STUDIES IN *FUZZINESS*
AND *SOFT COMPUTING*

Studies
in Fuzziness
and
Soft Computing

Yap-Peng Tan · Kim Hui Yap · Lipo Wang
Editors

Intelligent Multimedia Processing with Soft Computing

Springer

Contents

Video Compression by Neural Networks

Daniele Vigliano; Raffaele Parisi; Aurelio Uncini

INFOCOM Department, University of Rome “La Sapienza” – Rome, Italy

Abstract. In this chapter a general overview of most common approaches to video compression is first provided. Standardization issues are briefly discussed and most recent neural compression techniques reviewed. In addition, a particularly effective novel neural paradigm is introduced and described. The new approach is based on a proper quad-tree segmentation of video frames and is capable to yield a considerable improvement with respect to existing standards in high quality video compression. Experimental tests are described to demonstrate the efficacy of the proposed solution.

Keywords: neural networks, cellular networks, fuzzy systems, MPEG standards, recurrent neural networks.

1 Introduction

“A picture is worth a thousand words”. This popular saying well synthesizes the different weight between visual and textual or linguistic information in everyday’s life. As a matter of fact, visual information has reached a primary and undisputed role in modern Information and Communication Technology. In particular, the widespread diffusion of telecommunications and networking offers today new opportunities to the transmission and processing of multimedia data. Nevertheless, the transmission of highly informative video contents imposes strict requirements in terms of bandwidth occupancy. A trade-off between quality and compression is thus searched for.

Compression of video data aims at minimizing the number of bits required to represent each frame image in a video stream. Video compression has a huge number of applications in several fields, from telecommunications, to remote sensing, to medicine. Depending on the application, some distortion can be accepted in exchange for a higher compression ratio. This is the case of so-called lossy compression schemes. In other cases (e.g. biomedical applications), distortion is not allowed (lossless coding schemes).

Video compression techniques have been classified into four main classes: waveform, object-based, model-based and fractal coding techniques [45].

Waveform compression techniques use time as a third dimension. Into this class one can find all the applications working in the time domain (e.g. Discrete cosine transform, Wavelets and also Motion compensation techniques [58][53]). *Object-based techniques* consider video sequences as collections of different objects

[62], that can be differently processed. Objects are typically extracted by a segmentation step [44]. *Model-based* approaches perform the analysis of the video input and the synthesis of a structural 3D or 2D model [66]. *Fractal-based techniques* extend to video applications the approaches successfully applied to image coding. In this framework images are expressed as the attractor of a contractive function system and then retrieved by iterating the set of functions [73]. Correspondingly, several standards have been also developed.

In recent years there has been a tremendous growth of interest in the use of neural networks (NNs) for video coding. This interest is justified by the well-known capabilities of NNs of performing complex input-output nonlinear mappings, in a *learning from examples* fashion. As a matter of fact, appropriate use of NNs can considerably improve the performance of all the four compression techniques above described.

This chapter is organized as follows. Section “Review of recent standards” provides a short description of most recent standards in video compression. Section “Neural video compression: existing approaches” presents an overview of most popular neural approaches to video coding, while section “Quad-tree segmentation and neural compression” describes two innovative and particularly effective solutions.

2 Review of recent standards

Compression of image and video data has been the object of intensive research in the last twenty years. The diffusion of a large number of compression algorithms has led to the definition of several standards. In particular, two international organizations (ISO/IEC and ITU-T) have been involved in the standardization of images, audio and video data. A complete overview of recent standards and trends in visual information compression is out of the scope of this work and can be found in [45][51][52]. A brief summary is provided here for convenience of description.

The standards proposed for general purpose compression of still images are JPEG [46][47], based on a block discrete cosine transform (DCT) followed by Huffman or Arithmetic coding, and the more recent JPEG2000 [48]-[50], based on discrete wavelet transform and EBCOT coding.

Concerning video compression, ITU H.261 suggests the use of hybrid schemes in order to reduce spatial redundancy by DCT and temporal correlation by motion compensated prediction coding [53]. This approach was designed and optimized for videoconference transmission over an ISDN channel, for a bit rate down to 64 kbit/sec.

H.263 [56] and H.263+ [54] have the same core architecture of H.261 but introduced improvements principally in the precision of motion compensation and in prediction. These standards allow for the transmission of audio video information at a very low bit rate (9.6 kbit/sec).

Most recent advances in video coding aim at developing new improved standards by exploiting all the suitable features previously used in video compression. An example is H.26L [77][55].

The first studies of the Moving Picture Expert Group (MPEG) started in 1988. They aim at developing new standards for Audio-Video Coding. The main difference with respect to the other standards is that MPEGs are “open standards”, in the sense that they are not oriented to a particular application.

MPEG-1 was developed to operate at bit rates of up to about 1.5Mbit/sec for the consumer video coding and video content storing on media like CD ROM, DAT. It provides important features including frame-based random access of video, fast forward/fast reverse (FF/FR) searches through compressed bit streams, reverse playback of video and editability of the compressed bit stream. MPEG-1 performs the compression by using several algorithms, such as the subsampling of video information to match the human visual system (HVS), variable length coding, motion compensation and DCT to reduce the temporal and spatial redundancy [57]-[59].

MPEG-2 is similar to MPEG-1 but it includes some extensions to cover a wider range of applications (e.g. HDTV and multi-channel audio coding). It was designed to operate at a bit rate between 1.5 and 35 Mb/sec. One of the main enhancements of MPEG-2 with respect to MPEG-1 is the introduction of syntactical rules for efficient coding of interlaced video. The Advanced Audio Coding (AAC) is one of the formats defined in the non back-compatible version of MPEG-2. It was developed to specifically perform multichannel audio coding. MPEG-2 AAC is based on the MPEG-2 layer III, where some aspects were improved (frequency resolution, joint stereo coding, Huffman coding) and some others (like spectral and time prediction) were introduced. The resulting standard is able to perform the coding of five audio channels [60][61].

Object-oriented techniques extensively developed in computer science have been successfully applied to video compression, leading to MPEG-4. In this standard the video signal can be considered as composed by different objects, each one with its own shape, motion and texture representation. Objects are coded independently, in order to allow for direct access and manipulation. The power of this coding approach is that different objects can be coded by different algorithms, with different compression rates. This approach is justified by the fact that in a video sequence different parts of the scene may accept different distortion levels. The original video is divided into streams: audio and video streams are separated and each object has its own stream, e.g. the information about object placement, scaling and motion (Binary Format of Scene).

Furthermore, in MPEG-4 synthetic and natural sounds are coded in a different way. In fact the Synthetic Natural Hybrid Coding (SNHC) performs the composition of natural compressed audio and of synthetic sounds (artificial sounds are created in real time by the decoder). In addition, MPEG-4 proposes also the distinction between speech and “non speech” sounds, since the former can be compressed by specific *ad hoc* techniques [62]-[65].

In modern information and communication technology, a fundamental issue is to guarantee that the information content of a message can be easily accessed and handled by the user. MPEG-7 (also named “Multimedia Content Description Tool”) provides a rich set of tools performing the description of audio-video contents in a multimedia environment. The application areas that benefit from audio-video content description are multiple, from web search of multimedia contents to media broadcasting, from services in arts (e.g. in art galleries) to home entertainment, to database (of multimedia data) applications [67]-[70]. Descriptions provided by MPEG-7 are independent of the compression method and have to be meaningful just in the context of the considered application. For this reason different types of features perform different abstraction levels.

More specifically, the MPEG-7 standard consists of several parts. In this section Multimedia Description Schemes, the Visual description tool and the Audio description tool are detailed. Multimedia Description Schemes (DSs) are metadata structures used to describe audio-visual contents. It is defined by the Description Definition Language (DDL), based on XML. Resulting descriptions can be expressed in a text form (TeM) or in a binary compressed form (BiT). The former one allows for human reading and editing, the latter one improves the efficiency in storing and transmission. In this framework tools are developed to provide DSs with information about the content and the creation of the multimedia document and DSs to improve the browsing and the access to the audio-visual content. The Visual description tool performs the description of visual categories like colour, textures, motion, localization, shape and face recognition. The Audio description tool contains low level tools (e.g. spectral and temporal audio feature descriptions) and high-level specialized tools (like musical instrument timbre description, melody description, speech tools and those for the recognition and indexing of general sounds). The MPEG-7 standard provides also an application to represent the multimedia content description named “Terminal”. It is important to underline that the Terminal takes care of both the ingoing and the outgoing transmissions, also taking into account specific queries from the end user.

MPEG standards aim at processing multimedia contents in a physical and in a semantic context (MPEG-7), but they do not address other issues like multimedia consumption, diffusion, copyright, access or management rights. MPEG-21 was introduced with the explicit goal of overcoming this limitation, by providing new solutions to access, consumption, delivery, management and protection processes of different types of contents. MPEG-21 is essentially based on two concepts: Digital Item and Users. The Digital Item (DI) represents the fundamental unit of distribution and transaction (e.g. video collections, music albums); it is modelled by Digital Item Declaration (DID), which is a set of abstract terms and concepts. A User is every entity (e.g. humans, communities, society) interacting with the MPEG-21 environment or making use of Digital Items. Management of Digital Items is permitted only to a restricted set of Users [71][72].

3 Neural video compression: existing approaches

The purpose of this section is to provide a summary of most popular neural approaches to video compression. In recent years, in fact, NNs have been successfully applied to video compression, for example in intra-frame coding schemes, object clustering, motion estimation and object segmentation. The power of NNs as learning systems was also exploited to remove artifacts and in post processing.

An important issue in video compression is computational complexity, since more complex algorithms usually require more expensive hardware implementations. As a matter of fact, the parallel architecture of NNs allows to considerably reduce the computational cost with respect to more conventional approaches. This is one of the reasons of the success of neural video coding techniques.

The following sections will focus on some of the most representative neural approaches to video compression, namely those based on vector quantization, singularity maps and human vision, motion compensation and fuzzy segmentation.

3.1 Vector quantization

Vector quantization (VQ) is a very popular and efficient method for frame image (or still image) compression and it represents the natural extension of scalar quantization to *n*-dimensional spaces [17]-[19].

Figure 1 shows a conceptual scheme of a VQ coder. Input vectors are quantized to the closest codeword of the codebook, so the coder's output is the index of the selected codeword. Codebooks are generated from a set of training images by using clustering algorithms. For example in [20] this optimization problem is approached by a Kohonen neural network having the same number of neurons as the number of pixels in a block (self organizing feature maps, SOFM). The number of clusters (output neurons) is set to the desired number of codewords.

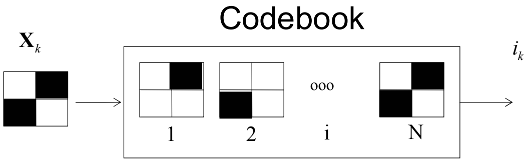

Fig. 1. Scheme of a VQ coder

Learning is based on evaluation of the minimum distance between outputs and inputs. The winner is the neuron having the smallest distance.

The advantages of using SOFM with respect to other clustering algorithms (k-means, LBG) include lower sensitivity to initialization, better performance in terms of rate distortion and faster convergence. In addition, during learning SOFM updates not only the winning class but also the neighboring one, since neurons unlikely to win are less frequently used.

For more details about general motivations justifying the use of SOFM in codebook design see [20]-[22]. Specific properties of SOFM can be used in performing more efficient codebook design, examples are APVQ (Adaptive Prediction VQ), FSVQ (Finite State VQ) and HVQ (Hierarchical VQ).

APVQ uses ordered codebooks where correlated inputs are quantized in adjacent codewords; an improvement in coding gain is obtained by encoding such codebook index with a DPCM (or some other neural predictor) [23].

FSVQ [24][27] introduces some form of memory in static VQ. It defines states by using previously encoded vectors. In each state the encoder selects a subset of codewords of the global codebook, the Side Match FSVQ [29], in which the current state of the coder is given by the closest side of the upper and left neighbouring vectors (i.e. the block of the frame image).

In order to reduce the computational cost, hierarchical structures can be also employed. In literature several techniques based on the cascade of multiple VQ encoders are described. Examples are two layer architectures or hierarchical structures [27] based on topological information [26].

Finally, in the VQ framework other neural approaches use a combination of different algorithms. As an example, [28] proposed neural principal component analysis (PCA) to generate the inputs to a SOFM.

3.2 Singularity maps and human vision

Emulation of the human vision system (HVS) inspired several solutions to video compression, yielding low compression ratios (about 1000:1) [30]-[32]. Due to its physiological nature, human eye does not focus on each single pixel of an image or a video stream but more on aspects like edges or intensity changes.

The retina in the human eye has two kinds of receptors: rods and cones. Rods are used for monochromatic light and cones for colours (RGB). Each receptor fires when it receives light, at the same time inhibiting nearby receptors. This behaviour is known as “lateral inhibition” and inspired some artificial neural architectures. For this reason the eye is able to detect edges better than smooth surfaces. Transmission through the optical nerve suffers from dispersion, so edges are smoothed and borders are broadened.

A Singularity Map (SM) [30]-[32] is obtained by labelling, with topological index and greyscale correspondence, the singular point of the border of the frame image. By this way the whole edge can be transmitted as a sequence instead of as an image. In practice, an SM collects all the multiresolution edges of a frame image. The extraction process requires a special care since ordinary edge extractors (like Sobel) typically broaden edges.

A typical HVS-based algorithm is composed by two main parallel steps:

- very low bit rate compression performed with a method that does not produce artifacts;
- singularity map (SM) computed from the original video, before the compression.

The second step corresponds to the application of singularity map on compressed frames. A block scheme of the proposed technique is shown in figure 2. Application of SM improves the performance with respect to more conventional video compression techniques (upper path in figure 2).

The algorithm performs two types of singularity maps: hard SM for daylight video sequences and soft SM for nightlight video sequences. In addition this approach takes into account the presence of noise in the original video sequence, because it is able to perform a more difficult estimation of singularity map.

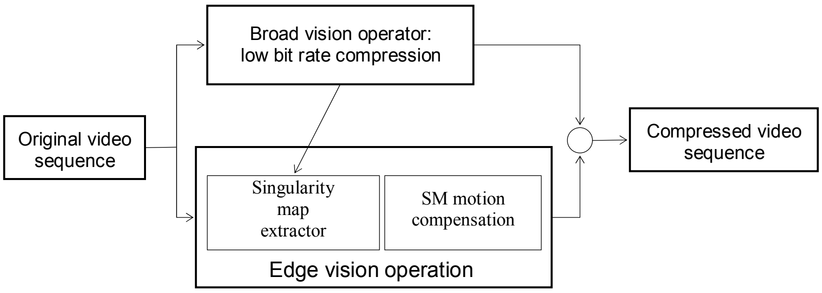

Fig. 2. Block scheme of HVS-based compressor

For hard SM iterative min-max was proposed, while soft SM can be performed by Cellular Neural Networks (CNNs), that can extract sharp edges in real time [31].

Once SM is computed, very low bit rate video compression is achieved by using Embedded Predictive Wavelet Image Coding (EPWIC [33]), Embedded Zerotrees of Wavelet coefficients (EZW [34]) or other wavelet-based compression techniques.

3.3 Motion compensation

Motion compensation (MC) is one of the most powerful techniques that can be used to reduce temporal correlation between adjacent frames. It is based on the assumption that in a large number of applications adjacent frames are usually highly correlated. Temporal correlation can be reduced by coding a block in a frame as a translated version of a block in a preceding frame. Of course the motion vector has to be transmitted too. In the following only translational motion will be considered.

Frames are typically segmented in macroblocks of 16x16 pixels, made of 4 blocks of 8x8 pixels (a reduced block representation error is obtained with finer segmentation but it produces a computational overhead). Figure 3 shows how in coding the block in frame *k*, the “best match block” of previous frame is computed and then the representation error is coded together with the information of the “motion vector”.

Several methods have been investigated in order to reduce the estimation error and to speed up the search for the best match. In particular, predictive methods perform the matching research only in the direction of previous frames, while bidirectional methods consider also future frames (bidirectional estimation).

In [35] a Hopfield neural algorithm is proposed to perform hierarchical motion estimation. It uses a classical best match method in order to reduce the number of possible macroblocks. Once obtained a subset of *D* candidates, a Hopfield network is used to obtain the best *vector of affinities* **v**. The optimal affinity vector **v** is the one minimizing the functional:

$$\frac{1}{2}\left\|\mathbf{f}-\mathbf{G}\mathbf{v}\right\|^2=\frac{1}{2}\sum_{p=1}^{L^2}\left(f_p-\sum_{i=1}^{D}g_{p,i}v_i\right) \qquad (1)$$

In (1) **f** is the vector of the current block to be estimated, **G** is a matrix whose columns are the *D* candidate blocks, **v** is the affinity vector (i.e. the one selecting the best match block) and L^2 denotes the size of the search windows. The architecture of the neural network performing the vector optimization is shown in figure 4.

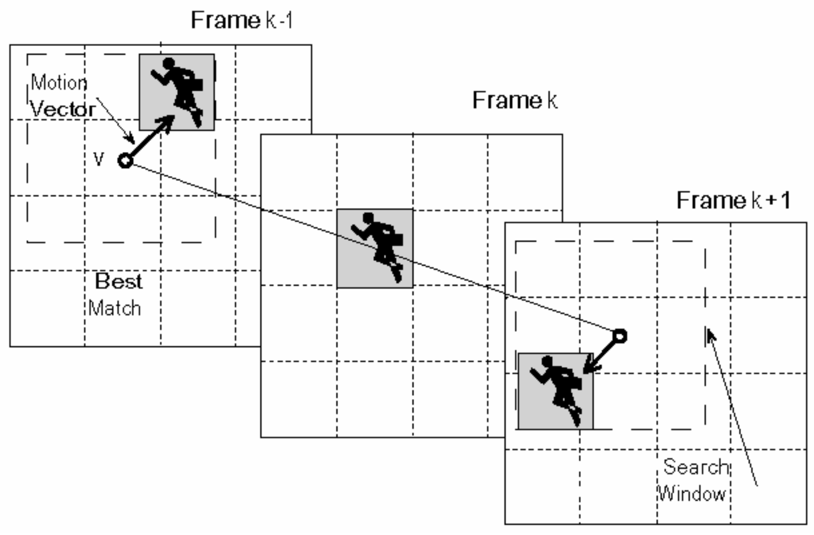

Fig. 3. Motion Compensation

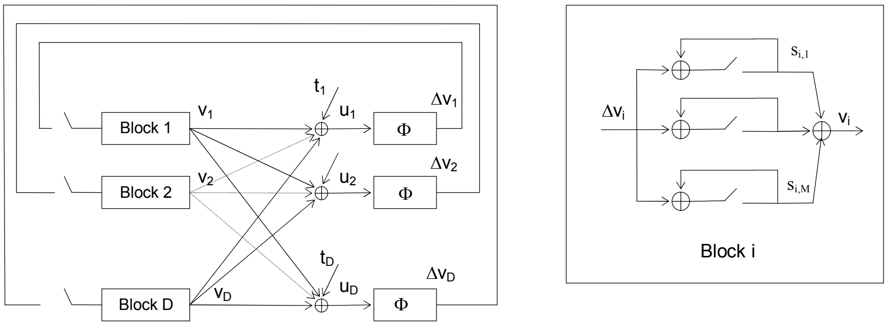

Fig. 4. Hopfield neural network for motion estimation

Other approaches to motion estimation include cellular neural networks (CNNs) [36]-[39][76]. These architectures can parallelize the computational flow required by both motion estimation and compensation, yielding faster and scalable computations. CNNs perform an optimization process based on their capacity to evolve toward a global minimum state.

Figure 5 shows the cell architecture of the network described in [36]. Cells are located in a $N \times M$ array; the generic cell C_{ij} has a state x_{ij}, a constant external input u_{ij} and an output y_{ij} and has r neighbor cells.

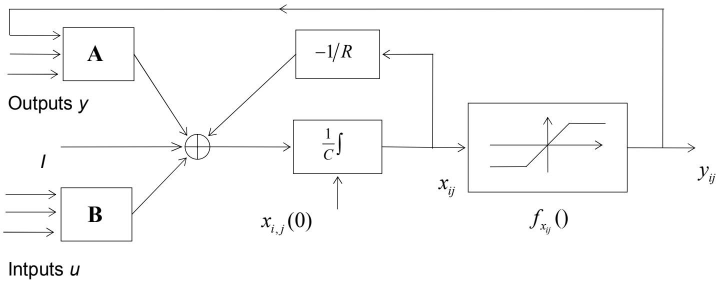

Fig. 5. Block diagram of cell C_{ij} of the Cellular Neural Network proposed in [36]

It is a graphical representation of the following difference equation:

$$C\dot{x}_{ij}(t) = -\frac{x_{ij}(t)}{R} + \sum_{k,l} A_{i,j;k,l} y_{kl}(t) + \sum_{k,l} B_{i,j;k,l} u_{kl}(t) + I \qquad (2)$$

Where C and R conform the integration time constant of the system, I is a constant scalar bias, **A** and **B** are $(2r+1)\times(2r+1)$ matrices; $A_{i,j;k,l}$ is the element k, l of the matrix **A** of the cell C_{ij}.

The dynamics of the CNN networks are described by a system of nonlinear ordinary differential equation (2) and by an energy function minimized during the computation process.

In [36] motion estimation is based on maximization of the *a-posteriori* probability (MAP) of the scene random field given the random motion field realization. It is possible to find similarity between MAP and CNN energy function.

For the scopes of this section it is enough to consider that MAP may be interpreted in terms of CNN architecture: feedforward input terms originates from matrix **B**, recurrent terms from the feedback matrix **A**. More details about the algorithm, stability and network design can be found in [36][37][40].

The capability of distributed computation, based on the parallel structure of CNNs, is exploited also in other contexts. For example, in [38][39] CNNs perform fast and distributed operation on frame images. The following mathematical formulation is used:

$$\begin{aligned}\dot{x}_{ij}(t) = x_{ij}(t) + \sum_{k,l} A_{i,j;k,l} y_{kl} + \sum_{k,l} B_{i,j;k,l} u_{kl} + \\ + \sum_{k,l} \hat{A}_{i,j;k,l}(y_{kl}) + \sum_{k,l} \hat{B}_{i,j;k,l}(u_{kl}) + I_{ij}\end{aligned} \qquad (3)$$

The cell architecture is similar to the one described in figure 5 adding nonlinear feedforward and nonlinear feedback blocks represented respectively by $\hat{B}_{i,j;k,l}(y_{kl})$

and $\hat{A}_{i,j;k,l}\left(y_{kl}\right)$. Motion compensation aims at determining what objects inside frame I_{n+k} are present also in frame I_n. Considering frame $n+k$, object positions in previous frame n are estimated by moving each object of frame n in a $p \times q$ -pixel window and comparing the result with the frame I_{n+k}.

Motion research is performed by following a "spiral" trajectory. All the processing is performed by the CNN, whose parameters (such as **A**, **B**, $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$, **x**, **I**, **u**, **y**) are preliminarily set to proper values in order to obtain the desired effect.

3.4 Neuro fuzzy segmentation of human image sequences

In order to achieve better compression ratios, modern video coding techniques apply different schemes to different objects in the same video stream (*object-based compression*). The advantages of using different compressions techniques for different objects are strictly tied to the capability of identifying and extracting the objects from the background. Classical tools for the generation of region-based representations are discussed in [44], where the state of art of this class of approaches is also described.

In [41] spatial and temporal information are combined to perform a neuro-fuzzy video segmentation of a videoconference video stream (one human speaker and background). The approach consists of three main steps:

- clustering
- detection
- refinement

In the first step a fuzzy self-clustering algorithm is used to group into fuzzy clusters similar pixels in the base frame of the video stream. Each frame image is divided into 4x4 pixel blocks, that are grouped in segments by the clustering algorithm. Segments are then combined together in order to form larger clusters. Each cluster is represented by Gaussian membership functions (one for the luminance and one for each chrominance), with a given mean value and variance.

After fuzzy clustering is completed, the detection step starts. In this step human face and body (i.e. "human objects") are detected and extracted from the background. Face segments are easily identified since they are characterized by chrominance values within a restricted range and luminance values having consistent variations. Once the area containing the face has been identified, the rest of body is assumed to lay in the area below the face.

On the basis of such analysis, clusters can be divided into foreground, background and ambiguous regions. A fuzzy neural network is employed to identify the ambiguous regions. The architecture of the network is shown in figure 6. Its operations are explained as follows.

Each pixel of each cluster yields three inputs x_1, x_2, x_3, that are the values of luminance and chrominances. The output of the network will be 1 if the cluster (or the pixel) is completely contained in the human object and 0 otherwise. The network layers are designed in the following way:

- *Layer 1*. The *input layer* contains the three inputs, that are directly transmitted to the next layer.
- *Layer 2*. The *fuzzification layer* contains *N* groups of three neurons each, being *N* the number of fuzzy clusters. The output is computed as a Gaussian function: $o_{ij}^{(2)} = \exp\left[-\left(\frac{o_i^{(1)} - m_{ij}}{\sigma_{ij}}\right)^2\right]$ where m_{ij}, and σ_{ij} are proper learning parameters.
- *Layer 3*. The *inference layer* contains *N* neurons. The output of each neuron is $o_j^{(3)} = \prod_{i=1}^{3} o_{ij}^{(2)}$.
- *Layer 4*. The *output layer* contains only one neuron, that performs the centroid *defuzzification*. Its output is: $o^{(4)} = \frac{\sum_{j=1}^{N} c_j o_j^{(3)}}{\sum_{j=1}^{N} o_j^{(3)}}$.

Parameters (m_{ij}, σ_{ij}, c_i) are trained from foreground and background blocks. The training algorithm is a combination of an SVD-based least squares estimator and gradient-based optimization (hybrid learning).

Other approaches to fuzzy neural segmentation are based on fuzzy clustering of more complex data structures. Data include both intra-frame information such as colour, shape, texture and contour, and inter-frame information, such as motion and object temporal shape.

In [42] good segmentation results are obtained by a two-step decomposition. The first step splits the image in subsets, by use of an unsupervised neural network. The frame image is then divided into clusters. The hierarchical clustering phase reduces the complexity of the object structure. Finally a PCA-based processing performs the refinement step, providing the final foreground-background segmentation.

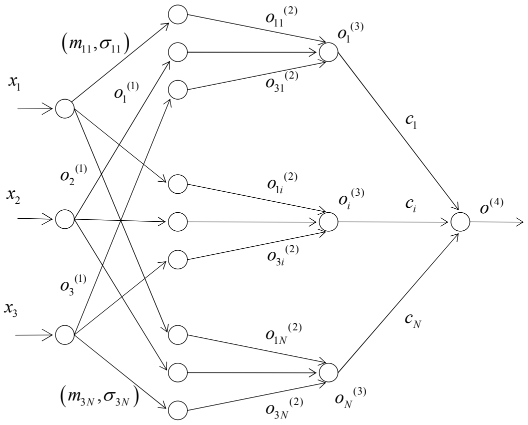

Fig. 6. Architecture of the fuzzy neural network for human object refinement

Other approaches are based on a subspace representation of the video sequence [43]. In this case video sequences are described by the minimum set of maximally distant frames that are able to describe the video sequence (*key frames*), selected on the basis of their semantic content. These frames are collected in a codebook. The core of the coding system is the video key frames codebook (VKC) definition, which is based on video analysis in the *vector space*. This definition is performed by an unsupervised neural network, through a *storyboarding* of the recorded sequence. Image feature vectors are used to represent images into the vector space. Clustering of all images in the feature vector space is employed to select the smaller set of video key frames for VKC definition.

4 Quad-tree segmentation and neural compression

The following sections describe in detail two *waveform* video compression algorithms, based on the use of feedforward and locally recurrent neural networks.

Techniques described so far were based on generalization of methods used for the compression of still images [75]. In particular, *transform coding* techniques achieve the desired compression by introducing proper transformations of images [51]. More specifically, given the set of coefficients representing a portion of an image or a video frame, transform coding produces a reduced set of coefficients such that reconstruction of the original image produces the minimum possible distortion. This reduction is possible since most of the initial block energy is concentrated in a reduced number of coefficients.

The optimal transform coder, in the sense of the minimum mean square error, is the one minimizing the mean-square distortion of the reconstructed data for a fixed quantization. In particular, the well-known Karhunen-Loève transform fulfils this constraint.

In the framework of video compression, techniques used for still images can be applied jointly with a temporal decomposition, thus calling for proper space-time processing. The following sections describe an effective video preprocessing technique and a feasible and particularly attractive solution to the design of neural transform coders.

4.1 Video preprocessing

Still images usually contain uniformly coloured areas, with poor informative content, and highly detailed areas. Different compression schemes can be adopted on areas with different activity levels, thus providing a better quality on detailed areas, and higher compression ratios on more uniform areas.

Frame images are decomposed in blocks that are individually processed. In particular, higher activity blocks can be extracted on the basis of their orientation: horizontal, vertical, diagonal. Blocks are then divided into subclasses and coded with different coders, in order to improve the performance of the compression [14]-[16].

Several papers described this kind of approach. Very good performance were obtained in [15], where blocks were grouped according to nine possible orientations: two horizontal (one darker on the left, one darker on the right), two vertical, four diagonal and the last shaded.

Figure 7 shows a picture splitted in blocks of different size by means of a *quad-tree* approach, based on a measure of the pixels variance: the bigger is the dimension of the block, the lower is the content detail, and viceversa.

Blocks having the same mask size carry about the same information and are processed by the same neural network, requiring specific training sets.

In video sequences, areas can be segmented also on the basis of changes in images, thus identifying sub-sequences where limited action takes place. Useful video representation can be obtained by identifying adjacent frames with reduced dissimilarities (group of frames, GOF). Each GOF collects frames having the same Depth of Activity (*DA*), by comparison of a pre-set threshold *th* with the variance between pixels of several adjacent frames. These frames have the same quad-tree segmentation structure.

The choice of the proper threshold is a critical issue in determining the *DA*. Higher values of the threshold yield lower quality of the reconstructed video, since frames are not represented by their own quad-tree structure. On the other hand, too low values of the threshold yield better quality but higher bit rate.

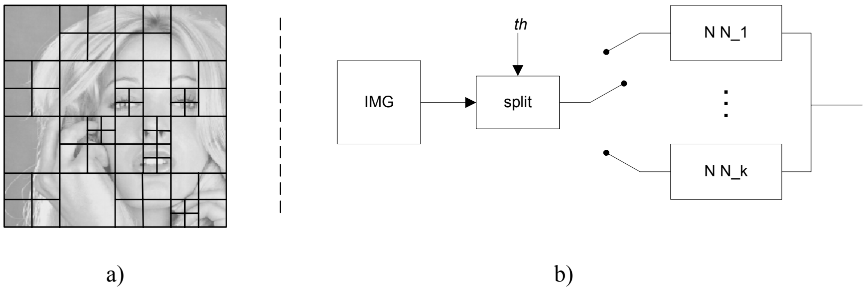

Fig. 7. a) Quad-tree segmentation; b) Adaptive size mask splitting block

The GOF generation algorithm consists of the following steps (figure 8):

1. the first frame (keyframe) is selected as a reference image of the *i*-th GOF;
2. a subsequent frame *n* belongs to the *i*-th GOF if the variance of the image difference between frame *n* and the keyframe is below *th*. The number of frames for which this condition is verified gives the *DA*, i.e. the length of the identified sub-sequence;
3. the final extracted sub-sequence consists of the keyframe *I* and the frames obtained by subtracting each subsequent frame to the keyframe (*D*-frames).

Images contained in every GOF are coded by a set of properly trained neural networks. The keyframe *I* and the last frame of the GOF *D1* will be coded with a fitted *quad-tree* structure, as shown in figure 9. For each sub-block of the keyframe *I* and of the frame *D1* in addition to the compressed data it is necessary to code also the quad tree segmentation, the network used for coding the sub-block, the sub-block mean value, the quantization and finally the number of frames internal to the GOF.

Sub blocks of *D2* (the residual frames of the GOF) only require information about the compressed data, since they have the same segmentation of *D1*.

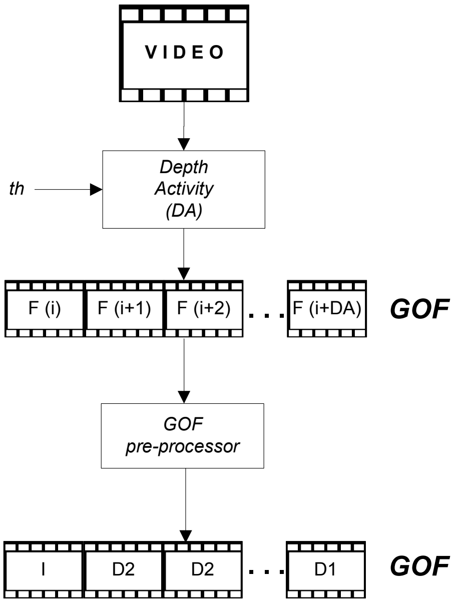

Fig. 8. Group of frames generation

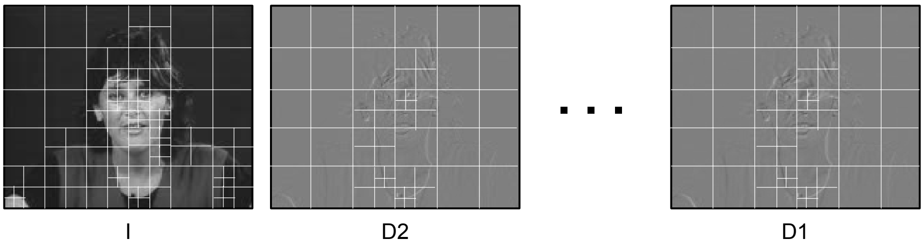

Fig. 9. Quad-tree schemes applied to frames within the GOF

The advantage of using the *D2* frames lies in the fact that frames near to *D1* have the same quad-tree segmentation structure (figure 9). In addition, these images are mostly made of uniform areas, so that the mask applied will be principally constituted by large blocks (e.g. 16×16), thus reducing the bit-rate. Figure 10 presents an overall scheme of the proposed approach.

The video preprocessor, given the original video stream, establishes the value of the *DA*. The GOF preprocessor computes the differences between frames, while the controller selects the keyframe and frames *D1* and *D2*, to be segmented in different ways.

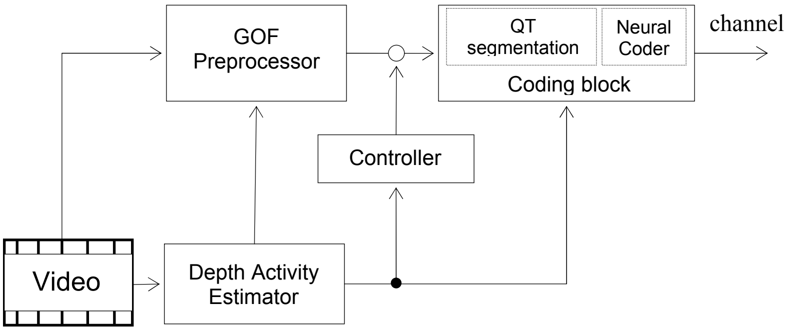

Fig. 10. Scheme of the proposed neural quad-tree video coding

4.2 Feed-forward neural compressor

Once segmentation has been performed, the next step is compression of each image block. In the transform coding framework, the Karhunen-Loève transform is commonly exploited to represent signals on the basis of their *principal components*.

In particular, it is possible to use a reduced set of principal components (reduced rank approximation), then obtaining a reconstruction error which depends on the variance of the eigenvalues of discarded eigenvectors. In more detail, given an *N*-dimensional vector signal **x**, the Karhunen-Loève transform represents it by using a basis **W** formed by the eigenvectors of its autocovariance matrix:

$$\mathbf{y} = \mathbf{W}\mathbf{x} \tag{4}$$

In this case no compression is performed [21]. A reduced rank (i.e. compressed) approximation of **y** is obtained by using the *M* eigenvectors corresponding to the *M* larger eigenvalues:

$$\hat{\mathbf{y}}=\hat{\mathbf{W}}\mathbf{x} = \sum_{i=1}^{M<N} w_i x \tag{5}$$

The representation error is bounded by the sum of the squared eigenvalues corresponding to the discarded eigenvectors [1]. It can be shown that the output vector coefficients are uncorrelated and therefore the redundancy due to correlation between neighbouring pixels is removed. Unfortunately application of KLT to video compression is not fully effective since it exploits second order statistics only.

The calculation of the estimate of the covariance of an image may be unwieldy and may require a large amount of memory; moreover the eigendecomposition implies a high additional computational cost due to the often large image size. These issues are important since KLT basis must be updated continuously during the video sequence.

A possible alternative to KLT is the discrete cosine transform (performed via FFT), which yields performance similar to KLT [51][45].

Another possible alternative to avoid these problems is the use of iterative neural techniques. Neural approaches require a reduced storage overhead giving a faster and computationally more convenient solution to the compression problem. Moreover neural networks are able to adapt over long term variations in the frame image statistics [3]. In the following, linear and nonlinear PCA are described.

Linear PCA: Hebbian learning

Linear PCA is an efficient solution to eigendecomposition computation. In [2] a mechanism inspired to neurobiology was proposed, where synaptic connections between neurons are modified by learning. Hebb's assumption consists in reinforcing the synaptic connection between two neurons if they are both active at the same time. Figure 11 shows the architecture of the artificial neuron used to perform the principal component extraction by Hebbian learning .

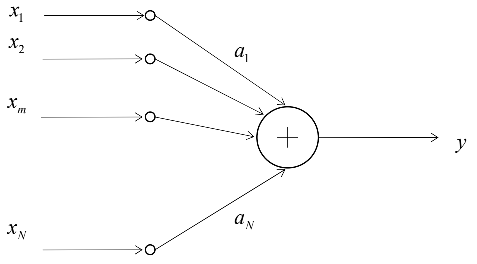

Fig. 11. Hebbian Neuron

The neuron's output is:

$$y = \mathbf{ax} \tag{6}$$

The Hebbian learning rule is given by the following recursive equation:

$$\mathbf{a}[n+1] = \frac{\mathbf{a}[n] + \mu \mathbf{a}[n]\mathbf{x}[n]}{\left\| \mathbf{a}[n] + \mu \mathbf{a}[n]\mathbf{x}[n] \right\|} \tag{7}$$

where μ is the learning rate and $\|\cdot\|$ is the Euclidean norm. Eq.(7) has been shown to converge to the first principal component.

Hebbian learning can be generalized to find the first M principal components. More specifically, the second principal component can be obtained by removing the first principal component from original data and performing PCA on updated data, and so on. The generalized Hebbian Algorithm includes also orthogonalization:

$$\mathbf{A}[n+1]=\mathbf{A}[n]+\mu\left[\mathbf{y}\mathbf{x}^T-LT\left[\mathbf{y}\mathbf{y}^T\right]\mathbf{A}[n]\right] \tag{8}$$

In (8) *LT* (i.e. lower triangular) is the matrix operator that sets to zero all the elements above the matrix diagonal. After convergence, matrix **A** contains the first *M* principal directions. An alternative is the APEX (*Adaptive Principal Component Extraction*) network, where hebbian synapses are used together with anti-hebbian ones.

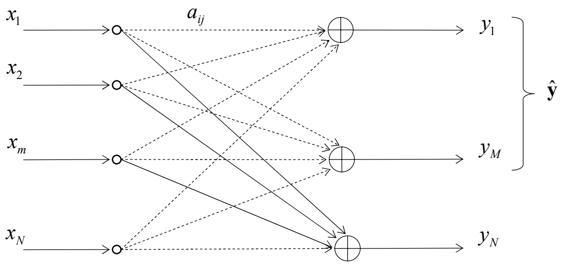

Fig. 12. Linear network for principal component extraction

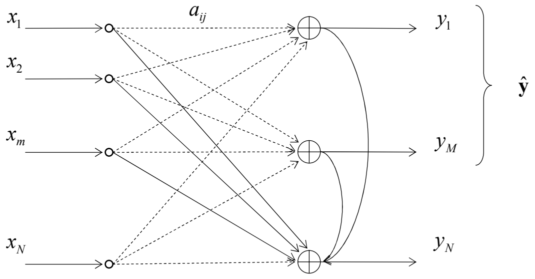

Fig. 13. The APEX network

Also this architecture has a biological justification. The *m*-th principal component can be computed on the basis of the previous *m-1*. More details can be found in [74].

Nonlinear PCA: Multilayer Perceptron

In 1988, Cottrel, Murno and Zipper applied a two-layer perceptron to the PCA problem [5]. The net was trained with the so called *autoassociative backpropagation*. This work opened the way to a large number of future developments.

Figure 14 shows the proposed architecture. In a first formulation a *linear neuron* was used. Its output is:

$$\hat{y}_i = \mathbf{a}^T{}_i\mathbf{x} \tag{9}$$

In matrix formulation:

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{A}^T\mathbf{x} \\ \hat{\mathbf{x}} &= \mathbf{A}\hat{\mathbf{y}} = \mathbf{A}\mathbf{A}^T\mathbf{x} \end{aligned} \tag{10}$$

Linear neural networks can achieve the same compression ratio as KLT without necessarily obtain the same weight matrices of the PCA transform: according to (10) given the optimum PCA solution **A**=**W**, different optimal solutions can be obtained by **A=WQ**$^{\mathrm{T}}$, being **Q** an orthogonal matrix.

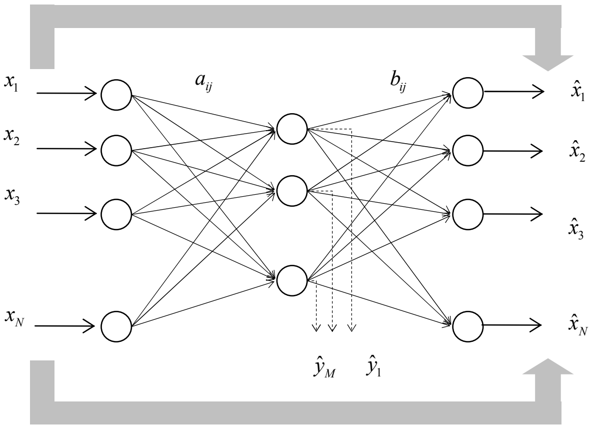

Fig. 14. Multilayer perceptron trained by autoassociative backpropagation

Other approaches developed neural networks with sigmoidal activation functions, yielding better results with respect to the linear network [3][4].

A critical issue in neural PCA is the fixed compression ratio of each processed block: the network performs the compression with a low distortion on uniform blocks but produces higher distortion on less uniform ones.

In order to overcome this problem, size-adaptive networks [6] can be employed to perform compression depending on block activity. This allows for higher compression of blocks with low activity level and good reconstruction of blocks with higher activity level.

As already described, the quad-tree algorithm segments images into several blocks of different size, on the basis of the activity level. An example of segmentation is shown in figure 15, where blocks of size 4x4, 8x8 and 16x16 are used.

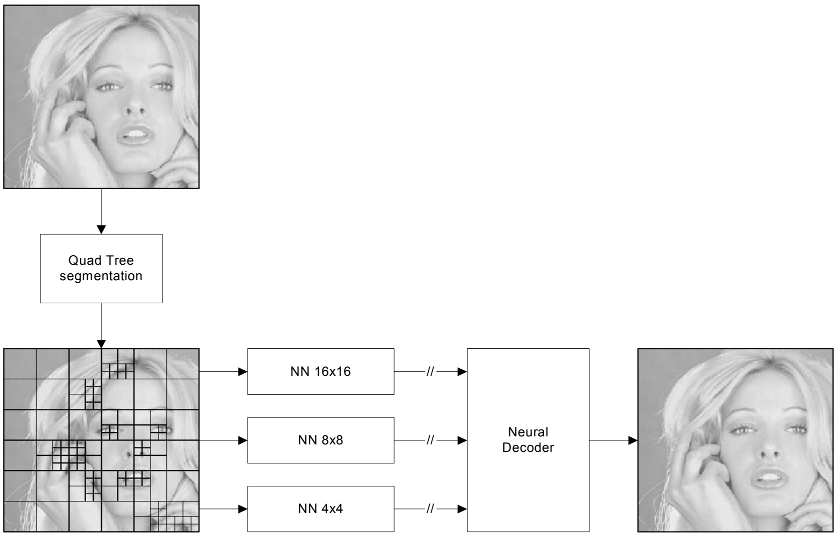

Fig. 15. Adaptive size mask compression of visual information

Three neural architectures were developed. They all have eight hidden neurons, while the number of inputs is equal to the number of pixels in a block. The output of each neuron is quantized with 4 bits.

Learning capabilities were improved by use of adaptable sigmoidal functions. In alternative, spline adaptive models were fruitfully employed [8].

Performance in video compression are usually evaluated on the basis of the Peak Signal to Noise Ratio (*PSNR*), defined as:

$$PSNR = 10 \cdot \log_{10}\left(\frac{256^2}{\frac{1}{M \times N} \cdot \sum_{m=1}^{M}\sum_{n=1}^{N}\left[pix_{org}(m,n) - pix_{comp}(m,n) \right]^2} \right) \tag{11}$$

where $pix_{org}(m,n)$ is the pixel of the current frame, and $pix_{comp}(m,n)$ is the compressed one and M and N are the frame dimension.

Figure 16 shows the PSNR values obtained on the Missa.avi benchmark file, by processing GOFs with different thresholds.

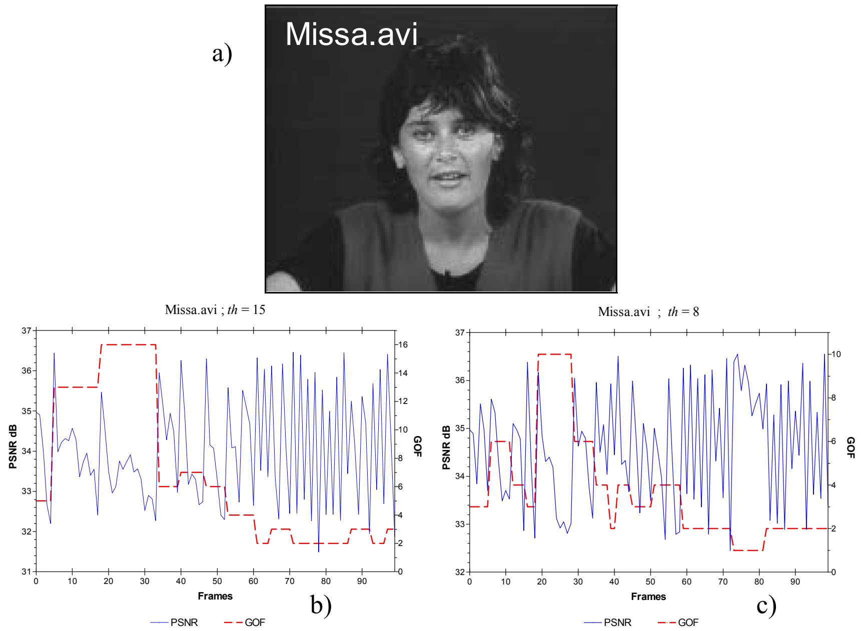

Fig. 16. a) Missa.avi movie segmented and compressed; b-c) PSNR and GOF evolution with two different thresholds

Table 1 shows different PSNR and bit rate values for different thresholds, for the Missa and Susi benchmark movies. It is easy to see that higher thresholds produce a gain in compression but decrease the quality of video reconstruction.

Table 1. Peak Signal to Noise Ratios (PSNR) and bit rate (br) for different thresholds in Missa and Susi videos

	th = 8		th = 15		th = 30	
	PSNR (dB)	br (kbps)	PSNR (dB)	br (kbps)	PSNR (dB)	br (kbps)
Missa	34,62	205,63	34,02	166,05	33,02	152,85
Susi	31,11	469,53	30,91	422,31	30,38	361,52

Hierarchical neural networks

As described, multilayer neural nets offer an attractive solution to video compression. Their success is due to several advantages, like short time encoding-decoding and no explicit use of codebooks. Nevertheless only information carried by contiguous pixels within the same segmented block is exploited. Better performance can be obtained by considering information on contiguous blocks.

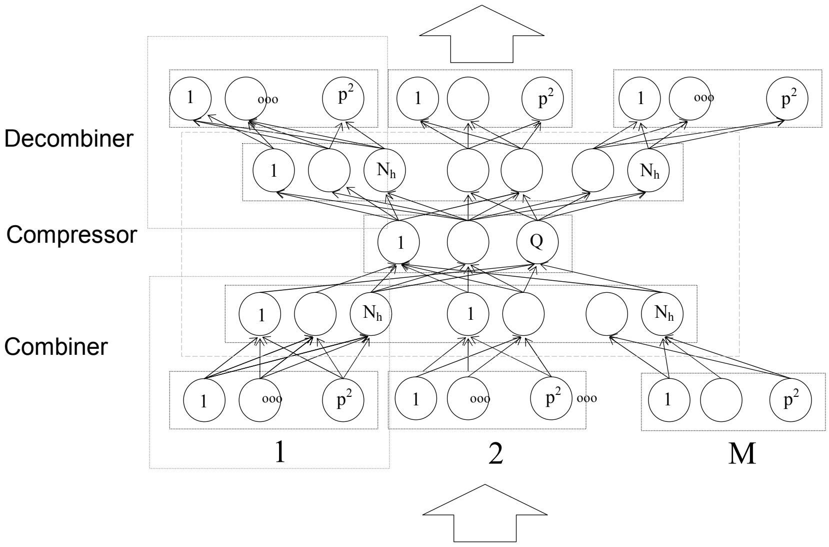

Fig. 17. Multilayer neural network for high order data compression-decompression

Hierarchical neural networks (HNNs) take into account the information about block contiguity [7]. The idea is to divide a scene into *N* disjoint sub-scenes, each one segmented in $n \times n$ pixels blocks. Blocks are processed together by the hierarchical structure shown in figure 17.

The HNN consists of input, hidden and output layers and it is not fully connected. The input and the output layers are single layers, composed by *N* input blocks (one for each section of the image), where each block has n^2 neurons. The hidden-layer section consists instead of three layers: combiner, compressor and decombiner layer. The connections between the input and combiner layers and between the decombiner and the output layers are not full.

Although learning in HNN could be performed by the classical back propagation algorithm, the so called *nested training algorithm* (NTA) provides better performances. NTA is a three phase training, one for each part of the architecture:

- *OLNN* (outer loop neural network). It performs the training of each fully connected network obtained by the corresponding sub blocks of input, combiner and output layer. Standard back propagation is applied. The target output is equal to the input. The training set is given by segmented blocks.
- *ILNN* (inner loop neural network). It performs the training of the hidden fully connected layers: combiner, compressor and decombiner.
- Once the *OLNN* and the *ILNN* have been separately trained, their weights are used to construct the overall network.

It is important to note that this hierarchical structure performs inter-block decorrelation in order to achieve a better compression level. About the same performance in terms of image quality and compression level were reached by use of adaptive spline activation functions, yielding a simpler structure.

4.3 Recurrent neural compressor

Multilayer neural networks can be properly adapted by introducing topological recurrency, in order to take into account the temporal dependence of video sequences. This allows either to improve the quality of the reconstructed video, for a fixed bit-rate, or to further reduce the compression level [78].

Dynamic behavior in multilayer perceptrons can be obtained by two different approaches:

- *Local approach*: a dynamical (e.g. ARMA) model of the neuron is employed.
- *Non local Approach*: external feedback is introduced.

In both cases the dynamical model is such that the input at time *n:* $x[n]$ may influence the output at time *n-h:* $y[n-h]$. In the case of asymptotic stability, the derivative $\partial y[n-h]/\partial x[n]$ goes to zero when h goes to infinity. The value of h for which the derivative becomes negligible is called *temporal depth*, whereas the number of adaptable parameters divided by the temporal depth is named *temporal resolution*.

An example of architecture used in this context is the IIR-MLP proposed in [10][11], where static synapses are replaced by conventional IIR adaptive filters, as depicted in figure 18.

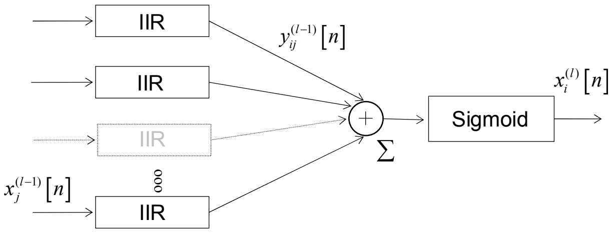

Fig. 18. Locally recurrent neuron for multilayer neural networks

Several learning algorithms for recurrent architectures exist in literature, although a comprehensive framework is still missing. In [9] a very effective algorithm was introduced for learning of locally recurrent neural networks. Learning is performed by a new gradient-based on-line algorithm [9], called causal recursive back-propagation (CRBP). It yields some advantages with respect to known online training methods and the well known recursive back propagation. CRBP includes backpropagation as a particular case [12][13]. This approach is based on the introduction of an ARMA model of synapses (figure 19). The forward phase at time n is described by the following equations, evaluated for layers $l=1,...,M$ and neurons $m=1,..,N_l$:

$$y_{km}^{(l)}[n]=\sum_{p=0}^{L_{km}^{(l)}-1} w_{km(p)}^{(l)} x_m^{(l-1)}[n-p]+\sum_{p=1}^{I_{km}^{(l)}} v_{km(p)}^{(l)} y_{km}^{(l)}[n-p] \qquad (12)$$

$$x_k^{(l)}[n]=sgm\left(\sum_{m=0}^{N_{l-1}} y_{km}^{(l)}[n]\right) \qquad (13)$$

where *sgm(.)* is the sigmoidal function.
If $\Phi^{(l)}[n]$ is the set of weights of layer *l* at time *n*, the updating rule is:

$$\Phi^{(l)}[n+1]=\Phi^{(l)}[n]+\Delta\Phi^{(l)}[n+1-D_l] \qquad (14)$$

where:

$$D_l=\begin{cases} 0 & \text{if } l=M \\ \sum_{i=l+1}^{M} \max_{n,m}\left(L_{nm}^{(i)}-1\right) & \text{if } 1\le l\le M \end{cases} \qquad (15)$$

and $\left(L_{nm}^{(i)}-1\right)$ is the order of the moving average part of the synapse of the *n*-th neuron of the *l*-th layer, relative to the *m*-th output of the (*l*-1)-th layer.

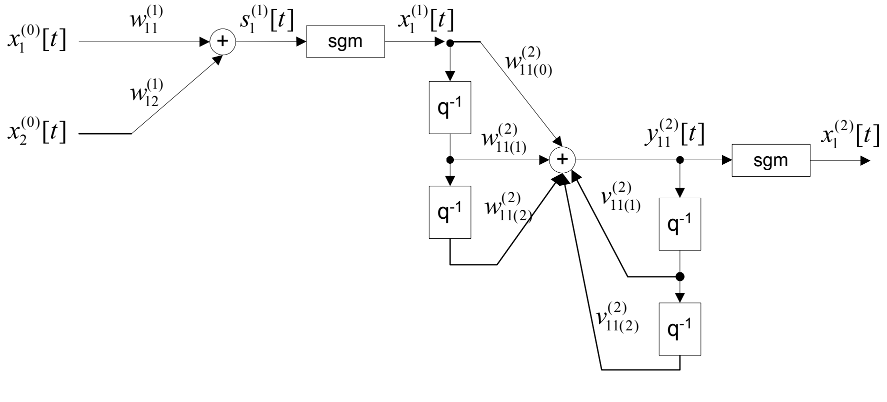

Fig. 19. Locally recurrent ARMA model for multilayer perceptrons

Referring to symbols in figure 19, the CRBP learning rules are:

$$\Delta\Phi_{km(p)}^{(l)}[n+1]=\mu e_k^{(l)}[n]s\dot{g}m\left[s_k^{(l)}[n]\right]\frac{\partial s_k^{(l)}[n]}{\partial\Phi_{km(p)}^{(l)}} \qquad (16)$$

$$e_k^{(l)}[n] = \begin{cases} e_k[n] & l = M \\ \sum_{q=1}^{N_{l+1}} \sum_{p=0}^{Q_{l+1}} \delta_q^{(l+1)}[n+p] \dfrac{\partial y_{qk}^{(l+1)}[n+p]}{\partial x_k^{(l)}[n]} & l = (M-1),\dots,1 \end{cases} \qquad (17)$$

The CRBP algorithm is computationally simple and can be fruitfully applied to the video compression problem. In particular, the proposed architecture was applied as neural coder in the coding block of figure 10.

Learning of locally recurrent neural networks for video compression is a critical issue since recurrent networks are typically sensitive to factors like choice of the proper training set, video length, or order by which the examples are presented. An inappropriate choice of these factors might compromise the correct learning of the network, typically producing artifacts in the reconstructed video. Most common artifacts are the so called "*regularities*" and "*memory effect*". An example of "*regularities*" is shown in figure 20. They can be avoided by reducing the length of the video training set.

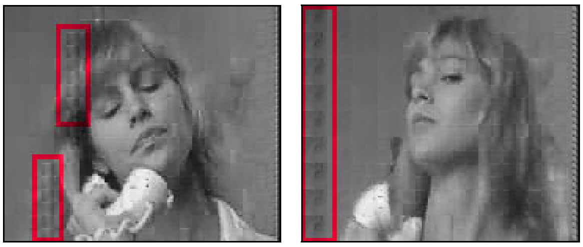

Fig. 20. Regularity effects in two frames of a video sequence

The "*memory effect*" is due to the delay lines in the synapse. It is typically detected by the presence in the reconstructed video of objects that are no longer present in the scene, especially on uniform color backgrounds (figure 21). This artifact can be avoided by carefully dimensioning the neuron dynamics and the number of taps of the ARMA filter.

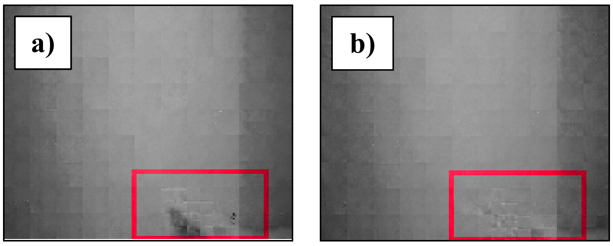

Fig. 21. Memory effect in two frames of a video sequence

Regularities and memory effects can be actually reduced if locally recurrent neurons only in the second layer of the structure of figure 14 are used.

It has been observed that most of artifacts are actually in the “background” of the scene. As a matter of fact, recurrent neural networks perform quite well on dynamical parts while they are not always effective on static background sections.

In order to overcome this limitation, an hybrid approach could be used after the scene segmentation. The idea is to use static neural networks on more static subscenes (the ones with the lowest activity), and to employ recurrent neural networks to code blocks with higher levels of detail. This solution requires a different processing for lower and higher activity blocks, in terms of network size, architecture and learning. Table 2 shows the performance typically obtained by a hybrid approach, where IIR synapses are used.

Table 2. Average bit rate and peak signal to noise ratio obtained with three different neural architectures

Reconstructed video	Susi_02	Susi_03	Susi_04
No. of hidden neurons	6	5	4
br (kbs)	433,38	372,51	319,32
PSNR (dB)	28,92	28,45	28,01

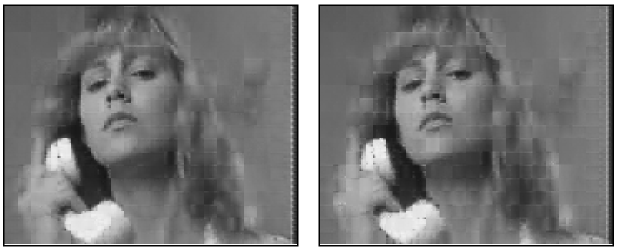

Fig. 22. Frames of the Suzi video compressed and recovered. Left: Suzi_02 (no block effect), right: Suzi_04 (block effect).

The improvement obtained by use of recurrent neural networks is not completely clear from a straight comparison between table 1 and 2, but it could be easily verified when watching at the reconstructed video sequence, where smoother and more natural motions and transitions among frames are actually performed.

References

[1] Jiang J (1999) Image compression with neural network – A survey. In: Signal Processing and image communications, vol 14, 1999, pp 737-760.

[2] Hebb D O(1949) The organizazion of behaviour. New York, Wiley, 1949

[3] Dony R D, Hykin S (1995) Neural network approach to image compression. Proc. IEEE 83, vol 2, February 1995, pp 288-303.

[4] Kohno R, Arai M, Imai H (1990), Image compression using neural network with learning capability of variable function of a neural unit. In: SPIE vol 1360, Visual Communication and Image processing ‘90, pp 69-75, 1990.

[5] Cottrel G W, Munro P, Zipser D (1988), Image Compression by back propagation and examples of extensional programming. In: Sharkey. N. E. (Ed.) Advances in cognition science (Ablex norwood, NJ 1988).

[6] Parodi G, Passaggio F (1994), Size-Adaptive Neural Network for Image Compression. International Conference on Image Processing, ICIP ’94, Austin, TX, USA.

[7] Namphon A, Chin S H, Azrozullah M (1996), Image compression with a Hierarchical Neural Network, IEEE Transaction on Aereospace and electronic System, vol 32, No.1 January 1996.

[8] Guarnirei S, Piazza F, Uncini A, (1999) Multilayer Feedforward Networks with Adaptive Spline Activation Function, IEEE Trans. On Neural Network, vol 10, No. 3, pp. 672-683.

[9] Campolucci P, Uncini A, Piazza F, Rao B D (1999), On-Line Learning Algorithms for Locally Recurrent Neural Networks. IEEE Trans. on Neural Network, vol 10, No. 2, pp 253-271 March 1999.

[10] Back A D, Tsoi A C (1991) FIR and IIR synapses, a new neural network architecture for time series modelling. Neural Computation, vol 3, pp. 375-385.

[11] Back A D, Tsoi A C (1994) Locally recurrent globally feedforward networks: a critical review of architectures, IEEE Trans. Neural Networks, vol 5, pp 229-239.

[12] Rumelhart D E, Ton G E, Williams R J, (1986) Learning internal representations by error propagation, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol 1, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Eds. Cambridge, MA: MIT Press.

[13] Widrow B, Lehr M A, (Sept 1990) 30 years of adaptive neural networks: perceptron, madaline and backpropagation, Proc. IEEE, vol 78, pp 1415-1442.

[14] Cramer C (1998) Neural Network for image and video compression: A review. European Journal of Operational research pp 266-282.

[15] Marsi S, Ramponi G, Sicuranza L (1991) Improved neural structure for image compression. In: Proceedings of the international conference on acoustic speech and signal processing Toronto, Ont., IEEE Piscataway, NJ, 1991 pp.2821-2824.

[16] Zheng Z, Nakajiama M, Agui T (1992) Study on image data compression by using neural network. In: Visual communication and image processing’92, SPIE 1992, pp 1425-1433.

[17] Gray R M (1984) Vector quantization. In: IEEE Acoustic and Speech Signal Processing. Apr. 1984, pp 4-29.

[18] Goldberg M, Boucher P R, Shliner S (1988) Image compression using adaptive vector quantization. In: IEEE Trans. Communication, vol 36, 1988, pp 957-971.

[19] Nasrabadi N M, King R A (1988) Image coding using vector quantization: A review. In: IEEE Transaction on communication, vol 36, 1988, pp 957-971.

[20] Nasrabadi N M, Feng Y (1988) Vector quantization of images based upon Kohonen self organizing feature maps. In: IEEE Proceeding of international conference of Neural Networks, S.Diego, CA, 1988, pp.101-108.

[21] Haykin S (1998) Neural Networks: A Comprehensive Foundation. In: Prentice Hall, 06 July, 1998

[22] Kohonen T (1990) The self organizing map. In: Proc. IEEE, vol 78, pp. 1464-1480, Sept 1990.

[23] Poggi G, Sasso E (1993) Codebook ordering technique for address predictive VQ. In: Proc. IEEE Int. Conf. Acoustic and Speech and Signal Processing '93, pp. V 586-589, Minneapolis, MN Apr. 1993.

[24] Liu H, Yum D J J (1993) Self organizing finite state vector quantization for image coding. In: Proc. of international Workshop on Application of neural networks in telecommunications, Hillsdale, NJ: Lawrence Erlbrume Assoc., 1993.

[25] Forster J, Gray R M, Dunham M O (1985) Finite state vector quantization of waveform coding. In: IEEE transaction on information Theory, vol 31, 1985, pp 348-359.

[26] Luttrel S P(1989) Hierarchical vector quantization. In : IEE Proc. (London), vol 136 (Part I), pp 405-413, 1989

[27] Li J, Manicopulos C N (1989) Multi stage vector quantization based on self organizing feature map. In: SPIE vol 1199, visual Communic and Image Processing IV (1989), pp. 1046-1055.

[28] Weingessel A, Bishof H, Jornik K, Leish F (1997) Adaptive Combination of PCA and VQ neural networks. In: Letters on IEEE Transaction on Neural Network, vol.8 no. 5, Sept 1997.

[29] Huang Y L, Chang R F (2002) A new Side-Match Finite State Vetor Quantization Using Neural Network for image coding. In: Journal of visual Communication and image reppresentation vol 13, pp 335-347.

[30] Noel S, Szu H, Tzeng N F, Chu C H H, Tanchatchawal S (1999) Video Compression with Embedded Wavelet Coding and Singularity Maps. In: 13th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls, Orlando, Florida, April 1999.

[31] Szu H, Wang H, Chanyagorn P (2000) Human visual system singularity map analyses. In: Proc. of SPIE: Wavelet Applications VII, vol 4056, pp 525-538, Apr. 26-28, 2000.

[32] Hsu C, Szu H (May 2002) Video Compression by Means of Singularity Maps of Human Vision System. In: Proceedings of World Congress of Computational Intelligence, May 2002, Hawaii, USA.

[33] Buccigrossi R, Simoncelli E (Dec. 1999) Image Compression via Joint Statistical Characterization in the Wavelet Domain. In: IEEE Trans. Image Processing, vol 8, no 12, pp 1688-1700, Dec. 1999.

[34] Shapiro J M (1993) Embedded Image Coding Using Zerotrees of Wavelet Coefficients. In: IEEE Trans. Signal Processing, vol. 41, no. 12, pp 3445-3462, Dec. 1993.

[35] Skrzypkowiak S S, Jain V K (2001) Hierarchical video motion estimation using a neural network. In: Proceedings, Second International Workshop on Digital and Computational Video 2001, 8-9 Feb. 2001 pp 202-208.

[36] Milanova M G, Campilho A C, Correia M V (2000) Cellular neural networks for motion estimation. In: International Conference on Pattern Recognition, Barcelona, Spain, Sept 3-7, 2000. pp 827-830.

[37] Toffels A, Roska A, Chua L O (1996) An object-oriented approach to video coding via the CNN Universal Machine. In: Fourth IEEE International Workshop on Cellular Neural Networks and their Applications, 1996, CNNA-96, 24-26 June 1996, pp 13-18.

[38] Grassi G, Greco L A (2002) Object-oriented image analysis via analogical CNN algorithms - part I: Motion estimation. In: 7th IEEE International Workshop Frankfurt, Germany 22 - 24 July 2002.

[39] Grassi G, Grieco L A (2003) Object-oriented image analysis using the CNN universal machine: new analogic CNN algorithms for motion compensation, image synthesis, and consistency observation. In: IEEE Transactions on Circuits and Systems I, vol 50, no 4 , April 2003, pp 488 – 499.

[40] Luthon F, Dragomirescu D (1999) A cellular analog network for MRF-based video motion detection. In: IEEE Transactions on Circuits and Systems, vol 46, no 2, Feb 1999 pp 281-293.

[41] Lee S J, Ouyang C S, Du S H (2003) A neuro-fuzzy approach for segmentation of human objects in image sequences. In: IEEE Transactions on Systems, Man and Cybernetics, Part B vol 33, no3, pp 420-437.

[42] Acciani G, Guaragnella C (2002) Unsupervised NN approach and PCA for background-foreground video segmentation. In: Proc. ISCAS 2002, 26-29 May 2002, Scottsdale, Arizona, USA

[43] Acciani G, Girimonte D, Guaragnella C (2002) Extension of the forward-backward motion compensation scheme for MPEG coded sequences: a subspace approach. In: 14th International Conference on Digital Signal Processing, 2002. DSP 2002 vol 1, 1-3 July 2002 pp 191 - 194.

[44] Salembier P, Marqués F (1999) Region-based representations of image and video: Segmentation tools for multimedia services. In: IEEE Trans. on Circuits and Systems for Video Technology, vol 9, no 8, pp 1147-1169, December 1999.

[45] Ebrahimi T, Kunt M (1988) Visual data compression for multimedia applications. In: Proceedings of the IEEE, vol 86, no 6, June 1998, pp 1109-1125.

[46] The International telegraph and telephone Consultative Committee (CCITT) (1992) Information technology - Digital Compression and coding of continuous – tone Still Image Requirements and guidelines. Rec T.81, 1992.

[47] Pennebaker W, Mitchell J (1992) JPEG Still Image Data Compression Standard. Van Nostrand Reinhold, USA, 1992.

[48] Christopoulos C, Skodras A, Ebrahimi T (2000) The JPEG2000 still image coding system: an overview. In: IEEE Transactions on Consumer Electronics, vol 46, no. 4, pp. 1103-1127, November 2000.

[49] ISO/IEC FDIS15444-1:2000 Information Technology – JPEG 2000 Image Coding System. Aug. 2000.

[50] ISO/IEC FCD15444-2:2000 Information Technology – JPEG 2000 Image Coding System: Extensions. Dec. 2000.

[51] Egger O, Fleury P, Ebrahimi T, Kunt M (1999) High-Performance Compression of Visual Information-A Tutorial Review-Part I: Still Pictures. In: Proceedings of the IEEE, vol. 87, no 6, June 1999.

[52] Torres L, Delp E (2000) New Trends in Image and Video Compression. In: EUSIPCO '2000: 10th European Signal Processing Conference, 5-8 September, Tampere, Finland,2000.

[53] CCITT SG 15, COM 15 R-16E (1993), ITU-T Recommendation H.261 Video Codec for audiovisual services at p x 64 kbit/s. March 1993.

[54] Côtè G, Erol B, Gallant M, Kossentini F (1998) H.263+: Video Coding at Low Bit Rates. In: IEEE Transaction on Circuits and Systems for video technology, vol 8, no 7, Nov 1998.

[55] Côtè G, Winger L (2002) Recent Advances in Video Compression Standards. In: IEEE Canadian Review, Spring 2002.

[56] CCITT SG 15 ITU-T Recommendation H.263 Version 2 (1998) Video coding for lowbitrate communication. Geneve. 1998.

[57] Noll P (1997) MPEG digital audio coding. In: IEEE Signal processing Magazine vol 14, no 5, pp 59-81, Sept 1997.

[58] ISO/IEC 11172-2:1993 Information Technology (1993) Coding of moving pictures and Associated Audio for digital Storage media at up to 1.5 Mbits/s. Part 2.

[59] Sikora T (1997) MPEG Digital Video coding Standard. In: IEEE Signal Processing Magazine, vol 14, no 5, Sept 1997 pp.82-100.

[60] ISO/IEC 13818-2, Information Technology (2000) Generic coding of Moving Pictures and Associated Audio Information. Part 2.

[61] Haskell G B, Puri A, Netravali A N (1997) Digital video: an introduction to MPEG-2. Digital Multimedia, standard Series. In: Chapman & Hall 1997.

[62] ISO/IEC 14496-2:2001 Information Technology. Coding of audio-visual objects. Part 2.

[63] Grill B (1999) The MPEG-4 General Audio Coder. In: Proc. AES 17th International Conference, Set 1999.

[64] Scheirer E D (1998) The MPEG-4 structured audio Standard. In: IEEE Proc. On ICASSP, 1998.

[65] Koenen R (2002) Overview of the MPEG-4 Standard-(V.21-Jeju Version). ISO/IEC JTC1/SC29/WG11 N4668, March 2002.

[66] Aizawa K, T. S. Huang, Model Based Image Coding: Advanced Video Coding techniques for low bit-rate applications. In: Proc. IEEE, vol 83, no 2, Feb. 95.

[67] Avaro O, Salembier P (2001) MPEG-7 systems: Overview. In: IEEE Transaction on circuit and system for video Tecnology, vol 2. no 6, June 2001.

[68] ISO/IEC JTC1/SC29/WG11 N3933, Jan 2001. MPEG-7 Requiremens document.

[69] Manjunath B S, Salambier P, Sikora T (2002) Introduction to MPEG-7: multimedia content description language. In: Jhon Wiley & Sons 2002.

[70] Martínez J M, MPEG-7 Overview (version 9), ISO/IEC JTC1/SC29/WG11N5525, March 2003

[71] Burnett I, Walle R W, Hill K, Bormans J, Pereira F (2003) MPEG-21: Goals and Achievements. In: IEEE Computer Society, 2003

[72] Bormans J, Hill K (2002) MPEG-21 Overview v5, ISO/IEC JTC1/SC29/WG11 N5231, October 2002

[73] Saupe D, Hamzaoui R, Hartenstein H (1996) Fractal image compression: An introductory overview. In: Technical report, Institut für Informatik, University of Freiburg, 1996.

[74] Kung S Y, Diamantaras K I, Taur J S (1994) Adaptive Principal component extraction (APEX) and application. In: IEEE Trans. Signal. Processing vol 42 (May 1994) pp 1202-1217.

[75] Piazza F, Smerilli S, Uncini A, Griffo M, Zumino R, (1996) Fast Spline Neural Networks for Image Compression. In: WIRN-96, Proc. Of the 8th Italian Workshop on Neural Nets, Vietri sul Mare, Salerno, Italy.

[76] Skrzypkowiak S S, Jain V K (1997) Formative motion estimation using affinity cells neural network for application to MPEG-2. In: Proc. International Conference on Communications, pp 1649-1653, June 1997.

[77] ISO/IEC JTC1/SC29/WG11, ITU-T VCEG: working draft number 2 of Joint Video team standard".

[78] Topi L, Parisi R, Uncini A (2002) Spline Recurrent Neural Networks for Quad-Tree Video Coding. In: WIRN-2002, Proc. Of the 13th Italian Workshop on Neural Nets, Vietri sul Mare, Salerno, Italy, 29-31 May 2002.