Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 5, pp. 719-726

A Homogeneous Algorithm for Motion Estimation and Compensation by Using Cellular Neural Networks

C. Grava, A. Gacsádi, I. Buciu

Cristian Grava, Alexandru Gacsádi, Ioan Buciu
University of Oradea
Faculty of Electrical Engineering and Information Technology
Oradea, Romania
E-mail: {cgrava,agacsadi,ibuciu}@uoradea.ro

Abstract: In this paper we present an original implementation of a homogeneous algorithm for motion estimation and compensation in image sequences, by using Cellular Neural Networks (CNN). The CNN has been proven their efficiency in real-time image processing, because they can be implemented on a CNN chip or they can be emulated on Field Programmable Gate Array (FPGA). The motion information is obtained by using a CNN implementation of the well-known Horn & Schunck method. This information is further used in a CNN implementation of a motion-compensation method. Through our algorithm we obtain a homogeneous implementation for real-time applications in artificial vision or medical imaging. The algorithm is illustrated on some classical sequences and the results confirm the validity of our algorithm.
Keywords: cellular neural networks, motion estimation, Horn & Schunck method.

1 Introduction

The motion estimation and compensation algorithms were developed for different applications as artificial vision, video information compression, medical imaging, digital and high-definition television, video-telephony, virtual-reality and multimedia techniques. Motion estimation allows one to reduce the temporal redundancy in a sequence of images in order to reduce the transmission rate and has been widely used in television signal coding (e.g. motion-compensated (MC) prediction, MC interpolation) and videoconference services [1]. To avoid this limitation in this paper we propose a fully parallel solution in order to realize the motion estimation and compensation, using CNN [2], as a competing alternative to classical computational techniques. The advantage of the algorithms that can be implemented on CNNs is that these kinds of neural networks already exists in hardware version [3], [4] and thus we can obtain real-time applications. In our case, because the motion estimation and compensation methods have generally a great computational cost, we develop homogeneous algorithms (that is the estimation part and compensation part are implemented in the CNN environment) for real-time applications that can be after that applied in artificial vision or medical imaging. After a small introduction, in the second part of this paper we present an overview of motion estimation and compensation methods, followed by a section that presents the CNN implementation of the Horn and Schunck motion estimation method and a section that presents the CNN implementation of the motion compensation. In the section dedicated to experimental results we present results that confirm the validity of our algorithm and we finalize our paper with a section of conclusions giving also some perspectives to our work.

2 Overview on motion estimation and compensation methods

The most used motion estimation methods are [5]:

- differential methods (or gradient methods); in this case the motion being estimated based on the spatial and temporal gradients of images [6], [7], [1];
- block-based methods (or correlative methods). These methods could be classified in phase-correlation methods and block-matching methods. In the case of phase-correlation methods, the motion is estimated based on the Fourier phase-difference between two blocks from two successive images. These methods are less used in practice because of the high noise-sensitivity. In the case of block-matching methods the location of the block (in the following or previous images) that best matches the reference block in the current image is searched, based on a certain matching or difference criteria. Both methods are usually applied in the case of a translation movement, but could be also adapted for other spatial models of the movement [8].

The principle of almost all motion estimation methods is that the brightness intensity of each pixel is constant along the motion trajectory or is modifying in a predictable way. This hypothesis of brightness intensity preservation of each point (x,y,t) along the motion trajectory can be expressed through the equation of *Displaced Frame Difference* (DFD), between the t and $t-1 = t-\Delta t$ instants [9]:

$$DFD(x,y) = \Phi(x-dx, y-dy, t-dt) - \Phi(x,y,t), \tag{1}$$

where $\Phi(x,y,t)$ denote the brightness distribution of the image at the moment t and $\mathbf{d} = [dx, dy]^T$ is the displacement vector between the moments t and $t' = t-\Delta t$ (dx and dy being the displacement vectors on x and y direction, respectively).

Differential motion estimation methods are based on spatial and temporal gradients of a sequence of images. If the brightness intensity of a pixel is not varying in time, then $d\Phi/dt = \Phi^t = 0$ [1]. The first order Taylor development of this last relation, has as result the "equation of movement constraint" EMC (or "optical flow equation" OFE) that links the spatial and temporal gradients of brightness intensity [9]:

$$\frac{\partial\Phi}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial\Phi}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial\Phi}{\partial t} = 0. \tag{2}$$

We can rewrite:

$$\Phi^x \cdot v_x + \Phi^y \cdot v_y + \Phi^t = 0, \tag{3}$$

where Φ^x and Φ^y are the spatial gradients, Φ^t is the temporal gradient of the brightness intensity and $v_x = dx/dt$, $v_y = dy/dt$ are the velocities on x and y directions [1].

As it can be observed in eq. 3, the optical flow equation has two unknowns (v_x, v_y), hence the system is under-determined leading to an ill-posed issue. In order to obtain both movement components (v_x, v_y), it has to be introduced a second constraint, to obtain a fully determined system (two equations with two unknowns). One of the possible constraints is offered by the well-known Horn & Schunck motion estimation method [1], which assumes that all the neighbor pixels have similar movement (we say that the velocity field is uniform or smooth). It results that it has to be minimized an energy:

$$E^2 = E^2_{flow} + \gamma E^2_{uniformity} \tag{4}$$

The first term correspond to the difference related to the projection of the velocity vectors on the spatial gradient (as in the EMC) and the second term correspond to the difference related

to a smooth field, g being the weighting term between the two terms. The uniformity constrain is expressed by the equation:

$$E^2_{uniformity} = \left(\frac{\partial v_x}{\partial x}\right)^2 + \left(\frac{\partial v_x}{\partial y}\right)^2 + \left(\frac{\partial v_y}{\partial x}\right)^2 + \left(\frac{\partial v_y}{\partial y}\right)^2 \quad (5)$$

$$E^2_{flow} = \left(\Phi^x \cdot v_x + \Phi^y \cdot v_y + \Phi^t\right)^2 \quad (6)$$

The solution is obtained after a Gauss-Seidel minimization [1]. Equation 6 will be minimized when the error (that means the difference between two successive values of (v_x, v_y), will be considered as being the minimal or when the maximum chosen number of iterations will be reached. This method is not limited to translations as block-matching method and the computations are shorter, but the movement amplitude has to be small (less than three pixels) because of the considerations regarding Taylor development. The solution to avoid the constraint regarding small amplitude of movement is to use the multi-resolution technique [1]. Using two consecutive frames ($\Phi_i(x, y, t_i)$ and $\Phi_f(x, y, t_f)$) of a sequence (Fig. 1), after the application of a motion estimation algorithm, as Horn and Schunck method, for each pixel it results an estimation of its movement in the both two directions (x,y) of the system of co-ordinates that is attached to image plane.

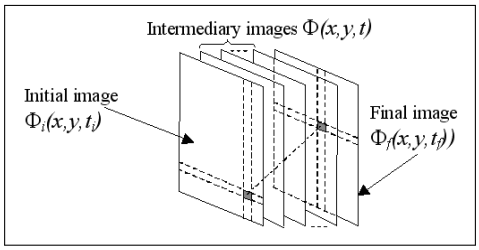

Figure 1: The illustration of motion compensation.

The purpose of motion compensation is that based on estimated motion information and starting from a reference image (the initial image Φ_i, in Fig. 1) to obtain an estimation of the real comparison image (the final image Φ_f) that was used in the process of motion estimation [10].

3 The CNN implementation of the Horn & Schunck motion estimation method

Regarding CNN gray-scale image processing generally, variational computing based template design is possible if the design constrains are respected [11], [12]. In order to analytically determine the template, cost functions or energies are used in the designing step. An important designing aspect is represented by the way to associate each energy function with one of the A, B, C or D template, as well as the way to chose the layers number of the CNN. Taking into account the characteristics of the existing CNN chip it is recommended to use only mono-layer CNN and only A and B templates. In the cost functions some weights can be introduced in order to maintain the state values in the linear zone of the state-output transfer characteristic. The motion estimation results using the two images, $\Phi_1(x, y, t)$ and $\Phi_2(x, y, t + \Delta t)$, a two-layer CNN structure and the *Hosch.tem* (see Fig. 2). After the cost function minimization [13], for the *Hosch.tem* it results:

- polarization images $Z_1 = \Phi^x\Phi^t$ and $Z_2 = \Phi^y\Phi^t$,

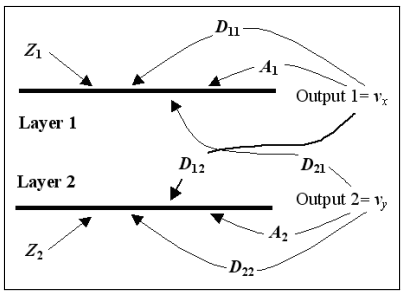

Figure 2: Two-layer CNN structure for the proposed Horn & Schunck motion estimation method.

- nonlinear templates $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$:

0	a	0
a	1-4a	a
0	a	0

where the parameter a also includes the γ parameter from the equation (6), obviously weighted with the constants that results at the energy minimization. Nonlinear $\mathsf{D}-\mathsf{type}$ template D is:

0	0	0
0	d_{kl}	0
0	0	0

where each element d_{kl} is expressed as follows: $d_{11}(v_x) = (\Phi^x)^2 \cdot v_x$, $d_{21}(v_y) = \Phi^x\Phi^y \cdot v_y$, $d_{22}(v_y) = (\Phi^y)^2 \cdot v_y$, and $d_{12}(v_x) = \Phi^x\Phi^y \cdot v_x$, respectively.

In Fig. 3 the images with the estimated motion in the case of the "taxi" real sequence are presented. The first two images represent two images of the well-known sequence in motion estimation reference, "Hamburg-taxi", and in the last two images the motion v_x and v_y (or displacement) images are presented. The two last images represent the displacement in the two spatial directions. The combination between these two images could be also represented as a single image with vectors corresponding to the displacement of each pixel of the reference image, as we will present in the section presenting other experimental results. For a better visualization, the motion images, v_x and v_y, are not calibrated in the CNN domain. If we want to use these images in image compensation we have to calibrate them in the CNN domain $[-1, +1]$.

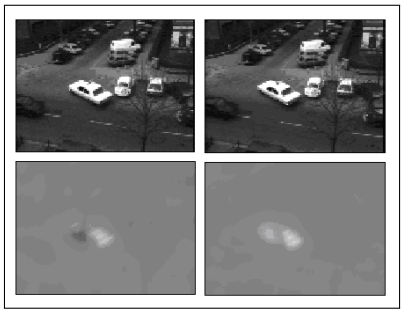

Figure 3: Two images of the Hamburg-taxi real sequence, used in our experiments.

4 The CNN implementation of motion compensation

In order to develop a motion compensation algorithm that can be directly implemented on a CNN chip, we have to decompose such an algorithm as elementary steps that can be then implemented on the existing hardware. As a result of the motion estimation process, a pixel could be stationary or can change its position in one of the eight elementary directions: N, N-E, E, S-E, S, S-W, W, N-W. After the application of a motion estimation technique, the pixels of the intermediary image frame $\Phi(x, y, t)$, that corresponds to any given moment $t \in (t_i, t_f)$, could be classified in the following categories (see Fig. 1), where t is the time-position of the intermediary image, between the initial image and the final image:

- Pixels of "*a*" type that has an identical position in the two consecutive images. These pixels does not change, at a given moment $t \in (t_i, t_f)$, neither their positions nor their values;
- Pixels of "*b*" type, that will moves as a result of the fact that the corresponding pixels in the two images that contains the motion information has a value greater than a current elementary value (that could be view as a quantum or a threshold). The value of these pixels is not changing, but inserting intermediary images between the initial and final image, Φ_i and Φ_f, their positions are changing successively with one elementary value (one quantum) corresponding to the spatial discretization. The maximum number of intermediary images that could be inserted equals the maximum number of elementary values (quantum) that could be identified in the images that contains motion (or displacement) information;
- Pixels of "*c*" type are those pixels that will change their values because will be covered by the pixels that will arrive in that position, overlapping the initial pixel:

$$c(t) = \mathrm{shift}(b(t)) \tag{7}$$

- Pixels of "*d*" type, with unknown values, that are the result of the displacement of "*b*" type pixels, that liberates a location but there is no pixel arriving in that location. In each step of the movement, the value of these pixels could be determined through spatial CNN spline-cubic interpolation [10]:

$$d(t) = \lfloor d(t-1) \cdot \overline{c}(t) + b(t) \cdot \overline{c}(t) \rfloor \cdot b(1) \tag{8}$$

- Pixels of "*e*" type that at the current time during the processing will have the same value as in the initial image Φ_i, due to the movement of the pixels (arrivals and departures of the pixels). The values of these pixels will be restored from the initial image:

$$e(t) = c(t-1) \cdot b(t) \cdot \overline{c}(t) \cdot \overline{d}(t) \tag{9}$$

Each intermediary step has as result an associate image and to create this intermediary image it has to be done the following operations:

- determine the "*c*" type pixels, that is the displacement with one pixel in the direction resulting from the motion information;
- interpolation, in order to determine the values of unknown pixels, that is the "*d*" type pixels.

For each intermediary image, the value of a pixel results after the determination of the type of that pixel. The state of a pixel could change during the processing. The initial and final image,

Φ_i and Φ_f, and the images that contain the motion information have the same dimensions. In this paper, all images are converted to standard CNN gray-scale images, taking values between -1 to +1 (see Fig. 5). The convention is that pixels with negative value are coding a displacement to the left (Fig. 5 a) or to up, respectively (Fig. 5 b) and the pixels having positive values are coding the displacements to the right (Fig. 5 a) or to down, respectively (Fig. 5 b). The values of the pixels coding the motion are multiples of the minimum detectable value of the motion. In order to detect the pixels that will change their position and to move the pixels, the *treshold.tem* and *shift.tem* templates family are used [14]. The determination of the value of pixels of "*d*" type could be made for each intermediate image or only to the final image. In order to avoid the modification of the pixels of "*a*" type or in order to restore the pixels of "*e*" type, some mask-images are created during the processing, using the equations (7), (8) and (9).

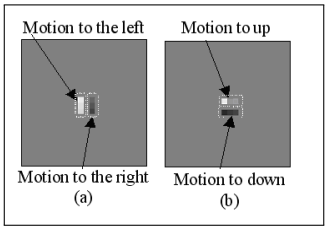

Figure 4: Conventions in the images containing the motion information.

5 Experimental results

In this section some experimental results obtained by using the "CadetWin" (CNN Application Development Environment and Toolkit under Windows [14]) are presented. The images containing the motion estimation, v_x and v_y, are calibrated in the CNN domain $[-1, +1]$. The processing time depends on the number of interpolations and on the number of the motion estimation quantum, that results after spatial discretization, that is on the total number of images inserted between the initial and final image, Φ_i and Φ_f. Due to parallel processing, this total processing time is independent by the dimensions of the original images or by the number of moving pixels. In order to illustrate the implemented method in the case of real images and in the case of a complex movement, starting from an initial image of a well-known "tennis-table" sequence, we have simulated a complex motion, using the Free Form Deformation principle, resulting (see Fig. 5) three real images of a sequence (Φ_1, Φ_2, Φ_3). In Figure 5 we also represented the "Motion Estimation Field" obtained after applying our CNN implementation of the Horn and Schunck motion estimation method. Starting from the first image of the real sequence (Φ_1) and this "Motion Estimation Field" we can obtain the "Motion Compensated image" ($\widehat{\Phi}_2$), that represent an estimation of the real image (Φ_2).

As it can be observed, the biggest errors between the real image and the motion compensated image ($\Phi_2 - \widehat{\Phi}_2$) are located in the region with a high gradient of intensity that usually corresponds to the regions with different motion. Another cause oh these errors could also be the discrete nature of the image spatial support and the interpolations that are necessary.

6 Conclusions

Generally, in the case of serial implementation of a motion compensation algorithm, the processing time depends on the image dimensions. In the case of our CNN motion estimation and compensation algorithm, that uses only 3×3 linear templates, the algorithm can be directly

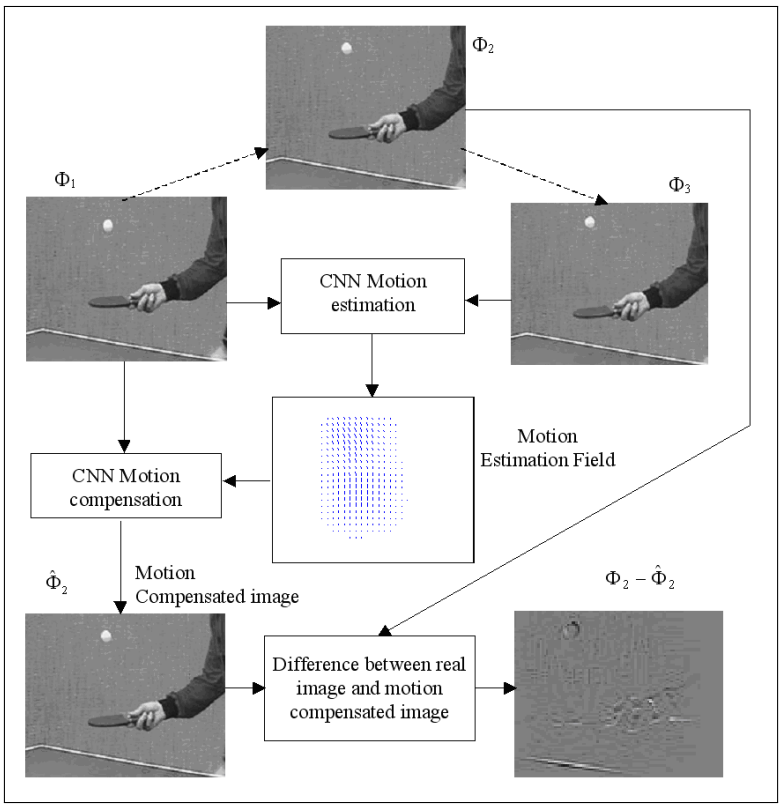

Figure 5: Motion estimation and compensation using CNN: principle and results.

implemented on the CNN-Universal Chip [4] and thus the image processing become completely parallel. The advantage of using the CNN hardware platform is that the total processing time doesn't depend on image dimensions, being dependent only on the number of displacement steps that has to be performed and thus we can obtain real-time applications with applications in artificial vision and medical imaging. Taking into account that our CNN motion estimation and compensation algorithm is based on the cost functions minimization (usually partially differential equations) resulting nonlinear templates, our attention is focused on the FPGA implementation of our algorithm, on a digital emulator of the CNN [11], [12].

7 Acknowledgement

This work was partially supported by a grant from the Romanian National University Research Council, PNCDI Program ID-668/2008.

Bibliography

[1] Horn B.K.P. and Schunck B.G, Determining Optical Flow, *Artificial Intelligence*, Vol.17, pp. 185-203, 1981.

[2] Chua L.O. and Yang L, Fuzzy Control Rules in Convex Optimization, *IEEE Transactions on Circuits and Systems*, Vol.35, pp.1257-1290, 1998.

[3] Cembrano G.L., Rodríguez-Vázquez A., Espejo-Meana S., and Domínguez-Castro R., ACE16k: A 128×128 Focal Plane Analog Processor with Digital I/O, *Int. J. Neural Syst.,* Vol.17, Issue 6, pp. 427-434, 2003.

[4] Roska T. and Chua L.O., The CNN universal machine: an analogic array computer, *IEEE Transactions on Circuits and Systems,* Vol.40, pp. 167-173, 1993.

[5] Konrad J., Motion detection and estimation, *Image Processing Handbook, Networking and Multimedia,* pp. 207-227, 2000.

[6] Bruhn A., Weickert J., Feddern C., Kohlberger T., Schnorr C., Real-time optic flow computation with variational methods, *Computer Analysis of Images and Patterns,* pp. 222-229, 2003.

[7] Brox T., Bruhn A., Papenberg N., Weickert J., High accuracy optical flow estimation based on a theory for warping, *ECCV,* pp. 25-36, 2004.

[8] Wei W., Hou Z.-X., Guo Y.-C., A displacement search algorithm for deformable block matching motion estimation, *Proc. of IEEE International Symposium on Communications and Information Technology,* pp. 457-460, 2005.

[9] Barron J.L., Fleet D.J., Beauchemin S., Performance of Optical Flow Techniques, *International Journal of Computer Vision,* Vol. 12, Issue 1, pp. 43-77, 1994.

[10] Grava C., Gacsádi A., Gordan C., Maghiar T., Bondor K, Motion Compensation using Cellular Neural Networks, *Proc. of the European Conference on Circuit Theory and Design (ECCTD),* Vol. I, pp. I-397-I-400, Krakow, Poland, 2003.

[11] Kincses Z., Nagy Z., Szolgay P, Implementation of nonlinear template runner emulated digital CNN-UM on FPGA, *Proc. of the 10th International Workshop on Cellular Neural Networks and Their Applications,* pp. 186-190, Istanbul, Turkey, 2006.

[12] Nagy Z., Vörösházi Zs., Szolgay P., Emulated Digital CNN-UM Solution of Partial Diferential, *Int. Journal of Circuit Theory and Applications,* Vol. 34, Issue 4, pp. 445-470, 2006.

[13] Gacsádi A., Grava C., Tiponut V., Szolgay P., A CNN implementation of the Horn & Schunck motion estimation method, *Proc. of the 10th International Workshop on Cellular Neural Networks and Their Applications,* pp. 381-385, Istanbul, Turkey, 2006.

[14] *** CadetWin, CNN application development environment and toolkit under Windows. Version 3.0, Analogical and Neural Computing Laboratory, Hungarian Academy of Sciences, Budapest, 1999.