Vision Guided Servoing using Neural Networks

Nadine Rondel, Gilles Burel

IEEE Int. Conf. on Computational Engineering in Systems Applications (CESA'96)
Lille, France, July 9-12, 1996

Vision Guided Servoing using Neural Networks

Nadine RONDEL & Gilles BUREL
Laboratoire d'Electronique, UFR Sciences
6 Av Le Gorgeu, BP 809, 29285 Brest cedex, France
e-mail: Gilles.Burel@univ-brest.fr

IEEE Int. Conf. on Computational Engineering in Systems Applications (CESA'96), Lille, France, July 9-12, 1996

***Abstract*— In a closed loop control system, a six-degree-of-freedom robot with a CCD-camera mounted on its end-effector, is operated. An object moves freely in 3D space (translation + rotation). The aim is to servo the camera on the object, so that the image of the object is always close to the "reference image", defined by a given reference position of the object with respect to the camera.**
Classical kinematics equations are first studied in order to determine the significant parameters of the problem. Two neural approaches are then proposed: in the first solution, a Multi-Layer Perceptron (MLP) is fed with the image coordinates of feature points and with previous robot commands. In the second solution, different neural input parameters are used, that are based on affine transformations between succeeding image coordinates. Results and comparisons with the classical approach (uniform and non-uniform translations of the object) are presented, and show the interest of the approach vs. classical methods.

***Keywords*— Visual Servoing, Neural Networks, Automatic Learning, Kinematics**

I. Introduction

This paper is concerned with the following visual task servoing problem: a 6 degree-of-freedom robot with a camera-in-hand is made 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 thereafter as the "reference image" associated to this "reference position" of the camera relative to the object. As the object moves, it is desired that the control system 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.

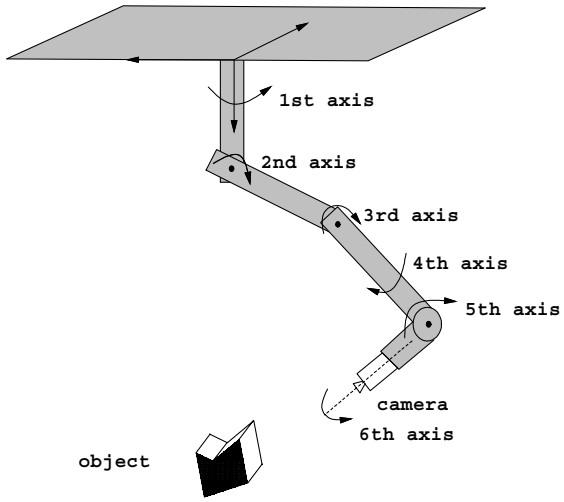

Figure 1: *A general overview of the system.*

In order to find a neural solution to this control problem, the pinhole camera model equations are first studied (Section 2). Then, in Section 3, the significant parameters of the problem are detailed, wherefrom two neural solutions are investigated (Section 4). Finally, Section 5 provides experimental results and comparisons with previous work. Section 6 draws conclusions on the neural approach.

II. Pinhole camera model

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 equations of classical mechanics.
The pinhole camera model can be defined as a convention defining the projection of points in 3D space onto

the image plane, following the notation of Figure (2). Its main equations are:

$$\left\{ \begin{array}{rcl} u & = & \frac{f}{S_X}\frac{X_C}{Z_C} + u_0 = x + u_0 \\ v & = & \frac{f}{S_Y}\frac{Y_C}{Z_C} + v_0 = y + v_0 \end{array} \right.$$

- 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 object point P_C in the camera coordinate system,
- parameters S_X and S_Y represent the pixel size, and f the focal length.

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

$$\left[\begin{array}{c} \frac{du}{dt} \\ \frac{dv}{dt} \end{array} \right] = \left[\begin{array}{ccc} \frac{f}{S_X Z_C} & 0 & -\frac{fX_C}{S_X Z_C^2} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{fY_C}{S_Y Z_C^2} \end{array} \right] \left[\begin{array}{c} \frac{dX_C}{dt} \\ \frac{dY_C}{dt} \\ \frac{dZ_C}{dt} \end{array} \right]$$

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

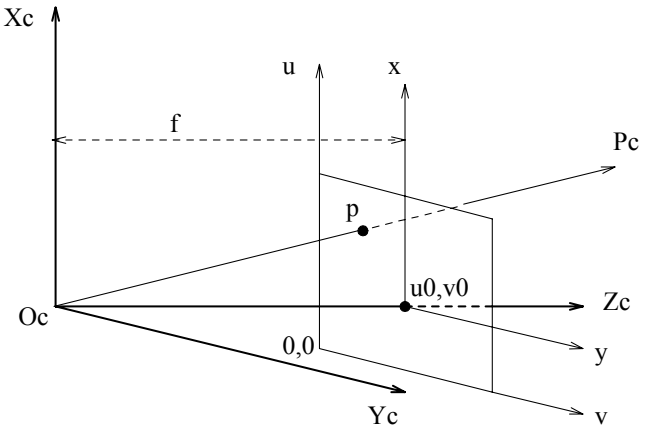

Figure 2: *Pinhole camera model.*

$$\frac{dI}{dt} = \left[\begin{array}{ccc} \frac{f}{S_X Z_C} & 0 & -\frac{x}{Z_C} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{y}{Z_C} \end{array} \right] \vec{V}_P^C \qquad (1)$$

Therefore $\vec{V}_P^C$, the speed of object point P_C in the camera frame, must be evaluated in order to write the system equations. Classical mechanics equations show that in the camera frame, the speed of an object point can be written as [5]:

$$\begin{array}{rcl} \vec{V}_P^C & = & \vec{V}_{O_O}^R + \vec{\omega}(O/R) \wedge \overrightarrow{O_O P} \\ & & -\vec{V}_{O_C}^R - \vec{\omega}(C/R) \wedge \overrightarrow{O_C P} \end{array} \qquad (2)$$

where for any frame $\mathcal{E}$, $O_\mathcal{E}$ stands as the origin of $\mathcal{E}$, and $\vec{\omega}(S/\mathcal{E})$ is the instantaneous rotation speed of object S with respect to $\mathcal{E}$.
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 frame, and the two last terms as the influence of the camera movement.

III. Problem parameters

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

A. 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 frame (Z_C), and its speed $\vec{V}_P^C$ in the camera frame.
Remember that we said from Equation (2) that $\vec{V}_P^C$ is the sum of two distinct terms: one term which is due to the absolute movement of the object in the fixed frame, and another which is due to the movement of the camera. Hence we can write:

$$\frac{dI}{dt} = \frac{dI_C}{dt} + \frac{dI_O}{dt} \qquad (3)$$

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

Let us explain the value of $\frac{dI_C}{dt}$: the movement of the camera in the absolute (or fixed) frame R produces a movement of the points in the image such that:

$$\begin{array}{rcl} \frac{dI_C}{dt} & = & \left[\begin{array}{ccc} \frac{f}{S_X Z_C} & 0 & -\frac{x}{Z_C} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{y}{Z_C} \end{array} \right] \\ & & \times \left(-\vec{V}_{O_C}^R - \vec{\omega}(C/R) \wedge \overrightarrow{O_C P} \right) \end{array} \qquad (4)$$

which can be rewritten in matrix form:

$$\begin{array}{rcl} \frac{dI_C}{dt} & = & \left[\begin{array}{ccc} \frac{f}{S_X Z_C} & 0 & -\frac{x}{Z_C} \\ 0 & \frac{f}{S_Y Z_C} & -\frac{y}{Z_C} \end{array} \right] \\ & \times & \left[\begin{array}{cccccc} -1 & 0 & 0 & 0 & -Z_C & Y_c \\ 0 & -1 & 0 & Z_C & 0 & -X_C \\ 0 & 0 & -1 & -Y_C & X_C & 0 \end{array} \right] \left[\begin{array}{c} \vec{V}_{O_C}^R \\ \vec{\omega}(C/R) \end{array} \right] \end{array}$$

which gives, using the definitions of x and y:

$$\frac{dI_C}{dt} = B \left[\begin{array}{c} \vec{V}_{O_C}^R \\ \vec{\omega}(C/R) \end{array} \right]$$

where B is the following 2×6 matrix:

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

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

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

and rewrite Equation (3) using this notation:

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

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

$$\begin{aligned} I^\bullet(t) &= B(t)\vec{u}_C(t) + I_0^\bullet(t) \\ I^\bullet(t-1) &= B(t-1)\vec{u}_C(t-1) + I_0^\bullet(t-1) \end{aligned}$$

If we assume that the time interval between two image acquisitions 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:

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

When we have N feature points, the matrices B are $2N \times 6$ matrices, and the size of vectors I is $2N$. 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:

$$\begin{aligned} \mathrm{B}(t)\vec{u}_C(t) &= I(t+1) - 2I(t) + I(t-1) \\ &\quad + \mathrm{B}(t-1)\vec{u}_C(t-1) \end{aligned} \quad (6)$$

B. Estimators and prediction

The original visual task problem can now be clearly defined, using the introduced notations, as the estimation of the instantaneous speed of the camera $\vec{u}_C(t)$ as a function of parameters B, I and $\vec{u}_C$ at past instants. The estimated speed vector $\vec{u}_C(t)$ is assumed to bring the camera into 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 $\vec{u}_C(t)$ using fundamental Equation (6) requires knowledge of $I(t)$. However, since $\vec{u}_C(t)$ 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:

$$\begin{aligned} \hat{I}(t) &= 2I(t-1) - I(t-2) \\ &\quad + \mathrm{B}(t-1)\vec{u}_C(t-1) - \mathrm{B}(t-2)\vec{u}_C(t-2) \end{aligned}$$

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

$$\begin{aligned} \vec{u}_C(t) &= \mathrm{B}^+(t)\{I_{ref} - 3I(t-1) + 2I(t-2) \\ &\quad - \mathrm{B}(t-1)\vec{u}_C(t-1) + 2\mathrm{B}(t-2)\vec{u}_C(t-2)\} \end{aligned}$$

where B^+ is the pseudo-inverse of B. Just before time t, parameters $I(t-1), I(t-2), \vec{u}_C(t-1)$ and $\vec{u}_C(t-2)$ 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 close to the "reference image". Therefore B is calculated using the reference image, and its values are kept thereafter. The simplified expression of $\vec{u}_C(t)$ is now estimated as:

$$\begin{aligned} \vec{u}_C(t) &= \mathrm{B}^+\{I_{ref} - 3I(t-1) + 2I(t-2)\} \\ &\quad - \vec{u}_C(t-1) + 2\vec{u}_C(t-2) \end{aligned} \quad (7)$$

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 judged that for full 3D tracking it is still an open problem.

IV. Neural solutions

We have seen that the classical solution to the visual task problem can be reduced to Equation (7), therefore merely requiring knowledge of $I_{ref}, I(t-1), I(t-2), \vec{u}_C(t-1)$ and $\vec{u}_C(t-2)$. Since these vectors clearly represent the significant parameters, it appears feasible 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 space of possible movements, such that the network responds correctly to any movement or any series of movements of the object.

A. The training set

The training set we want should consist of associated input and output vectors. Obviously, the output vector is a 6×1 vector representing the instantaneous

[1] For convenience, we write $t-1$, $t-2$, instead of $t-T$, $t-2T$, where T is the sampling period

speed $\vec{u}_C(t)$. In order to give the neural network the same data as in the classical method, the inputs will consist of a vector containing $I(t-1), I(t-2), \vec{u}_C(t-1)$ and $\vec{u}_C(t-2)$. This is a $2 \times 6 + 4 \times N$ vector, as I represents the x and y coordinates of N characteristic points.
The method used to create the training set is described in [5]. Basically, it consists in choosing random movements of the object and of the camera, under some constraints, as illustrated on Figure 3. Since the object movement is not chosen perfectly uniform, the network learns to follow an object, even when its speed is changing. $\vec{u}_C(t)$ is computed in order that the reference position is obtained at time $t+1$.

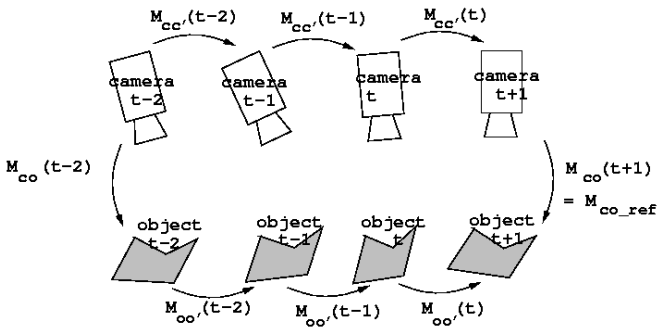

Figure 3: *Evolution of the relative positions between object and camera through time.*

B. A first network

Denoting $J(.) = I(.) - I_{ref}$, Equation (7) can be rewritten as:

$$\vec{u}_C(t) = \mathrm{B}^{+}\{2J(t-2) - 3J(t-1)\} - \vec{u}_C(t-1) + 2\vec{u}_C(t-2)$$

which shows that the significant parameters of this command equation are $J(t-1), J(t-2), \vec{u}_C(t-1)$ and $\vec{u}_C(t-2)$. These parameters are used as inputs to the first network. The training set previously mentioned comprises 2000 examples. An MLP with linear outputs is trained on this data: in our experiments, we used an object with $N = 5$ feature points (corners), presented in Figure (4). Therefore $2 \times N = 10$ values are needed to represent every image difference J, which makes a $2 \times 2 \times N + 2 \times 6 = 32$ input vector for the network. The output $\vec{u}_C(t)$ comprises 6 units. A 32-input, 6-output MLP 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 set

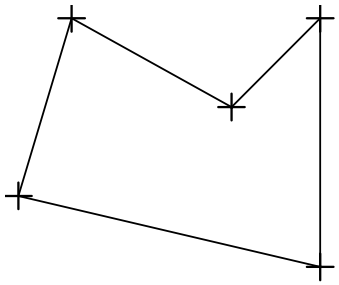

Figure 4: *Location of the feature points of the object.*

is smaller than the mean of the output errors of the classical approach [2] [3] on the same data.

C. Improving the neural approach

One disadvantage of the first neural approach, is that the size of the input layer is strongly dependent on the number N of characteristic points. Indeed, the size of input data such as $J(t-1) = I(t-1) - I_{ref}$, where each $I(t-1)$ represents the coordinates (x^i_{t-1}, y^i_{t-1}) of each of the N characteristic points, is obviously dependent on N.
It would be interesting to replace parameter vectors $J(t-1)$ and $J(t-2)$ by other, more compact, parameters, that would still represent $I(t-1), I(t-2)$ and I_{ref}, but would not depend on N. Instead of feeding the network with $J(t-2) = I(t-2) - I_{ref}$, for example, one could use the parameters of the affine transformation (2D space to 2D space) that transforms I_{ref} into $I(t-2)$.
Let P^i be a characteristic point of the object, and let P^i_{ref} be its reference position. The operation that transforms the 2D coordinates of P^i_{t-1} into the 2D coordinates of P^i_{ref} can be approximated by:

$$\begin{bmatrix} x^i_{ref} \\ y^i_{ref} \end{bmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{bmatrix} x^i_{t-1} \\ y^i_{t-1} \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

or, under matrix form:

$$I^i_{ref} = AI^i_{t-1} + D \tag{8}$$

If image $I(t-1)$ is not too different from image I_{ref}, the relation between points at time $t-2$ and points in the reference image, is the same for all points:

$$\forall i \in [1, N],\ I^i_{ref} = AI^i_{t-1} + D \tag{9}$$

Let G_{t-1} be the center of gravity of the 2D characteristic points in the image at time $t-1$ and let G_{ref} be their center of gravity in the reference position. We have:

$$G(t-1) \quad = \quad \frac{1}{N} \sum_{i=1}^{N} I^i_{t-1}$$

$$G_{ref} = \frac{1}{N}\sum_{i=1}^{N} I^i_{ref}$$

Relation (8) is still valid for the center of gravity:

$$G_{ref} = AG_{t-1} + D$$

therefore we can express the unknown 2×1 vector D as a function of A:

$$D = G_{ref} - AG_{t-1}$$

Replacing D in (9), we get:

$$\forall i \in [1, N],\ I^i_{ref} = AI^i_{t-1} + (G_{ref} - AG_{t-1})$$

that is to say:

$$\forall i \in [1, N],\ \left[I^i_{ref} - G_{ref}\right] = A\left[I^i_{t-1} - G_{t-1}\right]$$

Denoting

$$\begin{aligned} \bar{I}^i_{ref} &= I^i_{ref} - G_{ref} \\ \text{and } \bar{I}^i_{t-1} &= I^i_{t-1} - G_{t-1} \end{aligned}$$

yields a very simple matrix equation:

$$\forall i,\ \bar{I}^i_{ref} = A\bar{I}^i_{t-1} \tag{10}$$

Thus, with

$$\begin{aligned} K_{ref} &= \left[\bar{I}^1_{ref}, \bar{I}^2_{ref}, ..., \bar{I}^N_{ref}\right] \\ \text{and } K_{t-1} &= \left[\bar{I}^1_{t-1}, \bar{I}^2_{t-1}, ..., \bar{I}^N_{t-1}\right] \end{aligned}$$

where K_{ref} and K_{t-1} are $2 \times N$ matrices, Equation (10) can be rewritten more globally:

$$K_{ref} = AK_{t-1}$$

wherefrom we get

$$K_{ref}K^T_{t-1} = A\left[K_{t-1}K^T_{t-1}\right]$$

Since at least 3 of the N characteristic points are not aligned, matrix $K_{t-1}K^T_{t-1}$ is invertible, and we have

$$\boxed{A = K_{ref}K^T_{t-1}\left[K_{t-1}K^T_{t-1}\right]^{-1}}$$

Hence D is calculated as

$$\boxed{D = G_{ref} - K_{ref}K^T_{t-1}\left[K_{t-1}K^T_{t-1}\right]^{-1}G_{t-1}}$$

Similarly, it is possible to find the affine transformation (A', D') between $I(t-2)$ and $I(t-1)$:

$$\forall i \in [1, N],\ I^i_{t-1} = A'I^i_{t-2} + D'$$

we obtain:

$$\begin{aligned} A' &= K_{t-1}K^T_{t-2}\left[K_{t-2}K^T_{t-2}\right]^{-1} \\ D' &= G_{t-1} - K_{t-1}K^T_{t-2}\left[K_{t-2}K^T_{t-2}\right]^{-1}G_{t-2} \end{aligned}$$

Each matrix A or A' is composed of 4 parameters, and each vector D or D' of 2 parameters. Therefore, using (A, A', D, D') as neural inputs[2] instead of $(J(t-2), J(t-1))$, means using 12 parameters instead of $4N$ parameters as before (N being usually higher than 5, and always greater than 3).
The advantages are obvious: first, the size of the network input layer is reduced. Second, the size is no more dependent on N, therefore if the image processing unit should miss a point at one iteration, the system could still work. Finally, this new representation makes the neural inputs be more robust to noise (image processing errors).

V. Experiments and discussion

The first part of the experiments is concerned with uniform movement: the object translates at a constant speed, and the performance criterion is the average distance between the current position and the reference position of the $N = 5$ feature points over m samples:

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

Experiments have been done using simulation. In order to present more realistic experiments, the image pixels in $I(n)$ are rounded up to the nearest *even* value. This simulates the case where the image size is divided by 2 in each direction in order to speed up image processing. 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 samples/second, and each point of the graph represents the average over $m = 200$ samples. The case of non-uniform object movements is also investigated. In this non-uniform experiment, the position of the object along the X-axis is a sinusoidal function of the form:

$$X_{pos}(n) = A\ sin(2\pi f n) \tag{11}$$

where n is the number of the sample. The amplitude A of the movement is set to 30cm, while the frequency

[2] In practice, these values are normalized: the unit matrix is substracted to A and A', and D and D' are divided by 100

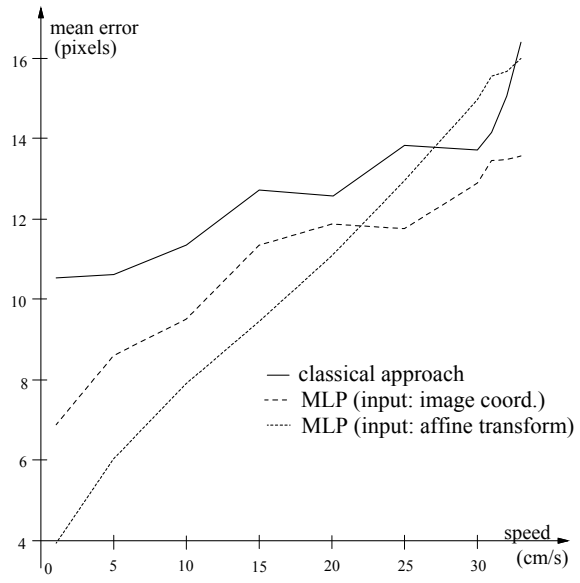

Figure 5: *Compared performances of the classical method and the neural approaches in the case of a uniform object translation.*

f varies. For each frequency, a trial is performed for 1000 samples. The graph showing the average pixel error over 1000 samples, with respect to the associated frequency, is depicted in Figure (6a). Sinusoidal

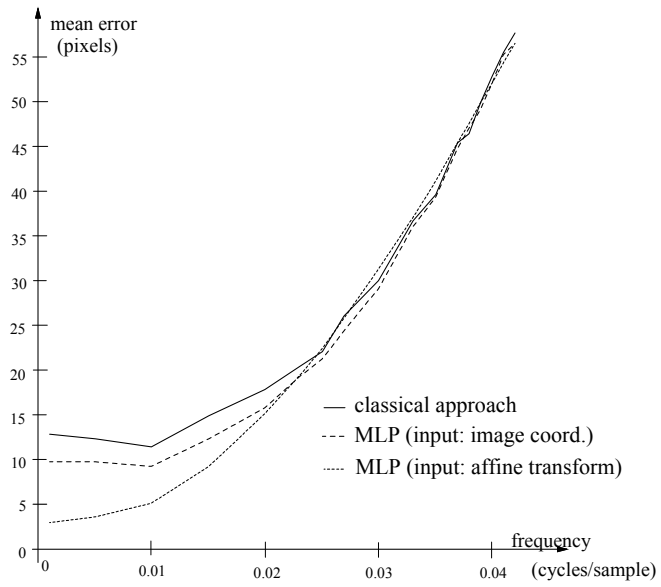

Figure 6: *Compared performances of the classical method and the neural approaches in the case of a non-uniform (sinusoidal) translation.*

(therefore non-uniform) movements appear 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.

VI. Conclusion

In a visual task-reference problem, a robot with a camera mounted on its end-effector is used to follow the movement of an object in 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. The approach can be further improved by replacing the image features on network input with parameters that model image transformations. Doing this yields to faster training and better performances. Experiments, including highly non-uniform movements for the object, have been presented and compared with the classical method, showing the interest of the approach.

References

[1] G. Burel, "Réseaux de neurones en traitement d'images: des modèles théoriques aux applications industrielles", Ph.D. Thesis, University of Brest, France, 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 in 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] N. Rondel & G. Burel, "Multi-Layer Perceptrons for Task Visual Servoing in Robotics", IEEE International Conference on Fuzzy Logic and Neural Networks, Funchal, Portugal, 3-6 Oct, 1995

[6] D. Rumelhart, & J. McClelland, "Parallel Distributed Processing", chapter 7, MIT Press, 1986