Moving Objects Detection based on CNNs and Clustering

G. COSTANTINI, D. CASALI, M. CAROTA
Department of Electronic Engineering
University of Rome "Tor Vergata"
Viale Politecnico, 1. I-00133 Rome
ITALY

Abstract: - An analog CNN algorithm is proposed for detection of moving objects in high-resolution gray-scale images, with multiple moving objects and with moderate a priori information on the scene. The algorithm is based on simple 3×3 templates followed by a clustering operation on the moving pixels. The effectiveness of the proposed algorithm is shown using some real-life indoor and outdoor images.

Key-Words: - Cellular Neural Networks, Clustering, Moving Objects Detection.

1 Introduction

Moving object detection is an important task in many real-time applications such as autonomous robotics, traffic control, driver assistance, surveillance systems. Since each pixel in the image may belong to a moving object, conventional sequential image processing algorithms, entail an impressive computation. An alternative approach is represented by Cellular Neural Networks (CNNs), owing to their ability to process the image in a parallel, local and therefore, distributed fashion [1,2]. CNN analog arrays can be realized, which process in real-time large scale, high resolution images. Moreover, since the CNN arrays are programmable, they can be repeatedly used to solve complex image processing tasks, decomposing them into several but simpler sub-tasks.

Moving object detection by CNNs has been investigated in the past, mainly from the viewpoint of image coding for low bit-rate transmission [3,4,5,6].

In this paper, we propose an algorithm for moving object detection suited for surveillance systems and traffic control; the algorithm can be implemented using exclusively CNN analogic processing, with 3×3 templates. The proposed algorithm works on real-life, gray-scale, noisy images, with multiple moving objects; not much a priori information is needed, namely the maximum speed of moving objects and the frame rate. A clustering procedure is performed on the moving pixels giving the number, size and position of moving objects.

The Paper is organized as follows. Section 2 presents an overview of CNN state equation and definitions. Section 3 describes the algorithm for moving object detection. Section 4 addresses the center point and size estimation of moving objects. The various steps of computation are illustrated by two real-life examples.

2 General Framework

The CNN is described by the following state equation [7]:

$$\dot{x}_{ij} = -x_{ij} + \sum_{C(k,\ell)\in N(i,j)} A_{ij,k\ell} y_{k\ell} + \sum_{C(k,\ell)\in N(i,j)} B_{ij,k\ell} u_{k\ell} + I \qquad (1)$$

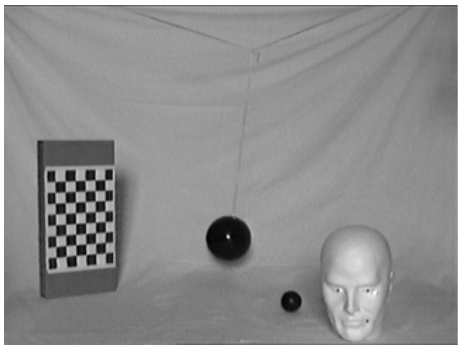
where N(i,j) represents the 3x3 neighborhood of cell C(i,j). A is the feedback template, B is the input template and I represents a bias. In eqn.(1) x_{ij} is the cell state, y_{ij} is the cell output and u_{ij} is the cell input, corresponding to an image pixel. The cell output y is related to the state x by the usual piecewise-linear function, $y = (1/2)(|x+1| - |x-1|)$.

For simulation purposes, we used real life image sequences taken both indoor and outdoor. The image size is 128×128 with 256 gray levels and the pixel intensity varies between –1 (black) and +1 (white). We extract a sequence of image pairs taken k frames apart, namely P(n), P(n+k), n = 0, m, 2m, …., where m is the interval between pairs. The value of k is related to the maximum speed of moving objects. Each pair of images is processed, as described in the following section. The task is to detect the position and size of each moving object in each image pair.

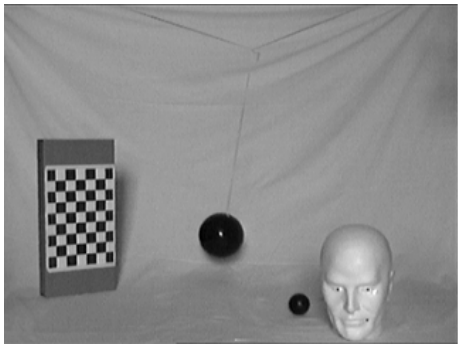
Two images taken from an outdoor sequence and two images taken from an indoor sequence are shown in Figure 1. There are three main moving objects, in both sequences. In the outdoor case there are two walking persons and a light van entering the scene in reverse gear. In the indoor case there are an oscillating pendulum and a little ball rolling toward the "head" after having bounced on the background fabric; also the fabric is slightly moving due to ball bouncing.

3 Algorithm for Object Detection

A scheme of the processing flow is presented in Figure 2.



P(n)



P(n+k)

Figure 1: Two image pairs P(n), P(n+k) used to illustrate the processing steps.

The input is represented by the two images P(n) and P(n+k). The first operation consists in the sum of image P(n+k) and the negative of image P(n). Then we apply the following operators: absolute value, noise filtering, thresholding, removal of isolated pixels and filling concave location. The output P* is a black image with white blobs, corresponding the moving objects. At this point we perform a clustering operation on the image P* to detect the objects that have moved.
The templates used in the proposed algorithm are presented in the following.

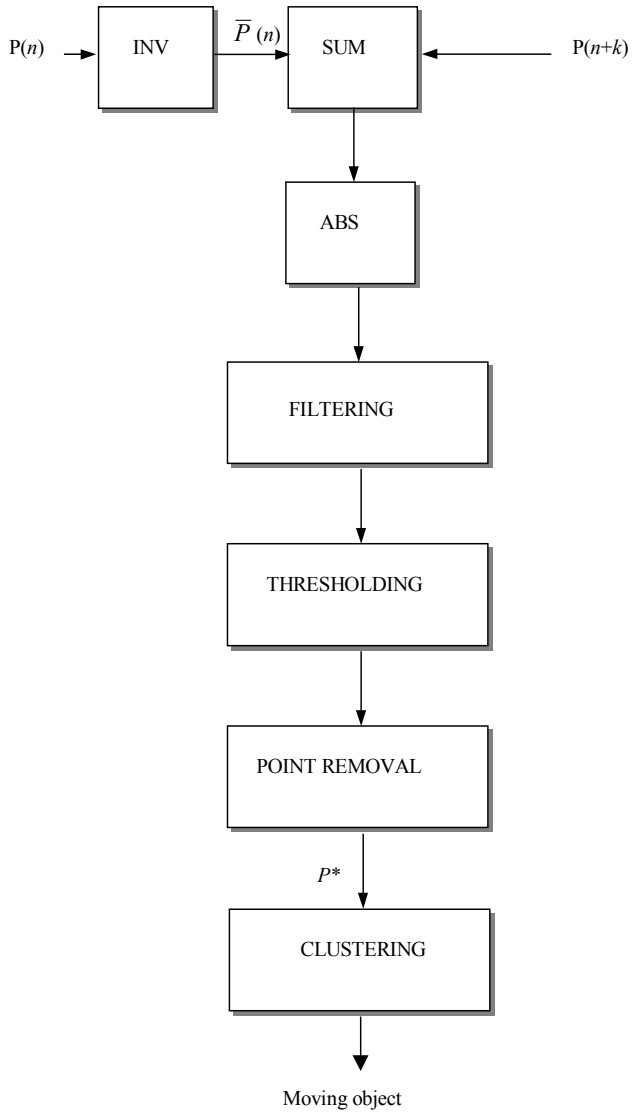


Figure 2: Flow chart of the proposed algorithm.

Step 1: Image inversion
Image $P(n)$ is transformed in its negative.
Templates and threshold:

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 0.95 & 0\\ 0 & 0 & 0\end{bmatrix}\quad \mathbf{B}=\begin{bmatrix}0 & 0 & 0\\ 0 & -2 & 0\\ 0 & 0 & 0\end{bmatrix}\qquad I=0$$

Input: $\boldsymbol{u}$ (t) = P
Initial state: $\boldsymbol{x}(0) = \mathbf{0}$
Output: $\boldsymbol{y}(\infty) = \overline{P}$.

Step 2: Image sum
Images $\overline{P(n)}$ and $P(n+1)$ are summed pixel wise. The result is image P_{SUM}.
Templates and threshold:

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 0.95 & 0\\ 0 & 0 & 0\end{bmatrix}\ \mathbf{B}=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}$$

I_{ij} = pixel *ij* of image P_1

Input: $\boldsymbol{u} = P_2$
Initial state: $\boldsymbol{x}(0) = \mathbf{0}$
Output: $\boldsymbol{y}(\infty)$ = sum of pixel intensities of P_1 and P_2.
Remark: the bias is cell dependent.

Step 3: Absolute value
The absolute value of each pixel is computed on P_{SUM}.
Templates and threshold:

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\ \mathbf{B}=\begin{bmatrix}0 & 0 & 0\\ 0 & f & 0\\ 0 & 0 & 0\end{bmatrix}\qquad I = 0$$

f represents the non-linear function shown in Figure 3.
Input: $\boldsymbol{u} = P$
Initial state: $\boldsymbol{x}(0) = \mathbf{0}$
Output: $\boldsymbol{y}(\infty) = f(P)$.

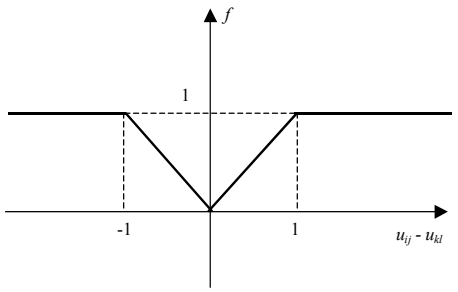


Figure 3: Non-linear function *f*.

Step 4: Filtering
A low pass filtering is performed to remove noise.
Templates and threshold:

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\ \mathbf{B}=\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}\qquad I = 0$$

Input: $\boldsymbol{u} = P$
Initial state: $\boldsymbol{x}(0) = \mathbf{0}$
Output: $\boldsymbol{y}(\infty)$ = each pixel intensity is the average of the input intensities in the neighborhood.

Step 5: Thresholding
Thresholding is performed on the gray-scale filtered image to get a binary image.
Templates and threshold:

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix}\ \mathbf{B}=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\qquad I = -\ threshold$$

Input: $\boldsymbol{u} = \mathbf{0}$
Initial state: $\boldsymbol{x}(0) = P$
Output: $\boldsymbol{y}(\infty)$ = binary image; white pixels correspond to the intensities of *P* greater than the threshold.

Step 6: Point removal
Isolated white pixels are removed.
Templates and threshold:

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}\ \mathbf{B}=\begin{bmatrix}1 & 1 & 1\\ 1 & 8 & 1\\ 1 & 1 & 1\end{bmatrix}\qquad I = -1$$

Input: $\boldsymbol{u} = P$
Initial state: $\boldsymbol{x}(0) = P$
Output: $\boldsymbol{y}(\infty)$ = binary image where isolated pixels of *P* have been eliminated.

2.1 Clustering on the image *P**
The final step consists of clustering of white pixels in the image *P**, i.e. of pixels that corresponds to a moving object; it is based on the two spatial coordinates. Each cluster will represent a distinct moving object. In our work, we used the clustering algorithm DBSCAN (Density Based Spatial Clustering for Application with Noise) [8] that relies on a density-based notion of clusters and was designed to discover clusters of arbitrary shape efficiently. Using DBSCAN the typical density of points inside clusters is considerably higher than outside the clusters. The control parameters of the algorithm are the maximum distance (in pixels) between two points belonging to the same cluster (D) and the minimum number of points in a cluster (M).

4 Simulations and Results
An example of the results obtained by the proposed method is shown in Figure 4. It concerns the images in Figure 1. The parameter values used for clustering are D = 4, d = 1, M = 8.

Figure 4 shows the image P*, which is the output of the point removal. For presentation purposes, we have evidenced the rectangles including the clusters created by DBSCAN. These rectangles correspond exactly to the moving objects.

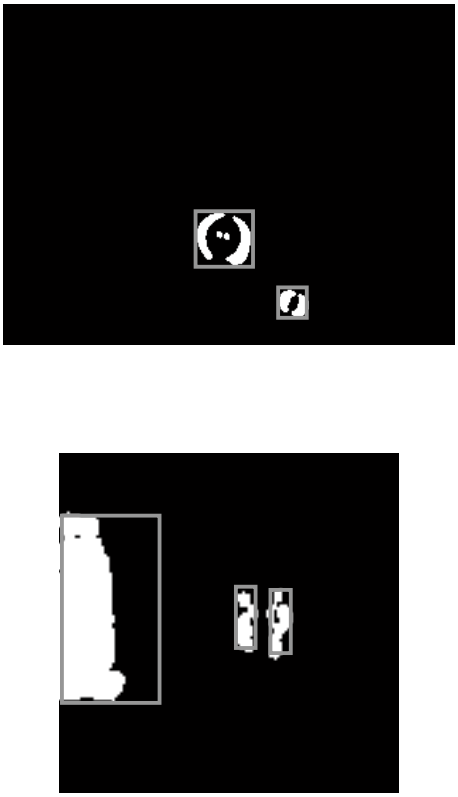


Figure 4: Result of the proposed algorithm on images in Figure 1.

5 Conclusion

A CNN algorithm has been proposed for position and size estimation of moving objects in high-resolution gray-scale image sequences. The method requires a fixed camera and it is suited for surveillance and control applications. Detection of moving objects is obtained through the sequential application of several space-invariant templates following by a clustering procedure on the moving pixels. The simulation results show the good behavior of the proposed method.

Acknowledgments

This work is supported by Italian Ministry for Education, University and Scientific Research by PRIN Project. The images used for simulations have been provided by the Italian National Agency for New Technologies, Energy and the Environment (ENEA) and by the Department of Information Engineering, University of Siena.

References:

[1] T. Yang, L. B. Yang, X. P. Yang, Application of a cellular neural network to facial expression animation and high-level image processing, *Int. J. Circuit Theory Applications*, vol. 24, pp. 425-450, 1996.

[2] A. Gacsadi, P. Szolgay, An analogic CNN algorithm for following continuously moving objects, Proc. of IEEE *Int. Workshop on Cellular Neural Networks and Applications*, CNNA-2000, Catania, pp. 99-104, 2000.

[3] T. Roska, T. Boros, A. Radvanyi, Detecting moving and standing objects using cellular neural networks, Proc. of *Int. J. Circuit Theory and Applications.*, vol. 20, pp. 613-628, 1992.

[4] T. Roska, Analogic CNN computing: architectural, implementation, and algorithmic advances: a review, Proc. of IEEE 5th *Int. Workshop on Cellular Neural Networks and Applications*, CNNA-98, London, pp. 3-10, 1998.

[5] G. Grassi, L. A. Grieco, Object-oriented image analysis via analogic CNN algorithms- Part I: motion estimation, Proc. of IEEE 7th *Int. Workshop on Cellular Neural Networks and Applications*, CNNA-02, Frankfurt, pp. 172-179, 2002.

[6] *Analogic CNN program library* 6.1, Analogical and Neural Computing Lab., Hungarian Academy of Sciences, Budapest, 1994.

[7] Chua, L. O. and Yang, L., Cellular neural networks: Theory, *IEEE Trans. Circuits and Systems*, Vol. 35, pp. 1257 -1272, 1988.

[8] M. Ester, H. P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Proc. of 2nd *Int. Con. on Knowledge Discovery and Data Mining* (KDD-96), pp. 226-231, Portland, OR, 1996.