NEW RESULTS AND MEASUREMENTS RELATED TO DYNAMIC IMAGE CODING USING CNN UNIVERSAL CHIPS

by

T. Kozek, C. W. Wu, A. Zarándy, H. Chen, T. Roska,
M. Kunt, and L. O. Chua

Memorandum No. UCB/ERL M96/58

8 October 1996

NEW RESULTS AND MEASUREMENTS RELATED TO DYNAMIC IMAGE CODING USING CNN UNIVERSAL CHIPS

by

T. Kozek, C. W. Wu, Z. Zarándy, H. Chen, T. Roska,
M. Kunt, and L. O. Chua

Memorandum No. UCB/ERL M96/58

24 April 1989

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

NEW RESULTS AND MEASUREMENTS RELATED TO DYNAMIC IMAGE CODING USING CNN UNIVERSAL CHIPS

T. Kozek*, C. W. Wu, Á. Zarándy*, H. Chen, T. Roska*, M. Kunt**, and L. O. Chua

University of California, Berkeley
also with the
*Computer and Automation Inst., Hungarian Academy of Sciences
and the
**Swiss Federal Institute of Technology, Lausanne

Abstract

Cellular neural networks are considered here for efficient implementation of the most computationally intensive steps of dynamic image coding. Several analogic CNN algorithms are presented for the generation of binary image masks and image decomposition. Measurement results for the first CNN universal chips executing an analogic algorithm for size classification are also presented. Based on measured execution times, the viability of the CNN implementation of efficient but computationally expensive compression algorithms such as dynamic image coding is assessed.

1. INTRODUCTION

Second generation video coding techniques [1] provide for sophisticated and efficient methods in low bit-rate video coding. Even their dynamic frame-by-frame optimization has been developed [2]. Using these techniques, high quality video can be transmitted over regular phone lines (16-64kb/s). Current development of the MPEG4 standards suggests the use of these techniques, as well. The underlying algorithms, however, require enormous computing power for the complex analysis of images. These analyses yield also several masks which identify qualitatively different regions of consecutive image frames. In order to apply second generation techniques in hand-held and portable devices, new, high-performance chips need to be used which can execute close to Tera (10^{12}) image analysis operations per second. The CNN Universal Machine (CNNUM) architecture [3,4,5] and its VLSI implementations (CNN universal chips) [12,13] are capable of delivering the required computational performance in a chip-set architecture [17], where handling of larger images and transparent interface to sensors and to digital environment is provided.

We are considering the CNN universal machine architecture as a new computing platform for dynamic image coding. CNN universal chips and chip-sets can be used as coprocessors for high-speed manipulation of 2-D analog signal arrays, i.e. gray-scale or color images. On a CNN chip, analog processors are arranged on a rectangular grid and perform operations defined by a translation invariant local interconnection pattern, called cloning template (or synaptic law). This template, consisting of a feedforward convolution

matrix (*B*-template) which computes a waited sum of the input values within the neighborhood, a feedback convolution matrix (*A*-template) which acts similar to an IIR filter, and a scalar bias term (*I*), is the basic instruction in the CNN computer. The CNN universal machine architecture facilitates quick reprogramming of templates to allow for stored-program operation of the analog processor array. Such instructions, plus local and global logic are combined on the universal chip to form analogic CNN algorithms. In recent months the first experimental CNN universal chips have appeared. Our measurements confirm the trillion operations per second computing speed.

In [9] it has been shown that some complex image segmentation techniques can be carried out via CNN algorithms. In this paper, we present results concerning the implementation of many computationally intensive steps of segmentation and coding algorithms via CNN technology. For the first time, we measured the execution of some algorithms on operational CNN universal chips. We also show how morphology operators for video coding can be implemented using linear templates only. These templates can be implemented on upcoming CNN universal chips and their algorithmic combination leads to implementation of complex systems on a single chip-set. These results prepare the way to develop dynamic image coding systems [2] incorporating CNN chip-sets and digital signal processing.

In Section 2, the implementation of binary morphological operators and a segmentation algorithm using a 20x22 CNN universal chip is presented. CNN implementation of two other important morphological operators, reconstruction and marker extraction is summarized in Section 3. Section 4 shows how some nonlinear CNN algorithms for object oriented image segmentation can be implemented using linear CNN templates only. In Section 5, the CNN implementation of a perceptual three-component image decomposition scheme is presented. Finally, in Section 6, we show how these techniques can be combined to solve a complex task.

2. IMPLEMENTATION OF A SIZE CLASSIFICATION ALGORITHM USING BINARY MATHEMATICAL MORPHOLOGY ON A 20×22 CNN UNIVERSAL CHIP

It was shown in [1] that both intra- and inter-frame image segmentation can be done by using morphological operators. The problem with the morphological tools is that they have high computational demand which is hardly met by any digital machine in hand-held, on-line, video coding applications. It was shown in [8], that the basic operators of the mathematical morphology (erosion, dilation) can be effectively implemented on the CNN Universal Machine (CNNUM). In this section, we demonstrate the viability of this approach by presenting measurement results obtained from a VLSI implementation of a 20x22 CNN Universal Chip [12]. Later we show, that not only the basic, but even a more complex morphological operator, namely the reconstruction operation, can be implemented on the CNNUM. Due to its high computational power, the CNNUM can be a "morphology engine", which can execute the morphological operators in real-time.

During the test phase of the 20x22 CNN Universal Machine (CNNUM), we implemented several image manipulation operators (templates and template sequences). Here we show a complex morphological operator for size classification. There are two key-points in the algorithm:

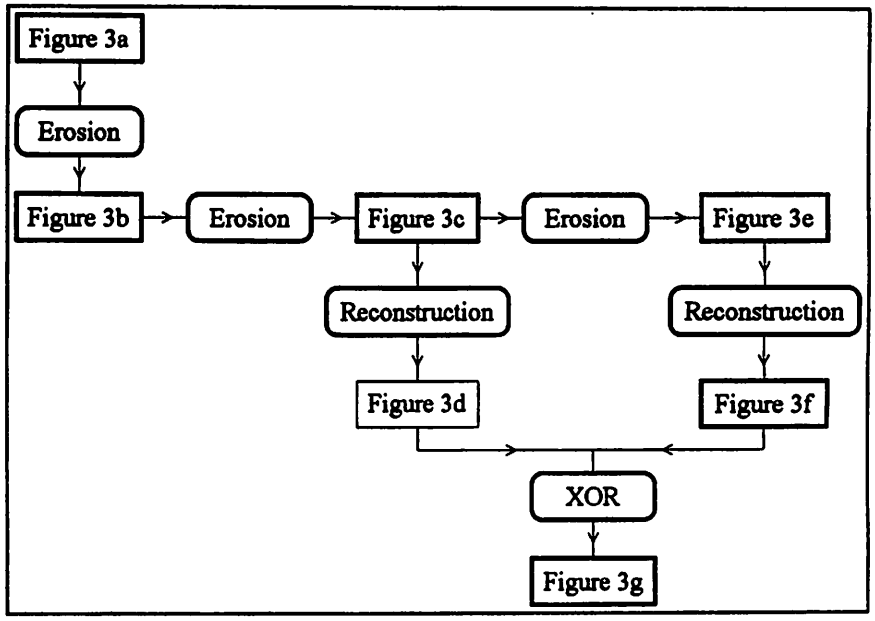
- The algorithm fully exploits the capabilities of the CNNUM chip: uses several templates, employs internal storage for intermediate results, and uses the internal logic of the processor array. Input images can either be captured by the optical sensors of the chip, or can be downloaded electrically. Both the templates and the images can be downloaded in advance to the chip, and then processing can be performed on the chip without I/O operations. This provides for extremely high processing speed.
- This example proves that complex image processing tasks can be efficiently implemented on the CNNUM. Some simple operators (dilation, erosion, reconstruction, etc.) can be implemented with a single CNN template, while other more complex tasks (like size classification), can be implemented by using an analogic CNN algorithm.

2.1 Size classification algorithm

The goal of the algorithm is to extract objects larger than a given size (*size A*), but not larger than another given size (*size B*). This is done by the following steps:

1. With ultimate erosion, the markers of those objects which are larger than *size A* are extracted. In our example, this operation was realized by applying the erosion template (5) twice.
2. Using this marker image, we reconstruct those objects, which were larger than *size A*. This operation was realized by using the RECALL template (6).
3. Then, with ultimate erosion, the markers of those objects which are larger than *size B* are extracted. This operation was realized by applying the erosion template (5) once more to the previous marker image.
4. Using this marker image, we reconstruct those objects, which were larger than *size B*. This operation was realized by using the RECALL template (6).
5. We apply XOR operation to the two reconstructed images and we get the objects which are larger than *size A*, but smaller than *size B*.

The flow-chart of the algorithm is as second follows:



The algorithm uses two templates. The first is the *erosion* template. This realizes a binary erosion with a 3x3 plus shaped structuring element set:

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

The second one is the *reconstruction* template, which is the single template implementation of the binary reconstruction operator.

$$A=\begin{bmatrix}0 & 1 & 0\\ 1 & 2 & 1\\ 0 & 1 & 0\end{bmatrix},\ B=\begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix},\ I=4.5 \tag{6}$$

2.1.1 Implementation

We implemented the algorithm on a 20x22 CNN universal chip [12], using our CNN Chip Prototyping System [14]. The results of the implementation are shown in Figure 1. As it can be seen in the figure, the chip solved the problem properly, without any error. In this algorithm, two different types of templates were used. The first one (*erosion* template) uses local information on an image, while the second (*reconstruction* template) uses global information of the image. Hence, the transient time of the first is shorter than the later. The *erosion* template settles in 2 μs, while the *reconstruction* template settles in 12 μs.

As a comparison, we estimated the required number of instructions and the elapsed time for a digital microprocessor when solving the same problem on the same sized image. We got, that a single erosion operation takes about 10,000 assembler instruction, while

the reconstruction needs 120,000. Considering an up-to-date high performance microprocessor (like Pentium-Pro), the erosion would take 100 μs, while the *reconstruction* would take 1200 μs. This means, that the erosion is roughly 50 times faster, while the reconstruction is 100 times faster on this CNN chip. We also have to know, that the silicon area of this test CNN chip is much smaller ($0.3cm^2$) than the silicon area of the microprocessor mentioned above (which is over $3cm^2$), and this chip was fabricated with a conservative (0.8 μs) technology, while the above mentioned processor is fabricated with 0.35 μs) technology.

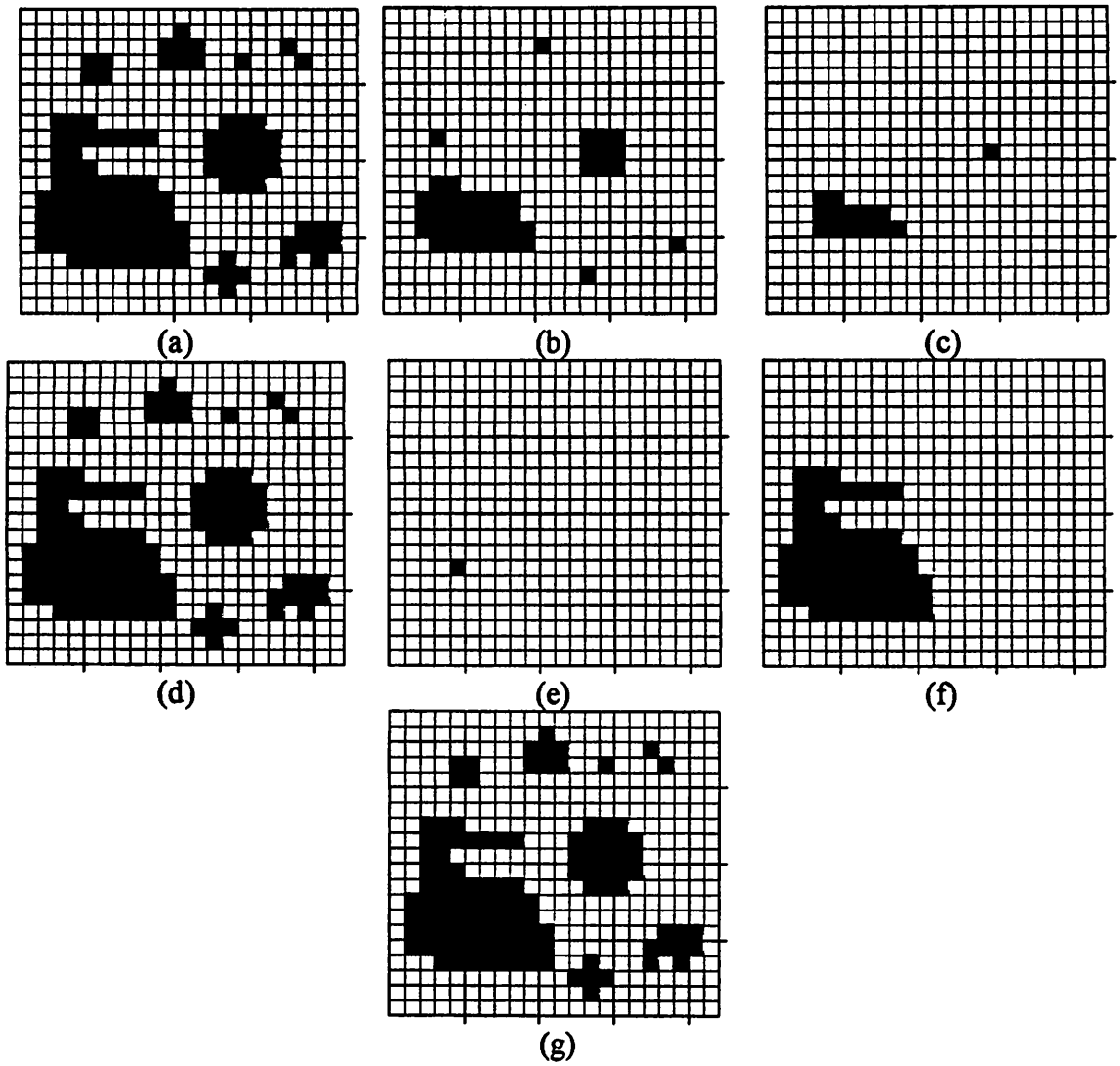


Figure 1. (a) original test image; (b) result of the first erosion operation; (c) result of the erosion operation; (d) reconstruction with (c); (e) result of the third erosion operation; (f) reconstruction with (e); (g) final result: XOR of (d) and (f).

3. MORPHOLOGY BASED IMAGE RECONSTRUCTION AND SEGMENTATION IMPLEMENTED ON THE CNN UNIVERSAL CHIP

In this section we show the major steps of the morphology based image segmentation [1]. These steps are the IMAGE SIMPLIFICATIONS, the MARKER EXTRACTION, and the DECISION. The first step deletes the high frequency parts of the image, but preserves the major edges and segment boundaries. This step prepares the image for the second step. In the second step, we extract the markers, which mark the individual regions. After this step, there are still some pixels in the image, which does not belong to any regions. The last step works on these uncertain regions, and melts each pixel to one of the regions.

3.1 Image simplification

During the image simplification phase

- the secondary non-relevant information is removed;
- and flat zones are produced in the interior of homogenous regions.

Usually the *h-minima*, *h-maxima* operators [1, p. 110] are used for image simplification. These operators are grayscale reconstruction operators, where the mask image (f) is the original image, and the marker image is ($f \pm h$). (h is a constant.) A simple example for the h-maxima operator can be seen in Figure 2. On Figure 2, the *h-maxima* operator is applied to a 1D signal (for example to a line of a grayscale image). The interesting property of these operators that they simplify the image by removing small or poorly contrasted regions without corrupting the contour information. Moreover, they produces flat zones on the interior of the regions, which is very useful in the Marker extraction step (described below). Here, we describe the CNN implementation of the *h-maxima* operator. The *h-minima* operator can be implemented on very similar way.

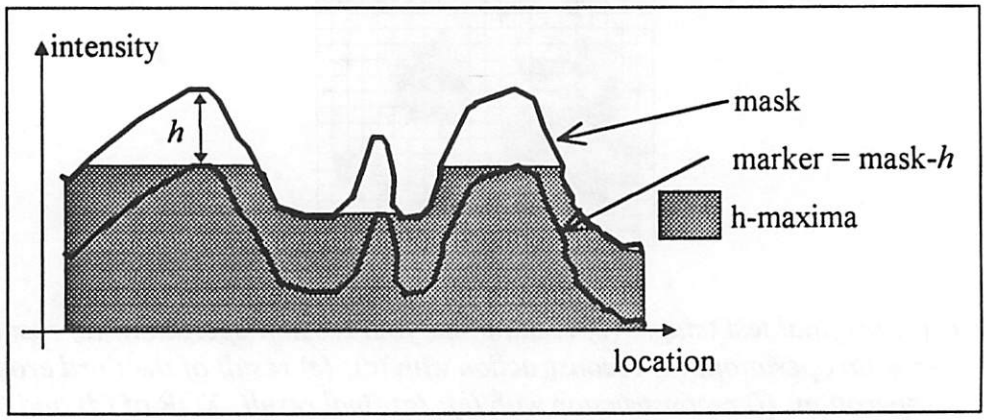


Figure 2. The h-maxima operator for a 1D signal. The h- operator files up the large domes, and creates flat areas under them.

The *h-maxima*, *h-minima* operators can be directly implemented with single CNN templates. The idea is the following. The marker image is loaded to the initial state, and a global maximum template (GLOBMAX in [10]) is applied to it. The state values of the

cells will start growing until they have brighter direct neighbor. The mask is loaded to the input, and a pixel-wise difference of the input and the state is calculated permanently. If the result of the subtraction is negative in a cell, (that is the state larger than the input), a strong negative current source will be activated, and it will stop the state growing.

With other words, the state of the cell (marker) is increasing while it has a larger direct neighbor, and it is not larger than the input (mask). If the mask image is *f*, and the marker image is (*f-h*), than the *h-maxima* operator is implemented. In the (*f-h*) image all pixel values are decreased with *h*. The template is as follows:

$$A=\begin{bmatrix} a & a & a \\ a & 1 & a \\ a & a & a \end{bmatrix},\quad D=[b],\quad I=0. \tag{1}$$

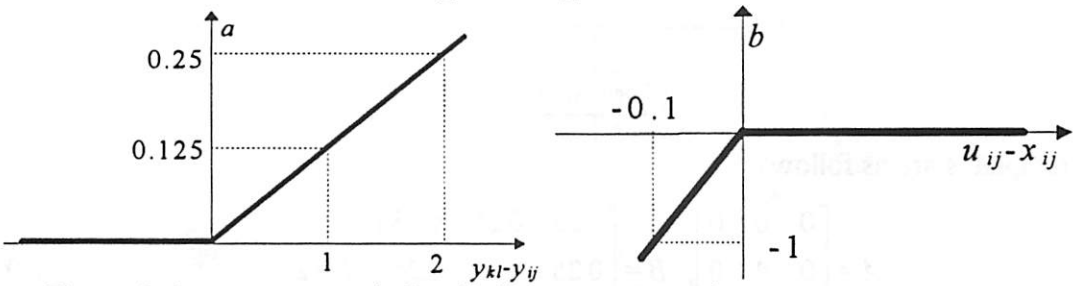


Figure 3 shows an example for the *h-maxima* operation.

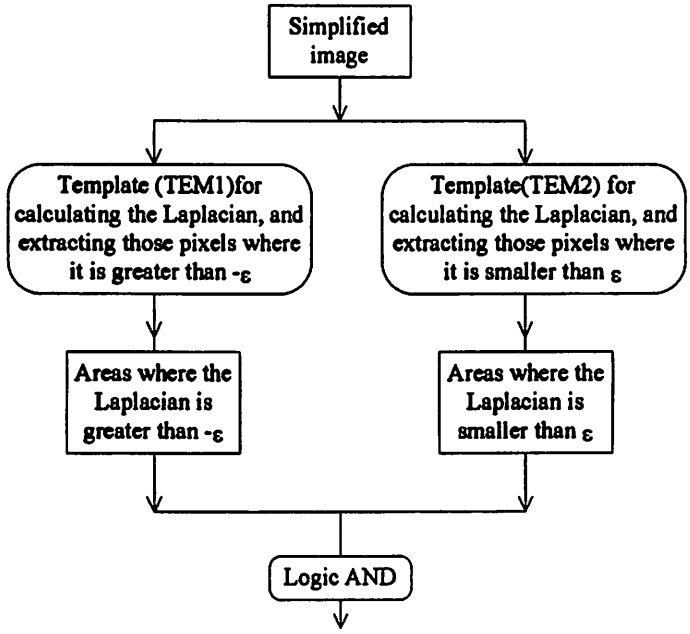
(a) (b) (c)

Figure 3. The h-maxima *operator. (a): the mask image* f, *(b): the marker image (*f-h*), (c) the result (h-maxima).* h *was 0.25, and the values were between -1 and +1.*

3.2 Marker extraction

In the previous section, we used the marker word for identifying one of the argument of the h-maxima function. In that case, the marker meant a grayscale 1D or 2D signal (image). In this section, the marker will mean a black-and-white image, which contains black objects for marking regions.

The goal of this step is to produce markers identifying a major part of the interior of the regions which will be the segments. The simplification process produces flat zones in the interiors of the regions. These flat zones can be extracted by applying a Laplacian operator to the image. We consider an area flat, if the result of the Laplacian operator is around zero. This can be tested with the following algorithm:



The templates are as follows:

TEM1:
$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 4 & 0\\ 0 & 0 & 0\end{bmatrix},\ B=\begin{bmatrix}0.25 & 0.25 & 0.25\\ 0.25 & -2 & 0.25\\ 0.25 & 0.25 & 0.25\end{bmatrix},\ I=\varepsilon \tag{2}$$

TEM2:
$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 4 & 0\\ 0 & 0 & 0\end{bmatrix},\ B=\begin{bmatrix}-0.25 & -0.25 & -0.25\\ -0.25 & 2 & -0.25\\ -0.25 & -0.25 & -0.25\end{bmatrix},\ I=\varepsilon \tag{3}$$

The input and the output of the algorithm can be seen on Figure 4a,b. As it can be seen, there are many single pixel (black) noise can be found in the image. These can be deleted with the filter template:

$$A=\begin{bmatrix}0 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 0\end{bmatrix},\ B=\begin{bmatrix}0.5 & 0.5 & 0.5\\ 0.5 & 4 & 0.5\\ 0.5 & 0.5 & 0.5\end{bmatrix},\ I=-5 \tag{4}$$

The extracted markers can be seen on Figure 4c.

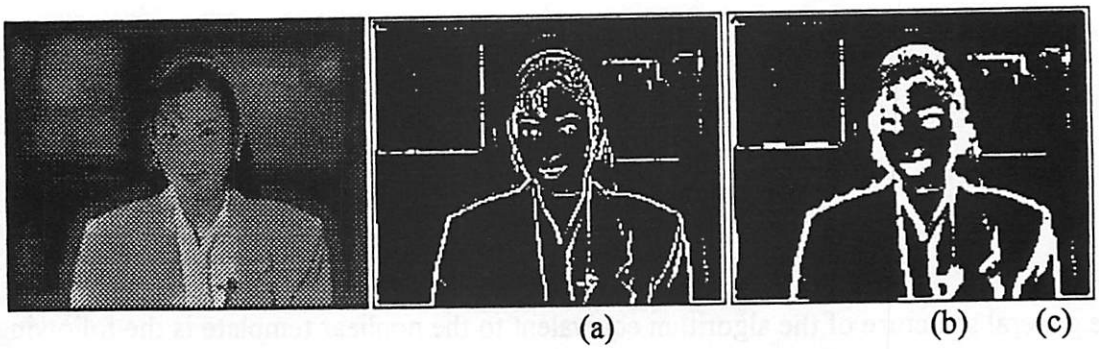


Figure 4. Marker extraction from the simplified image. (a): the simplified image; (b): the flat areas; (c): the filtered flat areas (markers of the segments). In our example, ε was 0.05.

Note: in the first segmentation level only the large segments are separated, and the details will be segmented in the subsequent segmentation levels.

3.3 Decision

In this phase, decision about the uncertainty areas (white areas in Figure 4c), which have not been assigned to any region must be taken. This is also image a local problem, hence a CNN algorithm can provide an effective solution for it.

4. IMPLEMENTING SELECTIVE GRADIENT NONLINEAR TEMPLATES WITH SEQUENCES OF LINEAR ONES

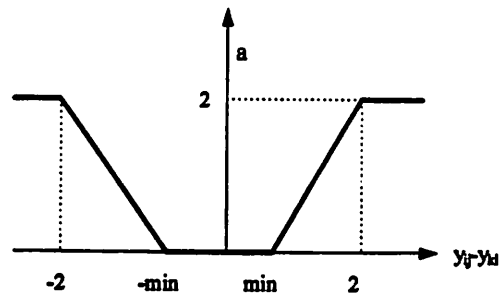
First generation CNN chips apply only linear templates. Fortunately, many of the nonlinear templates used for video coding can be implemented using a sequence of linear templates. Because of the high speed of the CNN, a series of linear template operations will still be well within constraints for real-time operation of video coding algorithms. Here we illustrate how the nonlinear selective gradient template used for detecting contours in an image can be implemented using a series of linear templates. For other templates see a general method in [18].

4.1 Implementation of Selective Gradient using Linear Templates

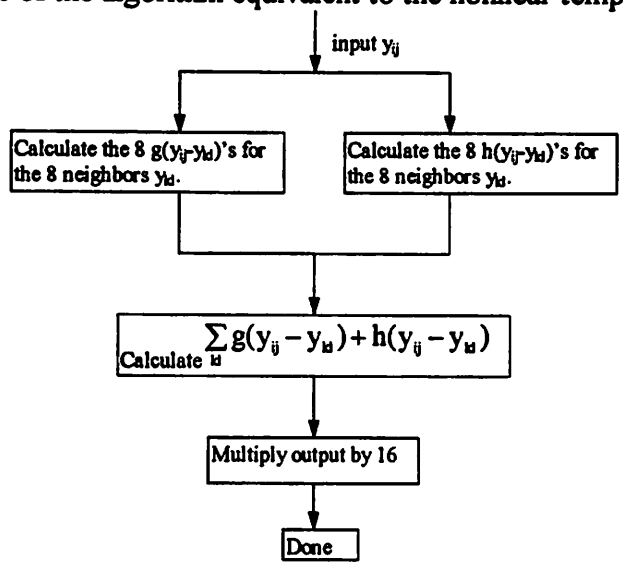
We want to implement the following nonlinear template using linear templates and a space-varying threshold.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} a & a & a \\ a & 0 & a \\ a & a & a \end{bmatrix}, I = -d$$

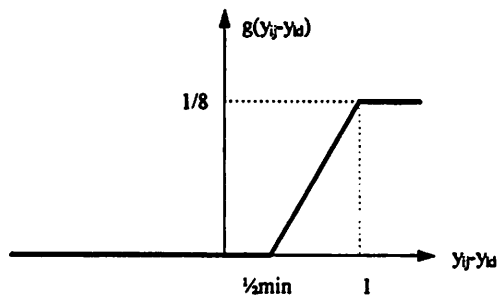
where a is a nonlinear function of the following form:



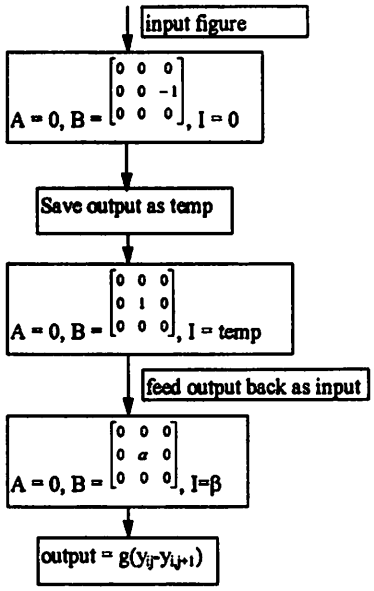
The general structure of the algorithm equivalent to the nonliear template is the following:



The first step is to construct $g(y_{ij}-y_{kl})$ for each y_{kl}, a neighbor of y_{ij}, where $g()$ is:

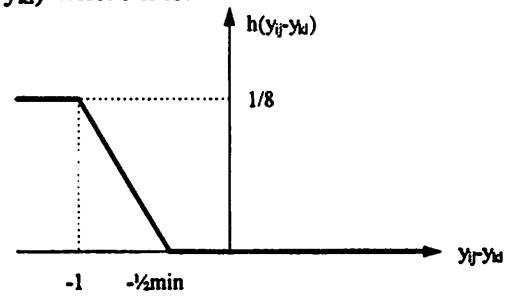


This is done with the following algorithm. Let $\alpha = 1/(4-2\text{min})$, $\beta = \frac{1}{2}(1+\frac{1}{2}\text{min})$.

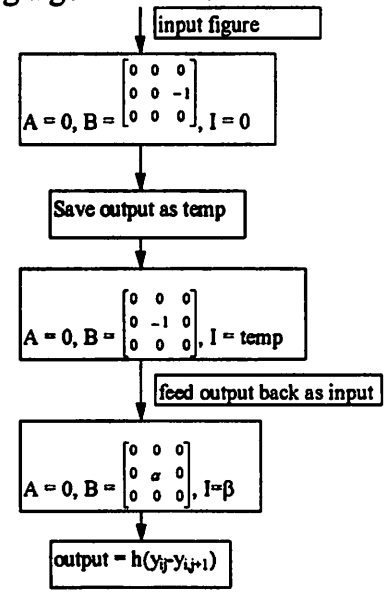


To calculate $g(y_{ij}-y_{i,j-1})$ etc., we just need to change the B-template in the first step of the above algorithm to $B = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, etc.

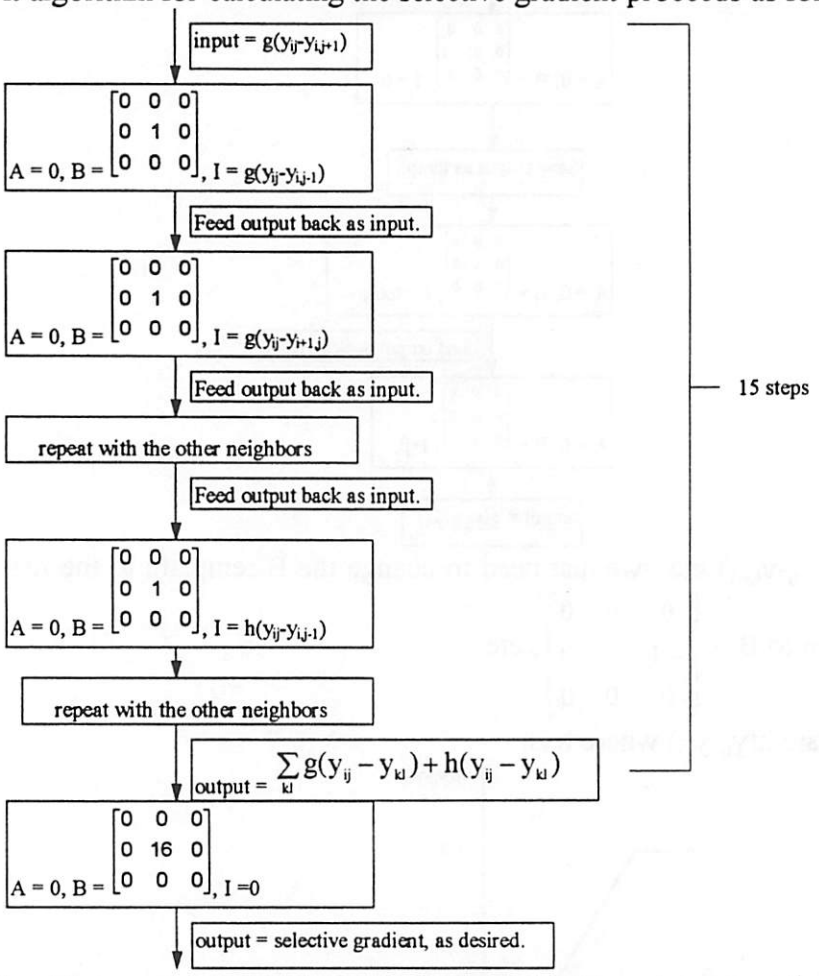
We next calculate $h(y_{ij}-y_{kl})$ where h is:



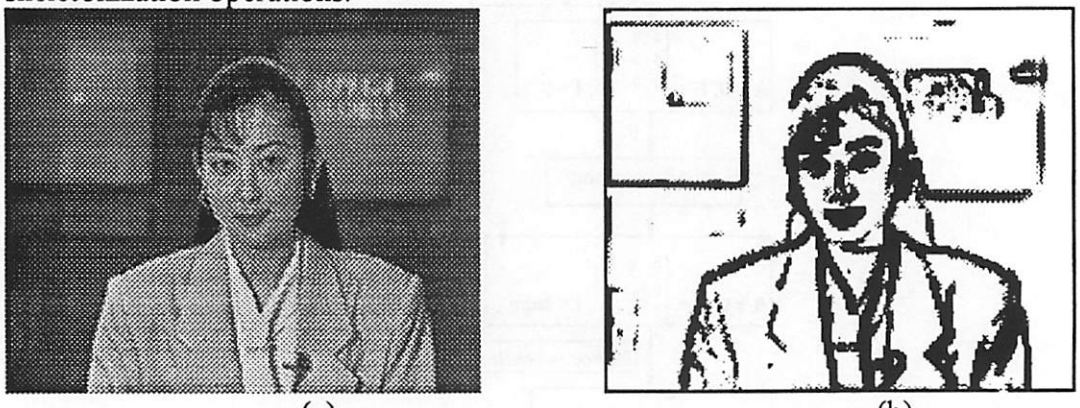
This is done with the following algorithm which is similar to the algorithm for g().



Then the next algorithm for calculating the selective gradient proceeds as follows:



The entire algorithm needs a total of (8+8)*3 +15+1 = 64 template operations. Using min = 0has .05 and d = 0.2, we obtain Figure 5b from the original image in Figure 5a. The next step in extracting contours will be a combination of edge detection and skeletonization operations.



(a) (b)

Figure 5: (a) original image. (b) after applying selective gradient with min=0.05 and d=0.2.

5. IMAGE DECOMPOSITION USING A PERCEPTUAL IMAGE MODEL

Based on a technique described in Chapter 9 of [1], a CNN algorithm has been developed for decomposition of gray-scale images into a so called primary (strong edge), smooth, and texture components. Such a decomposition originates from a perceptual image model which assigns primary importance to the edge information to be processed by the human visual system. While such a decomposition is attractive as part of a high performance compression scheme (the three components can be coded more efficiently then the original image), it is computationally expensive. The CNN implementation offers now the required performance through the unique capabilities of the CNN Universal Machine.
The flow-diagram of the algorithm is shown in Figure 6. While its basic structure is similar to that of [1], the implementation of the transformation blocks are tailored to efficient implementation on the CNN Universal Machine. It is worth noting that after coding, transmission, and decoding, reconstruction of the original involves only simple additions eliminating the need for a complex decoder.

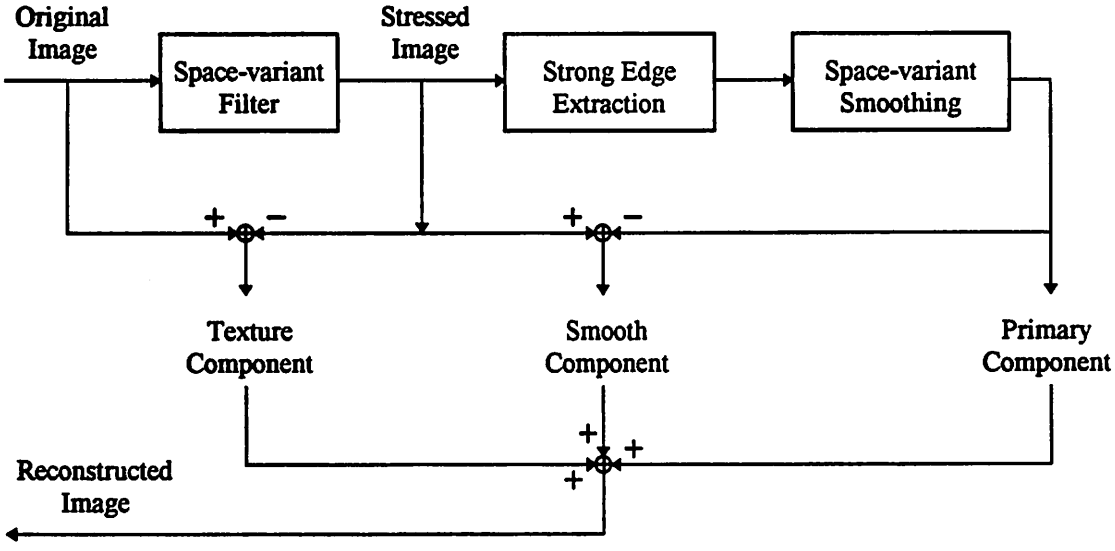


Figure 6: Flow diagram of the perceptually based decomposition algorithm

5.1 Space-variant low-pass filtering

First, a type of space-variant low-pass filtering is applied to the input image to produce a so called stressed image. The stressed image which contains accentuated edge and low spatial-frequency information can be used directly to obtain the texture component by subtracting it from the original. The flow-diagram of the iterative CNN algorithm implementing the space-variant filtering is shown in Figure 7. The main device of the transformation is the smoothing operator (15) which is essentially an approximation to the Poisson equation. It is applied in each cycle with new boundary conditions which are generated by the second step in the form of binary fixed state maps. Iteration starts with a blank fixed state map and stops when all cells are included in the map. In each cycle parameters α_i and β_i ($0 < \alpha_i, \beta_i \leq 1$) are changed such that for stronger edges (larger β_i)

weaker smoothing operators (smaller α_i) are chosen. The templates used in the subroutine are the following:

$$\text{Tem1: } A = \alpha_i \begin{bmatrix} 0.5 & 1 & 0.5 \\ 1 & -6 & 1 \\ 0.5 & 1 & 0.5 \end{bmatrix}, \qquad B = [1] \tag{15}$$

$$\text{Tem2: } B = \begin{bmatrix} 0.5 & 1 & 0.5 \\ 1 & -6 & 1 \\ 0.5 & 1 & 0.5 \end{bmatrix} \tag{16}$$

$$\text{Tem3: } A = [2], \qquad I = \beta_i \tag{17}$$

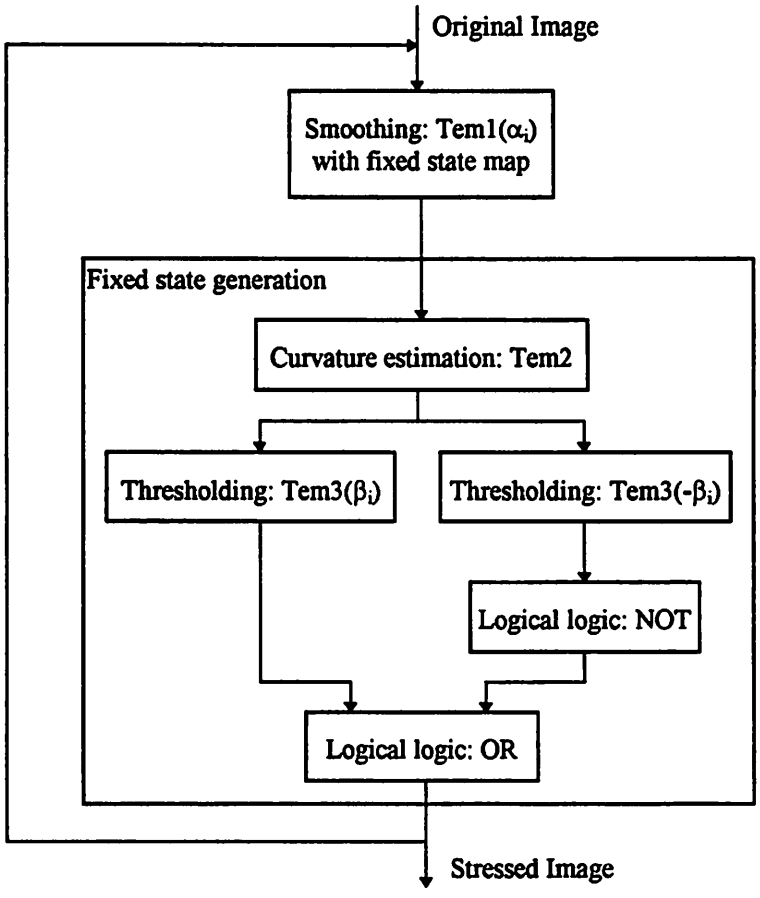


Figure 7: Flow-diagram of the space-variant filtering step

5.2 Strong edge extraction

Next, strong edges are located within the stressed image. Using another algorithm, a binary map is generated to mark the location of pixels corresponding to strong edge contours. The flow-diagram of the process is shown in Figure 8. In addition to the previous ones the following templates are used in the subroutine:

$$\text{Tem4: } A=[2], \quad B=\begin{bmatrix}0 & 0 & 0\\ 1 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}, \quad I=1 \tag{16}$$

$$\text{Tem5: } A=[2], \quad B=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}, \quad I=1 \tag{17}$$

$$\text{Tem6: } A=\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix}, \quad B=[2] \tag{18}$$

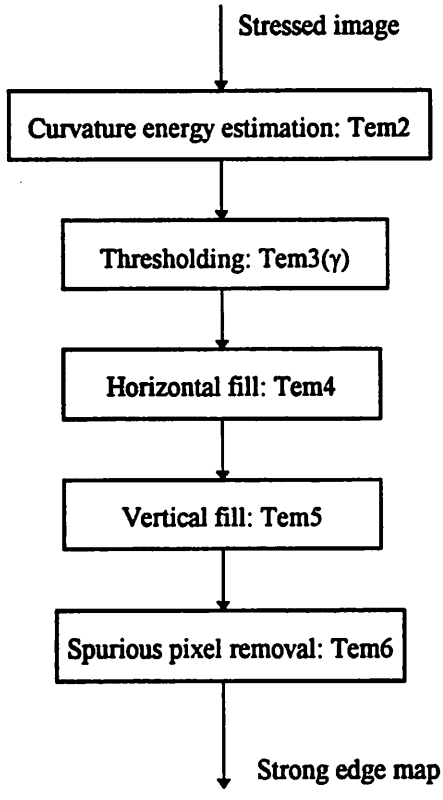


Figure 8: Flow-diagram of the strong edge extraction step

5.3 Space variant smoothing

Based on extracted edge information, space-variant smoothing is performed on the stressed image to obtain the primary image component which represents the visually most relevant edges. Template (19) is run on the stressed image using the strong edge map as a fixed-state mask to provide for appropriate boundary conditions for the smoothing process.

$$\text{Tem7:} \qquad A = \begin{bmatrix} 0.5 & 1 & 0.5 \\ 1 & -5 & 1 \\ 0.5 & 1 & 0.5 \end{bmatrix} \tag{19}$$

The difference image obtained by subtracting the primary image from the stressed image contains features represented by low spatial-frequencies and is called the smooth component.

Using these three subroutines, the three-component decomposition is easily and efficiently implemented on the CNN Universal Machine. Assuming a τ = 100ns time constant for the computing cells and n=10 recursions for the space-variant filtering step, the conservative estimate of $n\times 50\tau+100\tau=600\tau=60\mu s$ can be given for the overall processing time of a single input frame. Allowing another 40μs for I/O, a 64×64 cell CNN array, image sequences over 1000x1000 pixels/frame size can be processed at 30Hz video rate.

As an example, a segment of a video frame and its three-component representation generated by the above algorithm is shown in Figure 9.

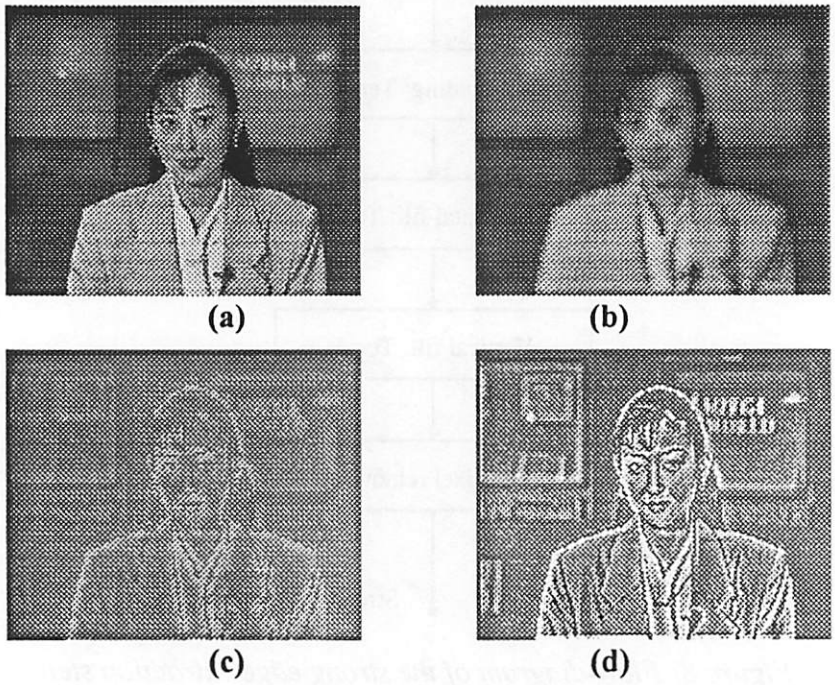


Figure 9: Original image (a) and its decomposition into primary (b), smooth (c), and texture (d) components.

It is worth noting that the above decomposition can be naturally extended into a multi-level representation. When in the first step the space-variant filtering is performed, by the particular choice of the parameters α_i and β_i, a decision is made about what will be considered primary information and what is regarded as detail/texture. Further processing of the texture component with another set of α_i and β_i parameters yields the next level of the representation. A multi-level representation generated by such a recursive scheme allows for more efficient coding while the computational complexity can be easily tackled by the CNN UM. It is also attractive that the proposed solution provides for dynamic control over the amount of detail transmitted and that it can naturally facilitate progressive transmission.

6. LUMINANCE COMPENSATED SEGMENTATION

This section presents a segmentation algorithm based on luminance information. Computational effort can be minimized by using an edge detection technique that creates regions where luminance changes in similar ways. These regions can be created by passing the gradient through two thresholds. By choosing a low threshold, major features in a picture are marked; a higher threshold will then fill in the smaller details. Each respective threshold is multiplied by a space varying scalar that depends on local luminance levels. In a final segmentation step, smaller segments are combined and all lines that do not form closed contours are removed. This overview is shown in Figure 10.

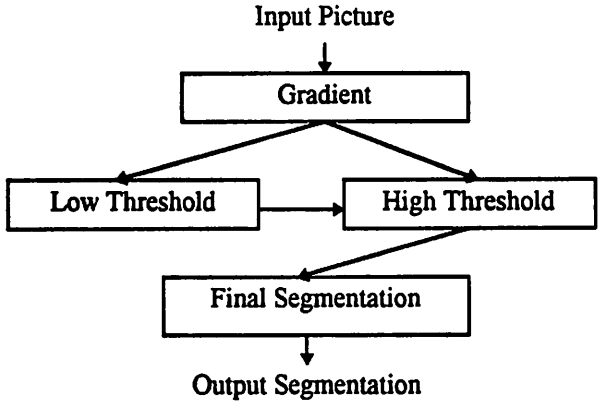


Figure 10. Overview of the segmentation scheme

6.1 Obtaining the Gradient

To approximate the gradient, we sum the absolute values of the directional derivatives. The directional derivatives can be calculated by using the A template as a FIR filter. In Figure 11, we summarize the steps taken to obtain the gradient.

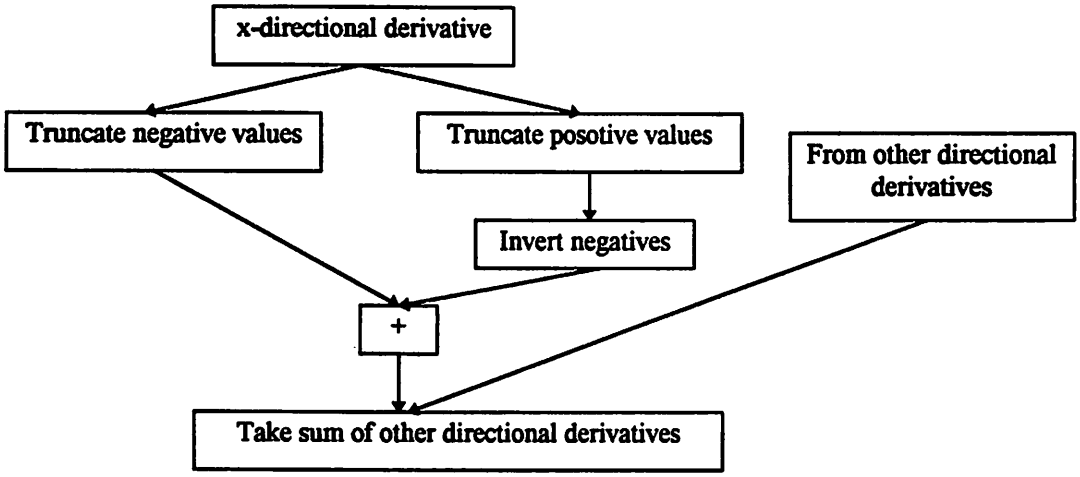


Figure 11. Flow chart of the algorithm for obtaining the gradient

Luminance Compensation

One optional step is the compensation for luminance levels. It is noted that the human eye is more sensitive to intensity changes in the medium luminance levels while less sensitive to changes in the high and low luminance ranges. Thus, the resulting gradient is weighted by a space varying scalar that depends on local luminance levels. This re-scaled version simplifies the segmentation in the extreme luminance levels.

6.2 Double Thresholding and Final Segmentation

After the low threshold is obtained, it is processed before being combined with the higher threshold. Such processing includes a small hole filling template (a variation of the hollow filling template) and binary edge detection. The final segmentation step comprises of one final hole filling followed by a skeletonization to remove small "bubbles" that occur on the boundaries of two segments. This is summarized in Figure 12.

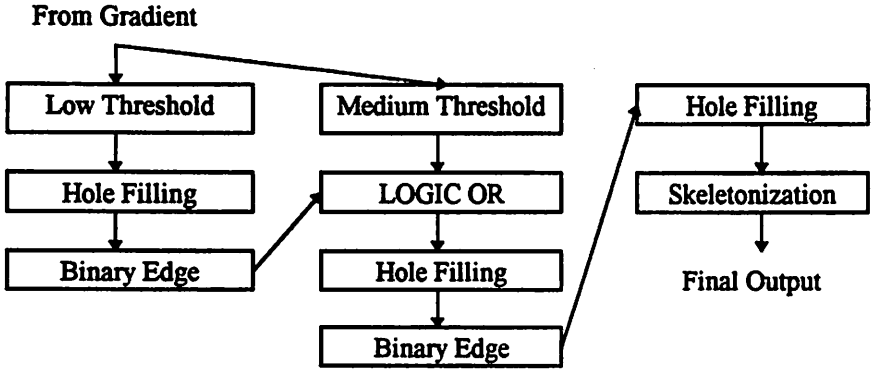


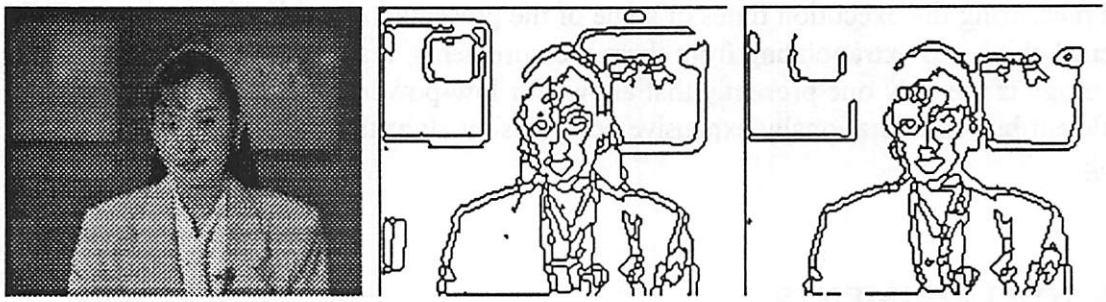
Figure 12. Thresholding and segmentation algorithm

The hole filling template is given below:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 6 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad , \quad B = 0 \quad , \quad I = 1$$

Other templates can be found in the CNN template library.

6.3 Simulation results



(a) (b) (c)

Figure 13. Comparison of compensated and uncompensated segmentation algorithms: input image (a), uncompensated output (b), and compensated output (c)

This algorithm does an adequate job segmenting the image into areas of comparable luminance. Many features on the face are clearly marked. The luminance compensated output provides some advantages in simplification compared to the uncompensated version, but some aspects of the background in the compensated version are also lost. For many applications, as long portions of the background are intact, the human viewer may be able to unconsciously supply the implied lines to recreate the background.

CONCLUSIONS

As it has been demonstrated in this paper, several prototype image segmentation and coding algorithms, as well as related mathematical morphology operators can be implemented using analogic CNN algorithms. The most time consuming operations of dynamic image coding are those involving generation of various masks. CNN is very efficient in doing such transformations while other, computationally less intensive and global operations can be better implemented using DSPs and specialized digital chips. Hence the combination of these two technologies via a CNN chip-set [17] which is fully transparent to digital technology, seems to be a viable solution. Below is a table comparing the performance of a CNN and two digital processors in executing image simplification using gray-scale morphology on a 500x500 pixel input image.

	Pentium Pro @200 MHz	C80 DSP @40 MHz	CNAPS @20	CNN UM 64x64 analogic
Number of processors on chip	1	5	64	4096
Technology	0.35 μ	0.5 / 0.6 μ	1 μ	0.8 μ
Processing time	1.4 s	2 s	0.2s	2 ms

From measuring the execution times of some of the presented algorithms on existing CNN universal chips and extrapolating from these measurements, it seems to be clear that CNN technology is the only one presently that allows for low-power, hand-held implementation of efficient but computationally expensive compression algorithms such as dynamic image coding.

ACKNOWLEDGMENTS

The authors would like to thank Ken Crounse for his input to Section 4. This work has been supported in part by the Office of Naval Research under grant N00014-89-1402, the Joint Services electronics Program under contract F49620-94-C-0038, by the joint grant of the National Science Foundation and the Hungarian Academy of Sciences INT-9413186, and by the Hungarian National Science Fund under grants No. T016528 and T017282.

REFERENCES

[1] L. Torres and M. Kunt, eds., "Video coding - The second generation approach", Kluwer, 1996.

[2] E. Reusens, T. Ebrahimi, and M. Kunt, "Dynamic coding of visual information", *IEEE Trans. Circuits and Systems for Video Technology*, to appear

[3] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory and Applications", IEEE Transactions on Circuits and Systems, vol. 35, pp. 1257-1290, October 1988.

[4] L.O. Chua and T. Roska, "The CNN Paradigm", *IEEE Transactions on Circuits and Systems - I*, vol. 40, pp. 147-156, March 1993.

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

[6] F. Werblin, T. Roska, and L.O. Chua, "The Analogic Cellular Neural Network as a Bionic Eye" *International Journal of Circuit Theory and Applications*, vol. 23, pp. 541-569, 1995.

[7] L. O. Chua, T. Roska, T. Kozek, and Á. Zarándy, "CNN universal chips crank up the computing power", IEEE Circuits and Devices Magazine, vol. ?, pp 18-28, 1996.

[8] Á. Zarándy, A. Stoffels, T. Roska, and L.O. Chua, "Implementation of Binary and Gray-scale Mathematical Morphological on the CNN Universal Machine" UCB Memo. No. UCB/ERL M96/19, 1996, submitted for publication to IEEE Trans. CAS

[9] A. Stoffels, T. Roska, L.O. Chua, "Object-oriented image analysis for very low bit-rate video coding systems using the CNN Universal Machine", Proc. of the fourth IEEE Int. Workshop on Cellular Neural Networks and their Application (CNNA-96), pp. 13-18, Seville, 1994, under publication in Int. J. of Circuit Theory and Applications

[10] T. Roska, L. Kék, "CSL-CNN Software Library (Templates and Algorithms)" Version 6.4, Analogic and Neural Computing Laboratory of the Computer and Automation Institute of the Hung. Acad. Sci., DNS-CADET-15 Budapest, Hungary 1995.

[11] P. L. Venetianer, F. Werblin, T. Roska, and L.O. Chua, "Analogic CNN Algorithms for some Image Compression, Decompression, and Restoration Tasks" *IEEE Trans. on Circuits and Systems-I*, Vol. 42, pp278-284, May 1995.

[12] R. Dominguez-Castro, S. Espejo, A. Rodriguez-Vazquez, R. Carmona, "A CNN Universal Chip in CMOS Technology", Proc. of the third IEEE Int. Workshop on Cellular Neural Networks and their Application (CNNA-94), pp. 91-96, December 1994.

[13] J. M. Cruz, L. O. Chua, and T. Roska, "A Fast, Complex and Efficient Test Implementation of the CNN Universal Machine", Proc. of the third IEEE Int. Workshop on Cellular Neural Networks and their Application (CNNA-94), pp. 61-66, December 1994.

[14] T. Roska, P. Szolgay, Á. Zarándy, P. L. Venetianer, A. Radványi, and T. Szirányi, "On a CNN Chip Prototyping System", Proc. of the third IEEE Int. Workshop on Cellular Neural Networks and their Application (CNNA-94), pp. 375-380, 1994.

[15] Á. Zarándy, "CCPS Analogic Machine Code (AMC)" technical report, Computer and Automation Inst., Hungarian Academy of Sciences, DNS-CCPS-13A, 1995

[16] J. Kishida, Cs. Rekeczky, Y. Nishio, A. Ushida, "An Iterative Approach of CNN to Image Processing-Feature extraction of Postage Stamps-", International Symposium on Nonlinear Theory and Its Applications (NOLTA'95) pp. 695-700, Las Vegas, December, 1995.

[17] T. Roska, "CNN Chip-set architectures and the visual mouse", Proc. of the fourth IEEE Int. Workshop on Cellular Neural Networks and their Application (CNNA-96), pp. 487-492, Seville, 1994.

[18] L. Kék, and Á. Zarándy, "Implementation of large-neighborhood nonlinear templates on the CNN Universal Machine", technical report No. DNS-9-1996, Computer and Automation Inst., Hungarian Academy of Sciences, 1996