section 6 draws a conclusion on the neural approach.

2 General equations

The pinhole camera model is first presented, in order to define which parameters are necessary in the equations of the system. These parameters will then be calculated using the classical equations of mechanics.

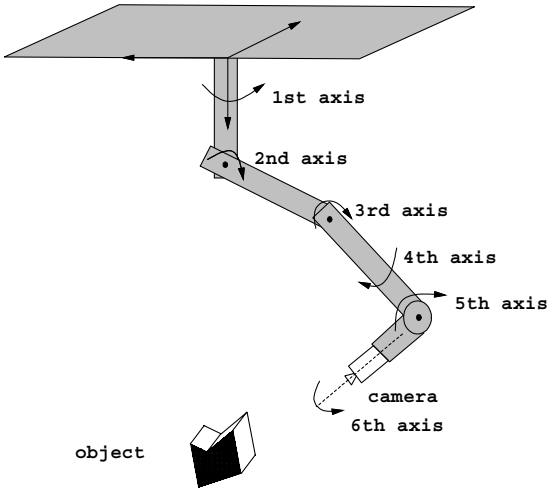

Fig. 1. *A general overview of the system.*

2.1 Pinhole camera model

The pinhole camera model can be defined as a convention ruling the projection of points of the 3D space onto the image plane, following notations of figure (2).

The main pinhole camera model equations are:

$$\begin{cases} u = \frac{f}{S_X}\frac{X_C}{Z_C} + u_0 = x + u_0 \\ v = \frac{f}{S_Y}\frac{Y_C}{Z_C} + v_0 = y + v_0 \end{cases}$$

- where (u, v) are the coordinates of point p on the image plane (in pixels),
- (u_0, v_0) are the coordinates of the optical center on the image plane,
- (X_C, Y_C, Z_C) are the coordinates of the objet point P_C in the camera coordinate system,
- parameters S_X and S_Y represent the pixel size, and f the focal.

Derivation of image parameters u and v with respect to time yields to:

$$\begin{cases} \frac{du}{dt} = \frac{f}{S_X}\left(\frac{1}{Z_C}\frac{dX_C}{dt} - \frac{X_C}{Z_C^2}\frac{dZ_C}{dt}\right) \\ \frac{dv}{dt} = \frac{f}{S_Y}\left(\frac{1}{Z_C}\frac{dY_C}{dt} - \frac{Y_C}{Z_C^2}\frac{dZ_C}{dt}\right) \end{cases}$$

that is to say, under matrix form:

$$\begin{bmatrix} \frac{du}{dt} \\ \frac{dv}{dt} \end{bmatrix} = \begin{bmatrix} \frac{f}{S_X Z_C} & 0 & -\frac{fX_C}{S_X Z_C^2} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{fY_C}{S_Y Z_C^2} \end{bmatrix} \begin{bmatrix} \frac{dX_C}{dt} \\ \frac{dY_C}{dt} \\ \frac{dZ_C}{dt} \end{bmatrix}$$

If we call $I = [x\ y]^T$, we have:

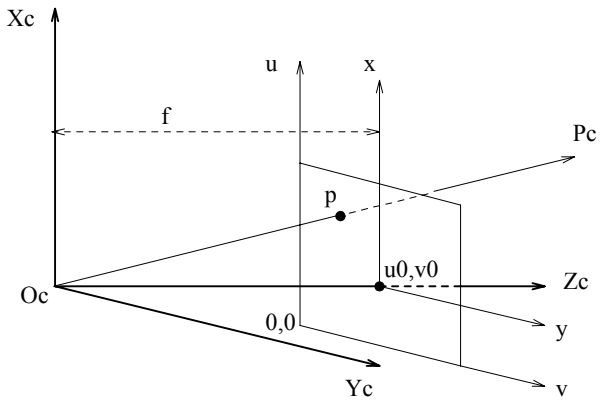

Fig. 2. *Pinhole camera model.*

$$\frac{dI}{dt} = \begin{bmatrix} \frac{f}{S_X Z_C} & 0 & -\frac{x}{Z_C} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{y}{Z_C} \end{bmatrix} \frac{dP_C}{dt} \tag{1}$$

Therefore $\mathbf{V}_P^C$, the speed of object point P_C in the camera referential, must be evaluated in order to write the equations of the system. This is where we use classical mechanics equations:

2.2 Classical mechanics equations

The intantaneous movement of an object can be defined by $C^{\mathcal{E}}(S)$:

$$C^{\mathcal{E}}(S) \equiv \begin{cases} \boldsymbol{\omega}(S/\mathcal{E}) \\ \mathbf{V}_P^{\mathcal{E}} \end{cases}$$

where $\mathcal{E}$ is the referential, S the object, and P a point of S. $\boldsymbol{\omega}(S/\mathcal{E})$ is the instantaneous rotation speed of S with respect to $\mathcal{E}$, and $\mathbf{V}_P^{\mathcal{E}}$ is the translation speed of point P in $\mathcal{E}$.

Let R_0 be a still referential, and R_1 a (possibly) moving referential. The relation describing the composition of movements is:

$$\mathbf{V}_P^{R_0} = \mathbf{V}_P^{R_1} + \mathbf{V}_{P\in R_1}^{R_0}$$

where the last term represents the speed in R_0 of the still point of R_1 that coincides with P at this very moment.

If we apply this relation to the still referential R of the robot and the referential C of the camera, we get:

$$\mathbf{V}_P^R = \mathbf{V}_P^C + \mathbf{V}_{P\in C}^R$$

which yields to the value we were looking for:

$$\mathbf{V}_P^C = \mathbf{V}_P^R - \mathbf{V}_{P\in C}^R$$

To help in the calculation of these two terms, let us write the fundamental relation of kinematics:

$$\forall (P,Q) \in S^2, \mathbf{V}_Q^{\mathcal{E}} = \mathbf{V}_P^{\mathcal{E}} + \boldsymbol{\omega}(S/\mathcal{E}) \wedge \overrightarrow{PQ}$$

and let us use it for $\mathbf{V}_{P\in C}^R$:

$$\mathbf{V}_{P\in C}^R = \mathbf{V}_{O_C}^R + \boldsymbol{\omega}(C/R) \wedge \overrightarrow{O_C P}$$

where O_C is the origin of referential C. For the same respect, we can write for a point P of the object:

$$\mathbf{V}_P^R = \mathbf{V}_{O_O}^R + \boldsymbol{\omega}(O/R) \wedge \overrightarrow{O_O P}$$

where O_O is the origin of the referential of the object.

Finally, in the camera referential, the speed of a point of the object can be written as:

$$\mathbf{V}_P^C = \mathbf{V}_{O_O}^R + \boldsymbol{\omega}(O/R) \wedge \overrightarrow{O_O P} - \mathbf{V}_{O_C}^R - \boldsymbol{\omega}(C/R) \wedge \overrightarrow{O_C P} \tag{2}$$

It is interesting to note that the two first terms can be interpreted as the influence of the absolute movement of the object in the camera referential, and the two last terms as the influence of the movement of the camera.

3 Problem parameters

It is the fusion of the pinhole camera model approach with the classical mechanics approach that enables the writing of the problem equations.

3.1 Fundamental equation

The pinhole model described in equation (1) shows that the speed of an object point P_C in the image depends only on the following parameters: its position in the image (x and y), its depth in the camera referential (Z_C), and its speed $\mathbf{V}_P^C$ in the camera referential:

$$\frac{dI}{dt} = \begin{bmatrix} \frac{f}{S_X Z_C} & 0 & -\frac{x}{Z_C} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{y}{Z_C} \end{bmatrix} \mathbf{V}_P^C$$

Remember that we said from equation (2) that $\mathbf{V}_P^C$ is the sum of two distinct terms: one term is due to the absolute movement of the object in the still referential, and the other term is due to the movement of the camera. Thence we can write:

$$\frac{dI}{dt} = \frac{dI_C}{dt} + \frac{dI_O}{dt} \tag{3}$$

where $\frac{dI_C}{dt}$ is the term associated to the object movement in the image due to camera movements only, and $\frac{dI_O}{dt}$ is the term associated to the object movement in the image due to the object own movements.

Let the value of $\frac{dI_C}{dt}$ be explained: the movement of the camera in the absolute (or still) referential R produces a movement of the points in the image such that:

$$\frac{dI_C}{dt} = \begin{bmatrix} \frac{f}{S_X Z_C} & 0 & -\frac{x}{Z_C} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{y}{Z_C} \end{bmatrix} \left(-\mathbf{V}^R_{O_C} - \boldsymbol{\omega}(C/R) \wedge \overrightarrow{O_C P} \right) \tag{4}$$

Yet a simple computation gives:

$$\forall (\mathbf{a}, \mathbf{b}),\ \mathbf{a} \wedge \mathbf{b} = \tilde{a}.\,\mathbf{b}$$

where

$$\tilde{a} = \begin{bmatrix} 0 & -a_Z & a_Y \\ a_Z & 0 & -a_X \\ -a_Z & a_X & 0 \end{bmatrix}$$

therefore we can write

$$\begin{aligned} \boldsymbol{\omega}(C/R) \wedge \overrightarrow{O_C P} &= -\overrightarrow{O_C P} \wedge \boldsymbol{\omega}(C/R) \\ &= -\widetilde{O_C P}.\,\boldsymbol{\omega}(C/R) \end{aligned}$$

Equation (4) then rewrites, under the matrix form:

$$\frac{dI_C}{dt} = \begin{bmatrix} \frac{f}{S_X Z_C} & 0 & -\frac{x}{Z_C} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{y}{Z_C} \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 & -Z_C & Y_c \\ 0 & -1 & 0 & Z_C & 0 & -X_C \\ 0 & 0 & -1 & -Y_C & X_C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{V}^R_{O_C} \\ \boldsymbol{\omega}(C/R) \end{bmatrix}$$

which gives, using the definitions of x and y:

$$\frac{dI_C}{dt} = B \begin{bmatrix} \mathbf{V}^R_{O_C} \\ \boldsymbol{\omega}(C/R) \end{bmatrix}$$

where B is the 6×6 following matrix:

$$\mathrm{B} = \begin{bmatrix} -\frac{f}{S_X Z_C} & 0 & \frac{x}{Z_C} & \frac{S_Y}{f} xy & -\left(\frac{f}{S_X} + \frac{S_X}{f} x^2 \right) & \frac{S_Y}{S_X} y \\ 0 & -\frac{f}{S_Y Z_C} & \frac{y}{Z_C} & \frac{f}{S_Y} + \frac{S_Y}{f} y^2 & -\frac{S_X}{f} xy & -\frac{S_X}{S_Y} x \end{bmatrix} \tag{5}$$

Let us call $\mathbf{u}_C$ the 6×1 vector defined by:

$$\mathbf{u}_C = \begin{bmatrix} \mathbf{V}^R_{O_C} \\ \boldsymbol{\omega}(C/R) \end{bmatrix}$$

and rewrite equation (3) using this notation:

$$I^{\bullet}(t) = \frac{dI}{dt} = B\mathbf{u}_C + \frac{dI_O}{dt}$$

At times t and $t-1$ we can write:

$$I^{\bullet}(t) = B(t)\mathbf{u}_C(t) + I_0^{\bullet}(t)$$
$$I^{\bullet}(t-1) = B(t-1)\mathbf{u}_C(t-1) + I_0^{\bullet}(t-1)$$

If we assume that the time interval between two snapshots is short enough, we can make the hypothesis that the movements of points in the image due to object movements are uniform, that is to say $I_0^{\bullet}(t)$ is constant over time. Therefore, the two preceeding equations can be subtracted and simplified:

$$I^{\bullet}(t) - I^{\bullet}(t-1) = B(t)\mathbf{u}_C(t) - B(t-1)\mathbf{u}_C(t-1)$$

Finally, with notations $I^{\bullet}(t) = \frac{I(t+1)-I(t)}{T}$, and $\mathrm{B} = T.B$, we get the fundamental equation of the problem:

$$\mathrm{B}(t)\mathbf{u}_C(t) = I(t+1) - 2I(t) + I(t-1) + \mathrm{B}(t-1)\mathbf{u}_C(t-1) \tag{6}$$

3.2 Estimators and prediction

The original task visual problem can now be defined clearly, using the introduced notations, as the estimation of the instantaneous speed of the camera $\mathbf{u}_C(t)$ as a function of parameters B, I and $\mathbf{u}_C$ at times gone by. The estimated speed vector $\mathbf{u}_C(t)$ is assumed to bring the camera in a position such that, at time $t+1$, the image $I(t+1)$ on the camera screen is the "reference image" I_{ref}. Furthermore, an estimation of $\mathbf{u}_C(t)$ using fundamental equation (6) requires the knowledge of $I(t)$. However, since $\mathbf{u}_C(t)$ is to be calculated *before* time t, and since $I(t)$ is obviously unknown before time t, an estimation $\hat{I}(t)$ has to be made. Here again, an expression of the fundamental equation (6) at time $t-1$ is needed:

$$\hat{I}(t) = 2I(t-1) - I(t-2) + \mathrm{B}(t-1)\mathbf{u}_C(t-1) - \mathrm{B}(t-2)\mathbf{u}_C(t-2) \tag{7}$$

Finally, $\hat{I}(t+1) = I_{ref}$ and the estimation of $\hat{I}(t)$ are introduced in (6) to give the estimation of $\mathbf{u}_C(t)$:

$$\mathbf{u}_C(t) = \mathrm{B}^{+}(t)\{I_{ref} - 3I(t-1) + 2I(t-2) - \mathrm{B}(t-1)\mathbf{u}_C(t-1) + 2\mathrm{B}(t-2)\mathbf{u}_C(t-2)\} \tag{8}$$

Just before time t, parameters $I(t-1), I(t-2), \mathbf{u}_C(t-1)$ and $\mathbf{u}_C(t-2)$ are perfectly known. Moreover, it can be seen from equation (5) that $\mathrm{B}(t)$ depends on parameters $x(t), y(t)$ and $Z_C(t)$ only. $x(t)$ and $y(t)$ are easily deduced from $I(t)$, but the depth $Z_C(t)$ is more difficult to estimate. This is why most articles (see [2] and [3] for example) consider matrix B as a constant, using the fact that, when the camera follows the object nicely, the current image is constantly equal to the "reference image". Therefore B is calculated using the reference image, and its values are kept thereafter. The simplified expression of $\mathbf{u}_C(t)$ is now estimated as:

$$\mathbf{u}_C(t) = \mathrm{B}^{+}\{I_{ref} - 3I(t-1) + 2I(t-2)\} - \mathbf{u}_C(t-1) + 2\mathbf{u}_C(t-2) \tag{9}$$

In [4] Papanikolopoulos proposed a method to estimate matrix B at each iteration when the distance between the camera and the object is fixed and known, but reckoned that for full 3D tracking it is still an open problem.

4 A neural solution

We have seen that the classical solution to the task visual problem can be reduced to equation (9), therefore merely needs the knowledge of $I(t-1), I(t-2), \mathbf{u}_C(t-1)$ and $\mathbf{u}_C(t-2)$. Since these vectors clearly represent the significant parameters, it would be a good idea to train (see [6] and [1] for more details concerning the training of neural networks) a neural network (namely a Multi-Layer Perceptron) to learn this control task. Furthermore, having trained the MLP, it would be interesting to compare its performances to those of the classical solution. The main problem here, is to build a data base representative of the possible movements space, such that the network answers correctly to any movement or any series of movements of the object.

4.1 The training base

The training base we want to build is composed of associated input and output vectors. Obviously here, the output vector is a 6×1 vector representing the instantaneous speed $\mathbf{u}_C(t)$. In order to give the neural network the same chances as the classical method, the inputs will be composed of a vector comprising $I(t-1), I(t-2), \mathbf{u}_C(t-1)$ and $\mathbf{u}_C(t-2)$. This is a $2 \times 6 + 4 \times N$ vector, for I represents the x and y coordinates of N caracteristic points.

To begin with, we would like to remind that, for any pair of referentials R_0 and R_1, the transformation matrix between R_0 and R_1 is the 4×4 matrix defined by:

$$M_{R_0}^{R_1} = \begin{pmatrix} R & T \\ 0\,0\,0 & 1 \end{pmatrix}$$

where R is the rotation matrix between R_0 and R_1, and T is the translation vector between the origin of R_0 and the origin of R_1. $I(t-2)$ is the projection

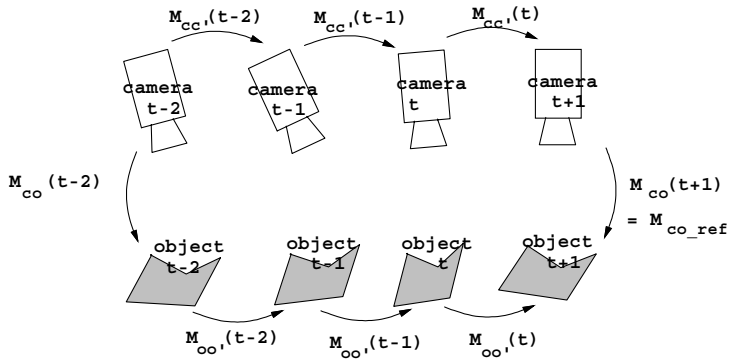

Fig. 3. *Evolution of the relative positions between object and camera through time.*

of the caracteristic points of the object onto the camera at time $t-2$. Therefore

to compute $I(t-2)$, we need $M_{co}(t-2)$, the transformation matrix between the referential of the object and the referential of the camera (see figure (3)). It is assumed that $I(t-2)$ and I_{ref} are close, hence $M_{co}(t-2)$ is considered as a "noisy" reference position matrix:

$$M_{co}(t-2) = M_{co\_ref} M_u(t-2)$$

where $M_u(t-2)$ is a "noise" random transformation matrix created by the following process: a "noise" rotation and translation vector $\mathbf{n}_u(t-2)$ is drawn uniformly, whereupon noise matrix $M_u(t-2)$ is calculated (see Appendix A) using Rodrigues formula [5].

Once this first position is set, a first translational and rotational object speed $U_0(t-2)$ is drawn uniformly at random. The transformation matrix towards the new object referential, $M_{oo'}(t-2)$, is calculated from $U_0(t-2)$ using Rodrigues formula as previously. The camera is supposed to follow this movement so as to keep its reference position with respect to the object: the "ideal" $M_{co}(t-1)$ matrix equals M_{co_ref} and:

$$M_{cc'}^{ideal}(t-2) = M_{co}(t-2) M_{oo'}(t-2) M_{co\_ref}^{-1} \tag{10}$$

The ideal camera speed $U_C(t-2)$ is extracted from $M_{cc'}^{ideal}(t-2)$ using the inverse Rodrigues formula. However, since usual movements are rarely perfect, a small random noise vector is added to $U_C(t-2)$: $\mathbf{u}_C(t-2) = U_C(t-2) + \mathbf{n}_C(t-2)$, from which the true $M_{cc'}(t-2)$ matrix is recalculated with Rodrigues formula. Equation (10) is then modified to let the new relationship between object and camera referentials appear:

$$M_{co}(t-1) = M_{cc'}^{-1}(t-2) M_{co}(t-2) M_{oo'}(t-2) \tag{11}$$

wherefrom $I(t-1)$ is deduced.

The whole process is rerun for the following iteration (from time $t-1$ to time t). However this time a constraint is applied, that comes from simple physical considerations: for an object moving at reasonable speed, and for sufficient sampling over time, it seems probable that the instantaneous speed of the object cannot vary tremendously between two iterations. Therefore $U_0(t-1)$ is assumed to obey:

$$U_0(t-1) = U_0(t-2) + \mathbf{d}_0(t-1)$$

where $\mathbf{d}_0(t-1)$ is a noise vector drawn at random between limits the tenth of the limits used in the drawing of $U_0(t-2)$.

The same process is rerun once more (from time t to time $t+1$): noise vector $\mathbf{d}_0(t)$ is drawn at random and $U_0(t) = U_0(t-1) + \mathbf{d}_0(t)$ yields to the computation of the object movement matrix $M_{oo'}(t)$. However now, since the network is supposed to be trained to cope perfectly with the various movements of the object, no spurious noise will be added to the "ideal" camera speed extracted from matrix

$$M_{cc'}(t) = M_{co}(t) M_{oo'}(t) M_{co\_ref}^{-1} \tag{12}$$

Therefore $\mathbf{u}_C(t)$ is calculated directly from $M_{cc'}(t)$ by the inversion of Rodrigues formula. We have now created randomly all the parameters necessary for the training of a network: inputs $I(t-2), I(t-1), \mathbf{u}_C(t-2), \mathbf{u}_C(t-1)$, and the output $\mathbf{u}_C(t)$. The drawing of parameters in random directions ensures the adaptativity of the network for any possible movement of the object. Furthermore, non constant object speed (speeds at time t, $t-1$ and $t-2$ are different) should provide the possibility to follow an object, even when its speed is varying (non uniform movement).

4.2 The network

The training base previously detailed comprises 1000 examples. A two-layer (linear) perceptron is trained on this base: in our experiments, we used an object with $N = 5$ caracteristic points (corners), presented on figure (4). Therefore $2 \times N = 10$ values are needed to represent every image I, which makes a

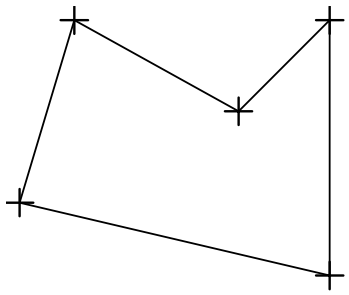

Fig. 4. *Location of the characteristic points of the object.*

$2 \times 2 \times N + 2 \times 6 = 32$ input vector for the network. The output $\mathbf{u}_C(t)$ comprises 6 units. A 32-input 6-output two-layer perceptron with linear activation function neurons is created, and trained on the database until the output error is sufficiently small: learning is considered optimal when the variance of the ouput errors of the network on the training base is smaller than the mean of the output errors of the classical approach [2] [3] on the same base. Once the network has learned, experiments are conducted on various movement cases: linear and non linear translations and rotations.

5 Experiments and discussion

The first part of the experiments is interested in uniform movement: the object translates or rotates at a constant speed, and the performance criterion is the mean square error between the current position and the reference position of the $N = 5$ caracteristic points over m iterations:

$$e_{MS}(m) = \frac{1}{m} \sum_{n=1}^{m} \sqrt{\frac{1}{N} \|I(n) - I_{ref}\|^2}$$

It is important to note that, in order to present more realistic experiments, the image pixels in $I(n)$ are rounded up to the nearest *even* value (instead of a usual round up to the nearest integer value). Moreover, a uniform pixel noise between -1 and 1 has been added to each pixel coordinate, so as to be as close as possible to real image processing system defects. In order to show the performances of both the classical method and the neural approach, various object speeds have been tested: figure (5a) shows the average error in pixels as a function of the translational speed of the object along the X axis (in cm/s). The sampling rate is 4 iterations/second, and each point of the graph represents the average over $m = 200$ iterations. The same experiment was conducted for a uniform rotation

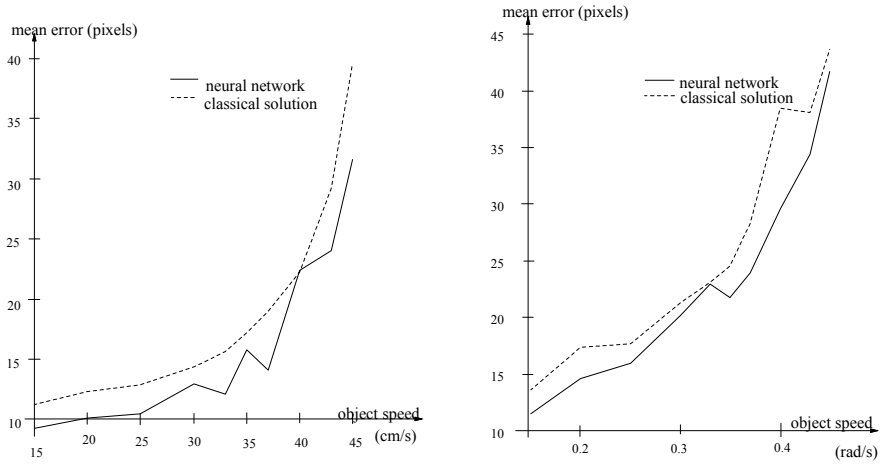

Fig. 5. *Compared performances of the classical method and the neural approach in the case of uniform object speeds: (a) translation, (b) rotation.*

of the object. As in the case of translation, it should be noticed that the neural network performances are very promising, and they even prove to be slightly better than the performances of the classical approach.

The case of non-uniform object movements is also investigated. In the first non-uniform experiment, the position of the object along the X-axis is a sinusoidal function following the form:

$$x_{pos}(n) = A\ sin(2\pi f n) \tag{13}$$

where n is the number of iterations. The amplitude A of the movement is set to 30cm, while the frequency f varies. For each frequency, an experiment is launched during 1000 iterations. The graph showing the average pixel error over 1000 iterations, with respect to the associated frequency, is depicted on figure (6a). The validity of the neural approach for non-uniform rotations is also illustrated: sinusoidal rotations around the X-axis are experienced (see figure (6b)) for a

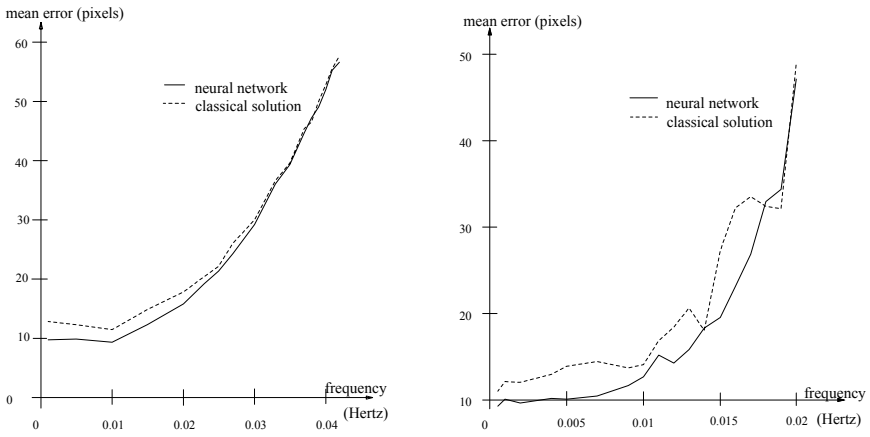

Fig. 6. *Compared performances of the classical method and the neural approach in the case of non-uniform object speeds: (a) sinusoidal translation, (b) sinusoidal rotation.*

rotation angle θ such that:

$$\theta(n) = \frac{\pi}{3} sin(2\pi f n) \tag{14}$$

Sinusoidal (therefore non-uniform) movements tend to prove that the neural approach can handle the problem in a satisfactory way. As is the case for uniform movements, the neural network can compete positively with the classical method, therefore showing its interesting and wide capabilities.

6 Conclusion and perspectives

In a visual task-reference problem, a robot with a camera at the extremity of its end-effector, is used to follow the movement of an object in the 3D space: the link between the camera and the object is desired to be rigid. After having identified the significant parameters, it is possible to design and train a two-layer perceptron dedicated to this control problem. Various experiments, including highly non-uniform movements for the object, have been presented and compared with the classical method, showing the interest of the approach. Furthermore, one advantage is the easiness to extend the neural approach to non linear loops, and/or control loops taking more information into account: such extensions merely require extra or new training.

Further studies include the extension to three-layer (non linear) perceptrons used in a particular way: once the two-layer perceptron has learned, a three-layer perceptron is trained to correct the two-layer MLP, that is to say its inputs are the same, but its outputs are the difference between the correct answer and the answer provided out of the two-layer MLP. This way the three-layer MLP

can only improve the performances of the two-layer MLP, though the learning phase is much quicker than a direct three-layer learning. Other extensions are interested in using more than two iterations in the prediction process, that is to say, for exemple, $I(t-3)$ and $\mathbf{u}_C(t-3)$ might be added to the network inputs.

References

1. G. Burel, "Réseaux de neurones en traitement d'images: des modèles théoriques aux applications industrielles", Ph.D. Thesis, Université de Bretagne Occidentale, Dec. 1991
2. F. Chaumette, P. Rives & B. Espiau, "Positioning of a Robot with respect to an Object, Tracking it and Estimating its Velocity by Visual Servoing", IEEE Int. Conf. on Robotics and Automation, Sacramento, CA, pp 2248-2253, Apr. 1991
3. B. Espiau, F. Chaumette & P. Rives, "A New Approach to Visual Servoing i Robotics", IEEE Trans. on Robotics and Automation, **8** (3), pp 313-326, June 1992
4. N.P. Papanikolopoulos, P.K. Khosla, & T. Kanade, "Visual Tracking of a Moving Target by a Camera Mounted on a Robot: a Combination of Control and Vision", IEEE Trans. on Robotics and Automation, **9** (1), Feb. 1993
5. O. Rodrigues, "Des loi géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire", Journal de Mathématiques Pures et Appliquées, 1st series, (5), pp 380-440, 1840
6. D. Rumelhart, & J. McClelland, "Parallel Distributed Processing", chapter 7, MIT Press, 1986

Appendix A

For a unit time the transformation matrix M_u associated to the 6×1 rotation and translation speed vector $\mathbf{u} = [\mathbf{t}, \boldsymbol{\omega}]^T$ is

$$M_u = \begin{pmatrix} R_u & t \\ 0\,0\,0 & 1 \end{pmatrix}$$

R_u is calculated with the help of Rodrigues formula:

$$R_u = I + \sin\theta U + (1 - \cos\theta)U^2$$

where I is the 3×3 identity matrix, θ is the norm (or length) of rotation vector $\boldsymbol{\omega}$, and U is the 3×3 antisymmetric matrix such that:

$$U = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}$$

where $\mathbf{v} = [a\ b\ c]^T$ is the unitary vector collinear to $\boldsymbol{\omega}$: $\mathbf{v} = \boldsymbol{\omega}/\|\boldsymbol{\omega}\|$.

This article was published in IEEE International Workshop on Fuzzy Logic and Neural Networks, Oct. 3-6, 1995, Funchal, Portugal