A NON PARAMETRIC METHOD FOR VIDEO CAMERA CALIBRATION USING A NEURAL NETWORK

Enrique Carlos Segura
Departamento de Computación - Universidad de Buenos Aires
Ciudad Universitaria, Pab. I - (1428) Buenos Aires - Argentina
Tel./Fax: 541-783-0729
E-mail: giren@dc.uba.ar

ABSTRACT

An efficient technique for video camera calibration is presented. Unlike the methods reported up to now, it does not require parameter estimation (lense distortion coefficients, focal distance, rotation and translation matrices, etc.). The method consists essentially of a mapping from the camera space onto that of real positions by means of a neural network (two-layer perceptron) trained with an original algorithm for neural network construction. This is an approximate technique, i.e. precision increases with the sample size and can be as large as needed by growing that size and the computing time. We show that it is possible to obtain good approximations with low computational costs (processor and memory).

Keywords: video camera calibration - artificial neural networks - multilayer perceptrons - function approximation.

1. INTRODUCTION

We present an efficient technique for the calibration of a video camera, with two main features:
- *Non parametric*: in advantage over other reported methods ([1],[2]), it doesn't require estimation of parameters (lense distorsion, focal distance, rotation and translation matrices,etc.).
- *Approximate*: precision increases with sample size and may grow as much as needed by increasing this size and the computation time.

Essentially the method consists of a mapping of camera space onto that of real positions by means of a neural network (two-layer perceptron) trained by the SAGA algorithm [3]. This is an approximate method, i.e. precision increases with the size of the sample and may be as high as desired by growing this size and the calculation time spent. It is shown by experiments that it is possible to obtain good approximations with a little computational effort and it is proved that the required memory is little too.

2. STEPS OF THE METHOD

The algorithm has, basically, two steps: 1)sampling and 2) approximation. The first one consists of obtaining a sequence of 5-uples (x,y,d,θ,h) where:

- x,y are the CCD coordinates;
- d is the distance from the observed point to the CCD;
- θ is the angle of the observed point with respect to the camera and
- h is the height of the point.

The equations relating those values are (for a 256 x 256 pixel CCD):

$$\theta = (\alpha/2)\ (x-128)/128 \qquad (1)$$

$$d = k\ h\ f\ /\ (y - 128) \qquad (2)$$

where f is the focal distance, α is the maximum angle of the camera and k is a normalizing factor. Hence, fixing h we can determine how θ depends on (x,y), and the same for d. We obtain:

$$\theta = \theta\ (x,y)$$

$$d = d\ (x,y)$$

The method provides this sampling for different values of h simultaneously.

As for the second stage, it consists of the approximation of functions θ and d. To do that we propose a neural network with a two-layer perceptron architecture, trained by means of
an original algorithm called SAGA (see below).

3. SAMPLING

The sampling system consists of a lattice of $m_x \times m_y$ sparkling points, separated by distances s_x and s_y respectively. Hence, knowing height h of the lattice with respect to the center of the camera and distance d to the CCD, we can take, with the same lattice, m_x samples (4-uples of the form (x,y,d,θ)) for each one of the m_y different heights defined, i.e., $m_x \times m_y$ 5-uples of the form (x,y,d,θ,h).

3.1 Sample uniformity

Suppose we want to take m samples, uniformly on the domain $n_x \times n_y$. Then we will need 1) a point for each $n_x/\sqrt{m}$ over the x-axis and 2) a point for each $n_y/\sqrt{m}$ over the y-axis. In order to the first, we divide the x-axis in intervals of size:

$$\alpha\sqrt{m}\ /\ n_x$$

For the second, and taking account of (2), if we want values for y in the form

$$n_y / \sqrt{m}$$

(for $k = 1,...,\sqrt{m}$) clearly it will be necessary the following intervals of distance:

$$d = h f \sqrt{m} / (k n_y) \qquad k = 1,...,\sqrt{m}$$

Nevertheless, it is easy to see that, for the sake of a real application, as distance increases, calibraton for large angles (those near the limits of maximum opening of the camera) becomes negligible, since the same scene looked at from a greater distance encompasses a less angular range.

4. APPROXIMATION

Once acquired a regular sample of the domain of values (x,y), the method will search for an approximating function implying the less possible computational cost when physically implemented. A *two-layer perceptron with sigmoid output* is a function of the form:

$$\sum_{i=1}^{N} d_i \phi(a_i x + b_i y + c_i) \qquad (3)$$

where ϕ is a differentiable, monotonic non decreasing function such that $\phi(+\infty) = 1$ and $(-\infty) = -1$. Well-known theorems ([4],[5],[6]) assert that such an architecture is able to approximate any continuous function in a compact domain with as much precision as desired, properly adjusting the number of terms *(hidden units)* in the sum and the values of a_i, b_i, c_i and d_i.

In our case

$$\phi(x) = \tanh(x) \qquad (4)$$

Previous experiments demonstrated that the computing time required by a 80486 microprocessor for a function in the form of (3) and (4) with 10 terms is about 0.3 msec. However, the number of 10 is a bad-case estimation and the simulations show that it is possible to obtain good approximations (on the order of under 5 % of mean relative error) with 5 terms, in which case the estimated time for the complete computation is in the order of 0.3 msec (0.15 for estimating the distance and 0.15 for estimating the angle).

As for the necessary amount of memory to store the perceptron parameters, it will be, if N is the number of hidden units:

$$m = 8 N$$

memory positions (words), since 4 values are needed for each term of the two perceptrons.

For N = 5 it would be m = 40, a negligible memory cost.

4.1 Perceptron parameters

In order to compute the adequate parameters (N and the a_i, b_i, c_i and d_i for i = 1,...,N) basically two classical strategies were tried: Backpropagation [7] and Simulated annealing ([8],[9]). The results of those approaches not being satisfactory, a hybrid algorithm was proposed, called SAGA (Simulated Annealing + Gradient descent + Adaptive growing). SAGA [3] has a basic SA structure concerning iterativeness and stochastic acceptance of changes to the variables in state space (in our case, the perceptron parameters), but it uses the method of gradient descent to propose the changes and an adaptive strategy for growing the net (i.e. to add hidden units).

More precisely, the SAGA algorithm consists of a SA where parameter changes are proposed according to the direction that maximizes the gradient. The acceptance, on the other hand, is subjected to the classical criterium of Boltzmann distribution ([8],[9]). As for N, it increases as the number of hidden units becomes inefficient to reduce the approximation error, according to a tolerance parameter.

5. EXPERIMENTAL RESULTS

We show approximations found by SAGA for a training set of 255 points, with h = 2 cm., tested on a regular lattice of points. In the case of the distance, the mean relative error was 3.4 % with N = 5 (see fig. 1). For the angle, it was 8.7 % with N = 6 (fig. 2). The difference between both errors may be explained by the greater linearity of θ (x,y), for
which a nonlinear function as the proposed ϕ is more difficult to adapt.

6. FUTURE WORK

First of all, it would be interesting to make the algorithm capable to find the proper correction for x and y, i.e. to which values of x and y would correspond, for an ideal camera, θ and d.

Another interesting question concerns the robustness of the method under errors in the acquisition of data (for example, those due to distances measured with a certain error or to the image digitalization in the CCD). It seems necessary an analysis of the effect of those errors on the final approximation, and of the ranges of tolerance for them, without significant effects when propagated over the final result (in comparison to the desired error
for the approximation).

At present, we are also working on the improvement of the precision of the technique.

7. CONCLUSIONS

We have presented an efficient technique for video camera calibration that**,** in advantage over conventional methods ([1],[2]), does not require the estimation of parameters (lense distortion coefficients, focal distance, rotation and translation matrices,etc.).

Essentially, the method consists of a mapping of the camera space onto that of real positions by means of an artificial neural network (two-layer perceptron), trained with the SAGA algorithm.

Besides of non parametric, the calibration is approximate, that is, the precision increases with the number of examples and may be as high as desired by increasing that number and the computation time. The experiments show that it is possible to obtain good approximations with a little computational cost, and it is proved that the required memory is little too.

The precision of the method may be improved. However, for many applications (such as vehicle navigation in a multitasking environment, for which the technique was initially intended), a mean relative error on the order of 3 %, as obtained for the distance, is very good.

8. REFERENCES

[1] R. Y. Tsai; *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses*, IEEE Journal on Robotics and Automation, vol RA-3, no. 4, 1987.

[2] D. Zhang, Y. Nomura y S. Fujii; *Error analysis and optimization of camera calibration*, IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS'91, Osaka, 3-5 Nov. 1991.

[3] E. Segura; *SAGA: a hybrid technique for neural network synthesis*; in preparation.

[4] K. Funahashi; *On the approximate realization of continuous mappings by neural networks*, Neural Networks, 2, 183-92, 1989.

[5] K. Hornik; *Approximation capabilities of multilayer feed-forward networks*, Neural Networks, 4, 251-257, 1991.

[6] M. Stinchcombe y H. White; *Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights*, Proc. IJCNN 1990, III-7-16.

[7] D. E. Rummelhart, G. E. Hinton y R. J. Williams; *Learning internal representations by error propagation*; En Parallel Distributed Processing, Vol. 1, ed. Rummelhart and McClelland, pp. 318-362, MIT Press, 1986.

[8] S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi; *Optimization by Simulated Annealing*; Science, vol. 220, pp. 671-680, 1983.

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. H. Teller y E. Teller; *Equation of state calculations by fast computing machines*; J. Chem. Phys., vol. 21, no. 6, pp. 1087-1091, 1953.