Object-Oriented Image Analysis Using the CNN Universal Machine: New Analogic CNN Algorithms for Motion Compensation, Image Synthesis, and Consistency Observation

Giuseppe Grassi, *Senior Member, IEEE*, and Luigi Alfredo Grieco, *Student Member, IEEE*

Abstract—Image-analysis algorithms are of great interest in the context of object-oriented coding schemes. With reference to the utilization of the cellular neural network (CNN) universal machine for object-oriented image analysis, this paper presents new analogic CNN algorithms for obtaining motion compensation, image synthesis, and consistency observation. Along with the already developed segmentation and object labeling technique, the proposed method represents a framework for implementing CNN-based real-time image analysis. Simulation results, carried out for different video sequences, confirm the validity of the approach developed herein.

Index Terms—Analogic CNN algorithms, cellular neural networks (CNNs), CNN based video coding, neural circuits for image processing, object-oriented image and analysis, spatiotemporal dynamics via cellular array.

I. INTRODUCTION

IN RECENT years, great efforts have been devoted to the study of coding techniques providing high compression ratios, while maintaining good picture quality. Among these techniques, the object-oriented coding approach represents an interesting and sophisticated method for video coding [2]–[5]. It mainly consists of a strong image-analysis stage (see Fig. 1), that can locate and label shape, color, and motion of objects appearing in a frame of a video sequence. In particular, objects of great importance for image understanding can be detected and coded using special parameter sets, whereas still background objects are not included in the coding procedure. This approach enables the transmission rate to be greatly reduced, since for reconstructing some objects the receiver can exploit the information already obtained from the previous frames. Namely, the “important” objects are fed adaptively into the channel until the upper bound of the transmission rate is reached, whereas the remaining parts of the frame can be obtained by restoring the information of the already received frames.

Since object-oriented coding schemes require very powerful and flexible devices [6], the idea recently proposed in [1], [7] is to exploit the great computational power offered by the cellular

Fig. 1. Block diagram of the image-analysis algorithm of an object-oriented coding scheme.

neural network universal machine (CNNUM) for obtaining accurate image analysis. Notice that the CNNUM has been already implemented in the form of a CNN universal chip [8]–[15], which represents an analog fully programmable supercomputer available on a single chip.

When applied to object-oriented coding schemes, the CNNUM is the engine responsible for the image-analysis operations, whereas the remaining coding and transmitting operations can be managed by digital coprocessors [7]. However, regarding the utilization of the CNNUM for image analysis, it should be pointed out that only few results are available in literature. This is because object-oriented image

analysis (Fig. 1) mainly consists of four parts: segmentation and object labeling, motion compensation, image synthesis and consistency observation. The results available in literature only concern the development of CNN algorithms for obtaining the segmentation and object labeling stage [1], [7]. No result is available concerning the utilization of CNNUM for motion compensation, image synthesis and consistency observation.

The aim of the paper is to bridge this gap by developing the remaining stages of the block diagram reported in Fig. 1. This objective is achieved by illustrating new *analogic* CNN algorithms, which are based on existing CNN templates [16] as well as on new ones. The paper is organized as follows. In Section II, the CNN algorithm for implementing the motion-compensation stage is described. This stage mainly consists of modeling the geometrical and motion properties of the objects, and encoding them using a special parameter set. In Section III, the CNN algorithm for implementing the image-synthesis stage is illustrated. In particular, the algorithm enables all parameterized image fragments to be linked in order to obtain a synthesized image. In Section IV, the CNN algorithm for implementing the consistency observation block is presented. This stage mainly consists of a comparison between the synthesized image and the original frame, with the aim of obtaining a “filtered difference image.” In Section V, simulation results and comparisons are reported to show the effectiveness of the proposed approach. These simulations have been carried out using *Miss America*, *Claire*, and *Stefan* video sequences. Finally, in Section VI, a discussion is reported, with the aim of illustrating strengths and weaknesses of the approach developed herein.

II. MOTION COMPENSATION

Before illustrating the proposed CNN algorithm for motion compensation (Fig. 2), we would briefly remind that the CNN model considered through the paper is the following [17]:

$$\begin{aligned} \dot{x}_{ij}(t) = & -x_{ij}(t) + \sum_{kl \in N_r} A_{ij,kl} y_{kl}(t) + \sum_{kl \in N_r} B_{ij,kl} u_{kl}(t) \\ & + \sum_{kl \in N_r} \hat{A}_{ij,kl}(y_{kl}) + \sum_{kl \in N_r} \hat{B}_{ij,kl}(u_{kl}) + I_{ij} \quad (1a) \end{aligned}$$

$$y_{ij}(t) = f(x_{ij}(t)) = 0.5(|x_{ij}(t) + 1| - |x_{ij}(t) - 1|) \quad (1b)$$

where $x_{ij}(t)$ is the state, $y_{ij}(t)$ is the output, $u_{ij}(t)$ is the input, I_{ij} is the bias, $A_{ij,kl}$, and $\hat{A}_{ij,kl}$ are the linear and nonlinear feedback parameters, respectively, $B_{ij,kl}$ and $\hat{B}_{ij,kl}$ are the linear and nonlinear control parameters, respectively, whereas $kl \in N_r$ is a grid point in the neighborhood within the radius r of the cell ij . Five different images can describe a CNN layer, that is, the input U , the state X , the output Y , the bias I , and the mask M . Along with the model, we would also remind that the CNNUM incorporates some advanced computational capabilities, such as [18] the following:

- capability to perform ADDITION and SUBTRACTION of grayscale images, also pixel by pixel;
- capability to combine, pixel by pixel, two binary images through logic operations such as AND, OR;
- capability to select which cells are going to be processed (the so called mask M).

Fig. 2. Block diagram of the proposed “motion-compensation algorithm.”

The motion-compensation approach proposed herein consists of determining in the frame I_{i+n} , the position of those objects belonging to frame I_i , which represent the result of the segmentation technique described in [1]. The position of these objects in the frame I_{i+n} can be computed by moving each object (belonging to frame I_i) in a $p \times q$ -pixel window and by comparing the result with the frame I_{i+n} . Notice that the dimensions p and q of the search-window depend on the motion features of the video sequence to be processed (see Example 2 in Section V). Finally, the position of the object that corresponds to the smallest error is stored for the next stage.

More precisely, the proposed CNN algorithm can be described as follows. For each object individuated by the segmentation stage, the first step consists in computing the difference between the frames I_i and I_{i+n} , with reference to the object extension. This can be easily done by the CNNUM, since it is possible to exploit the SUBTRACTION operation, along with the mask M_i (which describes the extension of the i th object belonging to frame I_i) [18]. The result (that is, the image $Y_{(SUB)i}$) is processed by the ABSOLUTE VALUE template [7]

$$A = 0 \quad B = 0 \quad \hat{A} = 0 \quad \hat{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (2a)$$

$$\begin{aligned} X &= 0 \quad I = 0 \quad U = Y_{(SUB)i} \quad M = M_i \\ Y &= Y_{(ABS)i} \quad T = 20\tau \end{aligned} \quad (2b)$$

where b is the nonlinear function illustrated in [7], T is the computation time, whereas $Y_{(ABS)i}$ is the result of the “absolute

value” for the i th object. From the implementation point of view, the problem of having nonlinear templates can be easily solved by exploiting the results in [19] and [20], where it is shown how nonlinear templates can be decomposed into a sequence of linear template executions.

The third step consists in computing the following “mean absolute error”:

$$\text{m_error} = \frac{\sum \text{pixel values } Y_{(\text{ABS})_i}}{\sum \text{black pixels } M_i}. \quad (3)$$

Equation (3) can be implemented onto the CNNUM by considering the following DIFFUSION template [16]:

$$A = \begin{bmatrix} 0.1 & 0.15 & 0.1 \\ 0.15 & 0 & 0.15 \\ 0.1 & 0.15 & 0.1 \end{bmatrix} \quad B = 0 \quad \hat{A} = 0 \quad \hat{B} = 0 \quad (4a)$$

$$X = Y_{(\text{ABS})_i} \quad I = 0 \quad U = 0 \quad M = M_i \\ Y = \text{average value of } Y_{(\text{ABS})_i} \quad T = 30\tau. \quad (4b)$$

Namely, the application of (4) to the image $Y_{(\text{ABS})_i}$ generates a diffusion process. It forces all the pixels to reach a unique graylevel value, which represents the average value of $Y_{(\text{ABS})_i}$.

Now, it is necessary to take into account that, in order to find the position of the i th object in the frame I_{i+n} , the object and its mask have to be moved pixel by pixel within the frame I_i . By choosing a spiral trajectory (that is, a trajectory with movements UP, RIGHT, DOWN, DOWN, LEFT, LEFT and so on), the fourth step of the algorithm consists in moving the object using the following UP template [16]:

$$A = 0 \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \hat{A} = 0 \quad \hat{B} = 0 \quad (5a)$$

$$I = 0 \quad X = U \quad U = Y_{(\text{OBJ})_i} \quad M = M_{\text{black}} \\ Y = Y_{(\text{OBJ})_i}^{\text{UP}} \quad T = 5\tau \quad (5b)$$

where $Y_{(\text{OBJ})_i}$ is the image of i th object (obtained from I_i using the mask M_i), $Y_{(\text{OBJ})_i}^{\text{UP}}$ represents the image where the i th object has been moved one pixel toward the UP direction, whereas the black mask M_{black} indicates that all the cells are in active state. Along with the object, its mask M_i is moved one pixel toward the UP direction, so that the mask M_i^{UP} is obtained. Successively, the image $Y_{(\text{OBJ})_i}^{\text{UP}}$ has to be compared with the frame I_{i+n} (referring to the object extension). This objective is achieved using the SUBTRACTION capability offered by the CNNUM. The result is the image $Y = [(I_{i+n})_{M_i^{\text{UP}}} - Y_{(\text{OBJ})_i}^{\text{UP}}]$, where $(I_{i+n})_{M_i^{\text{UP}}}$ is the image obtained from I_{i+n} using the mask M_i^{UP} .

Successively, after having computed the error using (2)–(4), both the object and the mask have to be moved, along the spiral trajectory, using the templates RIGHT, DOWN and LEFT reported in [16]. Therefore, the proposed algorithm is iterated until all the movements of the object have been carried out within the $p \times q$ -pixel search-window of frame I_i . The last iteration gives the image $Y = [(I_{i+n})_{M_i^{\text{last}}} - Y_{(\text{OBJ})_i}^{\text{last}}]$, where $Y_{(\text{OBJ})_i}^{\text{last}}$ and M_i^{last} are the object and the mask that correspond to the last

movement, respectively, whereas $(I_{i+n})_{M_i^{\text{last}}}$ is the image obtained using M_i^{last} . When all the movements have been carried out, a comparison among the errors is made, until the smallest error is found. The iteration that corresponds to the minimum error gives the image

$$Y = [(I_{i+n})_{M_i^{\text{COMP}}} - Y_{(\text{OBJ})_i}^{\text{COMP}}] \quad (6)$$

where $Y_{(\text{OBJ})_i}^{\text{COMP}}$ is the compensated image of the i th object whereas M_i^{COMP} is its compensated mask. These compensated images represent the results of the algorithm, which is applied to each object individuated by the segmentation stage described in [1]. If the segmented objects detected in frame I_i are N_i , the compensated images available for the next stages are $Y_{(\text{OBJ})_i}^{\text{COMP}}$ for $i = 1, \dots, N_i$ and M_i^{COMP} for $i = 1, \dots, N_i$.

III. IMAGE SYNTHESIS

In this section, a CNN algorithm for image synthesis is developed. The algorithm consists of two parts (Fig. 3). The objective of the first part is to link the objects belonging to frame I_i in their new positions in frame I_{i+n} (i.e., in those positions already detected by the “motion-compensation algorithm” developed in Section II). Then, the objective of the second part is to combine the previous result (called “object composition”) with the background.

A. Object Composition

In order to link the objects in their new positions, it is worth noting that the objects can be added to each other only if the common parts are taken into account. Roughly speaking, the behavior of proposed algorithm [Fig. 3(a)] can be explained as follows. Given, for instance, the fourth object detected in the frame I_i by the segmentation stage, let’s add it to the first, the second and the third object (using the block ADDITION). By considering that the fourth object and the remaining ones can share some parts, before adding the objects, it is necessary to delete these common parts from the fourth object. These parts can be deleted (using the block ZERO) only if a proper mask M_{AND} is computed. This mask is the result of the block AND between the compensated mask of the fourth object (M_4^{COMP}) and the “partial mask” obtained by combining the compensated masks of first, second, and third object (M_1^{COMP} , M_2^{COMP} , M_3^{COMP} , respectively) using the block OR. Therefore, the parts individuated by the mask M_{AND} can be deleted from the fourth object and the result can be added to first, second and third object.

More precisely, the proposed CNN algorithm works as follows. Given the i th object, let’s add it to the objects $1, 2, \dots, i-1$. This task can be carried out by computing the mask $M_{(\text{AND})_i}$, which takes into account the parts shared by the i th object and the remaining ones. This mask can be easily computed by exploiting the capability of the CNNUM of combining binary images through any user-selectable logic operations [18]. In particular, by applying the OR operation to the masks M_k^{COMP} ($k = 1, 2, \dots, i-1$), it is possible to derive the binary image $\bigcup_{k=1}^{i-1} M_k^{\text{COMP}}$. Successively, by exploiting the logic AND between the binary images M_i^{COMP} and $\bigcup_{k=1}^{i-1} M_k^{\text{COMP}}$, it is possible to obtain the desired mask $M_{(\text{AND})_i}$.

Fig. 3. Block diagram of the proposed “image synthesis algorithm”; (a) object composition; (b) combination with the background.

Now, the part individualized by the mask $M_{(AND)_i}$ can be deleted from the compensated image of the i th object $Y_{(OBJ)_i}^{COMP}$ using the following ZERO template:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = 0 \quad \hat{A} = 0 \quad \hat{B} = 0 \quad (7a)$$

$$I = 0 \quad X = U \quad U = Y_{(OBJ)_i}^{COMP} \quad M = M_{(AND)_i} \\ Y = Y_{(ZERO)_i} \quad T = 5\tau. \quad (7b)$$

Therefore, the image $Y_{(ZERO)_i}$ can be added to the image $\sum_{k=1}^{i-1} Y_{(ZERO)_k}$, which in turn combines the objects 1, 2, ..., $i-1$, using the ADDITION operation. The result is the image $\sum_{k=1}^i Y_{(ZERO)_k}$. The proposed algorithm is iterated until all the N_i compensated objects are processed. The resulting image is the “object composition,” that is

$$Y_{composition} = \sum_{k=1}^{N_i} Y_{(ZERO)_k}. \quad (8)$$

B. Combination With the Background

In order to combine the “object composition” with the background, the algorithm illustrated in Fig. 3(b) is proposed. First of all, it is necessary to compute the “mask composition” $M_{composition}$. This mask, obtained through the OR operation, is given by $M_{composition} = \bigcup_{k=1}^{N_i} M_k^{COMP}$. Successively, by using the ZERO template (7a) with

$$I = 0 \quad X = U; \quad U = I_i \quad M = M_{composition} \\ Y = Y_{BACKGR} \quad T = 5\tau \quad (9)$$

Fig. 4. Block diagram of the proposed “consistency observation algorithm.”

Fig. 5. Block diagram of the proposed “mask of significant difference” algorithm.

it is possible to delete the moving objects from the frame I_i , so that the background image Y_{BACKGR} is obtained. Finally, by exploiting the ADDITION, it is possible to combine the “object composition” with the background, so that the synthesized image Y_{SYNTH} is obtained

$$Y_{SYNTH} = Y_{composition} + Y_{BACKGR}. \quad (10)$$

IV. CONSISTENCY OBSERVATION

In this section, a CNN algorithm for consistency observation is proposed (Fig. 4). Since the objective of the algorithm is to compute the “filtered difference image,” the first step consists in making a comparison between the frame I_{i+n} and the synthesized image. This task, which can be carried out using the SUBTRACTION operation, leads to the “difference image” $Y_{DIFF} = (I_{i+n} - Y_{SYNTH})$. Successively, it is necessary to compute a proper mask, called “difference mask” M_{DIFF} . This mask, which individualizes the parts of the “difference image” characterized by values greater than a prefixed threshold, can

Fig. 6. Video sequence of *Miss America*. (a) Frame 3. (b) Segmented frame 3. (c) Frame 7.

be computed using the block called “mask of significant difference.” The algorithm for implementing this block (Fig. 5) works as follows.

Given the “difference image” Y_{DIFF} , it is necessary to compute the ABSOLUTE VALUE [see (2)], so that the image Y_{ABS} is obtained. Successively, in order to detect only those pixels where the “difference image” is significant, a threshold is required. This objective can be achieved by applying the THRESHOLD template reported in [16]. However, owing to the bias value reported in [16], the “threshold operation” cannot be directly applied to the image Y_{ABS} . Namely, an amplification of Y_{ABS} is required, with a gain that equals 10. Such amplification can be achieved by recursively applying the ADDITION operation. The result is the image Y_{X10} , which can be now used as input for the “threshold operation.” The result is the image Y_{THRES} . The last step for obtaining M_{DIFF} consists in removing small objects as well as groups of isolated pixels. This step can be carried out using the SMALL OBJECT REMOVER template reported in [16]. By considering Y_{THRES} as input, the result is the image M_{DIFF} .

Finally, it is necessary to delete from the “difference image” Y_{DIFF} those pixels that are not indicated by the “difference mask” M_{DIFF} . This objective can be achieved using the following ZERO* template:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \hat{A} = 0 \quad \hat{B} = 0 \quad (11a)$$

$$I = 0 \quad X = 0 \quad U = Y_{\text{DIFF}} \quad M = M_{\text{DIFF}} \\ Y = Y_{\text{filtered DIFF}} \quad T = 5\tau \quad (11b)$$

where $Y_{\text{filtered DIFF}}$ represents the “filtered difference image.”

V. SIMULATION RESULTS

Two examples are illustrated in this Section. The aim of the first one is to show how the proposed approach works, whereas the aim of the second one is to evaluate the performances of the proposed technique for different video sequences.

Example 1: Herein, the *Miss America* sequence in the common intermediate format (CIF) is considered [1]. We start by reporting in Fig. 6(a) and Fig. 6(c), the third (I_3) and the seventh (I_{3+4}) frame of the sequence, respectively, whereas in Fig. 6(b), the result of the segmentation stage is illustrated. From Fig. 6(b), it can be argued that the segmented objects detected in frame I_3 are $N_3 = 9$. By applying the motion-compensation algorithm to the ninth object, with a 15×15 -pixel window, the selected images reported in Fig. 7 are obtained. In particular, the first iteration of the algorithm (UP movement) gives the image $Y = [(I_7)_{M_9} - Y_{(\text{OBJ})_9}]$ reported in Fig. 7(a), whereas the mask M_9 is illustrated in Fig. 7(b). The last iteration of the algorithm gives the image $Y = [(I_7)_{M_9^{\text{last}}} - Y_{(\text{OBJ})_9}^{\text{last}}]$ illustrated in Fig. 7(c), whereas the mask M_9^{last} is reported in Fig. 7(d). Since all the movements have been carried out, a comparison among the errors enables the compensated images to be found. Namely, the iteration that corresponds to the minimum error (see (6)) gives the image $Y = [(I_7)_{M_9^{\text{COMP}}} - Y_{(\text{OBJ})_9}^{\text{COMP}}]$ reported in Fig. 7(e), whereas the final result of the algorithm, that is, the compensated object image $Y_{(\text{OBJ})_9}^{\text{COMP}}$ and the compensated mask M_9^{COMP} , are reported in Fig. 7(f) and Fig. 7(g), respectively.

Now, by exploiting the images $Y_{(\text{OBJ})_i}^{\text{COMP}}$ and M_i^{COMP} for $i = 1, \dots, 9$, the algorithm for obtaining the object composition

Fig. 7. Motion-compensation algorithm applied to the ninth object. (a) Image $Y = [(I_7)_{M_9} - Y_{(OBJ)_9}]$ (first iteration, m_error = 0.0464). (b) Mask M_9 (first iteration). (c) Image $Y = [(I_7)_{M_9^{last}} - Y_{(OBJ)_9^{last}}]$ (last iteration, m_error = 0.1535). (d) Mask M_9^{last} (last iteration). (e) Image $Y = [(I_7)_{M_9^{COMP}} - Y_{(OBJ)_9^{COMP}}]$, (iteration at minimum error, m_error = 0.0343). (f) Compensated object image $Y_{(OBJ)_9^{COMP}}$. (g) Compensated mask M_9^{COMP} .

is applied. Selected images resulting from its application are reported in Figs. 8 and 9. In particular, by considering the first six objects, their composition $\sum_{k=1}^6 Y_{(ZERO)_k}$ is reported in Fig. 8(a) whereas the corresponding mask $\bigcup_{k=1}^6 M_k^{COMP}$ is illustrated in Fig. 8(b). By applying the AND operation between $\bigcup_{k=1}^6 M_k^{COMP}$ and the mask of the seventh object M_7^{COMP}

[Fig. 8(c)], the image $M_{(AND)_7}$ is obtained [Fig. 8(d)]. Such mask is used for obtaining the image $Y_{(ZERO)_7}$ [Fig. 8(e)], which can be added to $\sum_{k=1}^6 Y_{(ZERO)_k}$ for obtaining the image $\sum_{k=1}^7 Y_{(ZERO)_k}$ [Fig. 8(f)]. Finally, by starting from the composition of the first eight objects $\sum_{k=1}^8 Y_{(ZERO)_k}$ [Fig. 9(a)] and by considering the AND operation between the

Fig. 8. Object composition algorithm. (a) Image $\sum_{k=1}^6 Y_{(ZERO)_k}$ (i.e., composition of the first six objects). (b) Image $\bigcup_{k=1}^6 M_k^{COMP}$ (i.e., mask of the first six objects). (c) Image M_7^{COMP} (i.e., mask of the seventh object). (d) Image $M_{(AND)_7}$ (i.e., result of the AND operation). (e) Image $Y_{(ZERO)_7}$ (i.e., result of the ZERO operation). (f) Image $\sum_{k=1}^7 Y_{(ZERO)_k}$ (i.e., composition of the first seven objects).

mask of the first eight objects $\bigcup_{k=1}^8 M_k^{COMP}$ [Fig. 9(b)] and the mask of the ninth object M_9^{COMP} [Fig. 9(c)], the image $M_{(AND)_9}$ is obtained [Fig. 9(d)]. The utilization of such a mask leads to the image $Y_{(ZERO)_9}$ [Fig. 9(e)], which can be added to $\sum_{k=1}^8 Y_{(ZERO)_k}$ for obtaining the “object composition” $Y_{composition} = \sum_{k=1}^9 Y_{(ZERO)_k}$ [Fig. 9(f)].

Now, in order to combine the “object composition” with the background, it is necessary to compute the “mask composition” $M_{composition} = \bigcup_{k=1}^9 M_k^{COMP}$ [see Fig. 10(a)], which is required for obtaining the background Y_{BACKGR} [Fig. 10(b)]. Finally, by applying the ADDITION, the synthesized image Y_{SYNTH} is obtained [Fig. 10(c)].

Now, the algorithm for consistency observation has to be applied for computing the “filtered difference image.” At first, by applying the SUBTRACTION, the “difference image” $Y_{DIFF} = (I_7 - Y_{SYNTH})$ is determined [Fig. 11(a)]. Then, the application of the algorithm in Fig. 5 leads to “difference mask” M_{DIFF} [Fig. 11(b)]. By deleting from Y_{DIFF} those pixels that are not

indicated by M_{DIFF} , it is possible to obtain the “filtered difference image” $Y_{filtered\ DIFF}$ [Fig. 11(c)]. Finally, Fig. 12 shows the image obtained by adding the “filtered difference image” $Y_{filtered\ DIFF}$ to the synthesized image Y_{SYNTH} . Notice that in object-oriented video coding the receiver has to reconstruct Y_{SYNTH} and add it to $Y_{filtered\ DIFF}$, which is provided by the transmitter.

We would conclude this example by giving an estimation of the complete execution time of the proposed CNN algorithms on the CNUM. The results are summarized in Table I, where the estimated processing time has been given for each CNN algorithm without taking into account the I/O operations as well as the digital operations such as AND/OR. We would stress the τ dependence of the proposed approach, which will become more and more efficient, as a function of τ , in the future chip implementations.

Example 2: The aim of this example is to evaluate the performances of the proposed motion-compensation algorithm

Fig. 9. Object composition algorithm. (a) Image $\sum_{k=1}^8 Y_{(ZERO)_k}$ (i.e., composition of the first eight objects). (b) Image $\bigcup_{k=1}^8 M_k^{COMP}$ (i.e., mask of the first eight objects). (c) Image M_9^{COMP} (i.e., mask of the ninth object). (d) Image $M_{(AND)_9}$ (i.e., result of the AND operation). (e) Image $Y_{(ZERO)_9}$ (i.e., result of the ZERO operation). (f) Image $Y_{composition} = \sum_{k=1}^9 Y_{(ZERO)_k}$ (i.e., the “object composition”).

in terms of precision of the prediction. To this purpose, a comparison among different video sequences is carried out. At first, we apply the motion-compensation algorithm to the first 50 frames of the *Miss America* sequence, starting from frame 3 with a 15×15 -pixel window. Successively, we apply the motion-compensation algorithm to the first 50 frames of the *Claire* sequence, with a 19×19 -pixel window. We start by frame 3, where 7 objects are found [see Fig. 13(a)]. Notice that, since the motion in *Claire* is larger than the motion in *Missa*, a larger search-window has been chosen. Finally, we consider a worst-case benchmark, that is, the *Stefan* video sequence. We apply the algorithm to the first 10 frames by choosing a 31×31 -pixel window. We start by frame 3, where 12 objects are found [see Fig. 13(b)]. Notice that, since in the *Stefan* sequence camera motions are applied and several objects move in different way, it is necessary to consider search-window larger than 15×15 -pixel size. Regarding *Missa* and *Claire*, the results are reported in Table II, where for each sequence the energies of the difference image with motion compensation

and without motion compensation have been evaluated. Notice that the energy has been computed by considering that the pixel values of the “difference image” are within the range $[-1, 1]$. Moreover, the gains in decibels are reported in Fig. 14. The obtained results (that is, the improvements in dB) clearly highlight the performances of the proposed algorithm in estimating the scene transformation. Regarding *Stefan*, the results from frame 5 to frame 10 are summarized in Table III. By analyzing these results, it can be argued that the gain obtained for frame 5 is satisfying, whereas from frame 6 to frame 10, there is a performance degradation due to fast scene transformation induced by camera motion. We would point out that, in our opinion, this degradation does not depend on the behavior of the “motion-compensation” algorithm, but on the fact that the “area of the fans” in the *Stefan* sequence continuously changes due to camera motion. By taking into account the results in Table III, we can argue that for *Stefan* sequence the “segmentation and object labeling” stage has to be carried out approximately every three frames.

Fig. 10. Combination with the background. (a) Mask composition $M_{\text{composition}} = \bigcup_{k=1}^9 M_k^{\text{COMP}}$. (b) Background Y_{BACKGR} . (c) Synthesized image Y_{SYNTH} .

Fig. 11. Consistency observation algorithm. (a) "Difference image" $Y_{\text{DIFF}} = (I_7 - Y_{\text{SYNTH}})$. (b) "Difference mask" M_{DIFF} . (c) "Filtered difference image" $Y_{\text{filtered DIFF}}$.

VI. DISCUSSION

The aim of this section is to illustrate strengths and weaknesses of the proposed approach. In particular, the following issues are discussed.

- 1) In the context of object-oriented video coding, interesting block-matching and region-matching algorithms have

already been developed (for instance, some of them are currently used in standards such as MPEG4) [7]. These approaches have first been proposed for digital serial processor architectures. Successively, they have been improved through parallelization based on MMX [12]. On the other hand, the processor considered through the paper is a new analog dynamic processor array,

TABLE I
CNN-BASED IMAGE ANALYSIS: PROCESSING TIMES OF “SEGMENTATION,” “MOTION COMPENSATION,” “IMAGE SYNTHESIS,” AND “CONSISTENCY-OBSERVATION” ALGORITHMS

Segmentation	1118 τ
Motion Compensation	148895 τ
Image Synthesis	88 τ
Consistency Observation	104 τ

Fig. 12. The image obtained by adding $Y_{\text{filtered DIFF}}$ to Y_{SYNTH} .

(a)

(b)

Fig. 13. Video sequence comparisons. (a) Segmented frame 3 of *Claire*. (b) Segmented frame 3 of *Stefan*.

called CNN Universal chip [10]. Therefore we would point out that the algorithms proposed herein, designed for this new computational paradigm, represent a new implementation for CNN-based object-oriented video coding architectures.

- 2) Referring to the objectives to be achieved, the proposed algorithms could appear close to those used in digital processors. However, we would stress that the approach illustrated herein consists of *analogic* CNN algorithms. Notice that the key idea in CNUM processors is the concept of “*analog instruction*” [10]. It consists of CNN template sequence, which generates nonlinear spatio-temporal dynamics [10]. This idea is lacking in digital hardware processors. For this reason the algorithms proposed for digital processors cannot be used for CNN architectures. As a consequence, we feel that our paper presents “*new analogic CNN algorithms*” for motion compensation, image synthesis and consistency observation.
- 3) Advantages and drawbacks of standard block-matching algorithms and object-oriented motion-compensation algorithms are now discussed. Block-matching algorithms do not usually require complex image-analysis tasks [7]. At the same time, they are able to guarantee satisfying motion prediction. However, referring to scenes characterized by fast motion, block-matching algorithms usually require a large number of blocks to be compensated, if compared to the number of segmented objects [7]. This clearly represents a drawback for real-time hardware implementations of these algorithms. Furthermore, the presence of blocking artifacts can significantly decrease the quality of human visual perception. On the other hand, object-oriented motion-compensation algorithms usually require complex image-analysis tasks in order to detect object contour [7]. As shown through the paper, this drawback can be overcome by exploiting the great computational power offered by the CNUM. Additionally, object-oriented motion-compensation algorithms do not generate blocking artifacts and, consequently, do not affect the image quality.

Finally, referring to Fig. 14, it should be pointed out that those results do not underline if an object-based compensation improves the block-based approach used in the standard. We feel that this issue needs to be further investigated in the future.

- 4) Regarding the motion model, through the paper only a simple translation model has been considered. Nevertheless, this has led to a complex analogic CNN algorithm for motion compensation. However, it should be noted that when a region adapted to an object in the scene is used to estimate the spatio-temporal transformation between two frames, the simple translation model is less precise than more general affine motion models. This issue needs to

Fig. 14. *Missa* and *Claire* video sequences: gains in decibels.

TABLE II
MISSA AND CLAIRE VIDEO SEQUENCES: ENERGIES OF THE "DIFFERENCE IMAGE" WITH MOTION COMPENSATION, AND WITHOUT MOTION COMPENSATION (FRAMES FROM 5 TO 50)

MISSA

Frame #	Difference Image Mean Square pixel value with compensation (dB)	Difference Image Mean Square pixel value without compensation (dB)	Gain (dB)
5	-33.23	-28.54	4.74
10	-28.54	-26.58	1.96
15	-26.99	-25.53	1.46
20	-24.68	-22.68	2.01
25	-25.23	-23.19	2.04
30	-24.44	-22.22	2.22
35	-23.98	-22.22	1.76
40	-25.09	-23.77	1.32
45	-23.09	-21.87	1.23
50	-22.84	-20.51	2.33

CLAIRE

Frame #	Difference Image Mean Square pixel value with compensation (dB)	Difference Image Mean Square pixel value without compensation (dB)	Gain (dB)
5	-31.57	-27.45	4.12
10	-24.09	-21.49	2.60
15	-24.56	-22.22	2.34
20	-24.56	-22.22	2.34
25	-24.56	-22.01	2.55
30	-22.76	-20.51	2.25
35	-20.91	-19.28	1.63
40	-20.22	-18.83	1.39
45	-20.41	-18.83	1.58
50	-19.47	-17.57	1.89

be further investigated, with the aim of developing even more sophisticated analogic CNN algorithms for motion compensation.

VII. CONCLUSION

The CNN UM has proved to be a powerful tool for implementing image processing algorithms via elementary CNN in-

structions. In this paper, the attention has been focused on object-oriented image analysis for video-coding systems. In particular, new analogic CNN algorithms for motion compensation, image synthesis and consistency observation have been developed. Simulation results using different video sequences have been reported to show the effectiveness of the technique. It can be concluded that, along with the segmentation technique de-

TABLE III
STEFAN VIDEO SEQUENCE: ENERGIES OF THE "DIFFERENCE IMAGE" WITH MOTION COMPENSATION,
AND WITHOUT MOTION COMPENSATION (FRAMES FROM 5 TO 10)

STEFAN

Frame #	Difference Image Mean Square pixel value with compensation (dB)	Difference Image Mean Square pixel value without compensation (dB)	Gain (dB)
5	-14.22	-10.24	3.98
6	-10.62	-9.88	0.74
7	-10.12	-9.51	0.61
8	-9.81	-9.11	0.7
9	-9.63	-8.92	0.71
10	-9.59	-8.86	0.73

veloped in [1], the proposed approach make perceive the possibility of implementing the whole object-oriented image-analysis scheme on the CNN universal chip.

REFERENCES

- [1] A. Stoffels, T. Roska, and L. O. Chua, "On object-oriented video-coding using the CNN Universal Machine," *IEEE Trans. Circuits Syst. I*, vol. 43, pp. 948–952, Nov. 1996.
- [2] M. Kunt, A. Ikonopoulou, and M. Kocher, "Second generation image coding techniques," *Proc. IEEE*, vol. 73, pp. 549–575, Apr. 1985.
- [3] N. Diehl, "Object-oriented motion estimation and segmentation in image sequences," *Signal Processing: Image Communication*, vol. 3, pp. 23–56, 1991.
- [4] M. Hotter, "Optimization and efficiency of an object-oriented analysis-synthesis coder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 181–194, Apr. 1994.
- [5] J. Ostermann, "Object-oriented analysis-synthesis coding based on the source model of moving rigid 3D objects," *Signal Processing: Image Communication*, vol. 6, pp. 143–161, 1994.
- [6] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression: A survey," *Proc. IEEE*, vol. 83, pp. 220–246, Feb. 1995.
- [7] A. Stoffels, T. Roska, and L. O. Chua, "Object-oriented image analysis for very-low-bitrate video-coding systems using the CNN universal machine," *Int. J. Circuit Theory and Applications*, vol. 25, pp. 235–258, 1997.
- [8] T. Roska and L. O. Chua, "The CNN universal machine: An analogic array computer," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 163–173, Mar. 1993.
- [9] R. Dominguez-Castro, S. Espejo, A. Rodriguez-Vazquez, and R. Carmona, "A CNN universal chip in CMOS technology," in *Proc. Third IEEE Int. Workshop Cellular Neural Networks Their Applications (CNNA '94)*, Rome, Italy, 1994, pp. 91–96.
- [10] T. Roska, "Computer-sensors: Spatial-temporal computers for analog array signals, dynamically integrated with sensors," *J. VLSI Signal Processing*, vol. 23, no. 2/3, pp. 221–237, 1999.
- [11] A. Rodriguez-Vazquez, E. Roca, M. Delgado-Restituto, S. Espejo, and R. Dominguez-Castro, "Most-based design and scaling of synaptic interconnections in VLSI analog array processing CNN chips," *J. VLSI Signal Processing*, vol. 23, no. 2/3, pp. 239–266, 1999.
- [12] T. Sziranyi, K. Laszlo, L. Czuni, and F. Ziliani, "Object-oriented motion-segmentation for video-compression in the CNN-UM," *J. VLSI Signal Processing*, vol. 23, no. 2/3, pp. 479–496, 1999.
- [13] G. Linan, S. Espejo, R. Dominguez-Castro, and A. Rodriguez-Vazquez, "The CNNUC3: An analog I/O 64×64 CNN Universal Machine chip prototype with 7-bit analog accuracy," in *Proc. 6th IEEE Int. Work. on Cellular Neural Networks and Their Applications (CNNA 2000)*, 2000, pp. 201–206.
- [14] A. Zarandy, "ACE box: High-performance visual computer based on the ACE4k analogic array processor chip," in *Proc. Europ. Conf. Circuit Theory and Design (ECCTD 2001)*, vol. I, 2001, pp. 361–364.
- [15] G. Linan, R. Dominguez-Castro, S. Espejo, and A. Rodriguez-Vazquez, "ACE 16K: A programmable focal plane vision processor with 128×128 resolution," in *Proc. Europ. Conf. Circuit Theory Design (ECCTD 2001)*, vol. I, 2001, pp. 345–348.
- [16] AnaLogic Computers Ltd, "CSL-CNN software library, version 1.1," AnaLogic Computers Ltd, Budapest, Hungary, 2000.
- [17] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 147–156, Mar. 1993.
- [18] G. Linan, P. Foldes, A. Rodriguez-Vazquez, S. Espejo, and R. Dominguez-Castro, "Realization of nonlinear templates using the CNNUC3 prototype," in *Proc. 6th IEEE Int. Workshop Cellular Neural Networks Their Applications (CNNA 2000)*, 2000, pp. 219–224.
- [19] G. Linan, P. Foldes, A. Rodriguez-Vazquez, S. Espejo, and R. Dominguez-Castro, "Implementation of nonlinear templates using a decomposition technique by a $0.5 \mu\text{m}$ CMOS CNN Universal Chip," in *Proc. IEEE Int. Symp. on Circuits Systems (ISCAS 2000)*, vol. II, 2000, pp. 401–404.
- [20] L. Kek and A. Zarandy, "Implementation of large neighborhood nonlinear templates on the CNN universal machine," *Int. J. Circuit Theory Applicat.*, vol. 26, pp. 551–566, 1998.