INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS
Int. J. Circ. Theor. Appl. 2004; **32**:509–522
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cta.293

Analogic CNN algorithm for estimating position and size of moving objects

Giovanni Costantini[1], Daniele Casali[1] and Renzo Perfetti[2,*,†]

[1]*Department of Electronic Engineering, University of Rome Tor Vergata, Rome, Italy*
[2]*Department of Information Engineering, University of Perugia, Perugia, Italy*

SUMMARY

An analogic CNN algorithm is proposed for detection of multiple moving objects in high resolution, grey-scale images taken from a fixed camera. The algorithm, based on simple 3×3 templates, can be implemented using CNN hardware, providing the real-time operation required in surveillance and traffic control applications. Efficient separation of moving objects from the background is obtained through automatic threshold selection. The performance of the proposed method is shown using real-life indoor and outdoor video sequences. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: cellular neural networks; image processing; moving object detection

1. INTRODUCTION

Moving object detection is very important in many real-time applications such as autonomous robotics, traffic control, driver assistance and surveillance systems. It is a hard problem for several reasons. Usually high resolution grey-scale images must be processed; since each image pixel may belong to a moving object, pixel-wise processing is required in conventional digital methods, which gives an impressive computational burden. On the other hand, the grey-level intensity of moving objects varies greatly with the lighting conditions, and the images are usually cluttered and noisy, so moving objects could be almost undistinguishable from the background. Finally, real-time operation is needed in most applications. A promising alternative approach to overcome these difficulties is represented by cellular neural networks (CNNs) [1, 2]. Owing to the analogic fully parallel processing of the CNN hardware, like the CNN-UM chips [3–5], we can process large scale, high resolution images providing the required speed of operation. Moreover, since the CNN chips are programmable, complex image processing tasks can be realized by decomposing them into several but simpler sub-tasks.

*Correspondence to: R. Perfetti, Department of Information Engineering, University of Perugia, Perugia, Italy.
†E-mail: perfetti@diei.unipg.it

Contract/grant sponsor: Italian Ministry of Education; contract/grant number: PRIN Project

Received 30 January 2003
Revised 29 April 2004

Moving object detection by CNNs have been investigated in the past, also from the viewpoint of image coding for low bit-rate transmission [6–10]. In this paper, we propose a simple and robust algorithm for moving object detection in video sequences taken from a fixed camera. It is well suited for surveillance systems and for car or air traffic control. The algorithm can be implemented using exclusively CNN analogic processing with 3×3 templates, and works on real-life, grey-scale images with multiple moving objects. Moderate *a priori* information is needed on the scenario, namely the maximum speed of moving objects and the frame rate. The outputs of the algorithm are: the co-ordinates representing the centre of each moving object; an estimation of the overall size of each moving object.

The Paper is organized as follows. Section 2 presents the rationale of the proposed approach and its limitations. Section 3 describes the algorithm for moving object detection. Section 4 addresses the hardware realization and the processing time.

2. RATIONALE OF THE PROPOSED METHOD AND LIMITATIONS

Let us consider grey-scale images where the pixel intensity varies between -1 (black) and $+1$ (white). The task is to detect the position and size of each moving object. The simpler way to extract moving objects in a video sequence is to compare image pairs taken k frames apart, namely $P(n)$ and $P(n+k)$, with $k \geqslant 1$. To simplify the exposition assume a dark object (e.g. with grey level -0.8) moving on a white background, as shown in Figures 1(a) and (b). Taking the difference $P(n+k)-P(n)$ we obtain three classes of pixels: white pixels representing the positions filled by the object in frame n but not in the frame $n+k$ (difference $= 1-(-0.8) = 1.8$ which is saturated at $+1$); black pixels corresponding to the positions filled by the object in frame $n+k$ but not in frame n($-0.8-1 = -1.8$ saturated at -1); the grey pixels of the background corresponding to zero grey level.

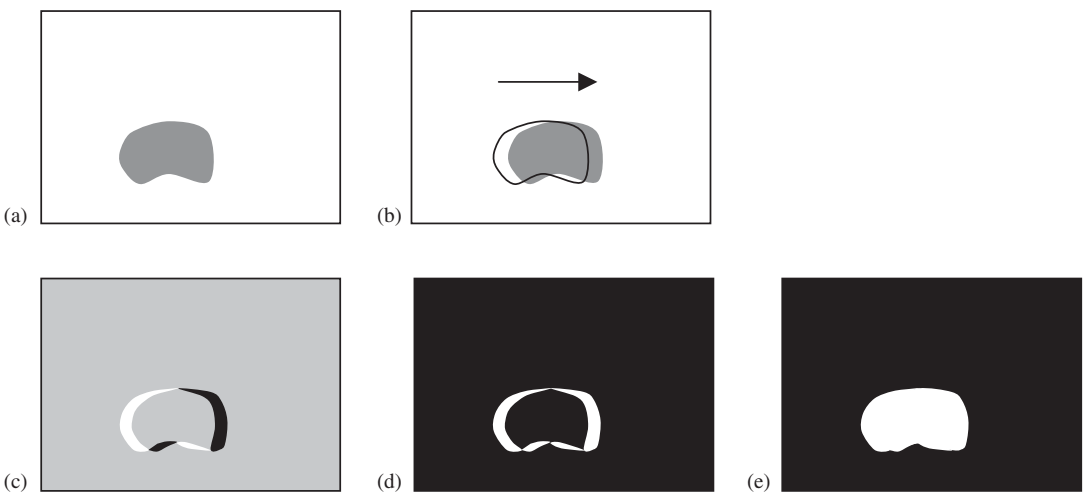


Figure 1. (a–b) An object is moving rightward; (c) difference between the images in a–b; (d) absolute value; and (e) filling of the contour in d.

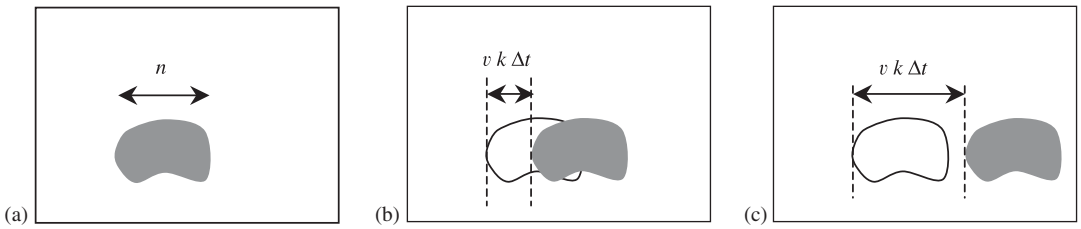


Figure 2. (a) Moving object with horizontal size n; (b) $v\ k\ \Delta t < n$; and (c) $v\ k\ \Delta t > n$.

If the movement of the object in the time interval corresponding to k frames is small enough, we obtain a ring shaped image on a grey background (Figure 1(c)). Taking the absolute value of the image in Figure 1(c), the pixel intensity varies between 0 (black) and 1 (white); now we have the white contour shown in Figure 1(d). Finally, filling the area inside the contour we have a white 'blob'; its size and position correspond to that of the moving object in the video sequence (Figure 1(e)).

In some cases the black and white parts in Figure 1(c) could be not connected together; nevertheless, filling of concave locations will merge the two parts giving a single connected and convex object, as shown in the simulation examples in Section 3.

A simple condition can be derived for the correct behaviour of the proposed approach. Let v represents the speed of the object in pixels/sec; Δt the inter-frame time separation; k the number of frames between the two processed images; n the size of the object (in pixels) measured in the direction of motion. If $v\ k\ \Delta t > n$, two separate objects will be present in the difference image (Figure 2). So, the value of k must be selected so that

$$vk\Delta t < n \ \Rightarrow\ k < \frac{n}{v\Delta t}$$

Moreover, in order to limit the *stretching* effect in the difference image (see Figure 1), it is convenient to further reduce the value of k with respect to the previous constraint. Since $k \geqslant 1$, the proposed method can be applied only if the ratio $n/(v\Delta t) \gg 1$. If n is too small or v too high, the above condition is not fulfilled.

The three basic operations illustrated in Figures 1(c) and (e) are not sufficient for proper operation in the various conditions. For example some noise is generally present in the original images, which is amplified taking the difference; hence, low-pass filtering of the difference image is required. Moreover, even if the camera is fixed, the background could be slightly changing due to some clutter (foliage, clouds) and to varying lighting. Owing to scarce illumination and to the presence of clutter, the moving object pixels in the difference image can have an intensity only slightly superior with respect to that of the background. To overcome this problem we introduce thresholding on the difference image: every pixel above the threshold is set to 1 (white); every pixel below the threshold is set to 0 (black). Thresholding can separate the moving objects from the rest; however, the threshold value must be carefully selected for each image sequence, or updated during the same sequence when the lighting conditions change. An original method is proposed in the paper to compute the threshold value automatically (the method is illustrated in detail in Section 3.1).

3. CNN ALGORITHM FOR MOVING OBJECT DETECTION

All the above mentioned operations namely, image difference, absolute value, low-pass filtering, thresholding and filling of concave locations, can easily be obtained using a CNN described by the following state equation:

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

In Equation (1) x_{ij} is the state of cell $C(i,j)$, y_{ij} is the cell output and u_{ij} is the cell input. The cells are arranged on a rectangular grid and correspond to the image pixels. $N(i,j)$ represents the 3×3 neighbourhood of cell $C(i,j)$, $\mathbf{A}$ is the feedback template, $\mathbf{B}$ is the input template and I represents a bias. The cell output y is related to the state x by the usual piecewise-linear function, $y = (1/2)[|x+1| - |x-1|]$.

We extract a sequence of image pairs taken k frames apart, namely $P(n), P(n+k), n = 0, k, 2k, \ldots$. Each pair of images is processed according to the flow diagram shown in Figure 3. Image inversion is obtained with the templates $\mathbf{A} = \mathbf{0}$, $\mathbf{B} = [0\,0\,0;\ 0\,-1\,0;\ 0\,0\,0]$, $I = 0$. Image sum is obtained using $\mathbf{A} = \mathbf{0}$, $\mathbf{B} = [0\,0\,0; 0\,1\,0;\ 0\,0\,0]$, $u_{ij} =$ pixel ij of image P(n), $I_{ij} =$ pixel ij of image $P(n+k)$. Filtering is obtained with the templates $\mathbf{A} = \mathbf{0}$, $\mathbf{B}_{ij,kl} = 1/9$ for each $kl, I = 0$. Absolute value and thresholding are described in Reference [10] and point removal in Reference [17]; filling of concave locations is performed using the template HOLLOW (see Reference [18]).

In Figure 3 P^* is a black and white image, with white blobs corresponding to the moving objects. A post processing gives the centre point and the vertical and horizontal extension of each moving object.

To show the behaviour of the proposed method, we use images taken from two different video sequences: an indoor sequence called 'pendulum' and an outdoor sequence called 'van'.

The indoor sequence was realized *ad hoc* to test the algorithm for moving object detection (two images from this sequence are shown in Figures 4(a) and (b)). There are a pendulum moving leftward 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. The image size is 320×240, with 256 grey levels. The effect of each processing step is illustrated in Figures 4(c) and (h). The frame separation is $k = 2$.

The difference image is computed by summing image $P(n+k)$ and the negative of image $P(n)$ (called $\bar{P}(n)$ and obtained with the INV template). In Figure 4(d) the difference image is shown. The light portions of the pendulum and those of the ball correspond to the old positions of these objects; the dark regions correspond to the new positions. After the absolute value computation (Figure 4(e)), the pixel intensities are in the range 0–1. Low-pass filtering is obtained by a simple averaging on a 3×3 window (Figure 4(f)). Thresholding is performed on the grey-scale filtered image to get a binary image (Figure 4(g)); the threshold value is 0.023. In the flow diagram in Figure 3, thresholding is followed by the removal of isolated pixels, which can be useful to eliminate spurious white pixels not belonging to the moving objects; since there are no isolated pixels in Figure 4(g), in this case this template has no effect. After thresholding, each moving object presents separated parts. Finally, filling of concave locations merges the separated parts giving two convex objects (Figure 4(h)).

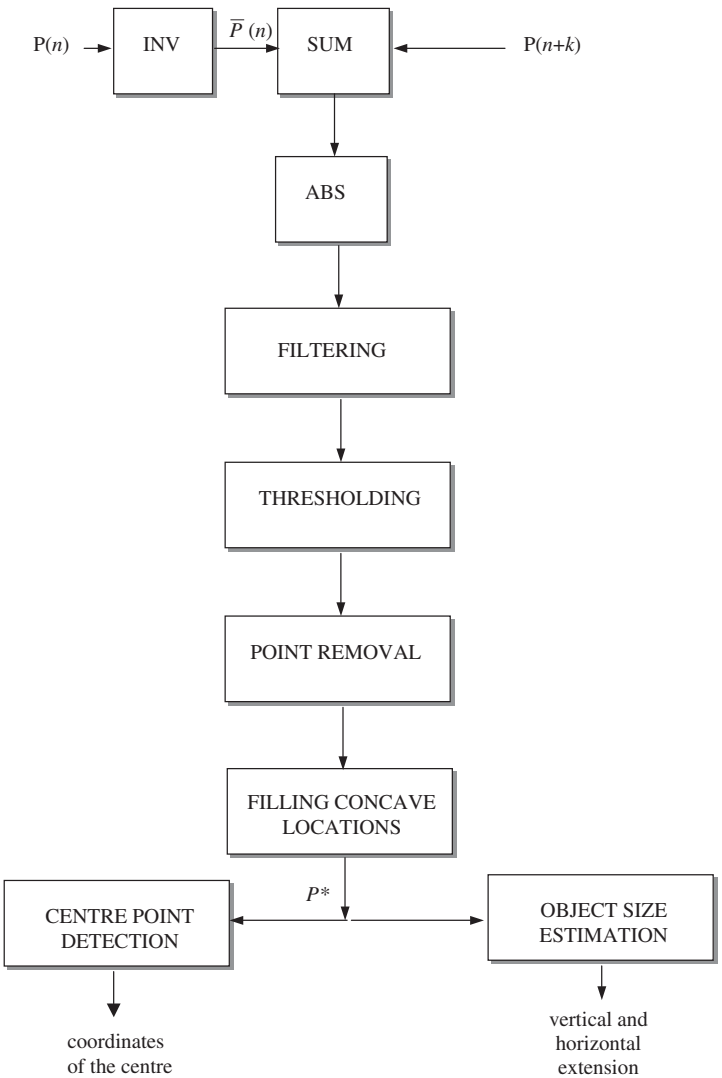


Figure 3. Flow diagram of the proposed algorithm.

The outdoor sequence is typical of a surveillance system (Figures 5(a) and (b)). There are three moving objects: two walking persons and a light van entering the scene in reverse motion. The image size is 128×128 with 256 grey levels. In this case we used the separation $k = 5$ frames. The effect of each processing step is shown in Figures 5(c) and (h). Note that the two walking persons are almost undistinguishable after filtering (Figure 5(f)). However, the pixel values of the persons are different from those of the background; this difference is evidenced by thresholding (Figure 5(g)). The threshold value used in this case is 0.015.

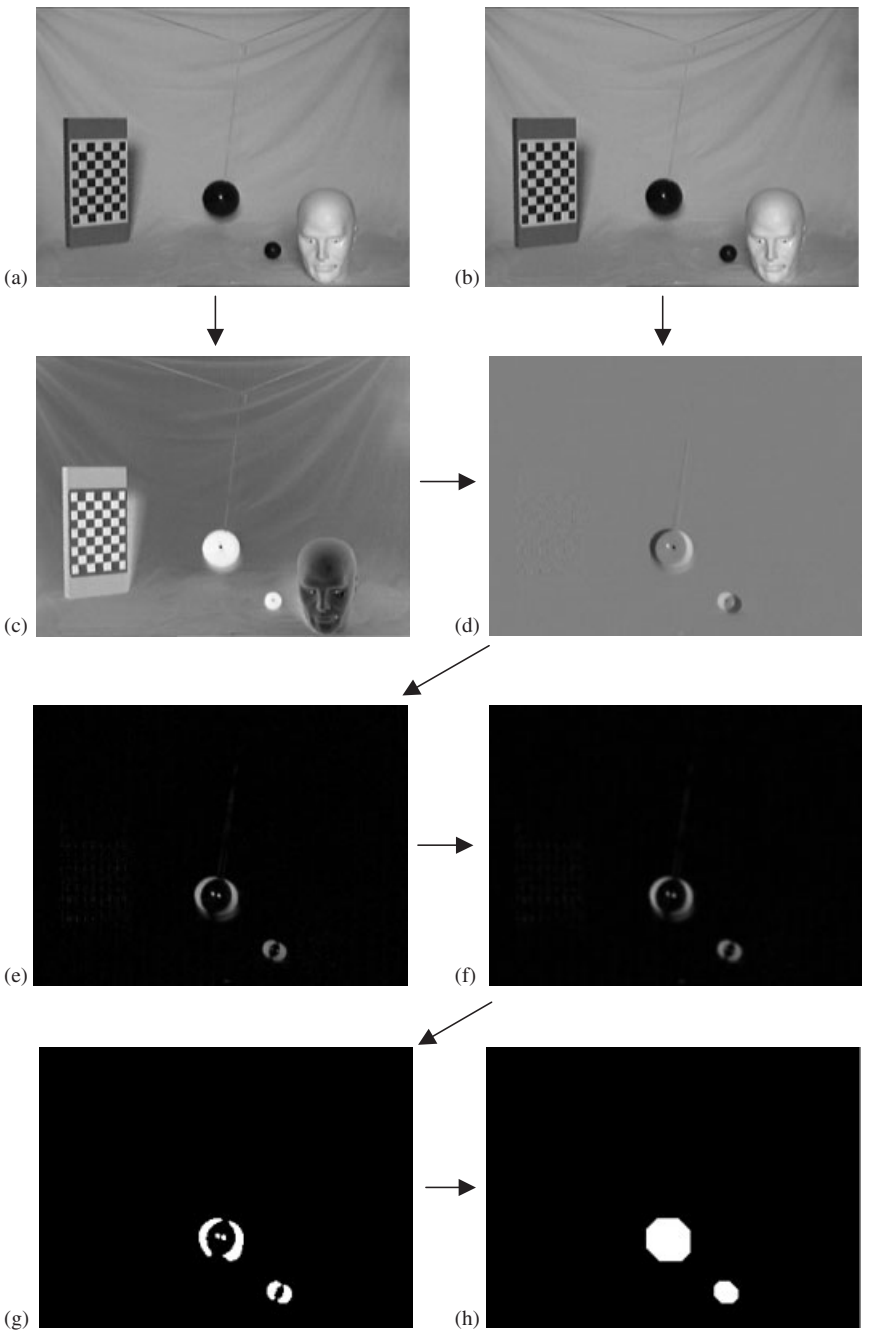


Figure 4. Detection of moving objects in the 'pendulum' sequence: (a) image $P(n)$; (b) image $P(n+k)$; (c) negative image $\bar{P}(n)$; (d) sum of $P(n+k)$ and $\bar{P}(n)$; (e) absolute value; (f) after averaging with a 3×3 window; (g) after thresholding; and (h) effect of concave locations filling (P^*).

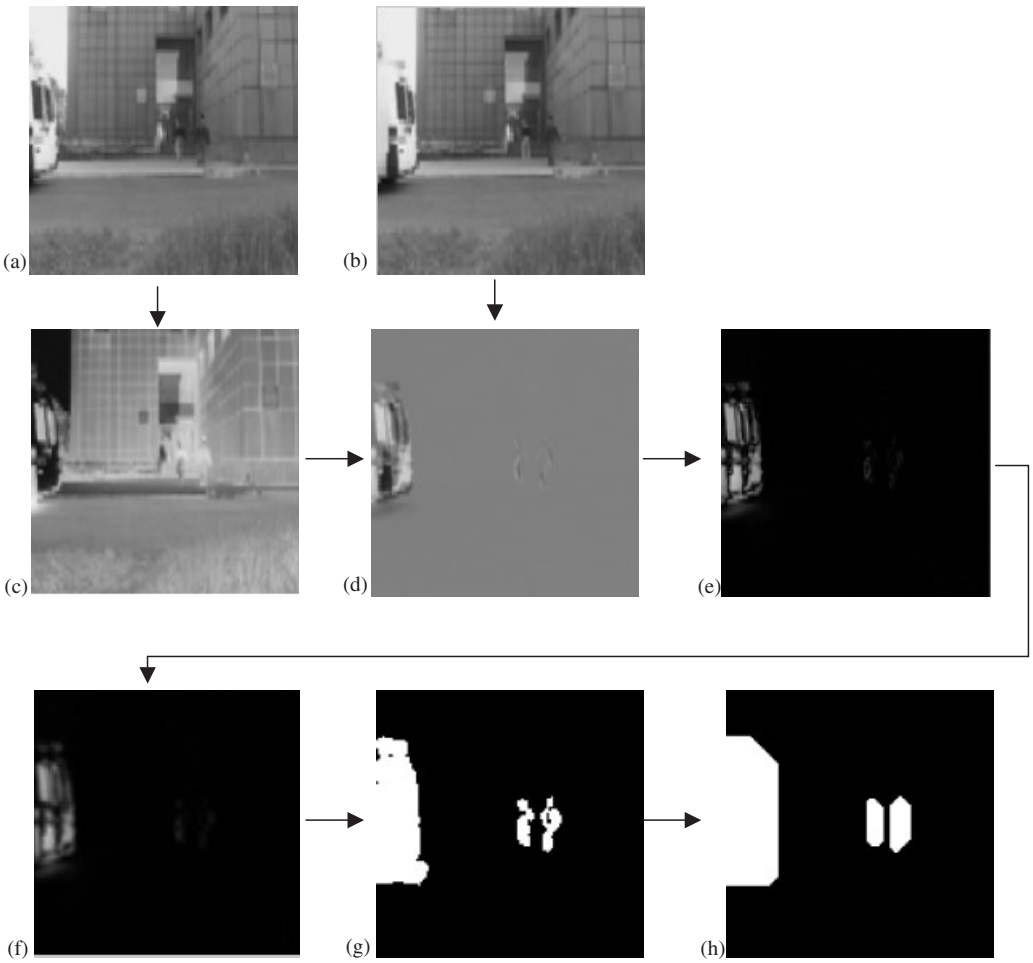


Figure 5. Detection of moving objects in the 'van' sequence: (a) image $P(n)$; (b) image $P(n+k)$; (c) negative image $\bar{P}(n)$; (d) sum of $P(n+1)$ and $\bar{P}(n)$; (e) absolute value; (f) after averaging with a 3×3 window; (g) after thresholding; and (h) after concave locations filling (P^*).

3.1. Threshold selection

Thresholding is a fundamental step in the proposed processing algorithm. It allows to discriminate between small slightly moving objects and the almost static background. For good operation, the threshold value is of crucial importance and we developed a systematic and reliable way to select it. In Figure 6 the difference images after filtering are shown, along with the grey level histogram (intensity values are in the range 0–1). The lines with larger magnitude correspond to the intensity levels of the background (intensity near zero). The small lines correspond to the pixels of moving objects. The arrow in Figures 6(b) and (d) corresponds to the threshold value we used in the simulations (0.023 and 0.015). Almost every grey level is present in the intensity distribution, so it is very difficult to distinguish between the grey

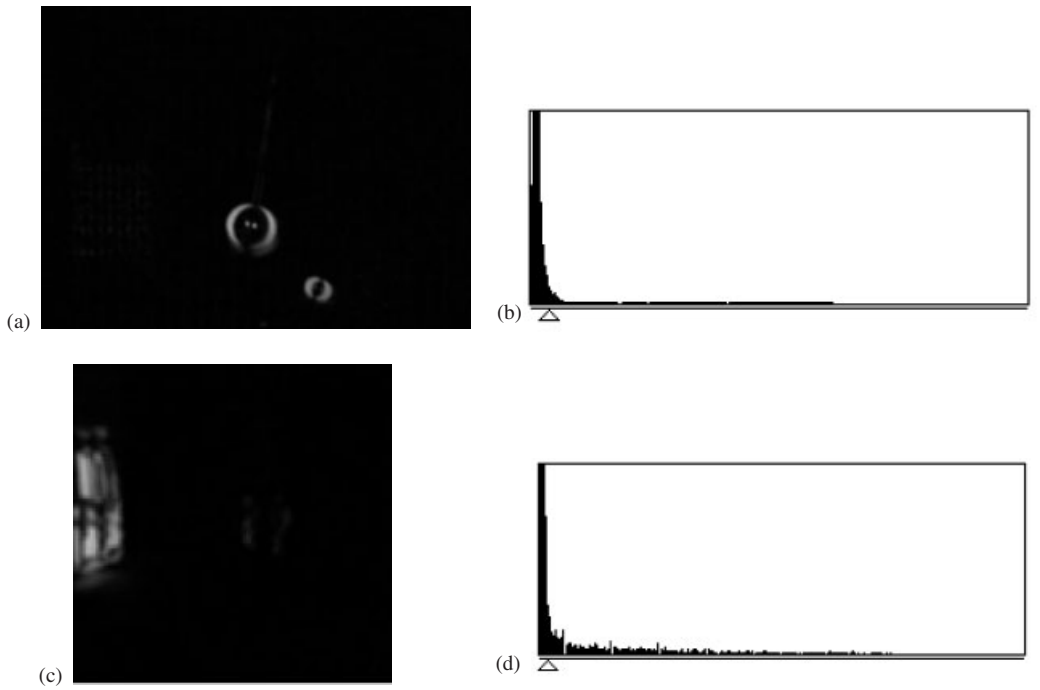


Figure 6. (a) Filtered difference image (Figure 4(f)); (b) the corresponding histogram of intensity levels; (c) filtered difference image (Figure 5(f)); and (d) the corresponding histogram of intensity levels.

levels of the moving objects and those of the background. A better separation is obtained by *histogram equalization*. This is a nonlinear pixel-wise transformation enhancing the contrast, and modifying both the appearance and the histogram of the image [14].

Let n_K be the number of pixels with intensity level I_K, $0 \leqslant I_k \leqslant 1$, and N the total number of pixels. Then the probability distribution of the intensity levels is

$$p(I_k) = \frac{n_K}{N} \tag{2}$$

Histogram equalization is performed by using, as non-linear transformation, the cumulative distribution function (CDF) of the intensity levels:

$$J_k = T(I_k) = \sum_{i=0}^{k} \frac{n_i}{N} \tag{3}$$

where J_k denotes the intensity level in the transformed image corresponding to level I_k in the original image.

Expression (3) can be computed from the image itself and gives a sort of contrast enhancement very useful for threshold selection. Figures 7(a) and (c) show the images in Figures 6(a) and (c) after histogram equalization; the corresponding histogram is shown in Figures 7(b) and (d). It has a characteristic structure with a few isolated lines, corresponding to as many

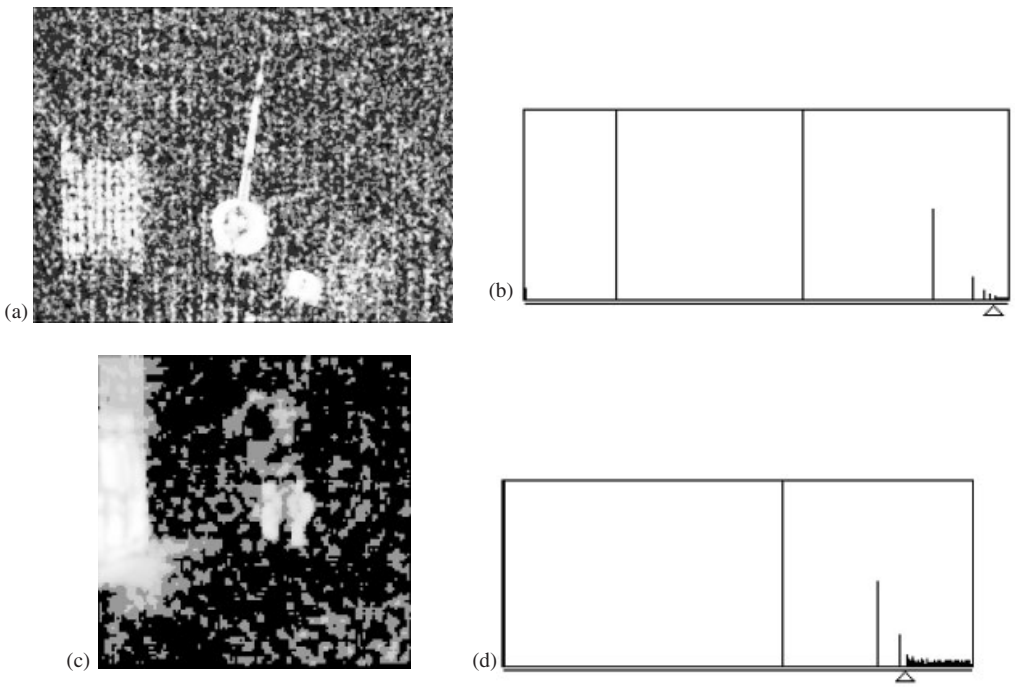
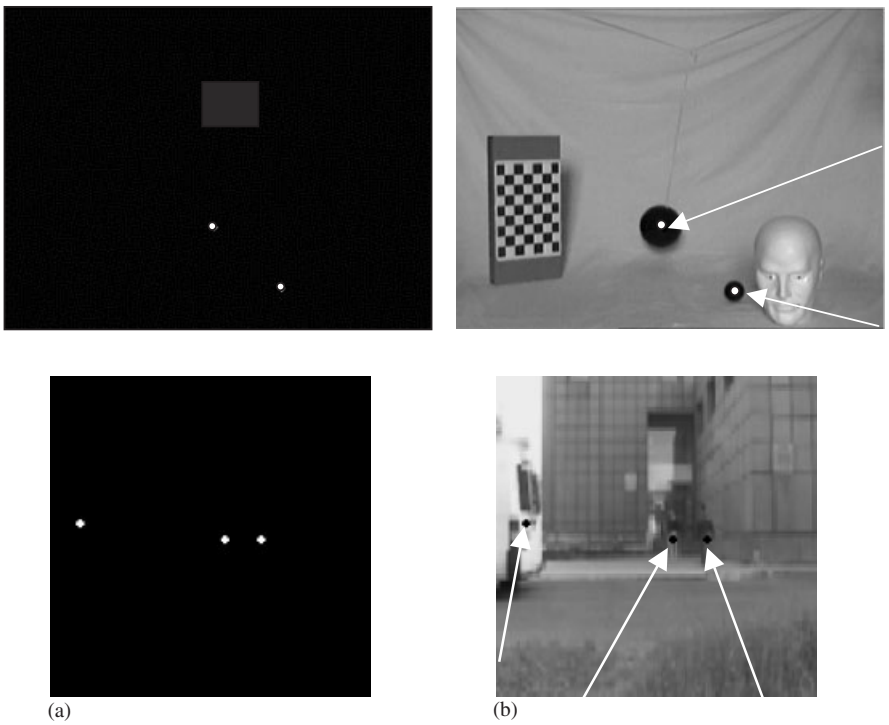


Figure 7. (a) Image in Figure 6(a) after histogram equalization; (b) The equalized histogram; (c) image in Figure 6(c) after histogram equalization; and (d) the equalized histogram.

quantization levels, and several adjacent small lines concentrated in the final part of the intensity axis. The arrows in Figures 7(b) and (d) represent the separation between these two zones. In all our tests on different image sequences, we verified that the isolated lines correspond to the background, while the rightmost continuous distribution represents the moving objects. So, an optimal choice for the threshold is the value $I_o = T^{-1}(J_o)$, where J_o is the level after the last isolated line in the equalized histogram (corresponding to the arrows in Figure 7). This value can be obtained by an automatic procedure, due to the following facts: (a) in the equalized histogram, the maximum intensity value is always present (due to the type of nonlinearity); (b) the number of adjacent lines is usually a small fraction of N. So, it is sufficient to explore the equalized histogram from right to left, until an isolated line is found. The value of the threshold I_o can be obtained by inversion of (3) by a look-up table. In Figure 7(b), it is $J_o = 0.97$ corresponding to $I_o = 0.023$ which is the threshold value used in the simulations. In Figure 7(d), it is $J_o = 0.85$ corresponding to $I_o = 0.015$.

The procedure for threshold selection must be performed at the beginning and should be repeated at fixed times for better results. The repetition rate depends on the distance of moving objects from the camera and on the application, however usually it is a small fraction of the rate with which image pairs are processed. In our simulations, histogram equalization has been obtained using conventional digital algorithms; however, it is worth noting that also this processing step could be realized using CNNs, as explained in Reference [12].



(a) (b)

Figure 8. (a) Results of centre point detection; and (b) correspondence with the initial image.

3.2. *Centre and size estimation*

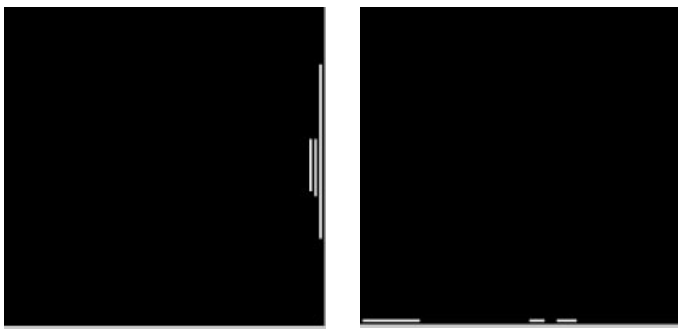
Once the image P^* is obtained, we perform *centre point detection* and *object size estimation*. Centre point detection leaves only one white pixel for each moving object, corresponding to the 'centre' of the object (for more details see Reference [19]). The input image and the initial state are represented by image P^* (Figures 4(h) and 5(h) for the two sequences of images). The effect of centre point detection is shown in Figure 8.

In order to estimate the size of moving objects, we perform the *connected component detection* (CCD, see [20]) both on the horizontal and vertical directions. The initial state is the image P^*. The results of CCD are shown in Figures 9 and 10.

Both in Figures 9(a) and (b) there are two projections in sequence; so, the width of each object can easily be obtained by pairing the vertical and horizontal projections: the association can easily be resolved by using the information on centre points in Figure 8. However, this is not the case in Figure 10. In Figure 10(b) there are three vertical projections in sequence, from the left to right, but the horizontal projections in Figure 10(a) are overlapped. In this case, we can estimate only the maximum vertical dimension of the three objects. The ambiguity may be only resolved by directly using the information in Figure 5(h).

(a) (b)

Figure 9. Effect of connected component detection in the 'pendulum' sequence: (a) horizontal projections; and (b) vertical projections.



(a) (b)

Figure 10. Effect of connected component detection in the 'van' sequence: (a) horizontal projections; and (b) vertical projections.

4. PROCESSING TIME AND HARDWARE IMPLEMENTATION

The proposed algorithm was tested by software simulation of the CNN dynamics, giving the results shown in the previous sections. An important issue concerning the applicability of our method is the processing time. In most of the templates used in the algorithm only the central element of **A** is not zero; in this case, all the cells evolve in parallel and the convergence is very fast. The most time-consuming steps of the algorithm are: filling of concave locations, connected component detection and centre point detection. The first two operations require a diffusion-type **A** template; as a consequence the convergence time depends on the image size. Centre point detection is a multi-step operation: eight feedforward templates must be repeatedly applied until the centre point is extracted. At each step the CNN must reach the steady-state; the number of steps and then the computation time depend on the size of moving objects. The processing time of the algorithm can be estimated in terms of analog time constants (τ) needed for convergence. Convergence of the CNN dynamics is assumed when the variation of the state at integration step n is less than 0.1% of the state at step $n-1$. The

Table I. Estimation of processing time.

Processing step	No. time constants
Inversion	4.5
Sum	4.2
Abs	5.1
Filtering	4.2
Thresholding	15.3
Isolated point removal	7.2
Filling concave locations	84.4
Centre point	185
CCD (hor. and vert.)	$185 + 153 = 338$

number of time constants required by each processing step are summarized in Table I for the image pair in Figures 5(a) and (b) (from the 'van' sequence).

We have 4–5τ for convergence in each processing step, except for the diffusion templates (filling of concave locations, connected component detection both horizontal and vertical) and centre point detection. Summing the values in Table I we have the processing time: $t_p = 648\tau$.

Another important aspect is hardware implementation. Since all the templates used in the algorithm are 3×3 and space invariant, practical implementation can be obtained using existing CNN analog, optical or mixed-type hardware, with slight modifications if necessary. In particular the 64×64 CNN chip designed in Seville (Spain) can be used [13, 14]. The chip is based on 0.5 μm CMOS technology and allows grey-scale analog electrical or optical input. Neighborhood is 3×3; the chip memory can store 32 templates and 64 switch configurations which are more than enough for the proposed algorithm, requiring 17 different templates (including the eight templates for centre point detection). Note that the SUM template, used to obtain the difference image between two frames, requires a cell-dependent bias I_{ij} which can be realized using an extra external input to each cell in addition to u_{ij}. A modified cell circuit is required for the realization of the absolute value template, which is nonlinear; the piecewise-linear nonlinearity can be realized as explained in References [15, 16].

The processing time of the hardware realization can be estimated assuming a time constant $\tau = 1.2$ μs [13], hence $t_p \cong 777$ μs. Assuming the minimum frame separation $k = 1$, the maximum frame rate compatible with the processing time is $1/t_p = 1286$ frames/sec.

Threshold computation has not been considered in this estimate because it is performed with a repetition rate that is far lower with respect to the frame rate.

5. CONCLUSIONS

A CNN algorithm has been proposed for position and size estimation of moving objects in high resolution grey-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 until the co-ordinates of

the centre and the overall size of the moving objects have been extracted. The algorithm uses thresholding to separate moving objects from standing ones, and an automatic procedure for correct threshold selection has been proposed. The simulation results show the good behaviour of the proposed method. The major limitation of the algorithm concerns the maximum speed of the objects. Moreover, if the lighting conditions or the background can change rapidly, the threshold computation must be repeated faster, with an increased computational cost.

ACKNOWLEDGEMENTS

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. Chua LO, Yang L. Cellular neural networks: theory. *IEEE Transactions on Circuits Systems* 1988; **35**: 1257–1272.
2. Chua LO, Yang L. Cellular neural networks: applications. *IEEE Transactions on Circuits Systems* 1988; **35**:1273–1290.
3. Roska T, Chua LO. CNN Universal Machine: an analogic array computer. *IEEE Transactions on Circuits and Systems* 1993; **40**(3):163–173.
4. Roska T. Analogic CNN computing: architectural, implementation, and algorithmic advances: a review. *IEEE 5th International Workshop on Cellular Neural Networks and Applications*, CNNA-98: London, 1998; 3–10.
5. Roska T, Zaradny A, Zold S, Foldesy P, Szolgay P. The computational infrastructure of analogic CNN computing—Part I: the CNN-UM chip prototyping system. *IEEE Transactions on Circuits Systems CAS-I* 1999; **46**:261–268.
6. Roska T, Boros T, Radvanyi A. Detecting moving and standing objects using cellular neural networks. *International Journal of Circuit Theory and Applications* 1992; **20**:613–628.
7. Gacsadi A, Szolgay P. An analogic CNN algorithm for following continuously moving objects. *Proceedings of the IEEE International Workshop on Cellular Neural Networks and Applications*, *CNNA*-2000, Catania, 2000; 99–104.
8. Kim H, Roska T, Chua LO, Werblin F. Automatic detection and tracking of moving image target with CNN-UM via target probability fusion of multiple features. *International Journal of Circuit Theory and Applications* 2003; **31**:329–346.
9. Yang T, Yang L-B, Yang X-P. Application of a cellular neural network to facial expression animation and high-level image processing. *International Journal of Circuit Theory and Applications* 1996; **24**:425–450.
10. Grassi G, Grieco LA. Object-oriented image analysis via analogic CNN algorithms—Part I: motion estimation. *Proceedings of IEEE 7th International Workshop on Cellular Neural Networks and Applications*, *CNNA*-2002, Frankfurt, 2002; 172–179.
11. Jain AK. *Fundamentals of Digital Image Processing*. Prentice-Hall: Englewood Cliffs, NJ, 1989.
12. Csapodi M, Roska T. Adaptive histogram equalization with cellular neural networks. *Proceedings of IEEE International Workshop on Cellular Neural Networks and Their Applications*, Seville, 1996; 81–86.
13. Liñán G, Espejo S, Domínguez-Castro R, Rodríguez-Vásquez A. The CNNUC3: an Analog I/O 64×64 CNN Universal Machine Chip Prototype with 7-Bit Analog Accuracy. *Proceedings of IEEE International Workshop on Cellular Neural Networks and Their Applications*, *CNNA'2000*, Catania, 2000; 201–206.
14. Szatmári I, Zarándy Á, Földesy P, Kék L. An analogic CNN engine board with the 64×64 analog I/O CNN-UM chip. *Proceedings of IEEE ISCAS 2000*, vol. II, Geneva, 2000; 124–127.
15. Liñán G, Földessy P, Rodríguez-Vásquez A, Espejo S, Domínguez-Castro R. Realization of nonlinear templates using the CNNUC3 prototype. *Proceedings of IEEE International Workshop on Cellular Neural Networks and Applications*, *CNNA-2000*, Catania, 2000; 219–224.
16. Liñán G, Földessy P, Rodríguez-Vásquez A, Espejo S, Domínguez-Castro R. Implementation of nonlinear templates using a decomposition technique by a 0.5 μm CMOS CNN universal chip. *IEEE ISCAS 2000*, vol. II, Geneva, 2000; 401–404.
17. *CNN software library*, Neuromorphic Information Technology Lab., Budapest, 2001, http://lab.analogic.sztaki.hu

18. Zarándy Á, Werblin F, Roska T, Chua LO. Novel types of analogic CNN algorithms for recognizing bank-notes. *Proceedings of IEEE International Workshop on Cellular Neural Networks and Their Applications*, Rome, 1994; 273–278.
19. Harrer H, Venetianer PL, Nossek JA, Roska T, Chua LO. Some examples of preprocessing analog images with discrete-time cellular neural networks. *Proceedings of IEEE International Workshop on Cellular Neural Networks and Their Applications*, Rome, 1994; 201–206.
20. Matsumoto T, Chua LO, Suzuki H. CNN cloning template: Connected component detector. *IEEE Transactions on Circuits Systems* 1990; **37**(5):633–635.

