OBJECT ORIENTED IMAGE ANALYSIS FOR VERY LOW BITRATE VIDEO CODING SYSTEMS USING THE CNN UNIVERSAL MACHINE

by

A. Stoffels, T. Roska, and L. O. Chua

Memorandum No. UCB/ERL M95/79

11 October 1995

OBJECT ORIENTED IMAGE ANALYSIS FOR VERY LOW BITRATE VIDEO CODING SYSTEMS USING THE CNN UNIVERSAL MACHINE

by

A. Stoffels, T. Roska, and L. O. Chua

Memorandum No. UCB/ERL M95/79

11 October 1995

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

OBJECT ORIENTED IMAGE ANALYSIS FOR VERY LOW BITRATE VIDEO CODING SYSTEMS USING THE CNN UNIVERSAL MACHINE

A. Stoffels*, T. Roska†, and L.O. Chua‡

*Institut für Allgemeine Elektrotechnik und DV-Systeme of the RWTH Aachen, University of Technology, and Nonlinear Electronics Laboratory of the University of California, Berkeley
(stoffels@fred.eecs.berkeley.edu)

†Computer and Automation Institute of the Hungarian Academy of Science, Budapest, and Nonlinear Electronics Laboratory of the University of California, Berkeley
(roska@fred.eecs.berkeley.edu)

‡Nonlinear Electronics Laboratory of the University of California, Berkeley
(chua@fred.eecs.berkeley.edu)

ABSTRACT

The CNN Universal Machine is applied to object oriented image compression algorithms and proves its universality for future applications in the field of very low bit rate coding. This proposal joins [24] in unfolding the enormous computational abilities for a wide class of video compression techniques. Here a novel image analysis technique has been considered and realized in form of analogic CNN algorithms. The specific features of the scheme, among them the extensive use of dynamic (finite running-time) CNN cloning templates, are outlined and discussed through different computer simulations. When implemented on the CNN Universal Machine, its performances outdo those of equivalent digital systems and qualify the CNN Universal Machine as a serious competitor for future video coding hardware.

I INTRODUCTION

On the eve of cellular videophone systems and videoconferencing facilities, the demand for coding techniques providing very high compression ratios while maintaining a good picture quality is steadily increasing. The already standardized methods do not cover this area and are basically focusing on video on demand and video storage architectures. The sole approach to videophone applications is ITU's H.261 standard which is the official predecessor of the recently released MPEG standards. This group of coding schemes is commonly referred as block based systems due to their intrinsic approach to motion estimation. In terms of image understanding and redundancy removal in the incoming videostream, these techniques have been proved to be suboptimal [22]. The H.261

standard offers its lowest transmission rate at 64 kbit/sec, which might satisfy the needs of low quality systems using regular twin-paired wires or even broadband channels, but this rate is far above the threshold allowed by cellular services. Further, the quality is even then fairly poor, and so improvements are necessary to open the door to future telecommunication services, as previously mentioned.

The *object oriented coding* approach offers therefore a more sophisticated approach to video coding. Its main difference to the block based systems is a strong image analysis stage that can locate and label the shape, color and motion of objects appearing in the original frame. Thus regions of great importance for image understanding, such as the facial expressions of a speaker, for example, can be identified and subsequently coded using special parameter sets. Still background objects that hardly change from one frame of the sequence to another can be detected and are not included into the coding step because the receiver can simply use the information used for the previous image. This scheme decreases the required transmission rates significantly and allows the use of smart concepts to even reduce these rates. If an upper boundary for image data may be given, assume a typical value of 10 to 15 kbit/sec, then data might be hierarchically ordered from higher to lower importance, e.g. by using the magnitude of their motion, and be placed on the channel until the given bound is reached. The remaining parts of the frame can then be built up by simply restoring the information of its predecessor.

Object based schemes are very likely to become part of the new emerging MPEG4 standard that is scheduled to be released in 1998 and is expected to cover the range of very low bit rate applications. Unlike the other, block based, MPEG standards which can be implemented by composing dedicated specialized hardware devices computing very specific task, such as the *discrete cosine transform* DCT, the probable MPEG4 basic architecture is composed of a multiple purpose video processing unit in its center, that is supported by various coprocessors. It is easy to realize that the efficiency of such an object oriented algorithm is directly related to the accuracy and speed of the available image analysis algorithms. But in the case of real time applications like videophone services, the hardware is not allowed to pass a certain processing time per frame, which may lead to important losses in output quality.

We therefore propose the *CNN Universal Machine* (CNNUM) [19] with its short computation times for convolution/deconvolution based nonlinear dynamic image processing as elementary instructions to attempt to overcome this drawback. Offering transfer rates of 50-100μsec per frame without I/O operations, a larger amount of image features may be detected and so, the quality of the segmented output increases as well. Being the main engine of an object oriented coding scheme, the CNNUM would be responsible for the image analysis operations. It is evident that this part of every object based system asks for the highest computational power and fits at the same time with the main talents of the CNN architecture that proved its importance for real time image analysis tasks throughout many reported contributions [3][4][5][6]. The remaining coding and transmitting operations can then be managed by digital coprocessors in order to offer a reliable interface to systems without the CNN core.

As many different approaches are proposed for image analysis in object oriented coding techniques, and as the emerging standard is expected not to fix this part of the algorithm, we show the universality of the CNNUM based image analysis for a popular image analysis algorithm. This method takes advantage of luminance contrasts and motion information to detect the contour lines of moving objects. Another method can be derived from the extensive use of mathematical morphology. [21] depicts such a model whose main morphological operations are described in [28]. Varying from the first scheme, this technique finds objects characterized by regions of homogeneous intensity distributions.

Generally speaking, we show complex analogic CNN algorithms using known templates (especially in [1], [24], and [26]), adjusted templates, and some new templates. The key point is the CNN algorithmic aspect and the speed advantage provided by the CNNUM architecture. Moreover, dynamic (finite running-time) templates are used extensively. In a more detailed description we can state that the detection of moving contours is based on a combination of a convolution with a gradient like operator to extract relevant contours and a subsequent skeletonization of the contour mask. The major advantage of this procedure is the fact that most operations are done in a purely binary mode. This enables the use of simple digital memory to store intermediate results outside the CNN Universal Machine without time consuming analog-digital conversion.

Section II provides a more detailed view at the algorithm. Section III discusses then the results of different simulations. The issue of future implementations of the presented algorithm on a CNN Universal Machine is finally discussed in Section IV.

II Object Oriented Image Analysis Algorithm

II.1 Introduction and General Structure

In the context of an object oriented coding scheme, the main task to be accomplished by an image analysis algorithm is the segmentation of a scene into different moving regions. These regions should coincide with the real image to guarantee that the modeling and, in particular, the description of the motion is efficient. Furthermore, the quality of such an object related segmentation depends also on its consistency throughout a sufficient number of image frames. This means that all segmented objects should have a global uniform velocity that enables their recognition during a significant time interval. Only this fairly strong condition allows the high compression rates of object oriented coding schemes.

In contrast to still image analysis algorithms, we deal with two different segmentation techniques that consider the spatial as well as the temporal character of a distinct object. We talk about *intraframe segmentation* when we consider solely the information provided by a single frame, for example contours and textures. The contours correspond to abrupt intensity changes in an image. The detection of reliable contour data is extremely

important for video coding applications. Errors resulting from a bad boundary detection are highly visible to the human eye and decrease the sequence's quality significantly. Therefore their correct localization is the most important task to be achieved by a good segmentation algorithm. The second semantic component to be considered is the texture of a region. Generally we interpret the region's intrinsic color (or gray-scale) and its pattern as its texture information. Compared to contour detection, inconsistencies in texture representation are far less disturbing and reduce the overall quality of the decoded output only slightly.

The third pillar of a segmentation algorithm is the motion analysis in a scene. This step is widely referred to as *interframe segmentation*. The interpretation of the motion occurring in a certain image sequence can be helpful at various stages of a segmentation. The most pertinent case is the distinction between changed and unchanged regions. The computation of the difference between two succeeding frames followed by an adequate thresholding operation determines those areas. Furthermore, the choice of another, higher threshold can extract those parts of a sequence whose changes are quick and abrupt. These regions are likely to be non-predictable by a predictive, motion compensating coding scheme. Therefore, their shape and content has to be transmitted for every frame of a sequence. The remaining parts of the scene, i.e. static background and slowly moving objects, are predictable and are fed into special coding algorithms.

Looking at the special character of videophone and videoconferencing scenes, we started from various assumptions to realize the segmentation. Most of these sequences show a moving person in front of a still background. The attention of the receiving person are likely to be focused on the movements of the facial regions, such as the eye and the mouth. Their recognition is mainly based on its motion and their shape can only be roughly estimated using contour information. The use of texture is even less productive, because most texture segmentation algorithms need a certain content of higher frequencies in the regions spatial frequency spectrum. Unfortunately, facial zones point out a very smooth and unicolored pattern, which is hard to extract in an unsupervised application like the one we are looking at. Finally we have to focus on the overall goal of all our coding endeavors. This is a significant coding gain and an output bitstream with a data rate below 15 kbit/sec. This implies a limited number of codable objects, which means at the same time that the objects have to be larger but still reliable in terms of a uniform motion. Such a result is only achievable if the segmented objects coincide exactly with the real world's objects.

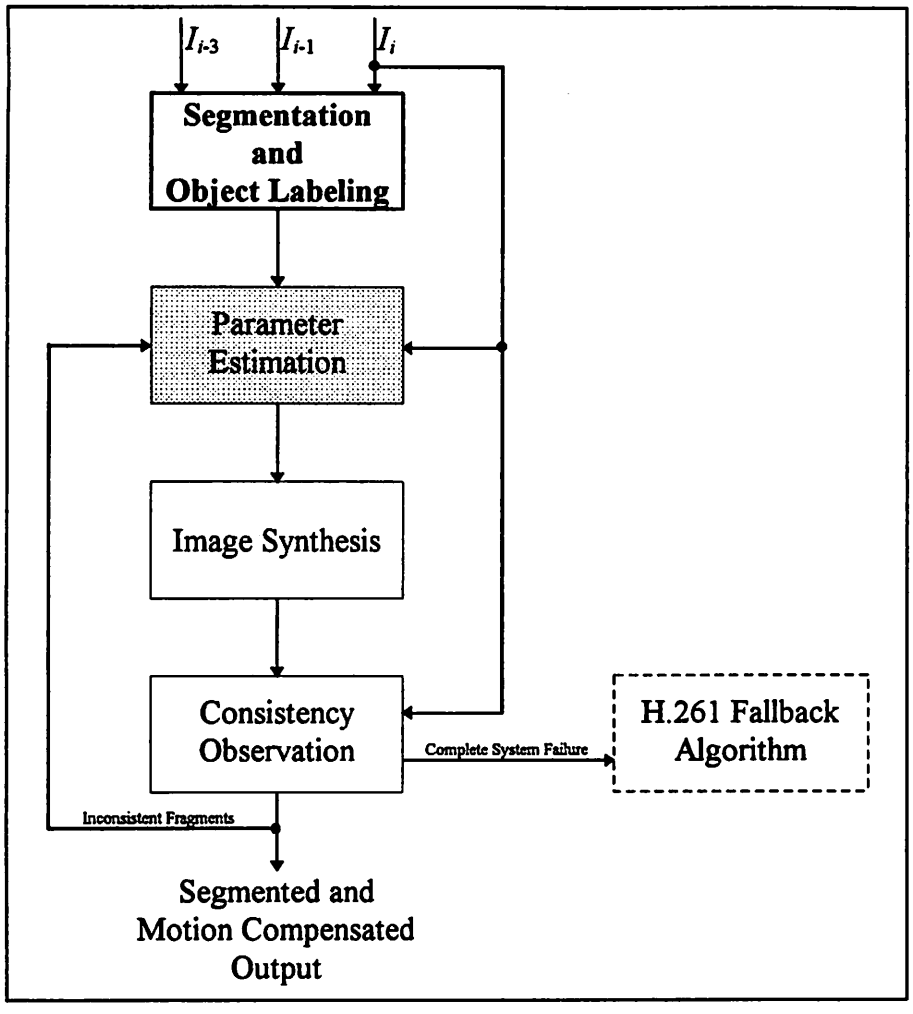


Figure 1. Block diagram of the *Image Analysis* algorithm of an object oriented coding scheme. CNNUM suited tasks are marked by blocks with a white background color. Our intention is focused on the emboldened block which is discussed in greater detail later in this section. A light gray background denotes a task to be processed by an additional microprocessor. Dashed lines symbolize procedures that are yet to be designed. The input is a group of three frames of a given sequence. The output are the segmented objects given in a parameter set coding the object shape, texture and motion. The H.261 Fallback System is not pretended to interact in the usual data flow of this object oriented coding scheme, but it is used in the case of a complete system failure, which may occur when the incoming frames vary extremely from the assumptions which are valid in the context of a regular videophone sequence.

Figure 1 presents the general skeleton of an object oriented image analysis block in which we aim to integrate our segmentation algorithm, here called *Segmentation and Object Definition*. This architecture is designed to fit into a wide range of proposed object oriented coding methods [13][14]. All object oriented approaches report important difficulties realizing a reliable segmentation digital hardware under the extremely strict time limitations. The CNN Universal Machine increases the computational speed enormously and offers at the same time an astonishing segmentation quality. After having computed the segmentation in the form of binary masks marking the locations of extracted

fragments, a second stage models the geometrical and motion properties of the objects and encodes them using a special parameter set. The main idea of this *Parameter Estimation* is to describe the mapping of the real world's three-dimensional (3D) space onto the two-dimensional (2D) image plane, the motion (2D motion with two degrees of freedom or 3D motion with 6 degrees of freedom), the surface shape and the local luminance signal which equals our understanding of the region's texture. Different approaches to parameterize this data set have been reported [9][11][14][15], and their different applications and performance features have been widely discussed throughout the related literature. The *Image Synthesis* stage assembles all parameterized image fragments to a synthesis image which is then compared to the original frame in the *Consistency Observation* block. The iterated structure of our model allows then a repeated processing of the parameter estimation, which occurs mainly if image fragments cannot be described by the previous, simpler parameter set and need to be refined. In the case of a complete system failure, which is likely to happen when our basic presumptions of slow motion videophone scenes are neglected, the system may switch over to a *H.261 Fallback Algorithm*. But these considerations are beyond the scope of this research, and the use of the CNN Universal Machine for block-based image compression schemes remains still very doubtful.

II.2 Segmentation and Object Labeling

In Figure 2 we have a closer look at the structure of the *Segmentation and Object Labeling* algorithm. In order to maximize the luminance information given in the image sequence, we convert the color input into the YIQ gray-scale measure, which is used in the American NTSC television. Here, the *Y* layer is subject to further processing. The *Edge Enhancing Low Pass Filtering* then removes disturbing noise patterns with a linear low pass operation. This operation maintains meanwhile the sharpness and contrasts of boundary zones in order to improve the following edge detection performances. At this point, the algorithm stores the filtered intermediate result. The segmentation of subsequent image frames uses it for motion related functions. It has to be emphasized here that this image is the only full gray-scale image to be stored while computing the whole segmentation routine. All other I/O operations occurring while processing a segmentation are of binary nature and therefore far less memory is required and the time consuming A/D conversions between the CNN Universal Chip and surrounding digital memory can be avoided. The detection and segmentation of fast moving areas is done in the *Remarkable-Feature Extraction* algorithm. These areas are expected to contain information like eye, nose, mouth and ears of a speaker in a videophone sequence. Generally it can be assumed that any deterioration of the facial expressions are highly visible and decrease the resulting image quality significantly. This mask is then merged with the result of the *Intraframe Segmentation*, which is the detection of the still image's contours, and yields the complete object oriented segmentation mask of a given image frame. Finally the information provided by the *Motion Detection* module aims to separate the moving objects from the unchanged background. In contrast to earlier contributions to the CNN paradigm [1][17][18], this module detects general motion in an arbitrary gray scale environment. This step enables the selective coding of only those fragments that are changed compared

to the preceding frames. The binary output is ordered by size and marks those areas that are subject to the succeeding parameter coding.

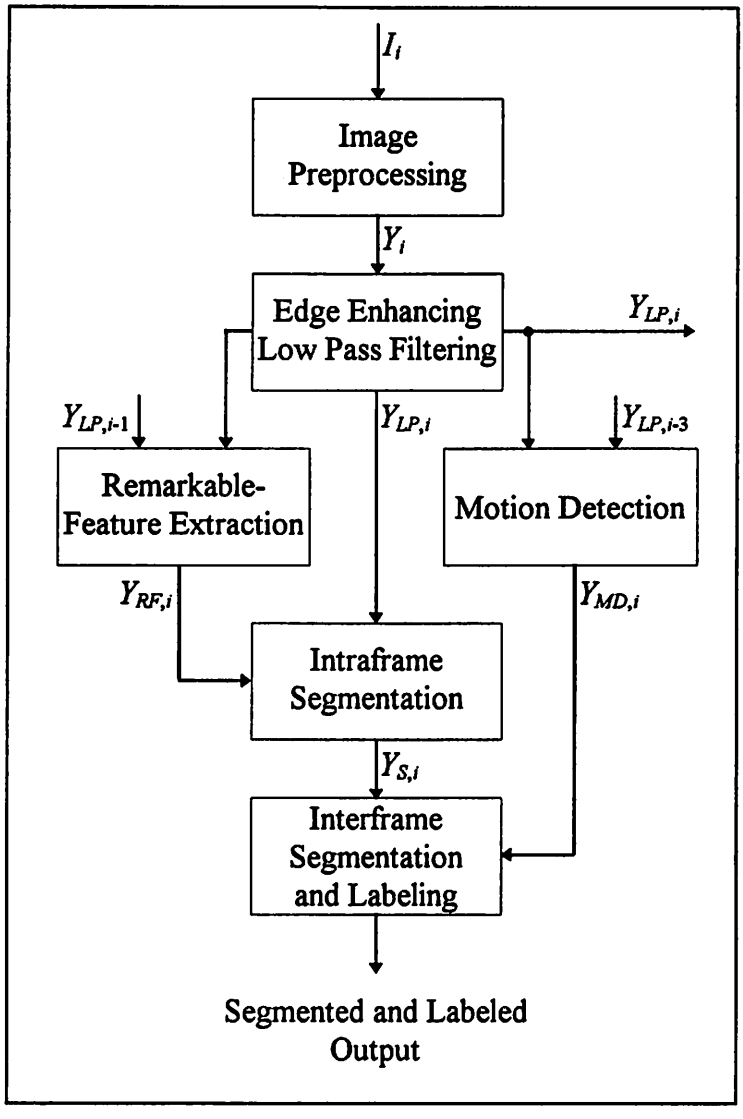


Figure 2. Block diagram illustrating the structure of the *Segmentation and Object Labeling* algorithm. This is the first block of the image analysis described in Figure 1. One can imagine the same architecture for the processing of all image frames. At certain levels these architectures interact mutually and exchange their low pass filtered intermediate results, this is $Y_{LP,i}$, $Y_{LP,i-1}$, and $Y_{LP,i-3}$, as it is demonstrated in this figure.

II.2.1 IMAGE PREPROCESSING

Dealing in general with color images, we have to decompose them first into a corresponding gray scale model. Generally, colors can be simulated by a superposition of three primary colors, typically red, green, and blue. When a color image is processed, three filters are used to extract the red, green, and blue (RGB) intensities in the image.

Later we perceive the recombination of the extracted intensity information exactly as the original color. Unfortunately, the direct use of the RGB components is not reasonable, because the human visual system is not particularly sensitive to pure color information.

CNN templates were already developed for transformations between different color representations [20]. Given a RGB triplet (R_i, G_i, B_i) for pixel i, we compute the YIQ values by

$$\begin{bmatrix} Y_i \\ I_i \\ Q_i \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}, \tag{1}$$

in a similar way as it has been presented in [20] for multi-layer CNN architectures. This yields the three YIQ channels: the Y, or luminance, signal measures the perceived brightness of the color and supports therefore a contrast and intensity based edge detection, and the less significant chrominance signals I and Q. Omitting the chrominance information we can develop the following cloning templates to realize the *RGB→Y Transformation* [22]:

$$\mathbf{A} = [0], \quad \mathbf{B}_R = [0.299], \quad I = 0, \tag{2}$$

$$\mathbf{A} = [0], \quad \mathbf{B}_G = [0.587], \quad I = 0, \tag{3}$$

$$\mathbf{A} = [0], \quad \mathbf{B}_B = [0.114], \quad I = 0. \tag{4}$$

The corresponding flow chart is sketched in Figure 3. Here we assume the *RGB* decomposition as already computed by the camera. This is a reasonable assumption when talking about cellular videophone or any kind of videoconferencing system.

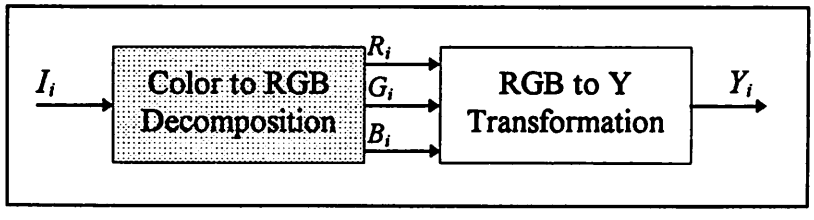


Figure 3. *Image Preprocessing* decomposes the color frames into the YIQ system and extracts the Y channel. The incoming image is first split off into a red R_i, green G_i, and blue B_i channel, commonly reported as the RGB system. This decomposition is usually

part of the camera itself and is therefore not realized on a CNN Universal Machine. The *RGB to Y Transformation* computes then the luminance information Y_i.

II.2.2 EDGE ENHANCING LOW PASS FILTERING

The complete body of the *Edge Enhancing Low Pass Filtering* is illustrated in Figure 4.

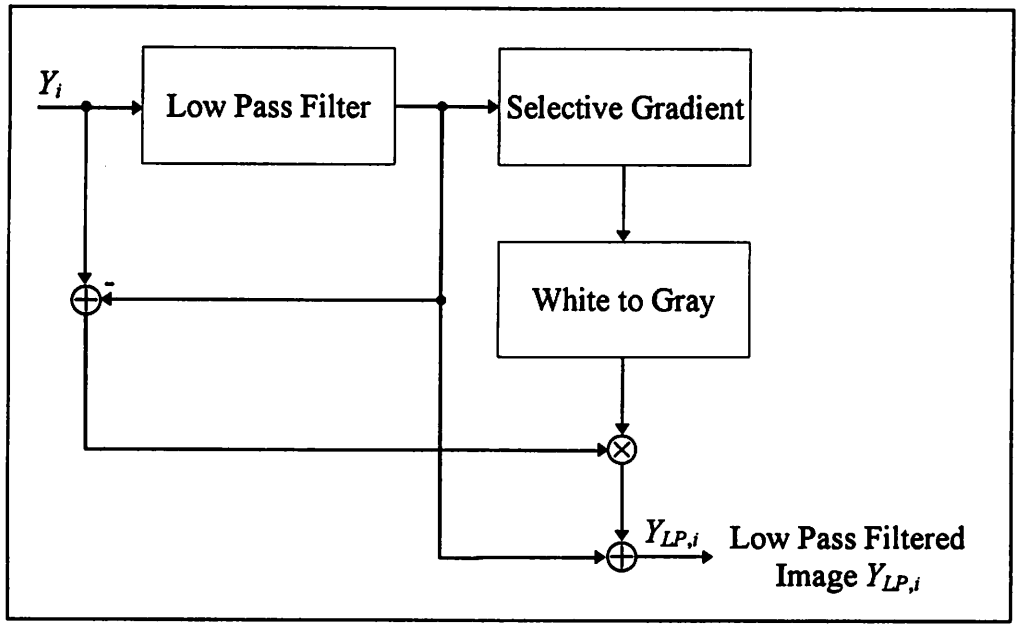


Figure 4. The *Edge Enhancing Low Pass Filtering* stage is intended to reduce the incoming image's noise with a special low pass filter operation. In order to maintain the sharpness of the scene's contours, an additional edge enhancement restores the most important features, this is the moving contours.

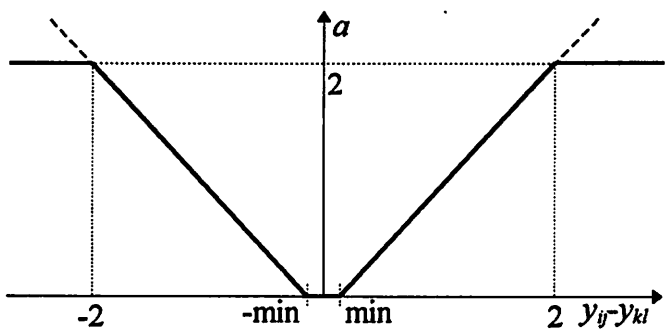
When considering the use of the object oriented coding scheme in wireless communication systems, one has to take care of the significant disturbances introduced by noise patterns. In addition, usual twisted telephone wires are far from being ideal. Therefore a general need for noise removing filter techniques is obvious. A look on the power spectral densities of natural images reveals a very high percentage of lower (and lowest) frequencies, this means spatial waves having a period length of 10 pixels and more. This leads to the assumption, that higher (and highest) frequencies usually belong to noise and have to be smoothed. This operation is done by a spatial *Low Pass Filter* template:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}, \quad I = 0 \tag{5}$$

This cloning template, a combination of a Laplacian-like **A** template and an image smoothing **B** template, obtained the best overall noise reduction while maintaining a good output quality in terms of sharpness. Other templates for low pass filtering can be designed using the knowledge given in [7] concerning linear image filtering.

The drawback of a general low pass filtering operation is that the blurring does not spare the contour information given in the image. This is the reason for an additional edge enhancement algorithm to restore the sharpness of the image's boundary zones. Following some ideas describing the use of the *aniosotropic diffusion* [16] and the CNN implementations (e.g. in [1]), we realized this task by calculating first the *Selective Gradient*, which detects important boundary data only and omits less relevant parts of the low pass filtered output by tuning its *minimum* bound. The template is similar to the Gradient template given in [1], but allows a more accurate detection of relevant boundary positions. Its result is a black and white output image. The threshold is fixed and depends on the source data. As the system is limited to certain applications, for example videophones, the threshold amplitude can be prescribed once and needs no further changes within the same application. The low pass filtered image is given to both input and state.

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix},\quad \mathbf{B}=\begin{bmatrix}a & a & a\\ a & 0 & a\\ a & a & a\end{bmatrix},\quad I=-\text{threshold}. \tag{6}$$



Then we convert the white areas of the binary into *zero-gray* values. In the CNN literature, the gray levels of a given gray scale images lay in the range of [-1,+1], where white is equivalent to -1 and black is equivalent to +1. Thus the zero-gray level refers explicitly to the half intensity in the image space. In order to compute the conversion, we apply the black and white frame to the input of the CNN, while a unicolored zero-gray image of the same size is given to the output state.

$$\mathbf{A}=[1],\quad \mathbf{B}=[0.5],\quad I=0.5 \tag{7}$$

Black and zero-gray masks are the appropriate choice when arithmetical operations have to be computed. Here we multiply the black and zero-gray gradient mask with the difference of the original frame and the low pass filtered one, which is the high pass filtered image frame. Thus we consider only those high frequency components that belong to real image contours and not to noise. Adding the result to the low pass filtered image produces therefore a significant edge enhancement effect.

II.2.3 MOTION DETECTION

The flow chart of the *Motion Detection* algorithm is given in Figure 5.

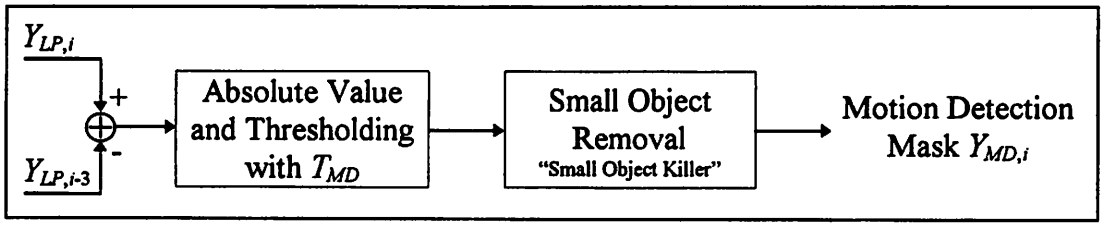
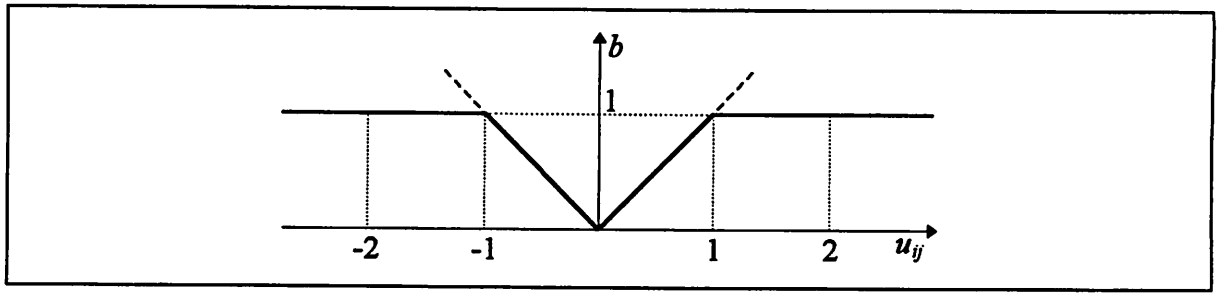


Figure 5. *Motion Detection* is done by computing the pixelwise difference of two incoming frames. The significant motion is extracted by a thresholding operation and an elimination of small detected moving areas that are mostly due to noise influences. The output is a binary mask where moving parts are marked in black.

In a first step the algorithm calculates the pixelwise frame difference of two fields i and i+3. Here, the period length is chosen in a way that every third frame of an original sequence of 30 frames/sec is encoded. This results in a coding scheme transmitting only 10 frames/sec, which denotes a compression ratio of 3:1. Other typical transmission situations, such like frame rates of 7.5 Hz or even 5 Hz are easy to adapt by increasing the number to 4 or 6, respectively. Generally it can be assumed that the longer the period length is selected, the more accurate the image analysis has to work in order to avoid important artifacts. This results from the simple presumption that the longer we choose the period to be, the more complex the scene's apparent motion will be. Then, the absolute frame difference is computed by applying the *Absolute Value* template as it is given in (8). The image frame showing an intensity distribution in [-1,+1] is fed to the input of the CNN, and the nonlinear **B** template maps it onto the range [0,+1].

$$\mathbf{A} = [0], \quad \mathbf{B} = [b], \quad I = 0 \tag{8}$$



Comparing the result to a certain threshold, the pixel is assigned either to the change state, that is black, if the threshold T_{MD} exceeded or to the unchanged state, this is white, if the result is under or equal T_{MD}. Such a thresholding operation is easy to implement on a CNN Universal Machine [1]. We download the image to be thresholded in the state and execute the following template operation until the transient is settled.

$$\mathbf{A}=[2],\quad \mathbf{B}=[0],\quad I=-T_{MD}. \tag{9}$$

The choice of the threshold is a crucial part of the whole algorithm. A very interesting approach to obtain a threshold amplitude fitting the purpose of detecting the contour motion might be the allocation of the standard deviation of the mean squared frame difference to the threshold T_{MD}. Nevertheless, in our simulation to be reported later we have chosen a fixed value for T_{MD} and reached a reliable extraction of the moving contours. Generally the threshold has to consider the chosen transmission rate, too. A higher frame frequency needs a lower threshold value and *vice versa*. It is important to note that neither an adapted nor a fixed threshold are able to extract only relevant data. Moving contours often show a large number of neighbored pixels detected after the thresholding. This is in a sharp contrast to insignificant objects of smaller size. Therefore we can eliminate these regions using a *Small Object Removal* or *Small Object Killer* template, as it has been reported in [26].

$$\mathbf{A}=\begin{bmatrix}1&1&1\\1&2&1\\1&1&1\end{bmatrix},\quad \mathbf{B}=\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix},\quad I=0. \tag{10}$$

The resulting binary output mask preserves the moving objects in their entirety and, after an intermediate storage, the algorithm uses it later in the interframe segmentation to divide changed and unchanged regions.

II.2.4 REMARKABLE-FEATURE EXTRACTION

Similar to the Motion Detection procedure, the *Remarkable-Feature Extraction* procedure also uses a frame difference to compute its result. The main purpose is the detection of image regions showing high speed changing. In the case of typical head-and-shoulder-scenes with a very detailed display of the person's facial expressions, it can be assumed that most of the attention a possible spectator is paying to the scene is focused on the speakers eyes, mouth, and even nose or ears. Artifacts of any kind in the transmission of these components decrease therefore the subjective perception of the sequence's quality in a very important manner. Furthermore, these changes are not predictable within a typical transmission rate of 10 frames/sec, this means, it is impossible to estimate the motion of these regions. Therefore they have to be coded and transmitted in a pure intraframe coding mode for every transmitted image frame. A more detailed consideration of this algorithm follows the presentation of the flow chart in Figure 6.

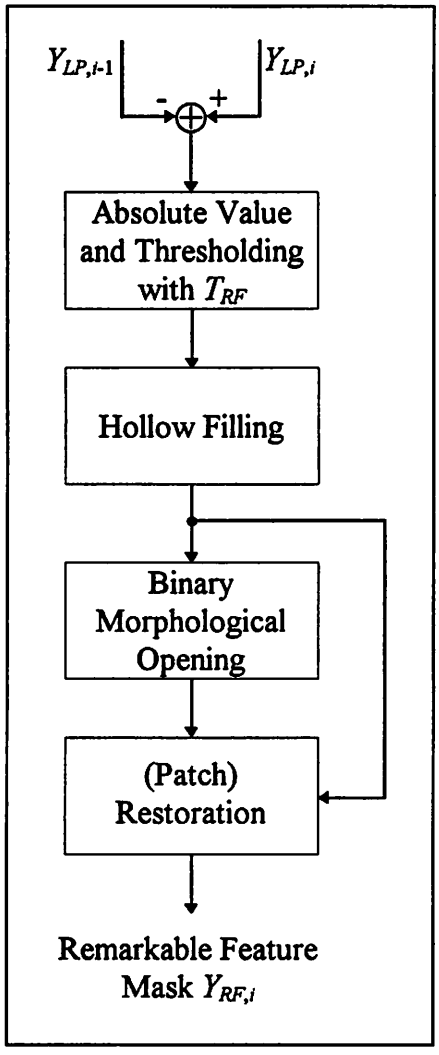


Figure 6. The aim of the *Remarkable-Feature Extraction* function is the detection of image parts fulfilling certain size and motion criteria at the same time. This result is computed by taking the difference of two directly succeeding frames, whose absolute pixel values are then compared to a relatively high threshold. The *Hollow Filling* intends to merge very small objects, such as the eye balls or the teeth, with the surrounding object.

The detection of a fast moving object is most likely when taking the difference of two successive image frames. The applied threshold T_{RF} is independent of the threshold T_{MD}. Here we operated also with a fixed value for different video sequences. Despite this simple approach, the related results illustrate a very good overall quality. Except the different threshold amplitude, the thresholding operation is the same as it was used at the beginning of the *Motion Detection* block. The binary output mask shows a rather scattered set of objects and is therefore hard to code or label. The task to be achieved now is simplifying the fragments in order to shorten their contours and to fill holes laying inside the objects themselves. The latter job is done by the *Hollow Filling* template, which has been presented and discussed in [26]:

$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = 3.5. \tag{11}$$

The problem raising when using this template is mainly its transient (a finite running-time template) character. Therefore one has to stop the template's transient after 10-15τ. This time interval proved to be sufficient to fill inside holes without deteriorating the outer shape. The output is subsequently processed by some operations belonging to the class of binary morphological image processing algorithms [27]. More details about this issue are presented in [28], where a short summary of the main ideas of the mathematical morphology is followed by its realization on the CNNUM. Generally, we simplify the input mask by a sequence of *morphological erosions* and *dilations* [10], which is known as a *Binary Morphological Opening*. The simplification effect is the greater the more iterations of erosions and dilations are computed. Notice that the features do not revert completely to their original shape during dilation, but instead take on the shape imposed by the *structuring element*'s shape. The structuring element is the mask which determines the way an image is eroded or dilated [10]. Continuing the dilation beyond the number of erosion cycles makes the features larger than their original size. The following flow chart given in Figure 7 explains this procedure graphically.

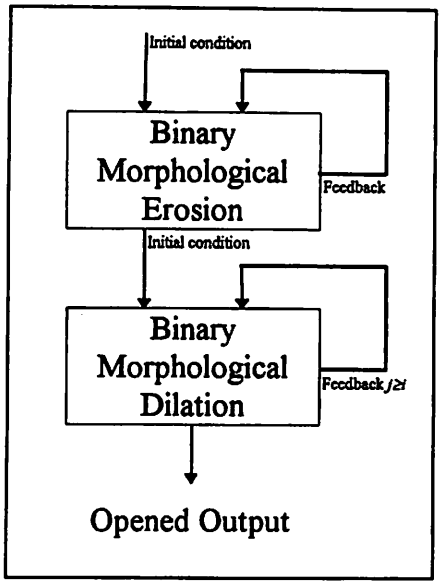


Figure 7. Flow chart of the iterated *Binary Morphological Opening* with a 3×3 square structuring element. The number of iterated erosions is equal or smaller the number of iterated dilations. A higher number of dilations enlarges those objects given in the original frame that are not completely erased after the erosion. The original frame is here referred as *initial condition* for the erosion, because it is only used for the first iteration. Following the same idea, the final output of the erosion is also fed to the dilation only while computing the first iteration.

Binary morphological erosion and dilation by means of a square structuring element are realized on a CNN Universal Machine with the templates (12) and (13) [28].

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad I_E = -9.5, \tag{12}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad I_D = 9.5, \tag{13}$$

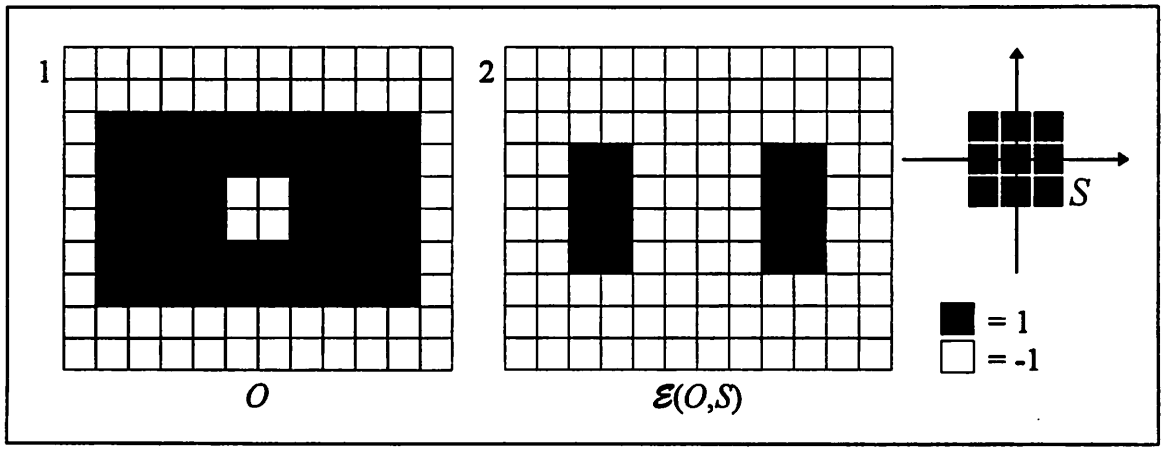


Figure 8. Binary erosion $\mathcal{E}(O,S)$ of an image O by a square structuring element S. Picture 1 shows the original image O. Picture 2 depicts the result of the binary erosion of

O by S. It is easy to understand one of the main applications of binary erosion, this is the separation of different main features of the image, here the two rectangular blocks, by deleting the connecting elements.

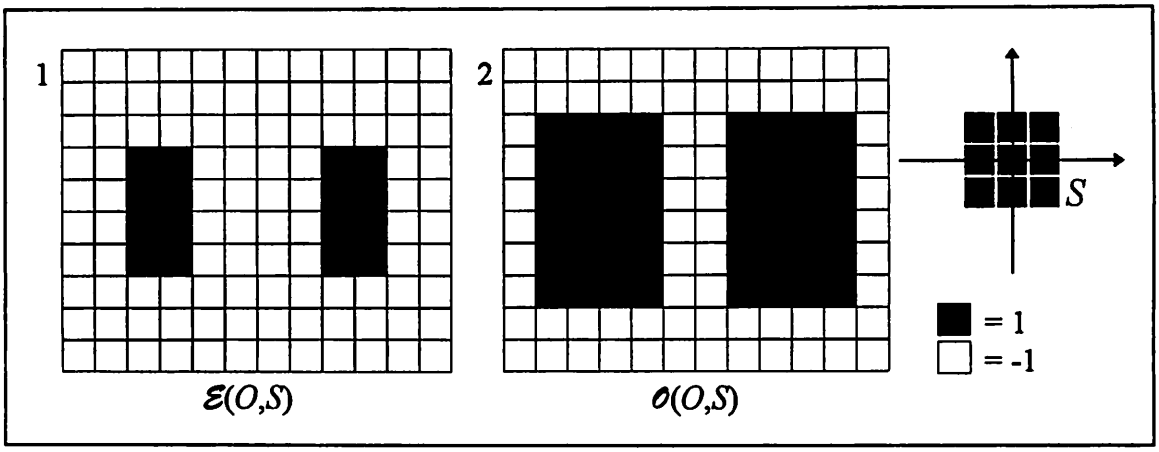


Figure 9. Binary opening $\mathcal{O}(O,S)$ of an image O by a square structuring element S. Picture 1 shows the eroded original image $\mathcal{E}(O,S)$. Picture 2 illustrates the result of the binary dilation of $\mathcal{E}(O,S)$ by S which equals the binary opening $\mathcal{O}(O,S)$.

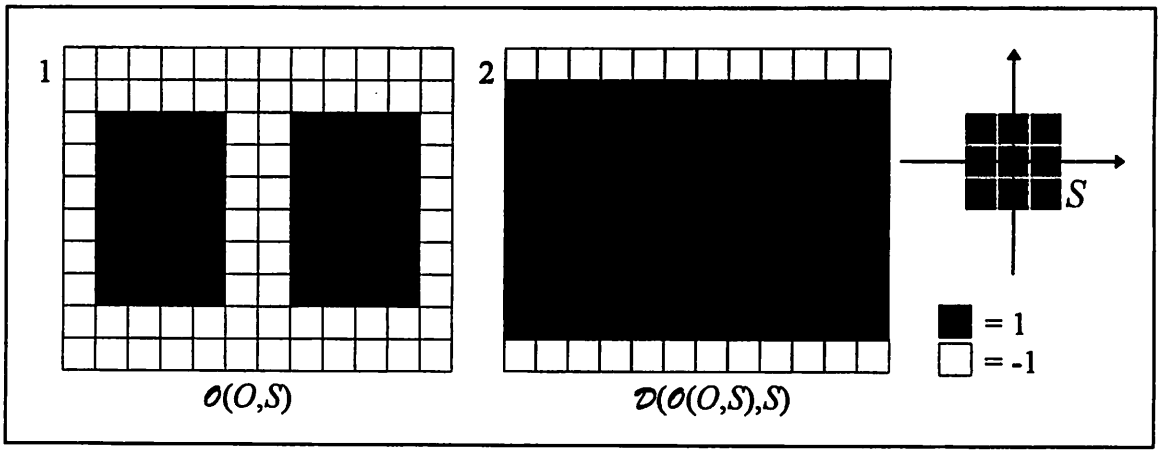


Figure 10. A second binary dilation $\mathcal{D}(\mathcal{O}(O,S),S)$ of the opening $\mathcal{O}(O,S)$ by a square structuring element S. Picture 1 shows the original image $\mathcal{O}(O,S)$. Picture 2 depicts the result of the binary dilation of $\mathcal{O}(O,S)$ by S. One can see how a dilation is able to join different features of an image and to fill cracks within an object.

Some of the most striking geometrical aspects of binary morphological erosion, dilation and opening are pictured and explained in the Figures 8, 9, and 10. The execution the *Binary Morphological Opening* requires the image to be processed to be given at the same time to the input and the state. The now simplified mask is finally compared with the result after the *Hole Filling* operation in order to eliminate objects that are smaller than N_{i-1}. This operation is done by the *(Patch) Restoration* template given in (14), [26]. The simplified image is loaded in the state of the CNN, while the image to be compared with is

given to the input. The final *Remarkable-Feature Mask* $Y_{RF,i}$ contains then only those fragments that fulfill the size requirements defined by the number of iterations of the morphological opening.

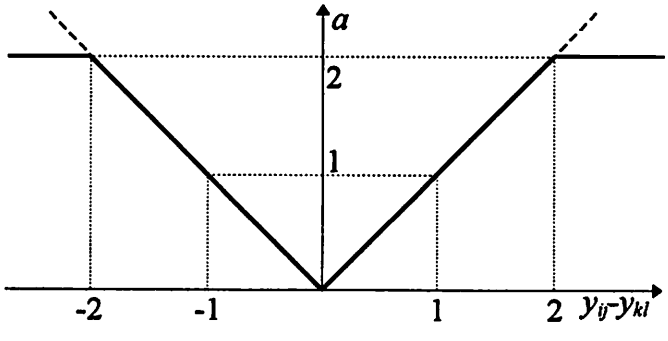
$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = -1.5 \tag{14}$$

II.2.5 INTRAFRAME SEGMENTATION

After having included two different aspects of motion into our segmentation algorithm, we focus now on the information coming with one single image frame. This part of a video compression scheme is commonly called *Intraframe Segmentation*. Our intraframe mode is not a pure one, because we include the *Remarkable-Feature Extraction* into the data flow of the segmentation. Nevertheless, this is not done to detect moving image parts for motion coding purposes, but to extract a special class of objects, here, the facial expressions of a speaker. The block diagram of the *Intraframe Segmentation* is pointed out in Figure 11.

The first step of the segmentation is the detection of the scene's contours. Like it has already been done earlier in this image analysis, we use also here a gradient operator based edge detection template. In contrast to the previously reported *Selective Gradient*, we take also very small pixel differences into our considerations. For the rest the use of the *Gradient* template is similar and needs no further explanations. The template's structure is the following [1]:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} a & a & a \\ a & 0 & a \\ a & a & a \end{bmatrix}, \quad I = -\text{threshold}. \tag{15}$$



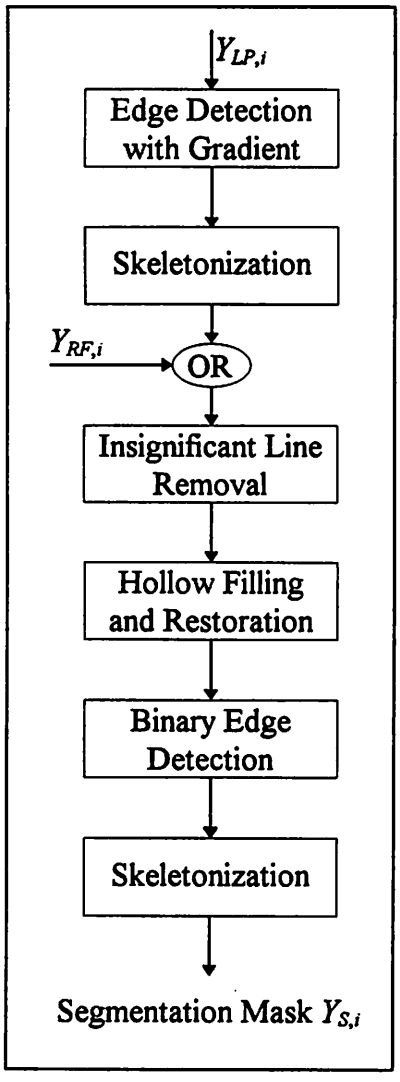


Figure 11. *Intraframe segmentation* uses the frames' intrinsic information, this is its contours, to detect the shapes of different image regions. *Skeletonization* is an iterated process that stops when all contours are reduced to one pixel thin lines. The algorithm allows a reliable and exact object segmentation. The results of the *Remarkable-Feature Extraction* are then added and the new contour lines are extracted. Finally a last *Skeletonization* takes place and insignificant lines are deleted.

The resulting output map includes all necessary contour pixels, but it is far from being sufficiently exact for segmentation purposes. The contours vary in width due to the different character of the detected edges. Sharp boundary zones have a fairly thin contour line, while smooth edges usually produce thicker lines. This effect can be reduced with a very accurate choice of the gradient threshold, but starting from the assumption of an unsupervised working coding scheme, we suggest another way to have a good result for every kind of edges. The *Skeletonization* proposed in [23] is a very powerful line thinning algorithm whose symmetrical approach reduces every type of line to its center pixels. This is ideal for our purposes because the contour lines produced by the gradient operation are widened on both sides almost identically. Our simulations proved that this assumption is valid for most cases and leads to very accurate segmentation results. *Skeletonization* is an circularly used set of 8 cloning templates, this means that the output of one operation is fed back to the input of the next one. This is repeated until the result remains constant for one complete loop. This interruption criterion is easy to detect by processing an 'XOR' between the most recent skeleton and the skeleton after the last complete cycle and using a global "all white" check.

$$\mathbf{A}=\begin{bmatrix}0&0&0\\0&3&0\\0&0&0\end{bmatrix},\quad \mathbf{B}_1=\begin{bmatrix}0.25&0.25&0\\0.25&-0.25&-0.25\\0&-0.25&0\end{bmatrix},\quad I=-0.75, \tag{16}$$

$$\mathbf{A}=\begin{bmatrix}0&0&0\\0&3&0\\0&0&0\end{bmatrix},\quad \mathbf{B}_2=\begin{bmatrix}0.25&0.25&0.25\\0&-0.25&0\\-0.25&-0.25&0\end{bmatrix},\quad I=-0.75, \tag{17}$$

$$\mathbf{A}=\begin{bmatrix}0&0&0\\0&3&0\\0&0&0\end{bmatrix},\quad \mathbf{B}_3=\begin{bmatrix}0&0.25&0.25\\-0.25&-0.25&0.25\\0&-0.25&0\end{bmatrix},\quad I=-0.75, \tag{18}$$

$$\mathbf{A}=\begin{bmatrix}0&0&0\\0&3&0\\0&0&0\end{bmatrix},\quad \mathbf{B}_4=\begin{bmatrix}-0.25&0&0.25\\-0.25&-0.25&0.25\\0&0&0.25\end{bmatrix},\quad I=-0.75, \tag{19}$$

$$\mathbf{A}=\begin{bmatrix}0&0&0\\0&3&0\\0&0&0\end{bmatrix},\quad \mathbf{B}_5=\begin{bmatrix}0&-0.25&0\\-0.25&-0.25&0.25\\0&0.25&0.25\end{bmatrix},\quad I=-0.75, \tag{20}$$

$$\mathbf{A}=\begin{bmatrix}0&0&0\\0&3&0\\0&0&0\end{bmatrix},\quad \mathbf{B}_6=\begin{bmatrix}0&-0.25&-0.25\\0&-0.25&0\\0.25&0.25&0.25\end{bmatrix},\quad I=-0.75, \tag{21}$$

$$\mathbf{A}=\begin{bmatrix}0&0&0\\0&3&0\\0&0&0\end{bmatrix},\quad \mathbf{B}_7=\begin{bmatrix}0&-0.25&0\\0.25&-0.25&-0.25\\0.25&0.25&0\end{bmatrix},\quad I=-0.75, \tag{22}$$

$$\mathbf{A}=\begin{bmatrix}0&0&0\\0&3&0\\0&0&0\end{bmatrix},\quad \mathbf{B}_8=\begin{bmatrix}0.25&0&0\\0.25&-0.25&-0.25\\0.25&0&-0.25\end{bmatrix},\quad I=-0.75. \tag{23}$$

The resulting skeleton is then merged with the *Remarkable-Feature Mask* $Y_{RF,i}$. Merging means in our case a logic 'OR' of both binary frames. The next operation deletes those contour lines that do not belong to a specific object. As an *object* is defined by the presence of its *closed* contour line, this *Insignificant Line Removal* removes all contour elements that do not belong to an object, i.e. open contour lines. Therefore it simplifies the image by turning all black pixels into white that do not have at least two neighbors. The

operation needs the merged image to be sent to both input and initial state of the CNN. Its computation time depends on the length of connected pixels of the longest broken contour.

$$\mathbf{A}=\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix},\quad \mathbf{B}=\begin{bmatrix}0 & 0 & 0\\ 0 & 5 & 0\\ 0 & 0 & 0\end{bmatrix},\quad I=-1. \tag{24}$$

This is followed by the execution of a finite running-time *Hollow Filling* template (11) followed by a *Restoration* template (14) in order to fill very small objects and merge them with larger neighbors. The product of this operation shows large black fragments whose contours have to be extracted. In the case of a binary image, this task is computed by the *Binary Edge Detection* template [1]. The input mask is here fed to both CNN input and state.

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix},\quad \mathbf{B}=\begin{bmatrix}-0.25 & -0.25 & -0.25\\ -0.25 & 2 & -0.25\\ -0.25 & -0.25 & -0.25\end{bmatrix},\quad I=-1.4. \tag{25}$$

A second *Skeletonization* erases finally those pixels that are redundant for the new contour description. Contrary to the previous application of this block, we need now only one iteration to receive a steady state.

The output of this process is the complete segmentation into distinct objects starting from the original image frame. The *Segmentation Mask* $Y_{S,i}$ contains all objects in the form of closed contour lines. Now it merely remains to define the moving ones among all objects.

II.2.6 INTERFRAME SEGMENTATION AND OBJECT LABELING

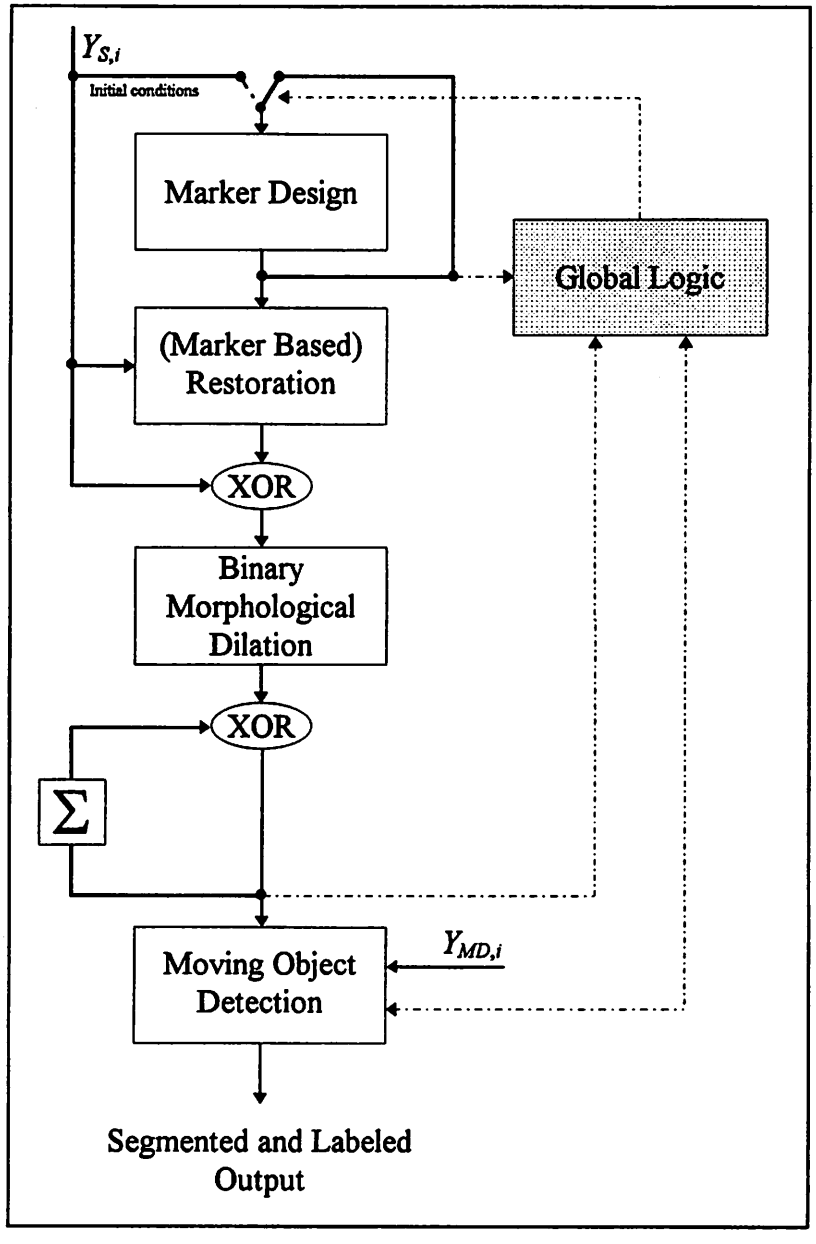


Figure 12. The *Interframe Segmentation and Object Labeling* operation sorts the fragments detected by the *Intraframe Segmentation* by size and transmits only those, which are also part of the motion detection mask $Y_{MD,i}$. This algorithm is based on *markers*, which are areas pointing out the location of segmented image objects. An external logic controls the data flow, such as switching the input of the *Marker Design* block after the initial conditions have been downloaded.

The main ideas of this concluding block of our segmentation algorithm are *marker based object reconstruction, labeling by size* and *detection of moving objects*. These functions figure also in the form of distinct blocks in the flow chart of the whole algorithm which is

shown in Figure 12. Here it has to be stressed that labeling by size is chosen to sort the outgoing data stream. This is of course an entirely random choice and certainly not part of the upcoming coding standard. A *marker* of a given object is defined as a black patch in the binary image space whose surface covers the entire area of the marked object, as it is explained in Figure 13 for an ensemble of two objects and a marker patch.

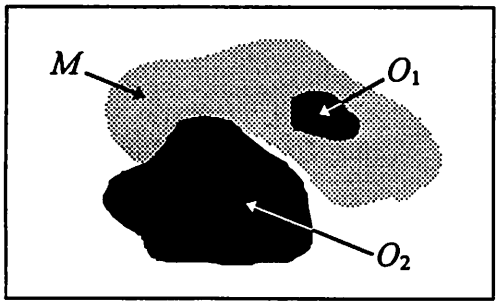


Figure 13. The marker M (for presentation purposes given in gray) marks only object O_1 but not object O_2. This results from the simple observation that O_1 is completely contained in M, while parts of O_2 do not coincide with M. In terms of mathematical set theory this means that $O_1 \in M$, but $O_2 \notin M$.

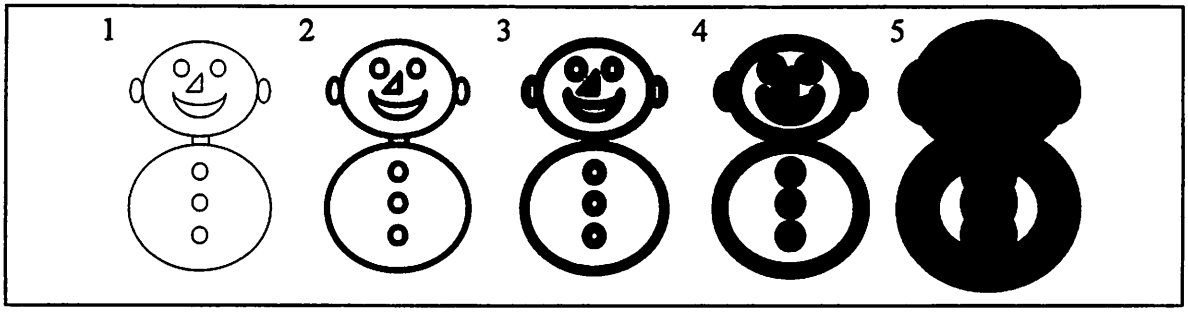


Figure 14. This sequence of 5 pictures explains in a graphical way how the *Marker Design* operation works. Picture 1 depicts the original image with its skeletonized contour lines. In picture 2 these lines are gradually widened, but none of the given closed contours is already filled up with black color. Therefore none of the objects is marked. In picture 3 the nose and the neck of our cartoon person are entirely filled with black color thus they are marked. Another line expanding operation adds the ears, the eyes, the mouth and the buttons to the set of marked objects. This is presented in picture 4. Finally, the whole head is marked in picture 5.

Additionally, a valid marker is not allowed to have holes or cracks, these are spots of white pixels inside the outer boundary line. Figure 14 sketches the case of a skeletonized image featuring several objects to be marked. Generally, the markers are produced by gradually expanding the existing contour lines. The resulting contours cover more and more parts of the image, until the whole set of objects is *marked*. The first input to the *Marker Design* operation is therefore the original skeleton $Y_{S,i}$, which is also given to the state of the CNN. The template has the following form [1]:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = 4.5. \tag{26}$$

After the transient has run for about 20τ, which we defined as the usual running length for one stage, it is stopped and the result is send to the next operation. In a parallel path of Figure 14 the result is fed back to the input for the next stage of this labeling by size. This step is repeated until no new objects are extracted. The *Global Logic* device verifies this condition and stops the computation. A simple 'XOR' between two frames can detect changed pixels and an external microprocessor automatically controls the data flow. Returning to the main path, the result of the previous operation is compared to the original skeleton $Y_{S,i}$. If the marker marks a region with one or several closed contours, then these objects will now be filled with black color. This step is referred as *Marker Based Restoration* and utilizes the same template as it is presented in (14). The marker mask is here fed to the initial state and the *Segmentation Mask* $Y_{S,i}$ goes to the input. The computational expenses are slightly higher than used for the marking operation. They depend in a very sensitive way on the scene that has to be segmented.

When the transient reaches the steady state, a logic 'XOR' separates the contour lines from the result. This segmentation mask features black objects separated by white contour lines, which do not contribute to any of its surrounding objects and remain therefore areas of uncertainty. A good example for such uncertainty areas can be considered in picture 5a of Fig 15. An easy way to segment these areas, too, is to allocate them to the most recently marked objects that are touching them. This is processed by a *Binary Morphological Dilation* with a cross-shaped **B** template (27) filling the (now white) areas that the contour lines covered before.

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix},\quad \mathbf{B}=\begin{bmatrix}0 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 0\end{bmatrix},\quad I=5.5\,. \tag{27}$$

It is easy to understand that with every larger marker mask, the already detected and segmented objects are once again segmented. The actual labeling by size is then realized by a simple logic operation. The segmentation results of the previous stages are summed up, which corresponds here to the logic 'OR', and compared to the latest segmentation using another 'XOR'. So, only the new segmented objects remain in the mask. These operations are illustrated in Figure 15 for the case of the previously used cartoon person. The pictures with emboldened numbers are those that satisfy the given size criterion applied at a certain labeling stage. The normal numbers indicate that these images are the segmentation result including all objects, even those that were already segmented during an earlier stage of the algorithm. It is easy to observe how the logic 'OR' extracts the new emerging objects. Finally one can see the importance of an additional dilation when considering picture 4 of Figure 15. The contour lines have also to be allocated to an object in order to be segmented, too. The dilation with the cross-shaped structuring elements does exactly this job [28].

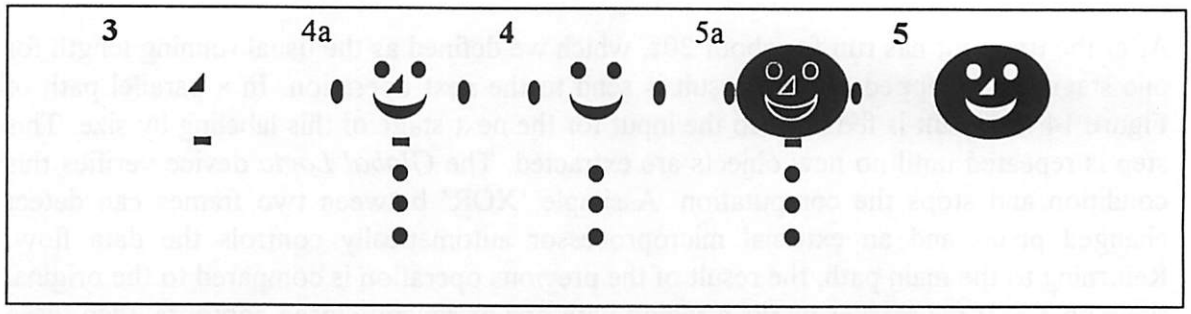


Figure 15. This sequence demonstrates final and intermediate results of the *Marker Based Object Restoration*. Picture **3** shows the first segmentation result which is directly correlated to picture 3 of Figure 14. There the nose and the neck of the cartoon person were marked for restoration. Picture 4a depicts all marked and restored objects of picture 4 of Figure 14. A logic 'OR' with the previously segmented objects (picture **3**) selects now the new objects that are subject to segmentation at this stage. The resulting segmentation mask is given in picture **4**. The use of the same operations produces then the pictures 5a and **5**, related to picture 5 of Figure 14.

Every object of every segmentation mask is finally checked whether it belongs to the *Motion Detection Mask* $Y_{MD,i}$ or to the unchanging portions of the scene. This *Moving Object Detection* template (28) requires that $Y_{MD,i}$ is fed to the state while the actual segmentation mask is applied to the input. Its processing time depends only on the object's size and shape.

$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = -0.5 . \tag{28}$$

Here solely those objects that show at least in some parts changes due to motion are subject to further coding. All other objects, i.e. still objects, are wiped out of the segmentation mask. This result is then the data base for the remaining parts of the image analysis system. After the segmentation of all moving objects of an image is finished, a last comparison, a logic 'XOR', between the mask showing all moving objects and the *Motion Detection Mask* $Y_{MD,i}$ detects those parts of $Y_{MD,i}$ that are not yet segmented. The result is given as an additional segmentation mask and concludes the *Interframe Segmentation*. A good visual example for such a case is given in the simulations done with the *Miss America* video sequence (see Figure 40). All parts of the image that are not segmented are contributed to the still background and remain uncoded while the moving parts are send to the *Parameter Estimation* module.

III SIMULATIONS

III.1 INTRODUCTION

The universality of the proposed analogic programs is proved by segmenting two standardized image sequences, *Claire* and *Miss America*. These sequences have been selected by international expert groups (ITU Study Group XV) as typical videophone sequences showing head and shoulder in motion. The sequence *Claire* has been provided by the *Centre National d'Etudes des Télécommunications* (CNET), France, while the sequence *Miss America* was designed by the *British Telecom Research Laboratories* (BTRL). These test sequences exist in different formats for different application aspects. In these experiments, the *Common Intermediate Format* (CIF) will be used which has been accepted for videophone applications by the ITU. Each picture consists of 288 lines and 352 pixels per line for the luminance component and of 144 lines and 176 pixels per line for both chrominance components. Each sample is quantized with 8 bits, and the frame rate of the sequences is 29.97 Hz.

Fig. 16. Typical frame of the test sequence *Claire*.

Fig. 17. Typical frame of the test seq. *Miss America*.

III.2 SIMULATION RESULTS

We show here the images we obtained after having tested the proposed algorithms. The images are given in the same order as it was already used when describing the analogic programs in greater detail. Additionally we refer at every stage to the notations introduced in Figure 2.

Skipping the *Image Preprocessing* operation, which is of minor interest at this time, we start the discussion of the simulation results with *Edge Enhancing Low Pass Filtering*. Both sets of images show first the results after a convolution with a simple low pass kernel as it is given in (5). The noise removal is efficient, but the quality of the images' contour lines is poor. Therefore we add in $Y_{LP,i}$ formerly removed high frequency

components to the contour areas and reach a significant improvement of the images' sharpness while maintaining the low background noise level.

Figure 18. Y_i after simple LP filtering.

Figure 19. $Y_{LP,i}$, output of Edge Enhancing LP Filter.

Figure 20. Y_i after simple LP filtering.

Figure 21. $Y_{LP,i}$, output of Edge Enhancing LP Filter.

The next operation depicts all moving parts of the image on a pixelwise base. These are the previously mentioned *Motion Detection Masks* $Y_{MD,i}$. Their information is stored and exploited later in this algorithm.

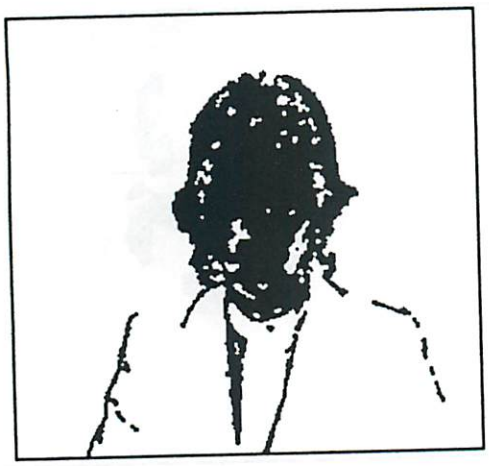


Figure 22. *Motion Detection Mask* $Y_{MD,i}$.

Figure 23. *Motion Detection Mask* $Y_{MD,i}$.

In a parallel path to the *Motion Detection* operation, the *Remarkable-Feature Extraction* is processed. After an initial thresholding of the absolute difference of two directly succeeding image frames, the next step extracts image fragments matching a specific size criterion. With an adequate set of running-time and threshold parameters, facial areas can be successfully extracted. This enables the *Parameter Coding* algorithm to distinguish between objects of predictable motion and those, as the remarkable features, that have to be coded independently. The images in our presentation show for both simulations the thresholded absolute difference image and the *Remarkable-Feature Mask* $Y_{RF,i}$.

Figure 24. Thresholded abs. $Y_{LP,i}$ - $Y_{LP,i-1}$ diff..

Figure 25. *Remarkable-Feature Mask* $Y_{RF,i}$.

Figure 26. Thresholded abs. $Y_{LP,i}$ - $Y_{LP,i-1}$ diff..

Figure 27. *Remarkable-Feature Mask* $Y_{RF,i}$.

The *Intraframe Segmentation* intends to decompose the image into a set of objects. Therefore it calculates first a contour map of the image by means of a *Gradient* template. The irregular contour lines are then thinned to a one pixel wide skeleton. The resulting output is subsequently merged with the *Remarkable-Feature Mask*. Insignificant lines, i.e. lines not belonging to a closed contour, are removed and an additional *Hollow Filling* and *Restoration* decreases the number of objects by merging small objects to larger neighbors. The final *Segmentation Mask* $Y_{S,i}$ results then from a simple *Binary Edge Detection* operation followed by one iteration of the *Skeletonization*.

Figure 28. Thresholded *Gradient* of $Y_{LP,i}$.

Figure 29. Skeleton after first *Skeletonization*.

Figure 30. Inclusion of $Y_{RF,t}$.

Figure 31. *Line Removal*, object merging and *Skeletonization* yield the *Segm. Mask* $Y_{S,t}$.

Figure 32. Thresholded *Gradient* of $Y_{LP,t}$.

Figure 33. Skeleton after first *Skeletonization*.

Figure 34. Inclusion of $Y_{RF,t}$.

Figure 35. *Line Removal*, object merging and *Skeletonization* yield the *Segm. Mask* $Y_{S,t}$.

The final *Interframe Segmentation and Object Labeling* separates still image parts from moving objects and sends the moving segments to the next unit, the *Parameter Estimation*. This data stream is labeled by means of a size criterion. After the last moving object was detected, the algorithm verifies if all information given in the *Motion Detection Mask* $Y_{RF,i}$ has already been allocated to image objects. If this check fails, an additional mask will send the missing image portions to the *Parameter Estimation*. A good example for such a mask is displayed in Figure 40.

Figure 36. Moving objects.

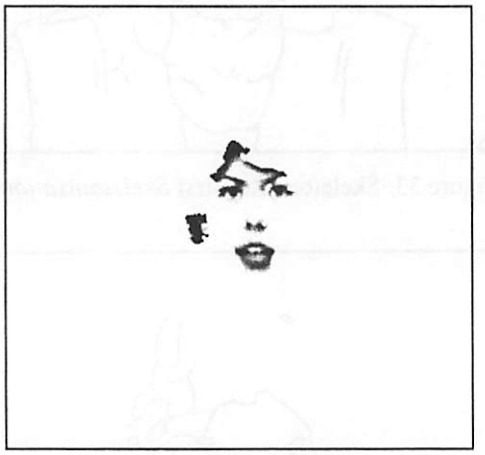


Figure 37. Fast moving objects (see $Y_{RF,i}$).

Figure 38. Still background.

Figure 39. Moving objects.

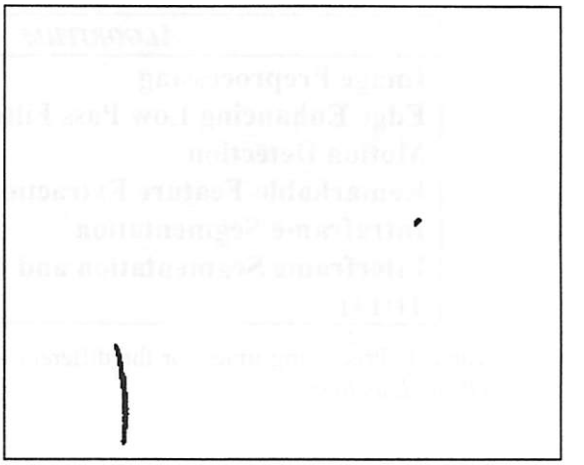


Fig. 40. Additional segmentation mask featuring the remaining, not yet considered parts of $Y_{MD,i}$ (see Fig. 23).

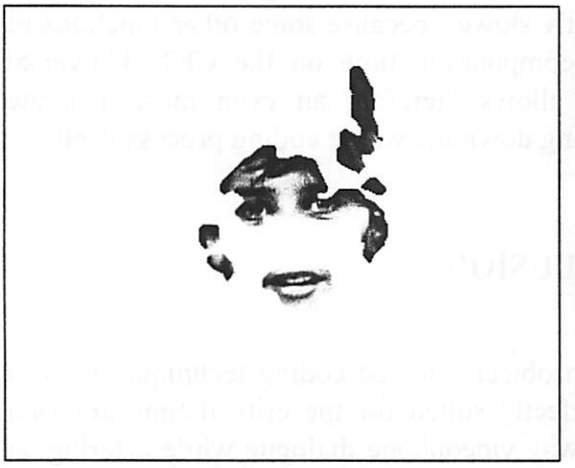


Figure 41. Fast moving objects (see $Y_{RF,i}$).

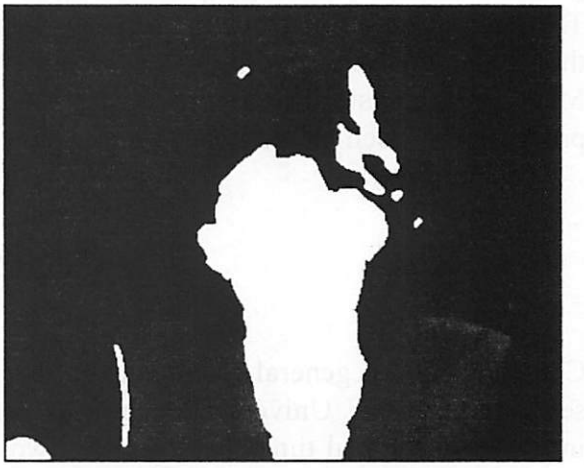


Figure 42. Still background.

IV IMPLEMENTATIONS

The complete execution time of the *Segmentation and Object Labeling* algorithm depends on the image size and the complexity of the given scene. The following survey features all estimated processing times of the different algorithms without I/O operations:

ALGORITHM	*TIME IN τ*
Image Preprocessing	3
Edge Enhancing Low Pass Filtering	50
Motion Detection	<15
Remarkable-Feature Extraction	<150
Intraframe Segmentation	<300
Interframe Segmentation and Object Labeling	<600
TOTAL	<1118

Table 1. Processing times for the different algorithms computing the *Segmentation and Object Labeling.*

For a usual CIF frame of 360×288 pixels computed on a recent design of the CNN Universal Chip, the usual time constant τ is equal to about 250 nsec [22]. The results given in Table 1 show therefore that up to 300 frames could be segmented per second. The real working frequency is probably slightly slower, because some other functions of the image analysis algorithm also require computation time on the CNN Universal Machine. This surplus in processing time allows therefore an even more accurate processing of each image frame without slowing down the whole coding process itself.

V CONCLUSION

Compared to the general requirements for an object oriented coding technique, we can see that the CNN Universal Machine is perfectly suited for the critical time criterion imposed by the real time character of a two way videophone dialogue while offering, at the same time, a reliable and exact image segmentation. The CNN Universal Machine is therefore able to adapt itself to every kind of transmission situation, depending on the overall goal of extremely low transmission rates or a high output quality. The first target is achievable due to the very accurate description of moving objects, allowing an exact distinction between predictable and unpredictable moving objects and still background. If on the other hand a high picture quality is demanded, it is possible to segment a video sequence into moving objects while maintaining the transmission rate of 30 frames/sec provided by the input sequence. This figure exceeds significantly the results reported for other related segmentation schemes [9][12], commonly being limited to transmission rates below 10 frames/sec. Thus we are able to conclude that the CNN Universal Chip is an ideal hardware component to compute the object oriented image analysis in very low bitrate video coding systems.

VI ACKNOWLEDGMENTS

The authors would like to thank K.R. Crounse, T. Kozek, and Á. Zarándy for their great interest and many helpful discussions. These acknowledgments also include Professor T.G. Noll, director of the Institut für Allgemeine Elektrotechnik und DV-Systeme at the RWTH Aachen, University of Technology, for his great support of this project.

Finally, we would like to acknowledge the financial support provided by the *Otto-Junker-Stiftung*, Aachen, and the *Konrad-Adenauer-Stiftung*, Sankt Augustin, Germany.

VII REFERENCES

[1] "Analogic CNN Program Library", DNS-7-1995, T. Roska and L. Kék, editors, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, April 1995

[2] C. Cafforio and F. Rocca, "The Differential Method for Image Motion Estimation", *Image Sequence Processing and Dynamic Scene Analysis*, T.S. Huang, ed., Springer Verlag, 1983, pp. 104-124

[3] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory", *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, October 1988, pp. 1257-1272

[4] L.O. Chua and L. Yang, "Cellular Neural Networks: Applications", *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, October 1988, pp. 1273-1290

[5] L.O. Chua and T. Roska, "The CNN Paradigm", *IEEE Transactions on Circuits and Systems - I*, vol. 40, no. 3, March 1993, pp. 147-156

[6] L.O. Chua, T. Roska, T. Kozek, and Á. Zarándy, "The CNN Paradigm - A Short Tutorial", *Cellular Neural Networks*, T. Roska and J. Vandewalle, editors, John Wiley & Sons, New York, 1993, pp. 1-14

[7] K.R. Crounse and L.O. Chua, "Methods for Image Processing and Pattern Formation in Cellular Neural Networks: A Tutorial", *IEEE Transactions on Circuits and Systems - I*, submitted for Special Issue on Nonlinear Waves, Patterns and Spatio-Temporal Chaos in Dynamic Arrays, vol. 42, no. 10, October 1995

[8] J.M. Cruz and L.O. Chua, "A Fast, Complex and Efficient Test Implementation of the CNN Universal Machine", *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and their Applications CNNA-94*, Rome, December 1994, pp. 61-66

[9] N. Diehl, "Object-Oriented Motion Estimation and Segmentation in Image Sequences", *Signal Processing: Image Communication*, 3, 1991, pp. 23-56

[10] C.R. Giardina and E.R. Dougherty, *Morphological Methods in Image Processing and Signal Processing*, Prentice Hall, Englewood Cliffs (NJ), 1988

[11] M. Hötter, "Optimization and Efficiency of an Object-Oriented Analysis-Synthesis Coder", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 2, April 1994, pp. 181-194

[12] M. Hötter and R. Thoma, "Image Segmentation Based on Object Oriented Mapping Parameter Estimation", *Signal Processing*, 15, 1988, pp. 315-334

[13] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second Generation Image Coding Techniques", *Proceedings of the IEEE*, vol. 73, no. 4, April 1985, pp. 549-575

[14] H.G. Musmann, M. Hötter, and J. Ostermann, "Object-Oriented Analysis-Synthesis Coding of Moving Images", *Signal Processing: Image Communication*, 1, 1989, pp. 117-138

[15] J. Ostermann, "Object-Oriented Analysis-Synthesis Coding Based on the Source Model of Moving Rigid 3D Objects", *Signal Processing: Image Communication*, 6, 1994, pp. 143-161

[16] P. Perona and J. Malik, "Scale Space and Edge Detection Using Anisotropic Diffusion", *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, 1987

[17] T. Roska, T. Boros, A. Radványi, P. Thiran, and L.O. Chua, "Detecting Moving and Standing Objects Using Cellular Neural Networks", *International Journal of Circuit Theory and Applications*, vol. 20, no. 5, September-October 1992, pp. 613-628

[18] T. Roska, T. Boros, P. Thiran, and L.O. Chua, "Detecting Simple Motion Using Cellular Neural Networks", *Proceedings of the IEEE International Workshop on Cellular Neural Networks and their Applications CNNA-90*, Budapest, December 1990, pp. 127-138

[19] T. Roska and L.O. Chua, "The CNN Universal Machine: An Analogic Array Computer", *IEEE Transactions on Circuits and Systems - II*, vol. 40, March 1993, pp. 163-173

[20] T. Roska, Á. Zarándy, and L.O. Chua, "Color Image Processing Using Multi-Layer CNN Structure", *Circuit Theory and Design 93*, H. Didier, ed., Elsevier, Amsterdam, 1993

[21] P. Salembier, L. Torres, F. Meyer, and C. Gu, "Region-Based Video Coding Using Mathematical Morphology", *Proceedings of the IEEE*, vol. 83, no. 6, June 1995, pp. 843-857

[22] A. Stoffels, "Object Oriented Image Analysis Techniques Using the CNN Universal Machine", MS Thesis, RWTH Aachen, University of Technology, December 1995

[23] P.L. Venetianer, F. Werblin, T. Roska, and L.O. Chua, "Analogic CNN Algorithms for Some Image Compression and Restoration Tasks", *IEEE Transactions on Circuits and Systems - I*, vol. 42, no. 5, May 1995

[24] P.L. Venetianer and T. Roska, "Image Compression by CNN", DNS-13-1995, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, August 1995

[25] T. Yang, "Application of Cellular Neural Network to Facial Expressions Animation and High-Level Image Processing", *IEEE Transactions on Circuits and Systems - I*, vol. 42, submitted for Special Issue on Nonlinear Waves, Patterns and Spatio-Temporal Chaos in Dynamic Arrays, no. 10, October 1995

[26] Á. Zarándy, F. Werblin, T. Roska, and L.O. Chua, "Novel Types of Analogic CNN Algorithms for Recognizing Banknotes", *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and their Applications CNNA-94*, Rome, December 1994, pp. 273-278

[27] Á. Zarándy, F. Werblin, T. Roska, and L.O. Chua, "Spatial Logic Algorithms Using Basic Morphological Analogic CNN Operations", *International Journal of Circuit Theory and Applications*, accepted for publication, 1995

[28] Á. Zarándy, A. Stoffels, T. Roska, and L.O. Chua, "Mathematical Binary and Gray-Scale Morphology on CNN", (manuscript, progress report), Electronics Research Laboratory, University of California, Berkeley, October 1995