Partial differential equation based preprocessing and immune response inspired algorithms implemented on CNN universal machine

György Cserey

A thesis submitted for the degree of
Doctor of Philosophy

Scientific adviser:
Tamás Roska, D.Sc.
ordinary member of
the Hungarian Academy of Sciences

Supervisor:
Csaba Rekeczky, Ph.D.

Faculty of Information Technology
Péter Pázmány Catholic University

Budapest, 2006

I would like to dedicate this thesis to my loving parents, Ilona and Mihály Cserey. They have always been there to love, support, and guide me. God has blessed me with two loving parents who worked hard to provide me with everything necessary. I am so very thankful for that blessing and for the example you both were to me over my life. Thank you, mom and dad, I love you both.

Acknowledgements

First of all I would like to thank Professor Tamás Roska, for his consistent help and support in very many ways, for his unbroken enthusiasm and fatherly guidance during my studies.

I am very greatful to Professor Wolfgang Porod for more than a years' time, that I could spend in Notre Dame Catholic University, Indiana, where as a visiting student I was posed to serious challenges. I thank Professor Árpád Csurgay for taking me under his wings, for his suggestions and our exciting converstations about science.

I thank Professor András Falus for sharing me the secrets of immunology. His explanations make biological processes clear to an engineer, too.

Special thanks to Csaba Rekeczky for providing projects and suggestions at the beginning of my studies and for his responsive help ever since.

I thank the reviewers of my thesis, Professor Péter Szolgay and Professor Marco Gilli for their conscientious reviews.

I am grateful to Dávid Bálya and Gergely Tímár for their friendly suggestions and encouragement and for our chats and colloquies without taking time-zones into account.

It was a pleasure to work with Professor Matthias Scheutz, John McRaven and Jasper Stolte.

I thank my older and younger colleagues for their advices and with whom I could discuss all my ideas: András Radványi, Péter Szolgay, Tamás Szirányi, Ákos Zarándy, Péter Földesy, Viktor Gál, Kristóf Karacs, László Kék, László Orzó, Szabolcs Tőkés, István Szatmári, Zoltán Szlávik, István Petrás, Zsolt Czeilinger, Levente Török, Péter Jónás, Viktor Binzberger, Gergely Soós, Anna Lázár, Róbert Wagner, Dániel Hillier, Zsolt Szálka, András Mozsáry, Attila Kis, Csaba Benedek, Gaurav Gandhi, Gábor Matyi, Gábor Hodász, Barnabás Hegyi, Zoltán Fodróczi és Gábor Vásárhelyi. For the younger ones I take the oppurtunity to wish success and endurance.

The support of the Hungarian Academy of Sciences (MTA-SZTAKI) and Péter Pázmány Catholic University, where I spent my Ph.D. years, is gratefully acknowledged.

I am indebted to Katalin Keserű and Gabi Kékné from MTA-SZTAKI, and Office of the Registrar at PPCU for their practical and official aid.

I had some really great teachers who should be mentioned here, as well I am greatful to Professor András Recski and Katalin Friedl from Budapest University of Technology and Economics, fr. Ákos Binzberger (OSB) from Benedictine Secondary School of Pannonhalma and Terézia Benkő from Hősök terei Primary School for the experiences in mathematics.

I am grateful to Zsófia Majoros, Bálint Farkas and Róbert Wagner for providing technical and readership help writing this document.

I am very grateful to my mother and father and to my whole family who always believed in me and supported me in all possible ways.

Abstract

This dissertation (i) describes parallel histogram modification techniques with embedded morphological preprocessing methods within the CNN-UM framework, describes and illustrates how the implementation of the algorithm results in an adaptive multi-thresholding scheme when histogram modification is combined with embedded morphological processing at a finite small number of gray-scale levels; (ii) presents an immune response inspired algorithmic framework for spatial-temporal target detection applications using CNN technology. The given algorithms can be implemented effectively only by using a computer upon which thousands of elementary, fully parallel spatial-temporal actions can be implemented in real time. Experiments demonstrate that the developed system can detect unknown patterns and dynamical changes in image sequences.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every day it is worth wondering over the beauties of Nature, life and its biological processes, came to that the complexity of only one cell. We can wonder and admire, however we will never be able to understand it fully. We often take courage, with our humbleness is on the small side, to copy or horn in its processes. The alibi of Medicine is simple: the protection of human life. An engineer can offer his knowledge to the doctor, and give better instruments to him or find ideas from studying medicine and apply them to provide more effective solutions of engineering problems.

During my research I worked on two problemfields, which are tightly connected to the domain of medical biology. One of them concentrates on helping the work of cardiologists; with the other, I would like to get ideas while studying the processes of our immune system.

The two fields / opened problem are the following:

- **the implementation of a real-time filter algorithm in the diagnostics of noisy ultrasound images**
- **the detection of real-time, multi target, spatial-temporal novelty detection in image flows**

In the case of cavity detection and processing of medical ultrasound images we have to tackle several problems like low contrast and heavy noise. The state of the art technologies (e.g. 3D echocardiography) need extremely high speed and real time processing. **In the course of my research I was searching**

for a method which, besides simultaneous contrast enhancement, noise filtering and shape enhancement, could be implemented on the input image with real-time processing. The output image can be an appropriate input to the next level of processing, where cavity and other object detection would be accomplished.

In medical image processing, numerous solutions were found for the problems arising in the preprocessing phase. (i.e., contrast enhancement, noise supression, shape enhancement). The most common solutions emerged from the fields of nonlinear diffusion and curve evolution, where the execution of the algorithms based on parial differential equations can not be solved in real-time with traditional computational methods because of the computational needs of the newest appliances.

Meanwhile, it is not a trivial engineering problem to make the analog implementation of a complicated algorithm either. It was my goal as well to show that an analog implementation was possible, and an adequate speed up can be achieved.

If we thoughtfully examine our immune system we could say it is a "great thinker". It often passes abstract and molecularly coded information between its billions of interconnected cells. It remembers past events and predicts future ones. Forming a dynamic model of its environment, its plasticity adjusts interactions of its cells, population sizes moreover genes. If we cut a piece out of a brain, we seriously influence its functioning contrary to the immune system, which continues to function even after suffering heavy damages. Neurons rarely multiply, but immune cells grow and are replaced constantly, because they war and guard our body against intruders, viruses, bacteria and harmful mutations. It does not need to be held in a protective bone case like the brain. It is distributed with no central controller, with no Achilles heel. It is intelligent, it monitors not only the world outside, but the trillions of cells that we are made from. It knows exactly what is going on inside us, even at molecular scales, while our brain is oblivious. Maybe its intelligence quotient is not measurable and we will never be able to lead a conversation with it, but while clever brains are optional in living creatures, clever immune systems are crutial.

The cell-level interaction of immune system is based on identification and recognition of 3D molecule patterns. During my research the object I proposed is a creation of a model, which, similarly to the 3D spatial pattern detection of the immune system, is able to detect and recognize dynamic objects in 2D image flows. **I intended to design topographical algorithms and their experimental realization where huge number of target objects are monitored in real time to detect previously unknown events. So my goal was spatial-temporal novelty detection.**

Novelty detection can be a challenge in several areas. Nowadays, one of these areas is robotics, where novelty detection - the differentiation of the general sensor input and the sensory pattern not yet experienced - provides useful knowledge to mobile robots in a dynamically changing environment.

Sensor-close computation can be crucial trom the point of view of utilization efficiency, since it could help solving some of the general problems of traditional systems, namely the reduction of the bandwidth of image transfer from the sensor to the computational unit and the time meeded to process images in real-time.

The dissertation is organized as follows. Chapter 2 describes parallel histogram modification algorithms with embedded morphological preprocessing methods within the CNN-UM framework. Chapter 3 presents experimental results processing real-life and echo-cardiographic images, measured on different hardware/software platforms, including a 64×64 CNN-UM chip. In Chapter 4 an immune response inspired algorithmic framework is presented for spatial-temporal target detection applications using CNN technology. Chapter 5 demonstrates that the given immune response inspired algorithms can detect unknown patterns and dynamical changes in image sequences. Chapter 6 summarizes the main results and highlights further potential applications, where the contributions of this dissertation could be efficiently exploited.

A number of appendices illustrate this work and summarize some of the theoretical background. In Appendix A and B a short summary of CNN technology as well as a new mathematical description of continuous machines on flows are given. Appendix C presents some basic results about level-set theory. In Appendix D, experimental results corresponding to Chapter 3 are given. For those who are not familiar with biology, some important functions of immunology are summarized in Appendix E.

The author's publications and other publications connected to the dissertation can be found at the end of this document.

Chapter 2

PDE Based Histogram Modification with the Level-Sets

This chapter describes parallel histogram modification techniques with embedded morphological preprocessing methods within the CNN-UM framework. The procedure is formulated in terms of nonlinear partial differential equations (PDE) and approximated through finite differences in space, resulting in coupled nonlinear ordinary differential equations (ODE). The I/O mapping of the system (containing both local and global couplings) can be calculated by a complex analogic (analog and logic) algorithm executed on a nonlinear array processor, called the cellular nonlinear network universal machine (CNN-UM, [20]).

2.1 Introduction

Nonlinear PDE-based image processing has been introduced to the field of computer vision by [69], and to the field of medical imaging by [70]. In the last years there has been an intensive research reaching from the mathematical foundations and properties of PDE-based image processing.

Quite some efforts have been undertaken in order to find reliable and efficient numerical schemes for nonlinear diffusion filtering: see e.g., [55, 72, 73, 74], and partial differential equations (PDE's) have dominated image processing research recently. The three main reasons for their success are: (i) their ability to transform a segmentation modeling problem into a partial differential equation framework and their ability to embed and integrate different regularizers into these models; (ii) their ability to solve PDE's in the level set framework using finite difference methods; and (iii) their easy extension to a higher dimensional space [73] .

Nonlinear diffusion filtering on current PCs or workstations can be achieved in the order of a second in 2D, and in the order of a minute for typical 3D data sets that arise in medical imaging [71].

Basically, contrast enhancement techniques are divided in the two groups, local and global, and their most popular representatives can be found in any basic book in image processing and computer vision.

The formalization of multiscale analysis given in [54] leads to a formulation of recursive, causal, local, morphological, and geometric invariant filters in terms of solutions of certain partial differential equations of geometric type, providing a new view on many of the basic mathematical morphology operations. One of their basic assumptions was the locality assumption, which aimed to translate into a mathematical language the fact that basic operations which were a kind of local average around each pixel or, in other words, only a few pixels around a given sample influence the output value of the operations. Obviously, this excluded the case of algorithms as histogram modification. This is why operations like those in [23, 24, 25] and described in my thesis work are not modeled by these equations.

I analyze a set of PDEs designed for simultaneous (i) contrast enhancement, (ii) noise suppression and (iii) shape enhancement[1]. Based on spatial approximations and using a discrete set of gray-values these PDEs will be decomposed to spatially interacting ODEs (discrete in space and continuous in time).

My aim was to realize a fast filter algorithm which can be used efficiently in ultrasound diagnostics. In the course of my research I was searching for a method which, besides simultaneous contrast enhancement, noise filtering and shape enhancement, could be implemented on the input image with real-time processing. The most common solutions emerged from the fields of nonlinear diffusion and curve evolution, where the execution of the algorithms based on parial differential equations can not be solved in real-time with traditional computational

[1]shape enhancement: the characteristic properties of an image are enhanced and the irrelevant parts are suppressed.

methods because of the computational needs of the newest appliances. The chosen computer architecture, the CNN-UM has several advantages (fast speed, local interaction, parallel and sensor-close processing) that makes this machine suitable for implementing the problem on. Meanwhile, it is not a trivial engineering problem to make the analog implementation of a complicated algorithm either. It was my goal as well to show that an analog implementation was possible, and an adequate speed up can be achieved.

Due to poor or changing lighting conditions image snapshots (or video-flows) are often captured at low contrast in different scenarios. On the other hand, the majority of known algorithms taking these images as input are in general fairly sensitive to huge (or rapid) contrast changes between consecutive images, which could significantly degrade the processing performance. Recent advances of different visual microprocessors resulted in new hardware platforms for parallel algorithms and are also the basis for the increased interest in the development of new parallel nonlinear techniques for contrast improvement.

This chapter is organized as follows. In Section 2, a short introduction is given to contrast enhancement through an example. Section 3 contains mathematical PDE formulations for histogram equalization with the level-sets. My PDE based histogram modification algorithms are presented in Section 4. Section 5 presents execution time data. In Section 6, the extent of the algorithmic schemes are summarized with morphological processing of the level-sets. In Section 7, conclusions are presented.

2.2 Contrast Enhancement Through Histogram Modification

The most common way to improve the contrast of an image is to modify its gray-value distribution, the image histogram. For example, histogram equalization (see Figure 2.1 (a) and (c)) is a basic method that drives the image pixel values into different gray-scale levels in order to achieve a uniform distribution [21, 22]. This global technique improves the contrast and does not modify the level-sets[2] of the image (see Figure 2.1 (b) and (d)). Histogram modifications can also be formulated in terms of PDEs [23, 24, 25] that give a hope to a fully parallel implementation. The PDE formulation makes it also convenient to combine other preprocessing schemes with histogram modification in order to build contrast invariant segmentation methods (see two different modified outputs in Figure 2.1 (e) and (g) with their corresponding level-sets in Figure 2.1 (f) and (h), respectively).

In this chapter, I investigate and modify some of these recent formulations, derive and implement efficient new algorithms for cellular nonlinear array processor architectures.

2.3 PDE Formulations

Let us consider the following PDE designed for histogram equalization [23, 24, 25]:

$$\frac{\partial\Phi(x,y,t)}{\partial t} = (N^2 - \frac{N^2}{M}\Phi(x,y,t)) - A\left[L(v,w,t)_{\Phi(x,y,t)}\right] \tag{2.1}$$

where $\Phi(x,y,t) : [0,N]^2 \times [0,T] \to [0,M]$ is the image intensity, N and M are constants, and $A[\cdot] : \Re^2 \to \Re$ stands for the area measure of a level-set.

The level-sets $L(v,w,t)_{\Phi(x,y,t)} : [0,N]^2 \times [0,T] \to \{0,M\}$ are the "binary shadows" of the image and can be described as follows:

$$L(v,w,t)_{\Phi(x,y,t)} = L_\Phi = \begin{cases} 1, & \text{if } \Phi(v,w,t) \geq \Phi(x,y,t), \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

Then the calculation of the area measure is defined as follows:

$$A[L_\Phi] = A[(v,w) : \Phi(v,w,t) \geq \Phi(x,y,t)] = \int_{v=0}^{N}\int_{w=0}^{N} L_\Phi dv dw. \tag{2.3}$$

A modified and extended version of Equation (2.1) that allows a simultaneous contrast enhancement with noise suppression [23, 24, 25] is as follows:

$$\frac{\partial\Phi(x,y,t)}{\partial t} = \alpha\kappa + (N^2 - H(\Phi(x,y,t))) - A[L_\Phi], \tag{2.4}$$

[2]In mathematics, a level set of a real-valued function f of n variables is a set of the form $(x_1,...,x_n)|f(x_1,...,x_n) = c$ where c is a constant. That is, it is the set where the function takes on a given constant value. When the number of variables is two, this is a level curve (contour line), also see Appendix C.

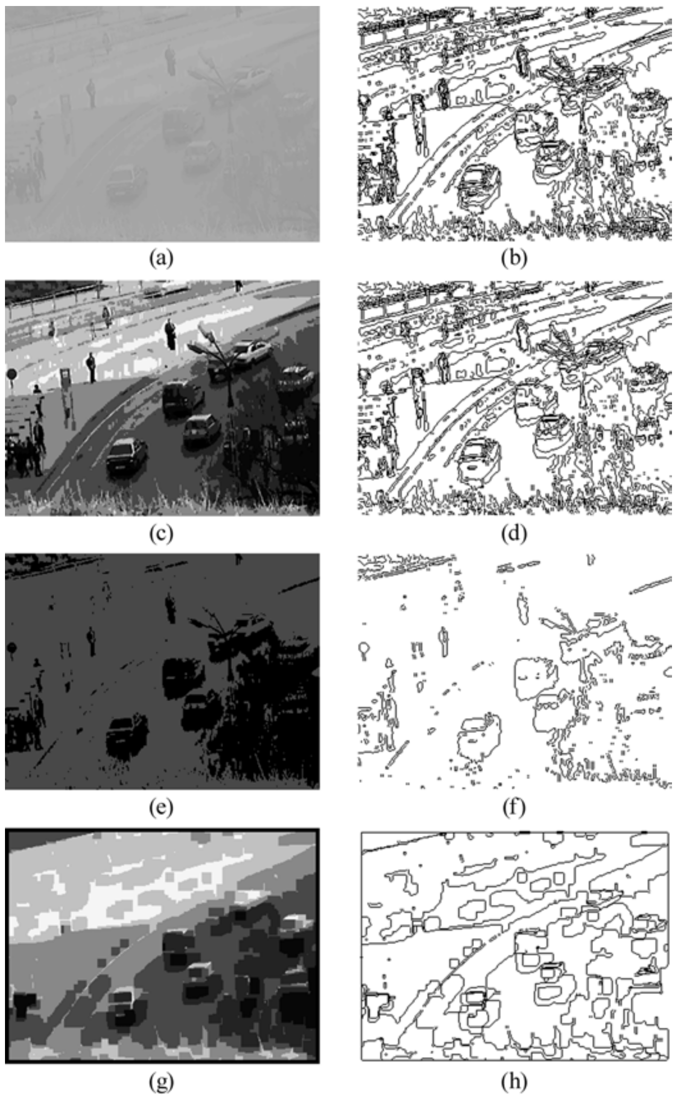


Figure 2.1: Constructive use of a programmable global PDE on a CNN architecture – the above example demonstrates complex histogram modifications where simultaneous contrast enhancement is connected with noise filtering and embedded morphological processing. (a) original, low contrast image, (b) level sets of (a); (c) histogram equalized image through a PDE based processing, (d) level sets of (c); (e) PDE based histogram modification output with embedded differential morphological processing at scale 3τ and 2 gray-scale levels, (f) level sets of (e); (g) PDE based histogram modification output with embedded differential morphological processing at scale 5τ and 8 gray-scale levels, (h) level sets of (g). Observe that the level sets are preserved in case of global histogram equalization and how embedded morphological processing modifies the level sets leading to a meaningful segmentation result.

where α is a constant, κ is a regularizing term, and $H(\cdot):\Re\rightarrow\Re$ represents a prescribed monotone increasing function.

A further novel generalization that adds shape enhancement by morphological processing of the level-sets is as follows:

$$\frac{\partial\Phi(x,y,t)}{\partial t}=\alpha\kappa+(N^2-H(\Phi(x,y,t)))-A\left[L_{g(\Phi)}\right], \tag{2.5}$$

where $L_{g(\Phi)}$ is a "threshold transformed" level set and $g(\cdot):\Re\rightarrow\Re$ is a general nonlinear function.

I assume the function $g(\cdot)$ to depend implicitly (explained later) on $\Gamma[U,B]$ that stands for a morphological processing of U with a structuring element B (a disk).

Two forms of morphological processing will be investigated, the n-step "erode-dilate" and "dilate-erode" operations [22] ($\oplus$ and $\otimes$ stand for dilation and erosion, respectively) described by the following set-theoretic formulations:

$$\Gamma[U,B]=((U\oplus B)^{(n)}\otimes)^{(n)} \text{ or } \Gamma[U,B]=((U\otimes B)^{(n)}\oplus)^{(n)}. \tag{2.6}$$

The input set U of these operations is either a level-set L_Φ or a level-set section: $L_{\Phi1,\Phi2}=L_{\Phi2}-L_{\Phi1}$.

2.4 Approximating the Histogram Modification PDEs

Mapping the PDEs introduced in the previous section into nonlinear ODEs I consider two cases corresponding to Equation (2.1), and Equation (2.5) respectively. All these approximations could be implemented as a complex analogic algorithm executable on an existing architecture [26, 27].

Case 1: Contrast enhancement through histogram equalization (Figure 2.1c and 3.2b)

Based on Equation (2.1) assuming $N^2=1; M=1$:

$$\frac{d\phi_{ij}(t)}{dt}=-\phi_{ij}(t)+(1-A_{ij}^{(1)}). \tag{2.7}$$

Case 2: Contrast enhancement, denoising and embedded morphological processing (Figure 2.1g and 3.2c-d)

Based on Equation (2.5) assuming $\alpha = 1; \kappa = div(grad(\phi)), H = N^2/M; N^2 = 1; M = 1$:

$$\begin{aligned} \frac{d\phi_{ij}(t)}{dt} &= -2\phi_{ij}(t) + (1 - A_{ij}^{(2)}) + \\ 1/4(\phi_{i-1,j}(t) &+ \phi_{i+1,j}(t) + \phi_{i,j-1}(t) + \phi_{i,j+1}(t)) \end{aligned} \tag{2.8}$$

In all cases $A_{ij}^{(1)} = A_{ij}^{(2)} = \text{const}$ during the evolution [24], therefore should only be calculated once. A_{ij} represents area (number of pixels in this discrete case). For the steady-state solution $\Phi_t = 0$, we have:

$$A_{ij} = A\left[(v, w) : \Phi(v, w) \geq \Phi(i, j)\right] = 1 - \Phi(i, j) \tag{2.9}$$

Then for $a, b \in [0, 1], b \geq a$,

$$A\left[(i, j) : b \geq \Phi(i, j) \geq a\right] = b - a \tag{2.10}$$

which means that the histogram is constant. Therefore the steady-state solution of Equation (2.1) gives a normalized image via histogram equalization.

Though A_{ij} is the output of a global transformation it is possible to give an approximation based on purely local (analog and logic) operations. This “spatial decomposition” will be discussed in the sequel.

$A_{ij}^{(2)}$ in Equation (2.8) differs from the first two versions, since it should include an embedded morphological processing [28]. Also, contrast stretching and further desirable smoothing and enhancement properties can be added to the algorithm.

The following operational notations and definitions will be used throughout the algorithm descriptions (all templates referenced can be found in the CNN Software Library [29]):

Definition 1: Threshold – thresholds a gray-scale input image at a given gray-scale level. The output is a binary image defined as follows:

$$Thr(\Phi_{ij}, \lambda) = \begin{cases} 1, & \text{if } \Phi_{ij} \geq \lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

CNN implementation: by using the THRESH template.

Definition 2: Area – calculates the area measure corresponding to a given level-set of the input image. The output is a scalar defined as follows ($\Phi_{ij,Bin}$ is the level-set calculated by the Thr function):

$$Area(\Phi_{ij,Bin}) = \sum_{i=1}^{N}\sum_{j=1}^{N}\Phi_{ij,Bin}. \tag{2.12}$$

CNN implementation: by using the DIFFUS template the average value of an image can be calculated in the specified domain. If the initial state is a constant image (all pixels are set to +1) over the specified level-set then the diffusion output at steady-state gives a normalized area measure (a value in the range of [0, 1]) of the level-set domain related to the entire image area.

Definition 3: Set level – sets the value of all pixels over a given mask to a specified gray-scale level. The output is a gray-scale image defined as follows:

$$SetLev(\Phi_{ij,Bin}, v) = \begin{cases} v, & \text{if } \Phi_{ij,Bin} = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

CNN implementation: by using the SETLEV template.

Definition 4: Erode – calculates erosion of a binary input image with a specified structuring element B. The set theoretical definition of the erosion based on Minkowski subtraction is as follows (- denotes translation):

$$Erode(\Phi_{ij,Bin}, B) = \Phi \otimes B = \cap\left\{\Phi_{ij,Bin} - b : b \in B\right\}. \tag{2.14}$$

CNN implementation: by using the EROSION template (single-step erosion) or PROPE (continuous erosion by a trigger-wave).

Definition 5: Dilate – calculates dilation of a binary input image with a specified structuring element B. The set theoretical definition of the erosion based on Minkowski addition is as follows (+ denotes translation):

$$Dilate(\Phi_{ij,Bin}, B) = \Phi \oplus B = \cup\left\{\Phi_{ij,Bin} + b : b \in B\right\}. \tag{2.15}$$

CNN implementation: by using the DILATION template (single-step erosion) or PROPD (continuous dilation by a trigger-wave).

Definition 6: Norm – calculates a normalized version of a gray-scale input image. The formulation of the operation is as follows (D_{min} and D_{max} stand for the minimum and maximum of the available dynamic range; Φ_{min} and Φ_{max} stand for the minimum and maximum of the input image, respectively):

$$\begin{aligned} Norm(\Phi_{ij}) = D_{min} + \frac{D_{max} - D_{min}}{\Phi_{max} - \Phi_{min}}(\Phi_{ij} - \Phi_{min}) = \\ D_{min} + \frac{\Delta D}{\Delta \Phi}(\Phi_{ij} - \Phi_{min}) \end{aligned} \tag{2.16}$$

CNN implementation: first Φ_{min} and Φ_{max} is calculated by using the THRESH template and global logic. Since ΔD is known a priori, the implementation of Equation (2.16) leads to an analogic algorithm based on template SCALE and simple arithmetics. Remark: the constant $b_0 = \Delta D/\Delta\Phi$ that is the central element of the B term in SCALE is image dependent, thus interaction with the digital environment is needed. It should be noted though that **Norm** is not an essential part of the histogram modification algorithm, it is included into the extended version (Algorithm 2, see later) in order to increase the robustness in a physical implementation with a limited analog precision (especially in case of very low input image dynamic range).

Definition 7: Diffuse – calculates a constrained linear low-pass filtered version of a gray-scale input image. The formulation of the operation is as follows ($*$ denotes convolution):

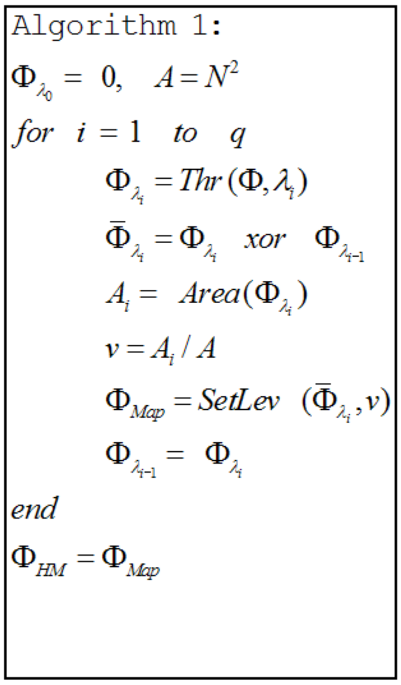
$$\text{Diff}(\Phi_{ij,1}, \Phi_{ij,2}) = \alpha\Phi_1 * G_{\sigma_1} + (1-\alpha)\Phi_2 * G_{\sigma_2} \tag{2.17}$$

where in 1D : $G_\sigma(\xi) = (1/\sigma\sqrt{2\pi})e^{-\xi^2/2\sigma^2}$

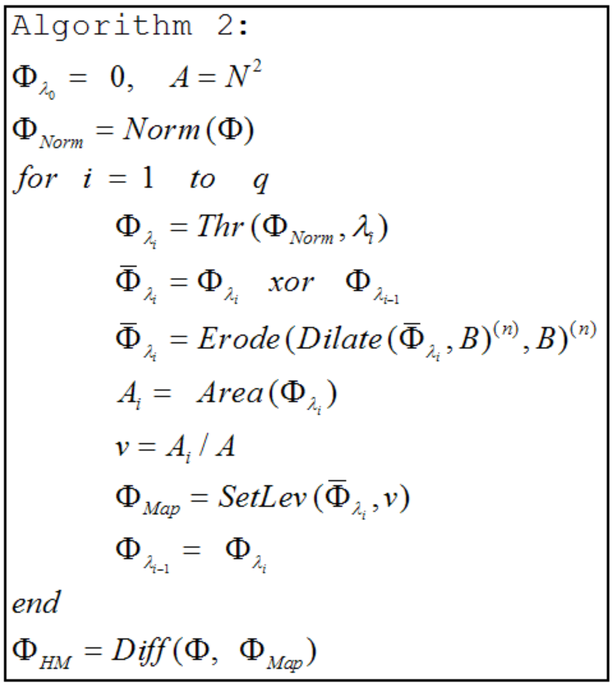
CNN implementation: Equation (2.17) describes a homotopy in between two different linear convolutions by a Gaussian kernel. Under fairly mild conditions at some time t this corresponds to the solution of a constrained diffusion type partial differential equation. After spatial discretization this can be mapped to a CNN structure [30, 31] programmed by template CDIFFUS. In this form the B term directly approximates G_{σ_2} , while the transient length is explicitly related to $G_{\sigma_1}(t \approx \sqrt{\sigma_1})$.

The pseudo-code description of the histogram modification algorithm can be found in Figure 2.2.

Algorithm 1 and 2 show the implementation steps of Equation (2.7) and Equation (2.8) including all necessary calculations. The UMF (Universal Machines on



```
Algorithm 1:
$\Phi_{\lambda_0} = 0, \quad A = N^2$
$for \quad i = 1 \quad to \quad q$
        $\Phi_{\lambda_i} = Thr(\Phi, \lambda_i)$
        $\bar{\Phi}_{\lambda_i} = \Phi_{\lambda_i} \quad xor \quad \Phi_{\lambda_{i-1}}$
        $A_i = Area(\Phi_{\lambda_i})$
        $v = A_i / A$
        $\Phi_{Map} = SetLev \ (\bar{\Phi}_{\lambda_i}, v)$
        $\Phi_{\lambda_{i-1}} = \Phi_{\lambda_i}$
$end$
$\Phi_{HM} = \Phi_{Map}$
```



```
Algorithm 2:
$\Phi_{\lambda_0} = 0, \quad A = N^2$
$\Phi_{Norm} = Norm(\Phi)$
$for \quad i = 1 \quad to \quad q$
        $\Phi_{\lambda_i} = Thr(\Phi_{Norm}, \lambda_i)$
        $\bar{\Phi}_{\lambda_i} = \Phi_{\lambda_i} \quad xor \quad \Phi_{\lambda_{i-1}}$
        $\bar{\Phi}_{\lambda_i} = Erode(Dilate(\bar{\Phi}_{\lambda_i}, B)^{(n)}, B)^{(n)}$
        $A_i = Area(\Phi_{\lambda_i})$
        $v = A_i / A$
        $\Phi_{Map} = SetLev(\bar{\Phi}_{\lambda_i}, v)$
        $\Phi_{\lambda_{i-1}} = \Phi_{\lambda_i}$
$end$
$\Phi_{HM} = Diff(\Phi, \ \Phi_{Map})$
```

Figure 2.2: Pseudo code description of two versions of the histogram modification. Algorithm 1 implements histogram equalization, while Algorithm 2 is the generalized histogram modification with embedded morphological processing of the level-sets.

Flows) description of the algorithmic core can be seen in Figure 2.3. The CNN flow-chart of the generalized histogram modification analogic algorithm (Algorithm 2) can be seen in Figure 2.4.

Algorithm 1 executes histogram equalization. In the first step, the initialization of the variables takes place. It is followed by a loop, whose variable is the same as the number of level-sets. The first step of the body of this loop is the thresholding of the input image with adequate values depending on the level-sets and get a result, called binary shadow. Taking the XOR opration of this result and a former binary shadow we have a binary image. The algorithm measures the area of this image. The black pixels of the binary image filled a grayscale value proportional to the measured area. The resulting picture is added to the output image (called global bias map), then the iteration is continued. At the end of the iteration, the output image gives the histogram-equalized result.

Algorithm 2 (as contrasted to Algorithm 1) is suitable for running morphological or wave operations within the body of the loop on the binary image. In Figure 2.2 and 2.3 we can follow the algorithm and its morphological operations can execute closing with the help of dilation and erosion of a variable number. This way, noise suppression and shape enhancement is executed. At the end the algorithm denoises the image with the help of diffusion.

In Figure 2.4, the fourth block of the flowchart, which executes the measurement of the area, can have other alternative solutions as well (e.g., counting the pixels). At the threshold step and other modules, further adaptive strategies can be embedded. $*$ mark: fuzzy decomposition instead of fixed threshold decomposition. In case of the results overlap SMAP images sub-quantum level adoptation can be done. $**$ mark: expand diffusion only to local areas instead of the entire image regions over a circular area calculated from TMapNew and TMapOld. $***$ mark: mebedded morphological processing the level-sets function SMap, which has been showed and done in Algorithm 2.

2.5 Execution time

In Figure 2.3, we can follow the time requriments of the independent steps of the algorithm. The execution time of the algorithm, depending on the number of

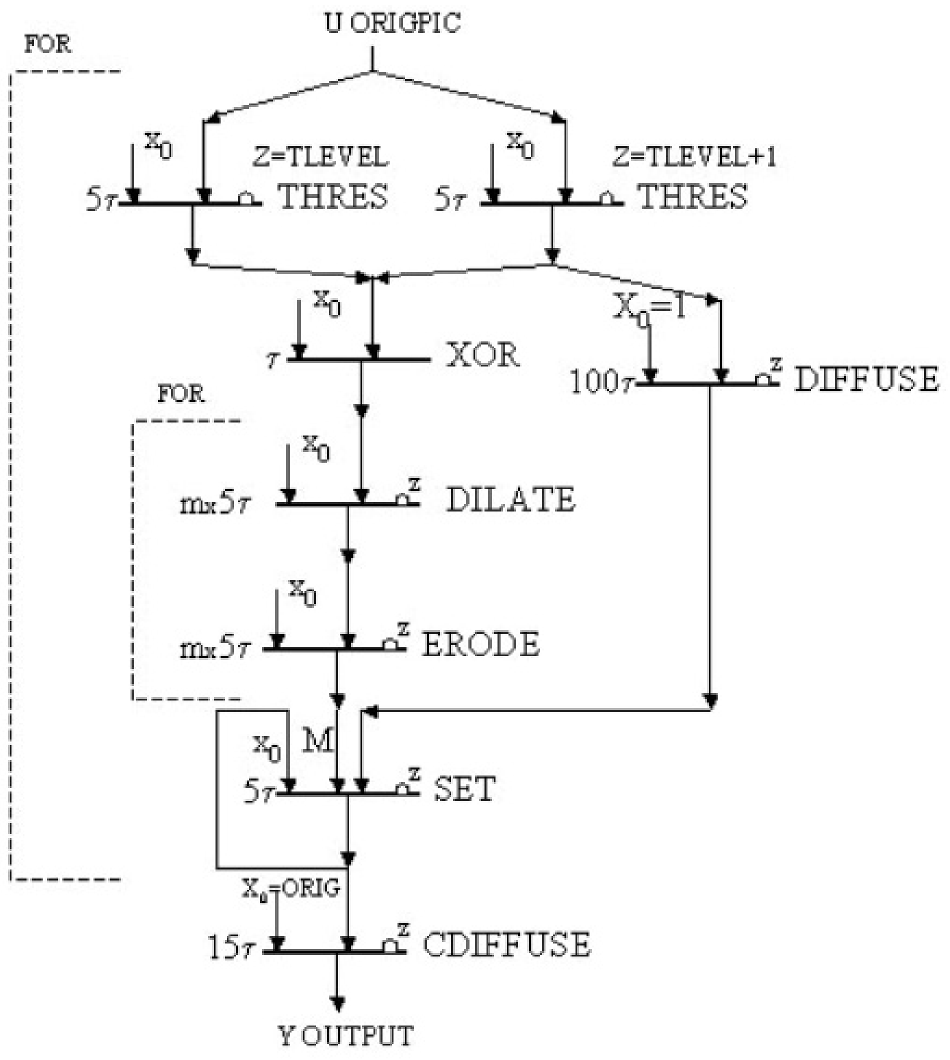


Figure 2.3: UMF (Universal Machines on Flows) description of Algorithm 2.

Nr. of level-sets	Nr. of morphology steps	Execution time (CNN τ)
2	0	242 τ
16	3	2276 τ
32	4	4852 τ

Table 2.1: Execution time depending on the number of level-sets and the number of morphological steps.

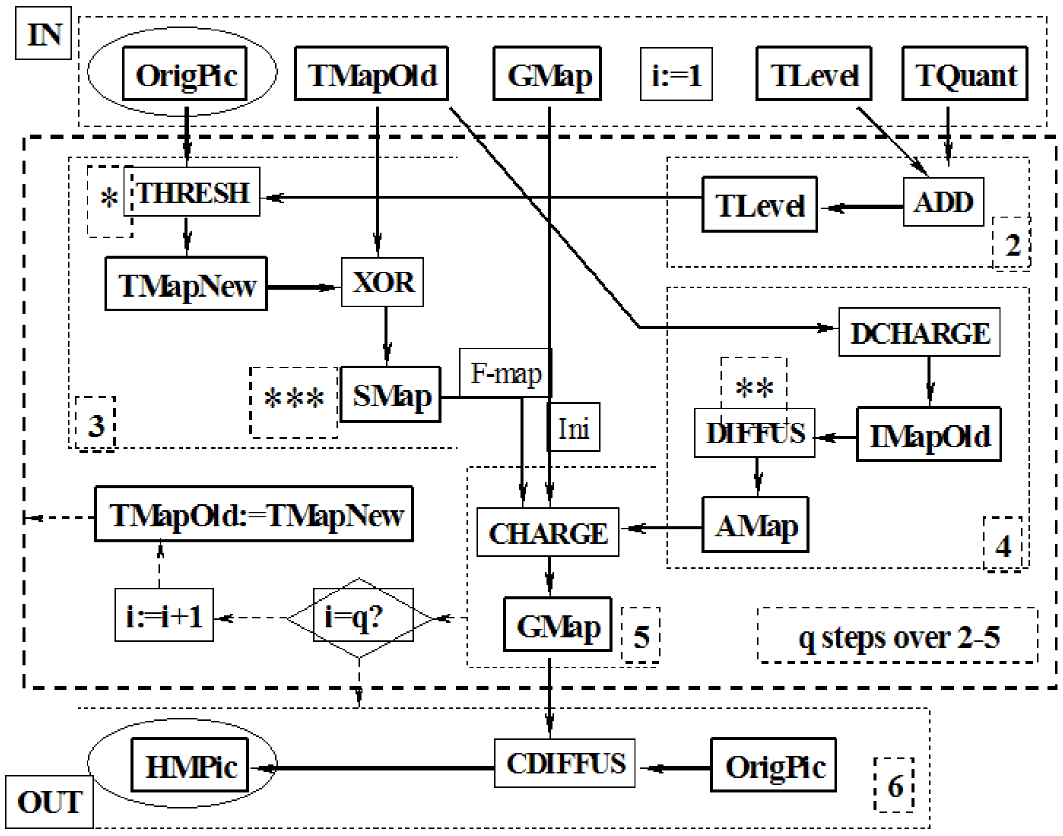


Figure 2.4: The CNN analogic algorithm approximating the histogram modification PDEs discussed in Section 4. The processing stages are as follows: (1) Initialization, (2) Set threshold level, (3) Detect regions above threshold, (4) Calculate area measure, (5) Set Global Bias Map pixels, (6) Modify image histogram with noise suppression (regularization). Blocks 2-5 should be repeated q times (the number of gray-scale levels that specify the accuracy of the approximation). In the flowchart *, **, and *** mark the stages where the subroutines implementing further adaptive strategies can be embedded. Morphological operations are embedded at *** processing the level-set function SMap.

level-sets (i) and the morphological steps (m) is $(20 + i(111 + 10m))\tau$, where τ is the time constant depending on the CNN implementation. The chosen optimal output has an execution time of 2276 τ (see Table 2.1).

2.6 Extended Algorithmic Schemes – Morphological Processing Of The Level Sets

An adaptive multi-thresholded output with shape enhancement can be obtained if a morphological processing is embedded at each gray-scale level considered. This could be implemented either through a multi-step erosion and dilation operations or using trigger-waves that approximate a continuous-scale binary morphology with flat structuring elements [32]. The description of the extended algorithm that contains the multi-step dilate-erode operations can be seen in Figure 2.2 (Algorithm 2).

2.7 Conclusions

In this chapter, I worked out nonlinear partial differential equation based parallel histogram modification algorithms for contrast enhancement and noise suppression. The morphological and wave processing steps of the CNN operate on a finite number of level-sets (Thesis 1.1). The steps of the algorithm are the following: for a given input image it produces XOR between two neighboring, thresholded level-sets and then morphological and wave operations are executed. This result is the current binary image. These binary images will be summed iteratively. The values of the pixels, covered by the current binary mask, are increased with a value proportional to the area of the current binary image (measured by diffusion). The outcome is provided after a diffusion filter (Figure 2.4).

Its application of medical imaging can give solutions (i) for real-time ultrasound image processing of echo-cardiographic diagnostics and (ii) MRI image evaluation. Experimental results can be found in the next chapter.

Chapter 3

Morphological Processing of the Level-Sets – Experiments

I describe and illustrate how the implementation of the algorithm results in an adaptive multi-thresholding scheme when histogram modification is combined with embedded morphological processing at a finite (small) number of gray-scale levels. This has obvious advantages if the further processing steps are segmentation and/or recognition. Comperative experimental results processing real-life and echo-cardiographic images are measured on different hardware/software platforms, including a 64×64 and 128×128 CNN-UM chips (ACE4k, ACE16k [27, 26, 33]).

3.1 Introduction

The resulting analogic (analog and logic) CNN [20, 26, 27, 34, 35, 36, 37] algorithms are implemented on different hardware-software platforms. Measurement results prove that various real-time image processing based solutions could use this technique as an efficient front-end scheme. Furthermore, it is demonstrated that the implementations accelerated by a CNN-UM chip [26, 27] are faster than the optimized solutions on a high-end DSP (Texas 6x).

Heavy engineering problem was to optimize the analog implementation of my algorithm to achieve as fast performance as possible. It was my goal to show that an efficient implementation was possible on existing CNN chips (Acex), and an adequate speed up can be achieved.

During the implementation, the optimalization of the use of analog memories had to be solved: minimalizing the outer data transfers (there is no need for extra data transfer), tuning parameters of the hardware needed for wave operations, and counting the linear combinations of images.

This chapter is organized as follows: In Section 2 experimental results are given for several different hardware-software platforms. Section 3 contains the parameters for CNN templates used in the algorithms. In Section 4 the implementation of the full algorithm is presented. In Section 5 conclusions can be found.

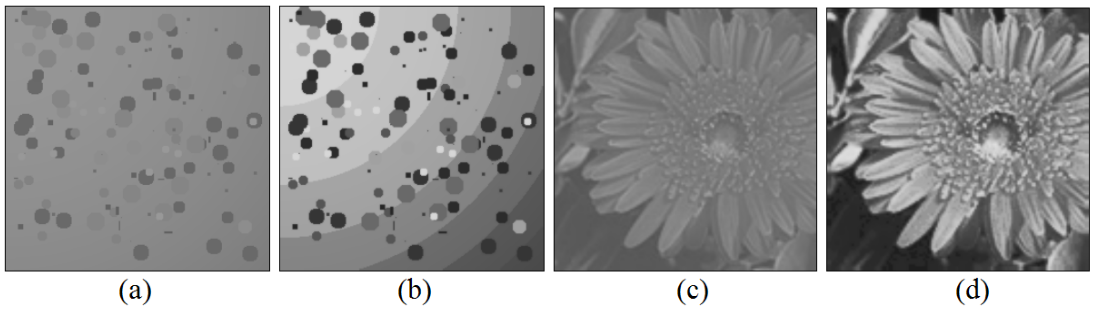


Figure 3.1: Programmable global PDE on a cellular nonlinear architecture – simultaneous contrast enhancement with noise suppression: original, low contrast image (a,c); enhanced – histogram equalized (b,d).

3.2 Experimental Results

For comparative analysis I have implemented the histogram modification method, discussed in Chapter 2, (Algorithm 2 without diffusion filtering, but including the morphological processing of the level-sets) in 6 different hardware-software configurations listed bellow (the 7th complete CNN-UM chip implementation is discussed in Section 3.4).

Version 1. in MATCNN MATLAB Toolbox simulating all analog CNN dynamics with an optimized C-code running on a CISC. μP: Pentium 1 GHz.

Version 2. as **Version 1**, except for simulating through the CNN fixed-points with an optimized C-code.

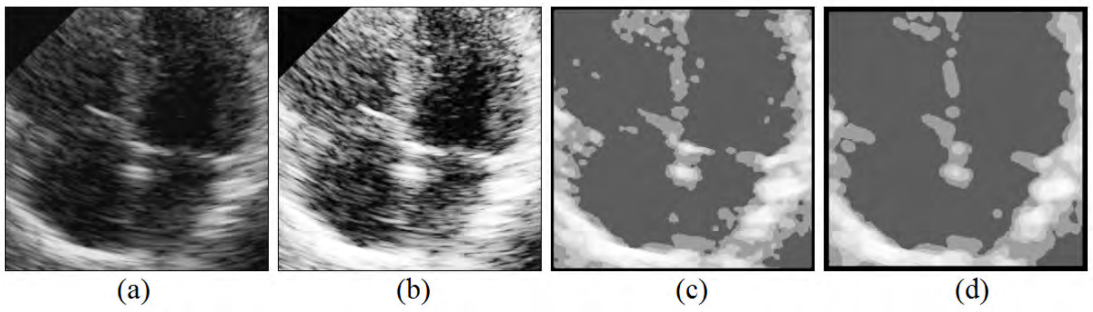


Figure 3.2: Programmable global PDE on a cellular nonlinear architecture – simultaneous contrast enhancement and noise filtering with embedded morphological processing of the level-sets: (a) original, low contrast image, (b) histogram equalized image, (c)-(d) histogram modification with embedded differential morphological processing (implemented through expanding and shrinking trigger-waves) at two different scales (3t and 5t) and 8 distinct gray-scale levels.

Version 3. as **Version 2**, except for the C-code contains the entire algorithm.

Version 4. in Aladdin Professional with an optimized C-code that contains the entire algorithm running on a DSP. μP: Texas TMS6x 250 MHz.

Version 5. as **Version 4**, except for the morphology operation is optimized at the assembly level.

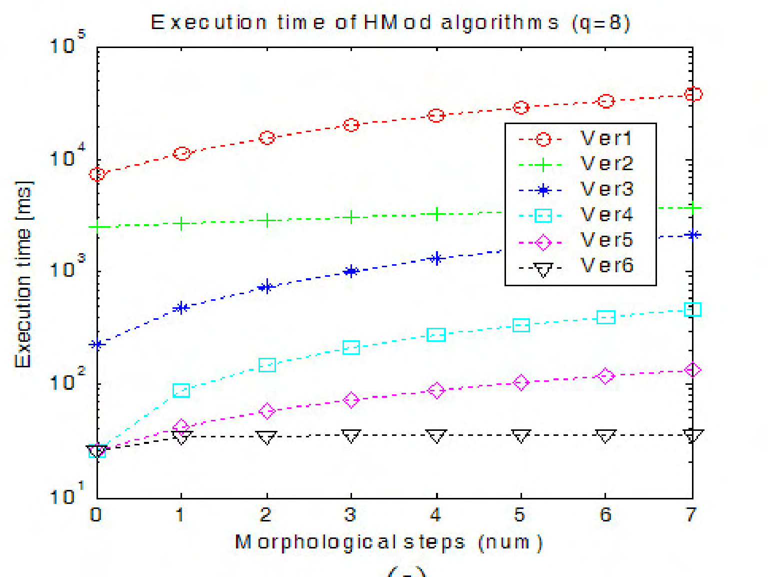
Version 6. as **Version 4**, except for the morphology operation is optimized for the CNN-UM chip. μP: ACE4k [26, 27].

Version 7. in Aladdin Professional running the entire algorithm on a CNN-UM chip. μP: ACE4k [26, 27].

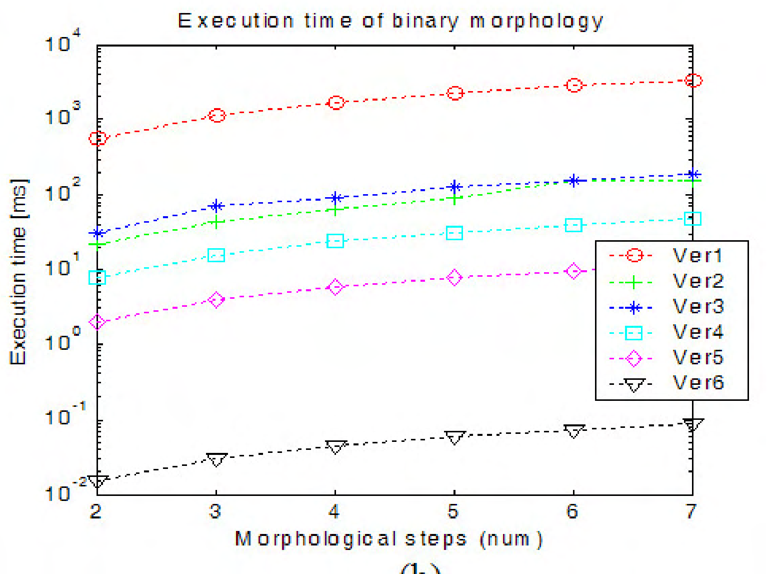
Detailed measurement results can be seen in Figure 3.3 Observe in Figure 3.3(b) that 7×2 binary morphology operations including logic are completed within 100 μs on the ACE4k CNN-UM chip.

3.3 CNN template parameters

The structure of linear templates used in the histogram modification algorithm can be seen in Equation (3.1). Matrix A and B are central symmetrical, some typical parameter values are in Table 3.1.



(a)



(b)

Figure 3.3: Comparison of different algorithmic implemenentations (Ver1-Ver6). (a) Execution time of Histogram Modification algorithms (q=8), (b) Execution time of binary morphology.

Table 3.1: Linear CNN templates used in the histogram modification algorithms. In boundary condition specification (B_c): ZF – zero flux, [-1,1] – constant.

	FeedBack (A)			Control (B)			Current	BCond
Template	a0	a1	a2	b0	b1	b2	I	Bc
THRES	2	0	0	0	0	0	0	1
SETLEV	0	0	0	1	0	0	0	0
EROSION	1	1	0	0	0	0	-4	-1
DILATION	1	1	0	0	0	0	4	-1
DIFFUS	0	0,15	0,10	0	0,15	0,10	0	ZF
CDIFFUS	0	0,15	0,10	0	0,15	0,10	0	ZF

$$A = \begin{bmatrix} a_2 & a_1 & a_2 \\ a_1 & a_0 & a_1 \\ a_2 & a_1 & a_2 \end{bmatrix}; B = \begin{bmatrix} b_2 & b_1 & b_2 \\ b_1 & b_0 & b_1 \\ b_2 & b_1 & b_2 \end{bmatrix}; z \tag{3.1}$$

I would like to present some results that show the effiency of the algorithm. The figures and their results executed with different parameters can be found in Appendix D in more detail.

3.4 ACE16k experiments and results

A number of related examples on artificial and real-life images can be found in Appendix D. The last experiments were performed on ACE16k chip whose 128×128 size and technical parameters make this chip unique on the market[33].

To take up a challenge, I attempted to implement the full algorithm on a CNN-UM chip without any external data transfer and with as many level-sets as possible. At a first step, I used AladdinPro software, where the AMC language provides access to both microprocessors of the Bi-i system, to ACE16k and Texas DSP. Some conventional operations, as multiplication or linear combination can be processed on DSP more easily. In this case, the external data transfer between the microprocessors had significant time loss, although improving the performance, the morphological and wave operations were running on the CNN chip. Experimental results can be seen in the middle column of Figure 3.5.

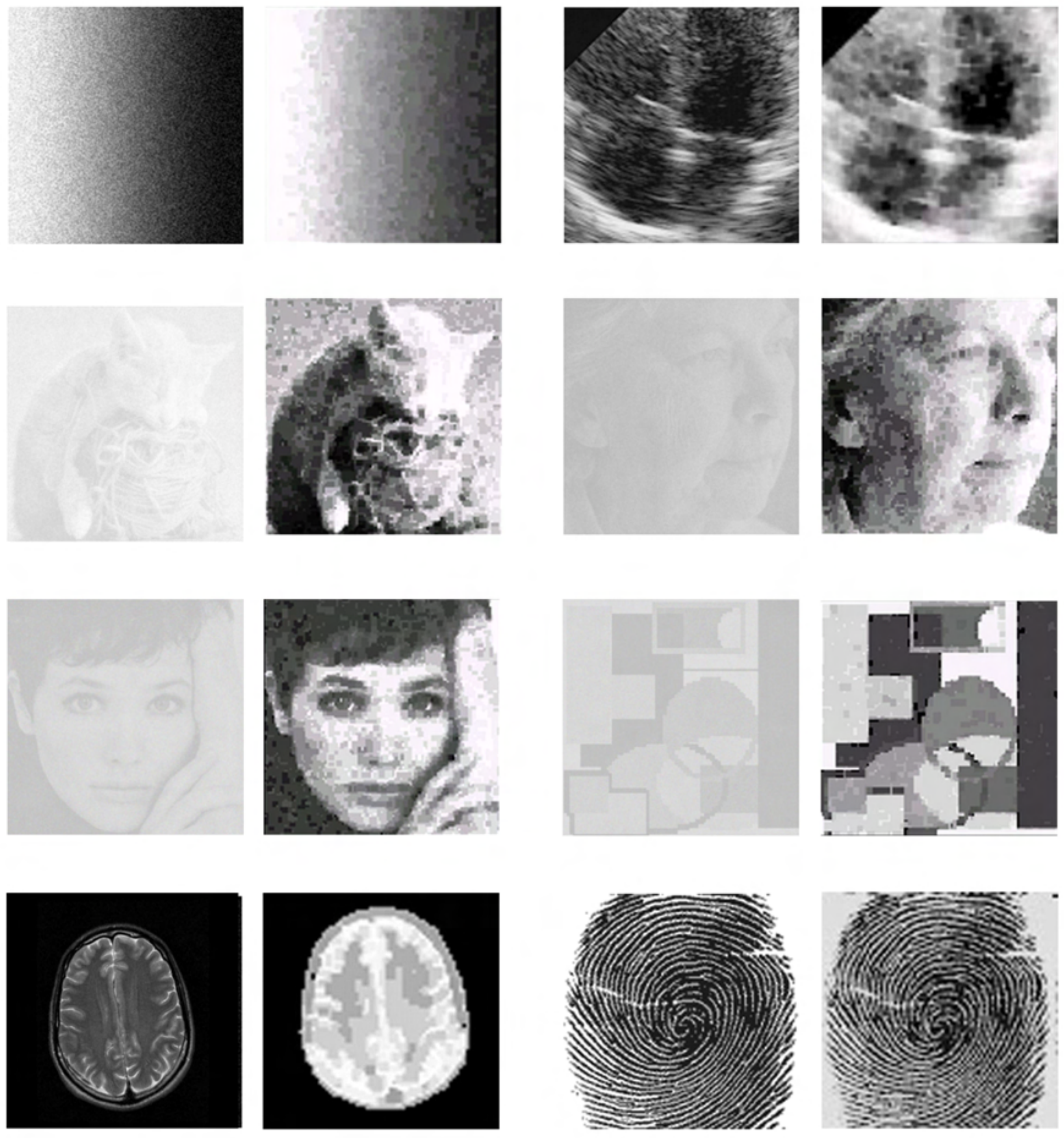


Figure 3.4: Simulation results for different inputs. More results can be seen in Appendix D. Left image is the input, right image is the result for each image pair.

After the morphological and wave operations were tested and tuned on the ACE16k chip, the next step was to implement the full algorithm on the chip. This implementation was a challenge, because in the course of long time data storage the memories of ACE16k lost data, so quality decreased and noise appeared in the stored images. The computing of linear combination of two images with adequate accuracy using only internal analog memories was a hard problem, too.

Having 128×128 resolution and 4 level-sets, the measured running speed of this implementation was 3.93 ms/frame. Results can be seen in Figure 3.5. Implementation codes can be found on the attached CD.

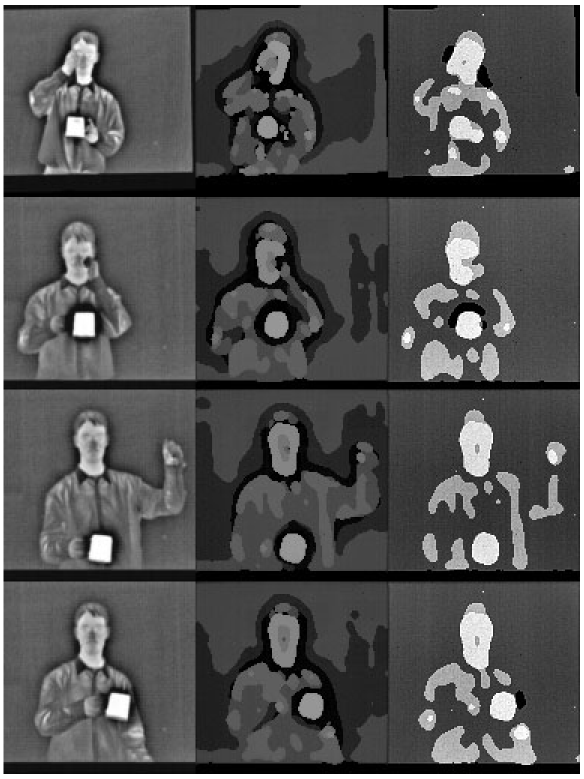


Figure 3.5: The results of the ACE16k chip experiments. In the first column some images of the original input image sequence can be seen. These were recorded by an infra camera. In the second column the results of the Bi-i system are presented in the case of 8 level-sets. This implementation combined the CNN-UM with a DSP. The third column shows those results which were the output of the ACE16k chip in the case of 4 level-sets. This implementation was achieved completely on the CNN-UM.

3.5 Conclusions

In this chapter, I have shown how a PDE, designed for simultaneous contrast enhancement, noise suppression, and shape enhancement could be implemented as an analogic algorithm relying on purely local operations. I have also performed an exhaustive comparative experiment of different implementations on various hardware-software platforms with CISC, DSP microprocessors and CNN-UM Ace4k and Ace16k chips.

I proved experimentally that it satisfies the theoretical expectation qualitatively and quantitatively to a good approximation. For 128×128 image resolution, a speed of 200 frame/s could be achieved (Thesis 1.2). I presented the typical parameters of linear CNN templates, which were used in the algorithm. I successfully implemented the full algorithm on the ACE16k chip without any external data transfer. Experiments on image sequences show the results.

Chapter 4

Immune Response Inspired Target Detection Algorithms

In this chapter I show that, similar to the nervous system and the genetic system, the immune system provides a prototype for a "computing mechanism". I present an immune response inspired algorithmic framework for spatial-temporal target detection applications using CNN technology [20, 53]. Unlike most analogic CNN algorithms [34, 53] here I will detect various targets by using a plethora of templates.

4.1 Introduction

During the last twenty years, the nervous system and later the genetic system have become useful "prototypes" for computer engineers in solving complex problems. "Neurocomputing" and "genetic algorithms" are now standard subjects taught in undergraduate curriculum. The third area having biological motivation and just emerging, is the application of the immune system, which may also play a similar role in the future [49, 50, 51].

Artificial Immune Systems (AIS) mimic the human immune system that has refined capabilities and methodologies, to build efficient algorithms that solve engineering problems. Moreover, our immune system possesses important properties, such as diversity, noise and fault tolerance, learning and memory and self-organization, which give it an advantage compared to other standard methods [49, 50].

During my research, I intended to design topographical algorithms and their experimental realization where huge number of target objects are monitored in real time to detect previously unknown events. So my goal was spatial-temporal novelty detection.

Novelty detection is the identification of new or unknown data or signal that a machine learning system is not aware of during training. Novelty detection is one of the fundamental requirements of a good classification or identification system since sometimes the test data contains information about objects that were not known at the time of training the model.

There are a multitude of applications where novelty detection is important including computer vision, signal processing, pattern recognition, data mining, and robotics [57, 58].

Several applications require the classifier to act as a detector rather as a classifier, that is, the requirement is to detect whether an input is part of the data that the classifier was trained on or it is in fact unknown. This technique is useful in applications such as fault detection [86], visual detection for mobile robots [83, 84], video surveillance [85], image region classification [56] and several others. Recently, there has been an increased interest in novelty detection as a number of research articles have appeared on autonomous systems based on adaptive machine learning.

Basically, there are two approaches of novelty detection: statistical based and neural network based approaches [57, 58].

Statistical approaches are mostly based on modeling data based on its statistical properties and using this information to estimate whether a test samples comes from the same distribution or not [57].

Neural networks have been widely used for novelty detection. Compared to statistical methods, some issues for novelty detection are more critical to neural networks such as their ability to generalize, computational expense while training and further expense when they need to be retrained [58].

To compare my work to the research areas of novelty detection, we can observe that it is rather statistical approach, because it has similar properties as statistical approaches have: easily re-trainable and the evaluation of the algorithm is based on the sub-patterns of the images.

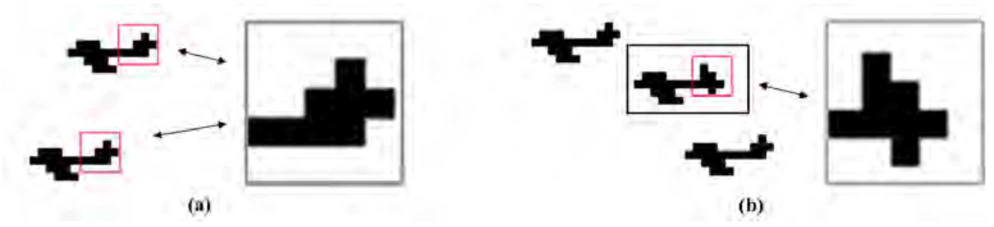


Figure 4.1: After the initialization phase (a), known input patterns keep the system tolerant and during the recognition phase (b) unknown objects can cause detection, if they can be differentiated from the unimportant noise.

The process of antigen presentation I used, gave several ideas for solving engineering problems, but, up to my knowledge, it has never been used for novelty detection problems based on image processing in a similar way. I have not found any similar processing methods in the literature of the AIS [82].

In this chapter I present immune response inspired algorithms for spatial-temporal target detection applications with extensions and a summary of my earlier work [13, 7, 8, 9]. Unlike in most analogic CNN algorithms [34, 53] here we are detecting various targets by using thousands of templates. These algorithms can be reasonably implemented only if we have a computer where thousands of elementary, fully parallel spatial-temporal actions can be implemented real-time. Fortunately, the recent CNN-UM [48, 40] based cellular wave computer and its visual microprocessor physical implementations, both CMOS [27, 38] as well as optical [75, 41], are ideally suited for this purpose.

In Figure 4.1., I show the basic idea. This is a simple example where the detection is based on the tail type of the planes. During the initialization the system was taught not to detect the planes with the first type of tail (Figure 4.1.a). Then, in the recognition phase, the planes which have the same tail, will not be detected, but any other type of tail, the unfamiliar ones (on Figure 4.1.b), will be detected.

The organization of this chapter is the following. First, I present the analogy of immune response and CNN algorithms for target detection. Section 3 introduces the specific target detection problem and its solution via a multi-template, immune response inspired CNN algorithm. In Section 4, I discuss some special feature extraction subroutines of my algorithm. Conclusions can be found in Section 5.

4.2 The Analogy of Immune Response and CNN Algorithms for Target Detection

A basic knowledge of immune systems [47] and cellular nonlinear networks [53] is recommended for reading this section. The reader may find more details in the Appendix.

The immune system is constantly in contact with its environment, and during this interaction, features are extracted and 3D molecule patterns are identified and detected by the process of immune response. This 3D dynamic pattern recognizer system has several other important properties (e.g.,memory), see Appendix E and [47, 46].

Humans might observe their environment through a visual process on 2D video flows recordeda and processed by their eyes and transmitted to the brain for further processing. Cellular Wave Computer based on the CNN-UM architecture is an efficient way for grasping the dynamics of the living visual system. The fast processing and computation performance of the vast numbers of immune cells can be mimicked by running a huge number of templates per second on the CNN-UM based physical implementation [38, 27].

My model is an attempt to use the analogy of 3D immune pattern recognizer, nature created immune system, to solve the dynamical object recognition and detection problem in 2D image flows.

4.2.1 Antigen Presenting

My model is based on the antigen presenting analogy. The short summary of this process is described in this section. Antibodies are complex molecules (created by B-lymphocytes and plasma cells,) that can bind to, deactivate and help remove antigens. After an antigen presenting cell (APC) e.g., B-lymphocyte, binds to an antigen through its antibody receptors, it ingests it. The antigen is digested into

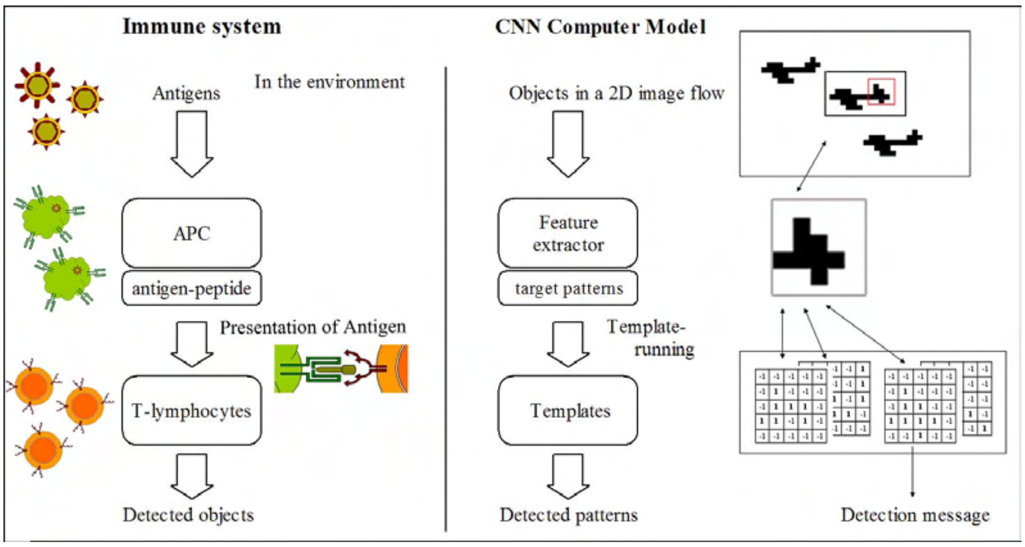


Figure 4.2: Antigen presenting and its CNN model

small pieces (called peptides), which are inserted into a groove in another type of molecule (MHC) and transported to the surface of the APC.

The T-lymphocyte also contains specific receptors on its surface, and it tries to bind to the APC, to its MHC-peptide. A successful binding sends chemical signals (e.g.,by cytokines) that stimulate the T-lymphocytes to divide.

On the left side of Figure 4.2, the procedure described above is illustrated. The antigens meet their match in APCs digested into peptides and presented to T-lymphocytes. If the T cells can bind to the APCs then it means that an antigen has been successfully detected.

The right side of Figure 4.2. is a diagram of my CNN model and the assigned notions are mapped. The 2D video flow is the input of the system – from which I extract the features using a feature extractor module. The extracted features depend on the current application, and exploit the advantages of the CNN-UM. In my model and example I worked with 5×5 or 3×3 binary patterns. In my work the chosen output gives a comfortable processing because the running of 5×5 templates on 5×5 patterns give unequivocal pattern recognition. The final output can be a binary value of detection. If any template fits into the pattern and recognizes it, then the target detection was successful.

My model defines the antigens and T-lymphocytes as two data items with

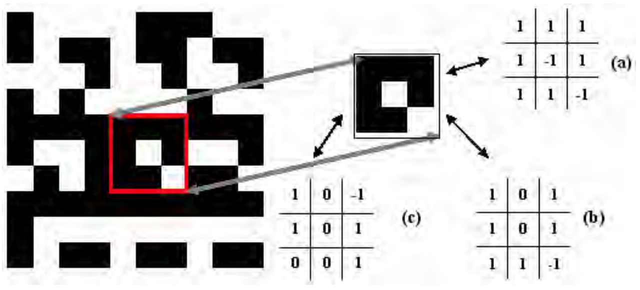


Figure 4.3: If in the position of *1*s black pixels and in the position of *-1*s white pixels are found, the matching is successful, while pixel color in the position of the *0*s is indifferent. Therefore (a) and (b) give successful matching while (c) is unsuccessful.

different characteristics and goals. The antigens can be represented by $n \times n$ sized binary (black and white) matrixes. Colors can be coded with *1* (black) and *-1* (white) numbers. Each antigen is usually a 3×3 or 5×5 subpattern of a binary picture which is extracted from the input image flow by a special feature extraction method. These patterns (2D-strings [81]) can be recognized by my T-lymphocytes, called match-templates [42]. They are usually 3×3 or 5×5 matrixes and contain *1*, *-1* and *0* numbers. During the interaction between templates and patterns, if in the position of *1*s black pixels and in the position of *-1*s white pixels are found, the matching is successful, while pixel color in the position of the *0*s (or otherwise "don't care" elements) is indifferent. An example can be seen in Figure 4.3.

4.2.2 The Parallel Notions

The essence of my model is based on the presentantion of antigen-peptides. For a better understanding, it is worth comparing the parallel concepts of my model again. The members of this mechanism correspond to the objects of my CNN algorithm. The summary of these parallel notions can be seen in Table 4.1.

The equivalent of the antigen is the 2D image flow with various objects. This may contain both non-pathogen objects and pathogen objects. The patterns extracted from the 2D video flow, which are parallel to the peptides, can be effectively processed by CNN-UM. The function of B-lymphocyte corresponds to

Table 4.1: Mapping between the immune system and the CNN algorithm.

Immune System	CNN algorithm
Antigen	Various objects or events in a 2D image flow
Non-Pathogen	Non-dangerous objects or events in a 2D image flow
Pathogen	Dangerous objects or events in a 2D image flow (targets)
Antigen-peptides	Patterns and/or properties of the objects or events, e.g., 5×5 patterns
APC	Feature extraction module
T-lymphocyte	Templates
Antibodies	(flags for targets on the objects in the 2D flow)
Complement	-
Cytokines, Tc, NK	Detection message
Memory cell	Specialized template with several recognition
Recognition	Template matching
Life of an organism	Number of interactions
Affinity measure	Number of “don’t care” elements

the feature extractor, which is a problem specific module. It converts the gray-scale or color input flow to binary. The T-lymphocytes – which carry out the recognition – are parallel to the templates. The role of the antibodies is not very important in my model, after the recognition of patterns they can be regarded as flags in the input flow.

The complement and the innate immune system (see Appendix E) do not play a significant role in my model. The message signal has little importance at this point. The most essential is the detection message which occurs during a successful detection. Memory cells are specified templates with several recognitions, where recognitions are successful template matchings between the data items. The life of an organism can be symbolized by the number of interactions. The number of “don’t care” elements can characterize the efficiency (matching affinity) of a template.

4.2.3 Artificial Immune Systems (AIS)

“Artificial immune systems (AIS) are adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving" [50]. For the AIS, immune cells and molecules are represented by data items which take apart in some general-purpose algorithm that models specific interactions and aspects of the immune systems.

In the AIS [49, 50], antigens, lymphocytes and any molecules have a generalized shape m in shape-space S. This generalized shape can be represented as an attribute string – set of binary or other coordinates – of length L. Therefore any molecule string m can be regarded as a point in an L-dimensional shape-space. The interaction of an antigen and a peptide is assessed via a common (Euclidean, Manhattan, Hamming, etc.) D distance measure, which is also called an affinity measure, between their proportional strings. Usually a relation is defined between the distance D and the recognition region V_ε proportionally to the recognition threshold or cross-reactivity threshold ε. If the D distance measure between data items is larger than ε, then a successful recognition is assumed between the items.

In my model the S shape-space has 9 or 25 dimensions, because the sub-pattern matrixes can be represented by 25 or 9 long binary vectors and the templates correspond to 25 or 9 long vectors (coordinates can be -1, 1, or 0). The distance measure between an antigen ($Ag = \langle Ag_1, Ag_2, ...Ag_L\rangle$) and a template $T = \langle T_1, T_2, ...T_L\rangle$ is

$$D = \sum_{i=1}^{L} \delta_i, \text{ where } \delta_i \begin{cases} 0, & \text{if } T_i = Ag_i \text{ or } T_i = 0, \\ 1, & \text{if } T_i \neq Ag_i \end{cases} \tag{4.1}$$

My match template class does the recognition if and only if the D distance is *0*. Contrary to common AIS, where the molecules usually are represented by similar vectors, the sub-patterns and template vectors generally differ in my model. There are "don't care" elements in the templates, whose position are fixed within their vectors. Therefore, I could not give a definition of affinity as other AISs have. If a match-template has d "don't care" elements, it can detect 2^d different sub-patterns. The more "don't care" elements it has, the more different sub-patterns are, which are detected. Therefore, the affinity of a template can be characterized by the number of the "don't care" elements. This affinity is called template affinity α. This affinity has a similar effect to the usual affinity or cross-reactivity threshold in AIS. A sub-pattern can be matched successfully by $2^L = \sum_{\alpha=0}^{L} \binom{L}{\alpha}$ different match-templates, where α is the affinity, defined formerly. The maximum number of sub-patterns that can be recognized by a template set is $\sum 2^{\alpha_i}$, where the α_i is the template affinity of ith template of the template set. During the recognition phase a successful template is cloned changing one of its *0* elements to *-1* or *1*, where the 0 element has been chosen by uniform random process, therefore the clones are more specific and two new templates are added to the template set. The lifetime of a template is extended if it is successful and its specificity reaches a given threshold ($d \leq 1$). Parallel to the recognition process, the actual template repertoire can be expanded with new templates using negative selection. It is also beneficial to refresh the template repertoire by replacing the unsuccessful old templates with low template affinity ones.

4.3 A Target Detection Problem and Its Algorithm

The immune system endeavors to solve a target detection problem where the objects to be detected are not predetermined. These objects can be numerous. The system has to react as quickly as possible to distinguish between non-pathogen and pathogen objects to protect the body from the pathogens.

The algorithm in Figure 4.4. is an attempt to provide a framework to solve the target detection problem below as effectively as the human immune system.

The algorithm has two parts – initialization and recognition which sequentially follow each other. In the course of initialization, a “non-dangerous” template set (T cells) is created. This template set contains templates which are not able to recognize the initial objects – the non-dangerous objects. This process is called negative selection [79, 80]. The output of the negative selection – templates – performs the recognition in the second part. The randomly chosen templates (lymphocytes in the bone marrow) are tested (in the thymus) against a pattern flow which is extracted (by the APCs) from the initial 2D image flow. Those templates, which do not match any of the patterns, are selected as the “non-dangerous” ones.

In the recognition phase, every member of the selected template-set tests the actual pattern extracted from the input image flow of the recognition part,

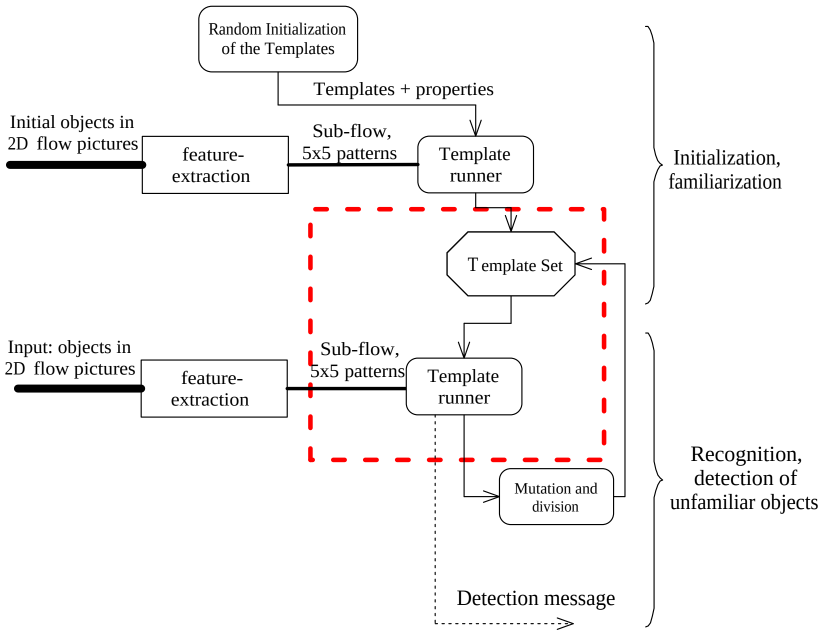

Random Initialization of the Templates
Templates + properties
Initial objects in 2D flow pictures
feature-extraction
Sub-flow, 5x5 patterns
Template runner
Initialization, familiarization
Template Set
Input: objects in 2D flow pictures
feature-extraction
Sub-flow, 5x5 patterns
Template runner
Recognition, detection of unfamiliar objects
Mutation and division
Detection message

Figure 4.4: Target detection CNN algorithm.

and if it recognizes the unfamiliar pattern then a detection message is generated. These templates are called prosperous and get higher priority, and thus have more opportunities to recognize patterns. The mutation and division module operates on the templates and its sequence to improve the effectiveness of the algorithm. The steps of the algorithm can be followed in the tail example in Figure 4.5.

4.3.1 Initialization part of the Algorithm

The details of the former immune response inspired CNN algorithm framework (Figure 4.4) is described by a Universal Machines on Flows (UMF) flow chart. The general UMF diagram of initialization or otherwise selection part of the algorithm can be seen on Figure 4.6. In my first experiments I worked with simple feature extraction and selection subroutines. Some more complex feature extraction subroutine can be found in section 4.4. In the first tests my feature extraction subroutine was empty (i.e. I began with binary images) and the selection subroutine contained different match templates, therefore, the recognition was solved by its pattern matches. In case of 3×3 matching templates, the initial template set

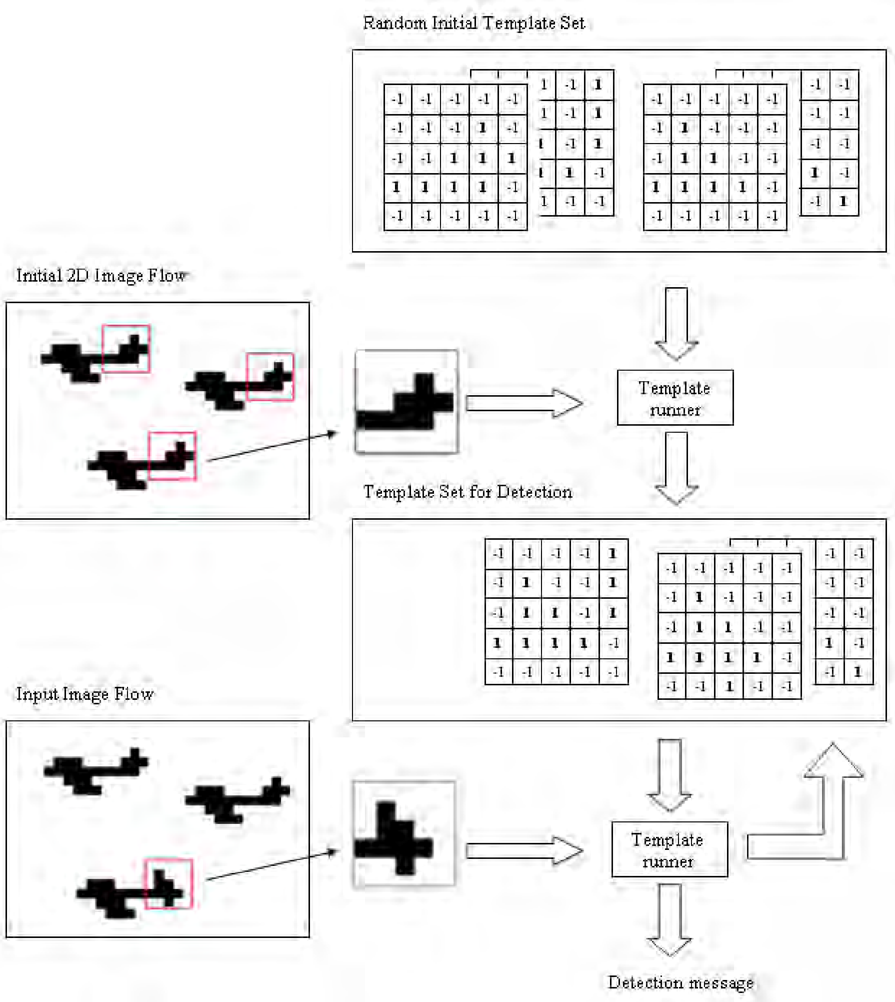


Figure 4.5: AIS based algorithm framework for a simple example. The random initial template set is generated by the bone marrow model. After negative selection of the initial 2D image flow, the template set is prepared for detection. In the recognition phase, in the event that any members of the template set are able to match the actual pattern with a given threshold, then a detection message is generated. The influence of mutation is provided by a loop-back from the template-runner to the template-set.

(H_{TEM_INI}) contains the following match templates [42]:

$$H_{TEM\_INI} = \left\{ \begin{array}{l} MATCH(p)\,|b_{11} = 1; \underset{i\neq 1 \wedge j\neq 1}{\forall} i,j\ b_{ij} = 0; \forall k,l\ 0 \leq k \leq 1, \\ 0 \leq l \leq 1,\ a_{kl} = 1 \vee a_{kl} = 0 \vee a_{kl} = -1;\ z = \sum\limits_{k,l} |a_{kl}| - 0.5 \end{array} \right\} \tag{4.2}$$

where p is the index of the elements and $|H_{TEM\_INI}| \triangleq Cp$ therefore $0 < p < Cp$. The goal of the initialization part is to obtain a set of only "non-dangerous" templates. These templates must not recognize the self input, the input of the initialization part. Thus, those templates which can recognize any of the input frames are H_{TEM_INI} . For the output set

$$G = \{p\,|MATCH(p) \in H_{TEM\_INI};\ Gp \neq 0\} \tag{4.3}$$

where Gp is the detection result of the selection subroutine and G contains the indexes of non-selected templates. (In Equation (4.3) Gp sometimes has to be bigger then a given threshold) The resultant non-dangerous template set is the following:

$$NDT = \{MATCH(p)\,|p \in G\} \tag{4.4}$$

therefore this set is the input of the recognition part: $H_{TEM\_REC} = NDT$ and $|H_{TEM\_REC}| \triangleq Cq$

4.3.2 Recognition part of the Algorithm

In both parts of the algorithm, the input pictures are processed by the template elements of the corresponding H set until the match. These loop mechanisms can be observed in Figure 4.6. and Figure 4.8 and helped by parameters p and q.

There is a difference between the recognition part and initialization part of the algorithm in the mechanism of the loop process. In the recognition part we do not need to process all the elements of the corresponding H-set because one match is enough for a successful recognition.

For generality the mutation and division subroutine is accomplished. One of its potential realizations can be seen in Figure 4.9.

In the mutation and division subroutine of my model, the winning template may change its position. The successful template and its clones will be placed

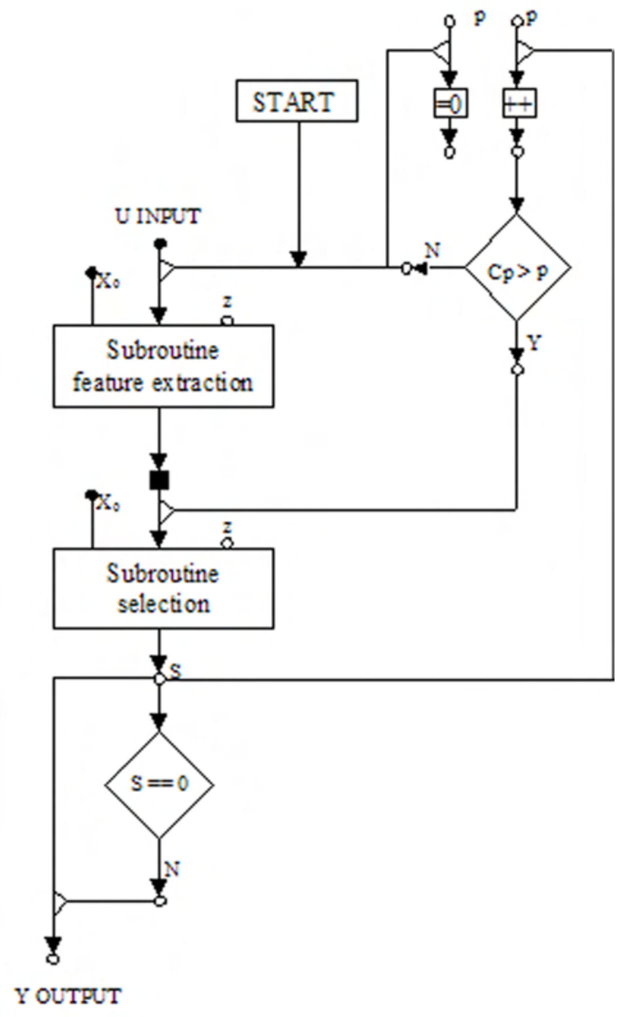


Figure 4.6: The UMF diagram of the selection part of my immune response inspired CNN algorithm.

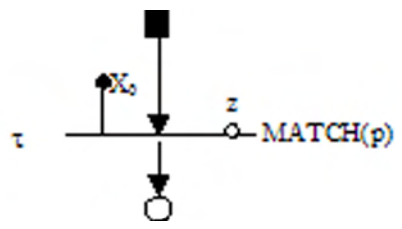


Figure 4.7: The UMF diagram of my subroutine of selection.

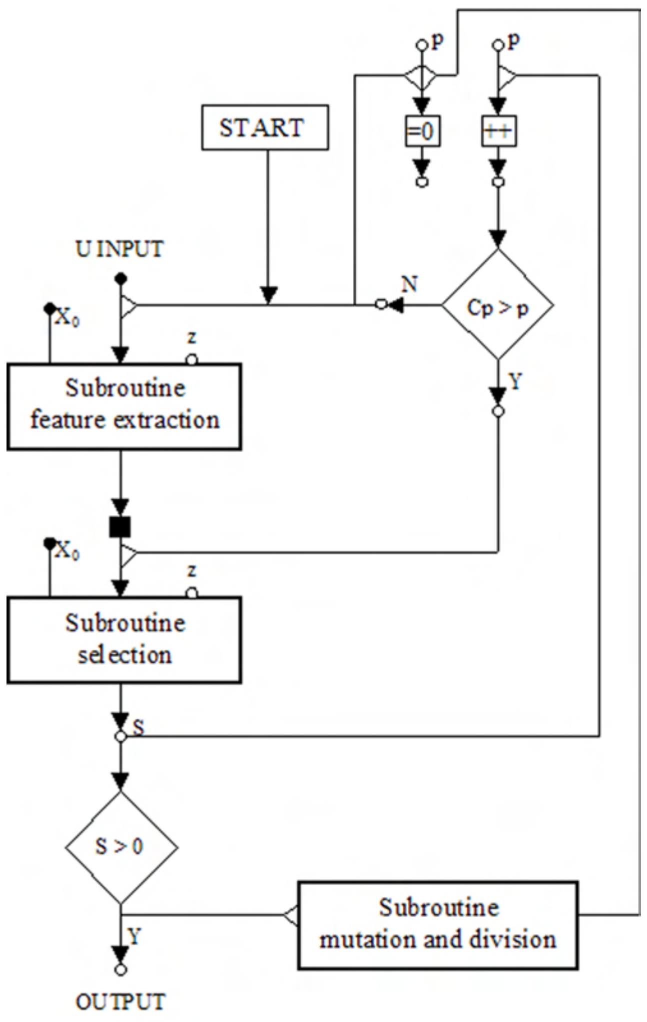


Figure 4.8: The UMF diagram of the recognition part of my immune response inspired CNN algorithm.

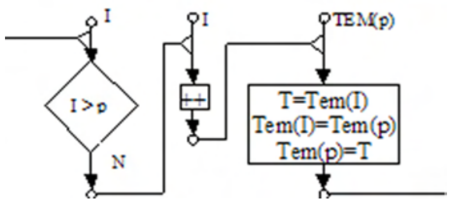


Figure 4.9: The UMF diagram of my subroutine of mutation and division.

at the beginning of the execution order, therefore efficient templates have an advantage over the other ones. They process the input earlier and help the entire algorithm to run faster.

4.4 Feature extraction subroutines

In my model, in the event that I have 3×3 match-templates in my selection subroutine, the feature extraction subroutine is responsible for converting grayscale or color input images to binary patters, where each 3×3 sub-pattern contains as much information as possible. First, let us suppose that the input image is grayscale and the templates are 3×3. In the algorithm, the input image is threshold nine times at different levels:

$$THRES(i) = \frac{2i}{9} - 1 \qquad i = 1, 2, ..., 9 \tag{4.5}$$

Each element in the 3×3 binary pattern is defined by a given threshold result. Practically, each threshold result is AND-ed to its mask to select its position in the 3×3 sub-pattern.

$$MASK(i)_{kl} = \begin{cases} 1, & k = i \bmod 3 + 3m;\ \ l = i\, div\, 3 + 3m; \\ & m \in Z;\ 0 \leq m \leq 21 \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

where k,l are the coordinates of pixels. All these masked binary images are logically OR-ed to give the output result. The UMF description of this algorithm can be seen on Figure 4.10b.

The second case is to combine different input images, that is the three color channels (red, green, blue) of the input images. The method is similar to the gray-scale case, only there are three threshold levels and each input set defines only three pixels in the pattern. The UMF description of this algorithm can be seen on Figure 4.10a.

Although in my experiments I used 3×3 or 5×5 matching templates in selection modules, it is possible to match and recognize different sizes of patterns. Using simulators, it is straightforward to use 5×5 templates but it is also possible to find and recognize binary templates of any size and shape[8]. This decomposition method is based on combining 3×3 templates. The templates can overlap

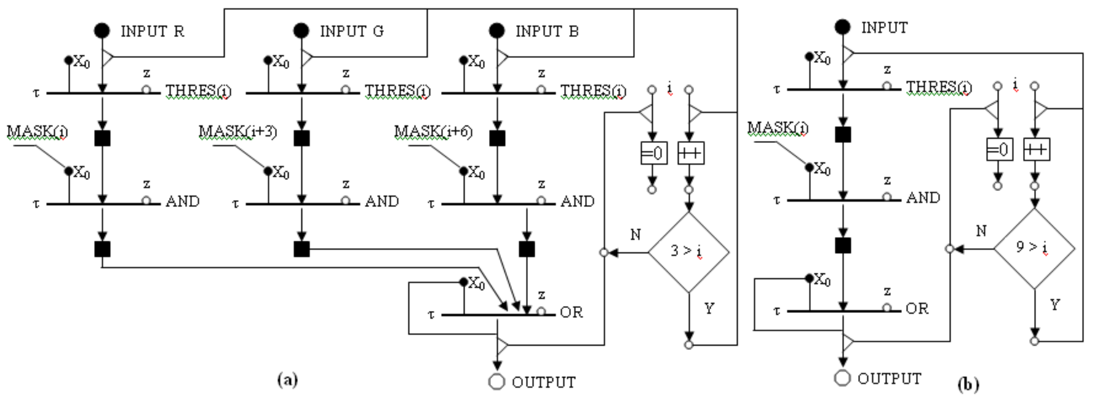


Figure 4.10: (a) The UMF diagram of the feature extractor subroutine for color, (b) for grayscale input images.

Table 4.2: Comparison of different matching templates. Note that same template combinations (arrangement can be seen on Figure 4.11) can cover different amount of pixels depending on their overlapping topology (3. and 4. row).

Size of templates	Nr. of templates	Nr. of pixels	Nr. of different patterns	Nr. of different templates
3×3	1	9	512	19683
5×5	1	25	33554432	847288609443
3×3	2	12	4096	531441
3×3	2	14	16384	4782969

each other, but no inconsistency is allowed in the overlapped area, which means that 1 and -1 elements are not allowed in the same pixel position. Only one of them can be other than 0 ('don't care').

For example an interesting and important case can be using two 3×3 templates overlapping each other and building a pattern which contains fourteen pixels, see Figure 4.11. Let us compare the different properties of these matching templates in Table 4.2.

The variations of 3×3 patterns may be less than the application needs, but 5×5 patterns give more variations than I could cover in real time. Note that these match templates contain "don't care" elements, so in order to recognize all variations of binary patterns, I do not need all templates. In the event that

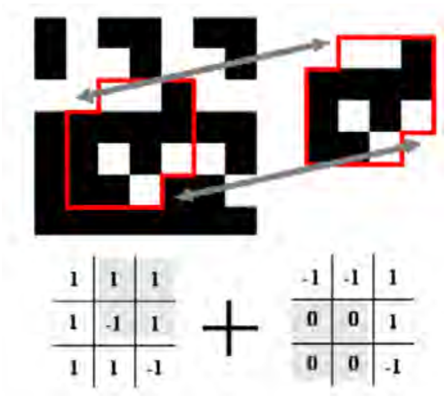


Figure 4.11: A potential decomposition for the bordered pattern by two 3×3 templates. Note that the overlapped area is darker and contains “don’t care” elements.

two or more templates are being used to recognize more complex patterns, the computational cost will be greater, because we need less template combinations.

4.5 Conclusions

The immune system is a source of inspiration for developing intelligent problem solving techniques [49, 50, 52]. In this chapter, I introduced an immune response inspired algorithmic framework and showed that the functional model of human immune response can be described in a cellular network (CNN) algorithmic framework (Thesis 2.1). I have presented the analogy of immune response and my CNN algorithm for target detection problems in image flows. After shortly summarizing the theory and mathematical description of my model from the point of AIS, the subroutines of my algorithm framework are described. Also sample feature extraction methods are presented.

Its application of surveillance can give solutions for real-time image processing of novelty detection. Experimental results can be found in the next chapter.

Chapter 5

CNN-UM Implementation and Experiments

The algorithms introduced in the previous chapter can be implemented successfully only by using a computer upon which thousands of elementary, fully parallel spatial-temporal actions can be implemented in real time. My tests show a statistically complete success rate, and I present a particular example of recognizing dynamic objects. Results from experiments in a 3D virtual world with different terrain textures are also provided to demonstrate that the system can detect unknown patterns and dynamical changes in image sequences. Possible applications of this algorithm include in explorer systems for terrain surveillance. Hardware experiments are also demonstrated on ACE16k CNN-UM wave-computer.

5.1 Introduction

In my experiments, the CNN-UM processor runs not only the sensory preprocessing algorithm, but the learning and recognition methods, too, which in cooperation with digital algorithms allows for fast image processing in real-time applications.

Section 2 presents an experiment with several hundreds of different targets and the use of several thousand templates to detect them. The statistical success rate is practically 100%. In section 3, I show an example which is a special application of my algorithmic framework for dynamic object detection. Section 4 describes a real time surveillance application implemented in the Aladdin Pro environment utilizing the ACE4k chip [27, 38] and gives time measurements. In Section 5, applied mutation subroutines are presented. Section 6 shows the result of my experiments, which were performed on the ACE16k chip. In Section 7, conclusions can be found.

5.2 An Experiment and Its Evaluation

In this first experiment my goal was to test the kernel of the algorithm which were introduced in Chapter 4, the template runner (in Figure 4.4.) and measuring its speed performance. One of the main points is to determine the necessary number of templates, which is a critical value of applicability.

Over the course of experiment, the defending "T-lymphocyte" templates were randomly chosen as 5×5 match-templates. The objects in the input flow (antigen-peptides) were 8×8 binary pictures, each containing randomly chosen 5×5 patterns. These patterns have to be recognized by my system.

In these first tests, all the input patterns are offensive objects. If an unidentified object is recognized by one of the templates then it is eliminated. If the number of unrecognized offensive objects is over a critical value, then the defense was unsuccessful. If all the offensive objects are eliminated, then the defense was successful.

There are 500 input patterns and 2000–5000 templates in each test. My goal was to set the system parameters to recognize all the input objects. My tests were performing in a CNN simulation environment called Aladdin Professional 2.4.

5.2.1 Results of the Tests

The pattern and template series were different. The results of the recognition phase can be seen in Table 5.1. In four cases the randomly generated match-templates could detect all the patterns. The maximum number of different templates is 3^{25} (=847288609443), while the maximum different patterns can be 2^{25} (=33554432). My result shows that 5000 randomly selected templates can recognize all the randomly generated patterns, because the zero, "don't care" value in the templates helps to successfully recognize the patterns.

Table 5.1: Result of first test. Number of unrecognized patterns are given for different pattern and template series.

Name of Pattern-series (500 in each)	Name of Template-series (2000 in each)	Number of unrecognized patterns
A	a	56
B	b	15
C	c	0
D	d	0
E	e	0
F	f	73
G	g	13
H	h	1
I	i	0

In the second test, I performed the same template series on different pattern series. As shown in Table 5.2, a template series gives similar result with different pattern series. Template series number c could recognize all the patterns in pattern-series A-F because this template set may contain some templates which have a lot of zero value.

In the third test, I tried to calculate the size of template series that are large enough to recognize all patterns in the pattern series where each pattern series contains 500 patterns. I can not promise that every randomly generated template series can cover the whole pattern target space but my result gives a statistical estimation of the number of templates which is usually practical.

The results in Table 5.3 show that at least 5000 different templates are necessary to match all the 500 different patterns. 5000 templates out of 3^{25} could detect all the 500 patterns out of 2^{25}.

5.2.2 Theoretical aspects

In this section, I give an approximation based probability analysis of the needed computational power, estimating the size of the template set. If a template contains k 0 (don't care) elements then it can recognize 2^k different patterns. The more 0 elements it has, the greater the number of patterns it recognizes,

Table 5.2: Result of second test. Number of unrecognized patterns are given for different pattern with same template series.

Name of Pattern-series	Name of Template-series	Number of unrecognized patterns
A	a	56
B	a	52
C	a	48
D	a	55
E	a	56
F	a	43
G	a	57
H	a	47
I	a	58
A	b	15
B	b	15
C	b	24
D	b	17
E	b	19
F	b	14
A-F	c	0

Table 5.3: Result of third test. A statistical estimation of the size of template series that are large enough to recognize all patterns in the pattern series where each pattern series contains 500 patterns.

Name of Template-series	Size of Template-series	Number of unrecognized patterns
a	2000	30
a	3000	9
a	4000	1
a	5000	0
b	2000	28
b	3000	7
b	4000	3
b	5000	0

but the specificity will be needed. If we have a set of templates, the maximum number of patterns that can be recognized by this set is

$$\sum 2^{\alpha_i} \tag{5.1}$$

where α_i is the number of the 0 (don't care) elements of the ith template of the set. I evaluated these summarizations for my different template series (a,b, etc.) and the results were around 6 million. As we know, the different patterns can be $2^{25} =$ 33 million. Let us examine this problem from another point of view. The probability that a random template contains 0 don't care elements, or in other words, that it can match only one pattern, is

$$\frac{2^{25}}{3^{25}} \tag{5.2}$$

The probability that a random template can match exactly 2^k patterns is

$$\frac{\begin{pmatrix} 25 \\ k \end{pmatrix} 2^{25-k}}{3^{25}} \tag{5.3}$$

If we summarize this formula by multiplying with the appropriate value (2^k), we can get the mean value ($\cong 1328.8$). The mean value is how many patterns will be covered by a template. This result shows, that 5000 templates can match with approximately 6.6 million patterns.

Comparing the experimental results to the theoretical analyzes, my result is different. This difference perhaps exist because 500 samples of 33 million is not a significant representation. Note that in my experiment also all the elements were chosen randomly. The theoretical value of the size of the template set is around 25000.

If I use 3×3 sized templates, based on a similar theoretical proof the size of the template set should be at least around 39. It is not proved that this set will cover all of the same sized sub-patterns, but it gives a good order of magnitude. It must be noted, however, that any sub-patterns can be recognized by special decomposition of 3×3 templates [39]. The templates can overlap each other without any inconsistency, which means that -1 and 1 are not allowed in the same overlapped position. Only one of the overlapped values can be other than

Table 5.4: Comparison of time necessities between Pentium PC and ACE4k.

	Simulation	Real hardware
Processor:	3Ghz Pentium IV.	ACE4k
Input size:	64×64	64×64
Number of templates:	5000	5000
Template size:	3×3	3×3
Software:	Aladdin 2.4	Aladdin 2.4
Time:	312 sec	0.4 sec

0. An example of overlapping decomposition can be seen in Figure 4.11. In this case where the pattern contains 14 pixels the size of the template set should be around 292.

5.2.3 Time measurement

The running time of my algorithm is a significant parameter. Therefore, I measured its running time of the selection subroutine both in simulation and an ACE4k chip environment. The comparisons of the different environments and my results can be seen in Table 5.4.

We can observe that the hardware requires significantly less time. These results show that my model can be implemented in real-time applications.

5.3 A simple example

My goal was developing a system which is able to recognize unexpected, irregular, dynamic events. Based on my former model and algorithmic frame I would like to show with a simple example that my theory is adaptable to real applications. This example explains how special, dangerous, or abnormal movements of dynamic objects can be detected. I used a simple car racing game where the movement of cars was followed on an a priori path. Normally, they do not go off their given path. In this example using my model the system was able to recognize abnormal

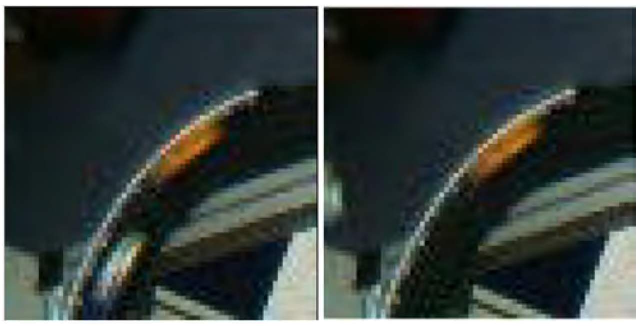


Figure 5.1: Sample pictures of the 2D input image flow. The difference is that on the second image the white car is missing.

events after a short learning period- if the car moved fast, it went off its path. My experiment was implemented in Aladdin Professional 2.4 simulator environment.

5.3.1 The algorithm

The algorithm frame of the example can be seen Figure 4.6. and Figure 4.8. The source of the 2D image flow input was a common web camera with 30 fps image flow. The moving objects were detected on the input flow and their centroid and position were calculated by an algorithm defined via AMC code. The coordinates of the position of each detected object have been converted into 3×3 patterns using their binary code.

These 3×3 patterns were transformed into pictures and the randomly generated match template set (contained 1, -1 and 0 numbers) was running on the set of these pictures in the initialization phase. The result of this phase was a template set that contained templates in which none of them recognized any of the input patterns.

In the recognition phase of my example, the input image flow was created as it was in the initialization phase but the speed of the cars was increased, which causes some accidents and this result gave that the cars went off their paths. All these “dangerous” events were detected successfully.

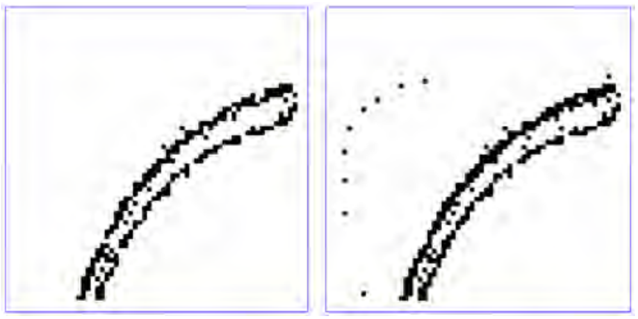


Figure 5.2: Cumulative figures of dynamic objects in initialization and recognition input flow.

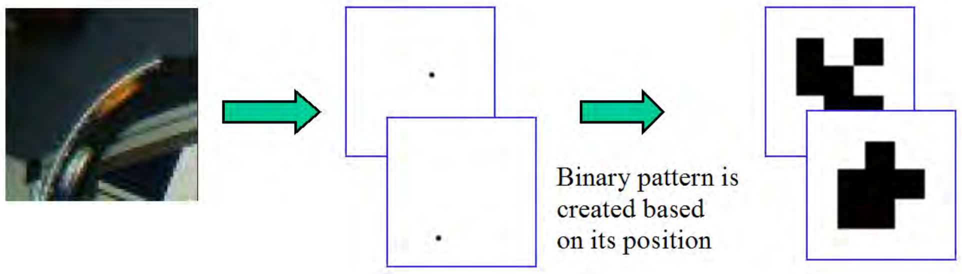


Figure 5.3: Steps of the feature extractor module.

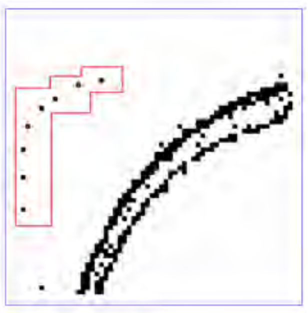


Figure 5.4: The centroids of detected objects are in the enclosed area. The system was able to recognize abnormal events after a short (e.g., few seconds long) learning period- if the car moved fast, it went off its path.

5.3.2 Result

The centroids of detected objects are in the enclosed area in Figure 5.4. This result shows that all the abnormal cases were detected successfully.

My example is simple and can be solved efficiently in other ways (e.g.,with neural networks or classifier methods), but my result proved that my immune response inspired model has efficient applicability with real-time speed and easy implementation.

The speed of the algorithm is not enough to solve high speed applications, but the car racing experiment gave a reasonable, real-time result.

5.4 Sample application, result and time measurements

My development focused on a real time application that is able to detect unknown objects, patterns and geological formations based on their textures. It can be used in visual systems where autonomous surveillance is needed or where there is no human presence. It is a helpful additional property of existing surveillance systems subject to unexpected occasions and give detection warnings. For example, it could be a useful, complementary function on Mars rovers because due to the long distances real time remote control is unfeasible.

This application was implemented in the Aladdin Pro environment utilizing the ACE4k chip [38, 27]. The input frames of the algorithm were generated in a

Table 5.5: The most relevant parameters in my AIS visual analyzer model.

Parameter name	Meaning and values	Experiment values
InitSetSize	The size of the template set, usually between 100 and 500.	200
InitMethod	The initial method can be random creation or loading stored data.	random creation
ImmuneProcessLevels	Learning phase, recognition phase or combined phase.	combined phase
Agelimit	Length of the non-active state after successful matching.	50
FeatureExMethod	Color or gray-scale feature extraction method.	color
MatchThreshold	Needed threshold value for successful matching, usually between 1 and 100.	30
Mutation	True or false.	true
MutationValue	Probability value of mutation . between 0 and 1	0.3

3D virtual world by a common PC. In this environment, the position, speed and movements were controllable manually. The gray-scale or color input pictures were converted to binary patterns by my presented feature extraction algorithm, see Figure 4.10a and 4.10b. The selection subroutine worked with a randomly generated match template set, which contained 100–500 templates. A texture or image was detected by a template if the number of the successful matching of the template with the sub-patterns was more than the MatchThreshold parameter value. The most relevant parameters in my real-time experiments are summarized in Table 5.5.

5.4.1 Experiments

In the first experiment, I used the initialization and recognition phases separately. In the initialization phase, all the templates of the initial set was run on some input images of the actual view of the virtual world. The size of these input images

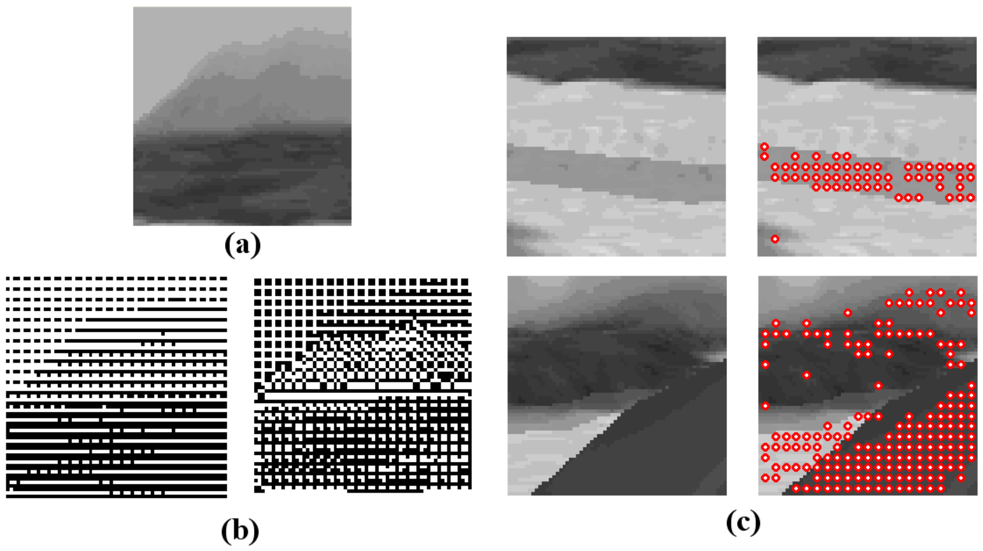


Figure 5.5: The input image is on (a). On (b) converted binary images can be seen. Dots on (c) show the result of the detection. Note that different colors can be detected with the same templates.

was 64×64. They usually contained some typical texture patterns, e.g.,texture of mountains, ocean or forest. During the process, in case of successful matching the template was selected out from the set. During the recognition phase, the algorithm detected all textures, which were not members of the initial input flow. But those sub-patterns, which the system has been taught with, have not been detected. Detection results can be seen in Figure 5.5.

In the second experiment, the initialization and recognition part periodically followed each other. If any template had a detection, it was selected out temporarily. For a longer period if this template has detection, it was not allowed to send detection message. I assumed that if something was detected by this template, then the next few detections would carry no new information and, therefore, it is not worth sending a warning message. The dynamics of this process can be seen in Figure 5.6.

In my experiments the success rate (SR) of the algorithm was 1.0 for 25-25 independent runs with the given experiment values in Table 5.5. The average number of evaluations to solution (AES) index was 17.32 for 25 independent runs, which means, on the average, 18 template runnings per frame were sufficient for a successful detection.

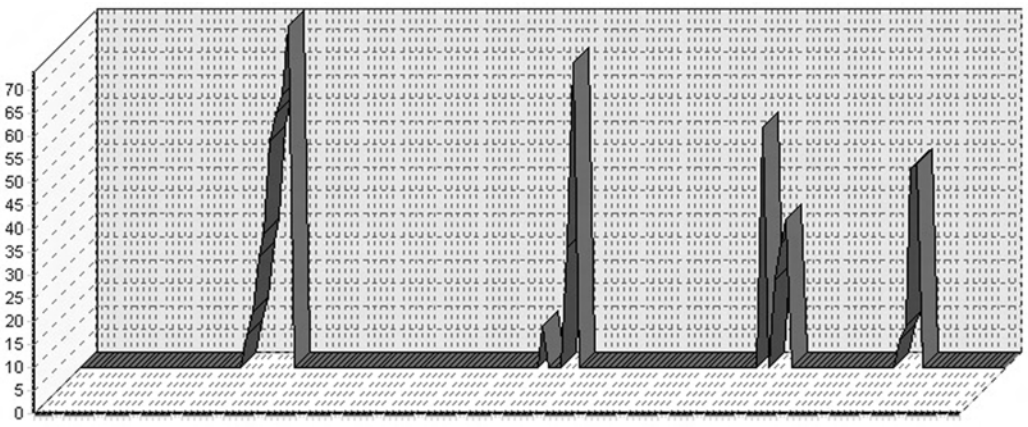


Figure 5.6: The horizontal coordinates show the time based on the the input frames. The vertical value is the label of the current template. Different peaks appear when the texture of the environment manly changed and at least one template has detected a new pattern. The first peak came when we reached the mountains, the second shows the water, third is the ground again and the last one the ocean again.

5.4.2 Observations

The time measurements of the algorithm are summarized in Table 5.6. The difference of the speed between the gray-scale and color method is caused by the data transfer in memory, as the color images are 3 times bigger than the gray-scale ones. These results show that my algorithm can run in real time even if I use the combination of two 3×3 templates for sub-pattern detection.

Along with the run of my application, we can follow the different parts of the processing on Figure 5.7. In the beginning, the system was in the recognition phase. The green line shows how the number of templates selected out was increasing when I changed it to the initialize (learning). The yellow line shows if there was any detection and which template was successful after I turned the recognitions phase back on.

If we evaluate the match number (number of those sub-patterns, which can match with the actual template) on an input image for each template of a given template set, than this statistical analysis will yield a kind of histogram, which

Table 5.6: Running time of the algorithm and number of matching with different input image flows (grayscale/color) and template set size (100/200/500), image size is 64×64, template size is 3×3

Feature extraction method	Nr. of templates	Frame /sec	Nr. of matches/sec
for gray-scale	100	46-48	18 million
for gray-scale	200	36-38	28 million
for gray-scale	500	20-22	40 million
for color	100	38-40	15 million
for color	200	32-34	25 million
for color	500	18-20	36 million

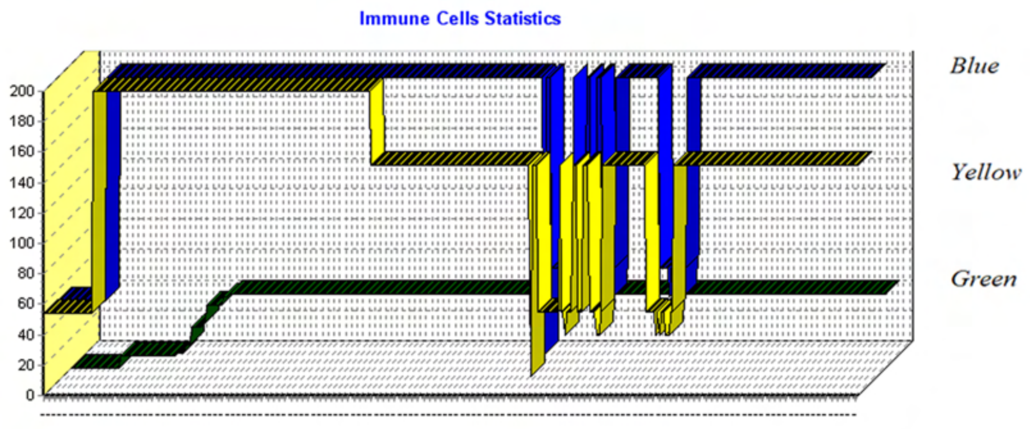


Figure 5.7: Dynamical statistics of the templates during the initialization and recognition phase. The horizontal coordinates show the time based on the input frames. The vertical value is the label of the actual template. The *yellow* line shows the number of the last or actually **successful templates**. *Blue* shows how many **templates were run** on the image. Green shows the number of **templates selected out** and the size of the template set is currently 200.

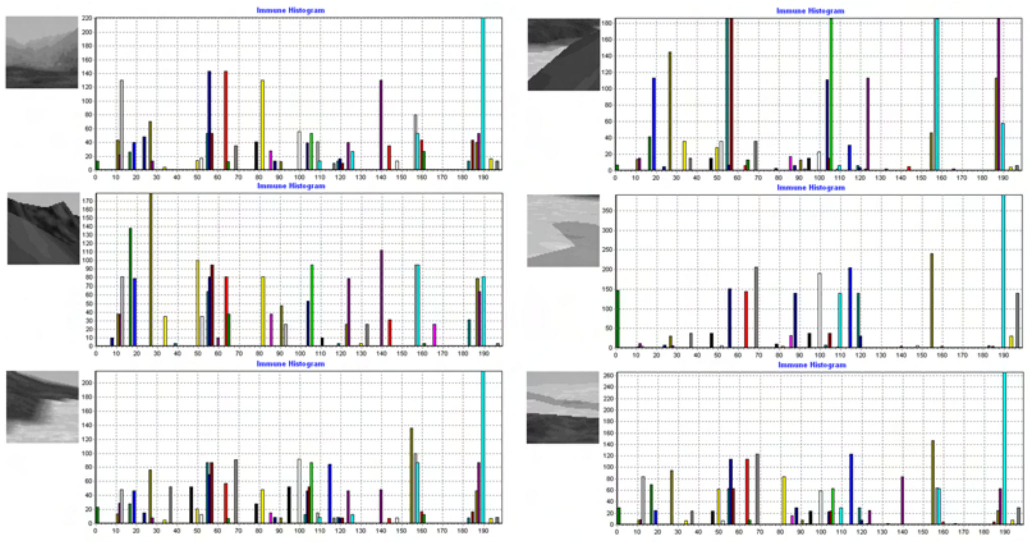


Figure 5.8: Immune histograms of different gray-scale inputs. The size of the template set is 200.

I decided to call the immune histogram. Moreover, because the matching templates can contain “don’t care” elements and the pixel of sub-patterns contain information about different pixels and the matching result is also dependent on the local neighborhood, my immune histogram is not equal to the well-known histogram.

Note that because the immune histogram can be different depending on the actual template set, it is an open question as to how we should choose the template elements and how we can interpret the result if the system has active mutation.

On Figure 5.8. some immune histograms can be seen. These results show that this method is able to distinguish between different types of inputs. If a template contains many “don’t care’ elements, its recognition ability is high, and it will recognize almost any kind of pattern. There is a bar on the right side of the immune histograms (Nr. 190), which shows that the match number of the appropriate template is always high. Observe also, that some templates have low match numbers in all cases, probably they contain few “don’t cares’, which make them very specific.

5.5 Mutation subroutines

Emphasizing the significance of the mutation of the immune system I would like to mention three motivations why it was applied in my implementation.

Increased affinity: Increased affinity will increase the chance of binding with the antigen, and therefore increase the efficiency of the attack by the immune cells. If a pathogen is associated with multiple patterns it is very well possible that a T-Cell that recognizes one of these shades does not recognize the other one.

Memory: Some of the most successful immune cells will mutate into long lived memory cells. This effect serves mainly to increase the speed of the immune response for latter invasions of the same pathogen. For this reason, in my model the T-Cells have to be equipped with some sort of activity flag showing how effective they have been in recognizing pathogenic patterns.

Set Completion: Due to the great number of different immune cells, it is hardly ever possible to cover the whole space with the actual immune cell set. Having some form of rotation in non-recognizing T-Cells by mimicking cell death and the spawning of new randomly created T-Cells will assure set completion over a larger span of time.

5.5.1 Implementation

In the next part of this section two methods of the implementation mutation are described.

Affinity Maturation: When a template is successful in the recognition of a pattern, there is a certain chance that it will generate a mutation. Mutation candidates are generated as follows: If a template is general (i.e., has multiple zeros) one of the zeros of the template will be mutated into either a +1 or a -1, making the mutation more specific. If it is already specific ($d < 2$), the mutation will just be random. The generated mutation candidate is tested on the input antigen to see whether it is more successful in recognition than the original version. The success of the template is determined by the number of matches found in the antigen. If the mutation candidate is found to be more successful than the original, it is tested to see whether it is triggering a response to the self set. In case the mutation is accepted, the original will be replaced. In case a test on the mutation candidate fails, the mutation will be rejected, and the candidate is removed (cell death). Since I want the system to be running in real time it is not feasible to run the whole self set each time we want to perform a mutation. Therefore this test is spread out over time. After every input image the mutation candidates will be tested on one of the images from the self set. Affinity maturation realizes increased detection efficiency, and makes the system more suitable for implementing pathogen recognition.

Set Completion: The total number of patterns recognized by the entire template set is kept in a variable. When this value drops below a threshold related to the total number of possible patterns, new random templates will be spawned. Of course, these templates are tested against the self set before being allowed into the pool.

5.6 Ace16k experiments

In this section, I describe two experiments, which were presented on Bi-i system and implemented on ACE16k chip. The inputs were real images transferred from the optical interface of the chip. The feature extraction module and the algorithmic template core were implemented on the chip. The evaluation of results, template storage, mutation and administration of their different properties were performed on the DSP. This combination gave better performance with real-time processing.

The first experiment shows dynamic adaptation of the system to the environment. In the rows, from left to right, the order of the images are the following. The first image is the gray-scale input. Its binary converted version is the second image. You can see the detection points of all the templates in the third image. The last image is the combination of the input and the detection points, where the detection points belong to a template whose number of detection points is higher than a value. From up to down the following can be observed: at the beginning, the system has several detection points, but during learning, the number of detection points decrease. Once objective is covered, the template set was enriched with new templates by the mutation routines. New detection points show that the input has changed again while the covering was removed (Figure 5.9).

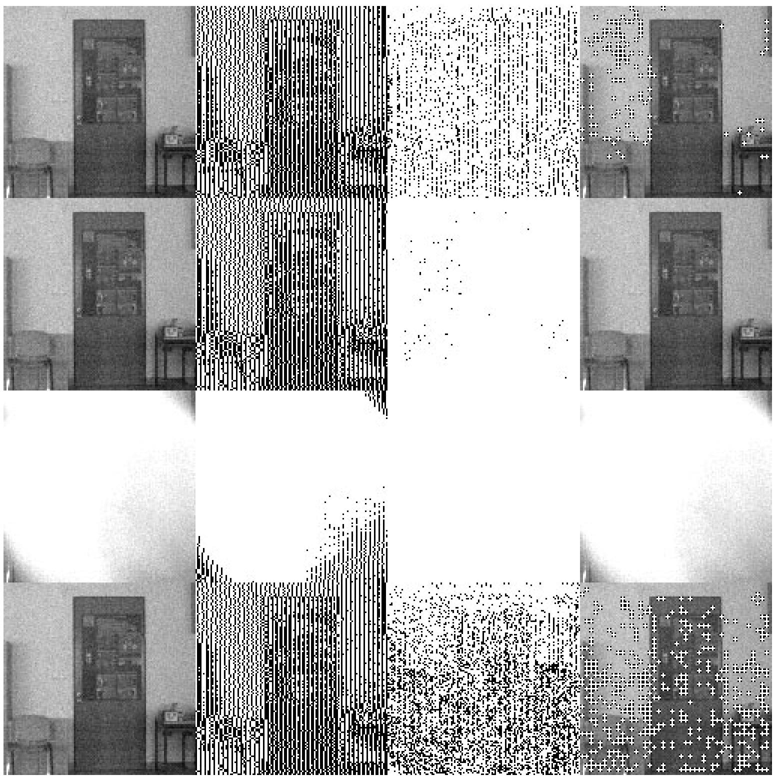


Figure 5.9: The system's dynamic adaptation to the environment. In the rows, from left to right, the order of the images are the following. The first image is the gray-scale input. Its binary converted version is the second image. You can see the detection points of all the templates in the third image. The last image is the combination of the input and detection points, where the detection points belong to a template whose number of detection points is higher than a given number. From up to down the number of frame indexes are 1, 23, 28, 64 in time.

In the second experiment, I show that the implemented system with particular parameters can be appropriate for object recognition and border estimation. The relation between the images is similar as in Figure 5.9. From up to down the following can be observed: the first image shows that the system is already adapted to the environment, there is no detection. In the second and third row, two results of an image sequence can be seen, where a palm in front of the camera shows the detection points, mainly on the fingers and upper part of the palm are detected Figure 5.10. The full video can be found on the attached CD.

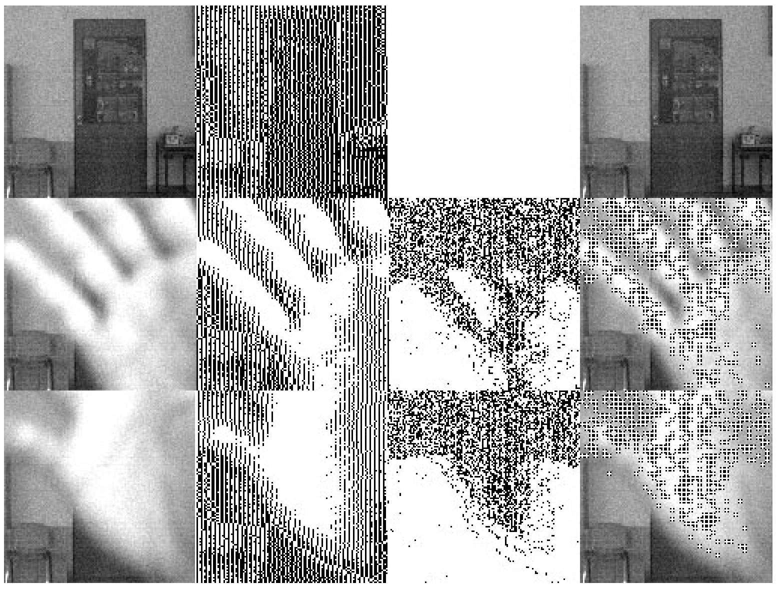


Figure 5.10: This figure shows that the implemented system with particular parameters can be appropriate for object recognition and contour estimation. The relation between the images is similar as in Figure 5.9. From up to down the number of the frames are 18, 21, 30 in time.

5.7 Conclusions

Nature has developed a powerful 3D pattern recognizer defense system. My model was inspired by this, and the results show that it is efficiently usable on 2D patterns (pictures). CNN's spatio-temporal dynamics with fast template processing is an effective tool for modelling the spatio-temporal dynamics of the immune system. The immune system provides a special algorithm, covers the target space, is able to learn and has memory. The proposed strategy has been successfully applied in a sample texture analyzer application and gave promising results. The CNN-UM chip (Acex) implementation of my algorithm is able to detect novelty events in image flows reliably, running 10000 templates/s with video-frame (25 frame/s) speed and on image size of 128×128 (Thesis 2.2).

Chapter 6

Conclusions

The results of research on preprocessing algorithms were presented in Chapter 2 and 3. The second chapter showed that the proposed PDE based algorithm relies on purely local operations in CNN environment. The CNN structure is ideal for image processing and executing the morphological- and wave-operations on the level-sets. In the third chapter, different implementations are presented and comparative experiments show that from several hardware-software systems the CNN-UM provides the fastest performance. In sample images, the advantages of the algorithm, noise suppression and contrast enhancement are demonstrated.

In Chapter 4 and 5, a biology-inspired algorithm is presented to mimic the immune response of human immune system. The designed framework and algorithm are capable of detecting previously unknown spatial-temporal events in natural or artificial video flows in real time. Theory and mathematical description are summarized in the framework of general artificial immune systems. Some of possible feature extraction methods are given, which are able to convert gray-scale and color images into binary ones. Experimental tests and results show a statistically complete success rate of the recognition.

Summary

6.1 Methods used in the experiments

In the course of my work, instruments of numerous disciplines were applied. The theory of partial differential equations and non-linear diffusion filtering was used for my preprocessing algorithms, where embedded morphological and wave operations were applied on different level-sets. The application of the theory of level-sets is often used in the case of object-segmentation and the implementations of problems needing the tracking of curve evolution. During my research I united the benefits of the binary morphological operators with the application of the method of level-sets. I compared both qualitative and quantitative measurements of my simulation and experimental results implemented in different hardver environments.

In the course of planning my human immune response inspired algorithms I became acquainted with the essential and functional notions of medical immunology.

The properties (pattern recognition, distributivity, noise and fault tolerance, resilience, immune learning and memory, robustness, self-organization) and computational power of the immune system stirred my intentions towards applying its effective methods.

During the planning of the algorithm I used the known methods of the immune response:

- the process of the maturation and negative selection of the immune cells in the thymus

The proposed system and algorithms have been also described in the theory of artificial immune systems (AIS). I measured the speed of its hardware implementation in case of different types of inputs. The mutation module of the algorithm has been described by standard measures of genetic algorithms.

Computer architectures based on the cellular nonlinear/neural network (CNN) paradigm and its 64×64 or 128×128 sized VLSI implementations offer adequate solution for both spatial modeling of discret PDE and high-speed pattern matching.

In my algorithms designed for CNN-UM wave computers I used already published template classes. It was an important aspect of the chosen templates being able to be executed reliably on available CNN-UM hardware systems. Besides using published templates, in some cases, usually in real-time hardware experiments I tuned templates.

In the course of the implementation of my analogical CNN algorithms I intended to raise efficiency, using both CNN and conventional digital solutions and implemented the most suitable algorithmic steps on the appropriate machine. In the first step of software development I used MATLAB environment with MatCNN simulation toolbox on Intel x86 PC. The real-time hardware experiments were done on ACE-BOX and Bi-i systems, where 64×64 sized Ace4k and 128×128 sized Ace16k analog/binary CNN-UM chips were built in.

The implementation independent description of the developed methods was completed in different “CNN languages” (UMF) ensuring their applicability on different hardware platforms.

6.2 New scientific results

1. Thesis: *Partial differential equation (PDE) based histogram modification, contrast enhancement and noise suppression with CNN al-*

gorithms using morphological and wave processing of the level-sets.

In the engineering practise of medical biology and in other difficult dynamical image diagnostical problems often preprocessing methods are needed which perform contrast enhancement, noise supression and other methods on the images providing better results for latter processing. In the course of my research I have designed and implemented a PDE based CNN-UM algorithm, which applies histogram modification and equalization while performing morphological and wave operations on certain level-sets. These morphological and wave-operations improve noise supression and shape enhancement. The output image could make the later processing of the object-detection more successful. See in [2], [13], [18] and Chapter 2 and 3.

1.1. I showed that nonlinear partial differential equation (PDE) described, morphological and wave operation-based parallel histogram modification can be realized with spatial approximation by operating on finite number of level-sets.

To sum up the algorithm, in a given input image it XOR two neighbouring, thresholded level-sets and morphological and wave operations are executed, this result is the current binary image. These binary images will be summed iteratively (with the current binary mask). The values of the pixels, covered by the current binary mask, are increased with a value proportional to the area of the current binary image (measured by diffusion). The outcome is obtained after a diffusion filter (Figure 2.3.).

The execution time of the algorithm, depending on the number of level-sets (i) and the morphological steps (m) is $(20 + i(111 + 10m))\tau$, where τ is the time constant depending on the CNN implementation. The chosen optimal output (see Figure 3.) has an execution time of 2276 τ.

I have implemented this complex analogic (analog and logic) algorithm containing both local and global couplings on a stored program nonlinear array processor relaying on purely local operations. The global coupling here is histogram equalization with contrast enhancement.

1.2. I showed that the chosen level-set based algorithm can be implemented on an analog CNN-UM chip (Acex) and I proved experimentally that it satisfies the theoretical expectation qualitatively and quantitatively to a good approximation. For 128x128 image resolution, a speed of 200 frame/s could be achieved.

I have done comparative experiments of different implementations of the algorithm on different hardware-software platforms, equally PC, DSP and CNN-UM microprocessors. The results show that the execution time of morphological operations could be a few thousand and the execution time of the full algorithm could be more than 100 frame/s (Figure 3.3.).

Figure 3.3a. shows that the execution time of the full algorithm provides the best results in case of ACE4k and increasing the number of morphological steps, contrary to the other systems, the speed is constant. In Figure 3.3b. it can be seen that the processing time of binary morphology in case of CNN-UM is much better as compared to the other hardware-software systems.

Experimental results can be seen with different parameters in Figure D.2. Observe, how the embedded morphological processing modifies the level-sets and provides a more reasonable result from the segmentation point of view. I have also implemented the full algorithm with wave-propagation on ACE16k-box hardware. In this case, having 128×128 resolution and 4 level-sets, the measured running speed was 3.93 msec/frame.

The achieved method can be parameterized on demand for all of real-time image processing problems, where the input image contains heavy noise and histogram equalization is needed.

2. Thesis: *Working out pattern recognition topographic algorithms inspired by the T-cell based immune response of human immune system, modelling on CNN-UM and experimental implementation for spatial-temporal novelty detection in case of a great many objects.*

I have worked out analogic algorithms, designed their models and experimentally implemented them on a CNN-UM wavecomputer. In these algorithms I have used the negative selection principle of the human immune system, processing the patterns in 2D image flows, to detect **previously unknown events in images** in real time.

The algorithms advantageously exploit the parallel processing potential of the architecture of CNN-UM wave computers. See in [1], [5-7], [12], [16] and Chapter 4 and 5.

2.1. I showed that the functional model of the immune response can be described in a cellular neural network (CNN) algorithmic framework and can be applied as an efficient image processing method.

Modelling the cell-based interactions of the 3D molecule pattern recognition and detection of the immune system I have developed topographic analogic algorithms for pattern detection, which are able to detect and recognize dynamic objects and patterns in 2D image flows. Real-time processing and the evaluation of a large number of target objects, similarly to the immune system, is the major benefit of the developed method.

The algorithm has two main parts: learning and recognition. The cores are very similar. During learning based on randomly chosen CNN Match template set, using the principle of negative selection, a template set is constructed, which will not recognize the already thought objects as dangerous. This output template set is responsible for the recognition in the latter part of the algorithm. The member templates of this set test the actual patterns of the image flow in the recognition part and in the case of an unfamiliar pattern, a warning message is generated. The template-mutation methods of the implemented algorithm provide a relatively simple, real-time adaptation to dynamic environments, which makes the system robust.

I have summarized the theory and the matematical description of my model in the framework of general artificial immune systems (AIS) too. Contrary to AIS, where the objects are represented by the same type of vectors, the patterns and templates are different here. I have defined a distance between the CNN template class and patterns, called template affinity. By the means of this affinity, I have proved that the system does not need to run all kind of templates to detect all the patterns. Moreover I have estimated the necessary size of the template set.

I have worked out analogic CNN algorithms, which are able to convert grayscale and color images into binary ones efficiently.

I have designed this conversion to be fast and preserve as much information as possible besides that there is no need to do any further image data transfer.

The algorithm subsamples the input image and thresholds at nine levels storing the binary values in 3×3 neighbouring subpatterns. The location of these subpatterns are arranged by binary masks and OR logical operations.

In case of color input image, the patterns can be determined by combining three color channels, each channel gives three binary values.

2.2. I worked out a real-time algorithm and its CNN-UM chip (Acex) implementation based on my model, which is able to detect novelty events in image flows reliably, running 10000 templates/s with video-frame (25 frame/s) speed and on image size of 128x128.

I have implemented the algorithms on general CNN-UM chips (ACE4K, ACE16K) and I have realised the full framework implementations (on ACE4K-box and Bi-i) as a real-time tool, which are able to run more than 10000 templates per second to be equivalent to around 40 million evaluated object interactions per second. Measurement results and computational power of the CNN-UM implementation can be seen in Table 1. as compared to an avarage PC. We can observe that at ACE4k-box, the required running time is significantly less.

I have defined a statistical property of the images, called immune-histogram, which is based on the subpatterns of the images.

The tests of this implemented system have been done in a virtual 3D environment and also in real environment with optical input. In the test application the system was tought by a given pattern-set and in the course of recognition the object-patterns, different form the given tought set, are detected in real-time as unfamiliar patterns (objects).

One of the major advantage of this system is that the learning mechanism, powered by mutation and selection methods, is fast and automatic. Experimental tests and results show a statistically complete success rate of the recognition.

6.3 Examples for application

All the developed algorithms and implementations offer solutions for real application problems.

My running time measurements prove that the histogram modification algorithm (first thesis) can be applied efficiently in real-time image preprocessing methods.

Its application in medical imaging can give solutions (i) for real-time ultrasound image processing of echocardiographic diagnostics and (ii) fMRI image evaluation.

In the second thesis the presented model and its algorithms were designed to be able to be used in complex surveillance systems, to learn fast, to be adaptive to dynamic environments and to send alarm messages based on different rules, if necessary.

This system can be well-applied in any situation, where human presence is beyond its ability or the supervision can not be real-time, but immediate decision based on visual input is required.

All the applied templates can be executed on every commercial CNN-UM chips, moreover, they are available on the EYE-RIS[1] systems which will be soon on the market.

[1]A new chip and OEM system of Anafocus Ltd. (Sevilla) (with 176×144 QCIF resolution and 10000 frame/sec image-speed besides 100mW consumption)

Appendix A

The CNN Computer (a Cellular Wave Computer) – Notions and Dynamics

Cellular nonlinear/neural networks (CNNs) are regular, single or multi-layer, parallel processing structures with analog nonlinear dynamic units (cells). The state value of the individual processors is continuous in time and their connectivity is local in space. The program of these networks is completely determined by the pattern of the local interactions, the so-called template. The time-evolution of the analog transient, "driven" by the template operator and the cell dynamics, represents the elementary computation in CNN (results can be defined both in equilibrium or non-equilibrium states of the network). The standard CNN equation[34] contains only first order cells placed on one layer of a regular grid and the interconnection pattern is linear.

A cellular wave computer architecture that includes CNN dynamics as its main instruction, is the **CNN Universal Machine** (**CNN-UM**, [20]). The CNN-UM makes it possible to efficiently combine analog array operations with local logic. Since the reprogramming time is approximately equal to the settling time of a non-propagating analog operation it is capable of executing complex analogic (analog and logic) algorithms. To ensure stored programmability, a global programming unit is added to the array and for an efficient reuse of intermediate results, each computing cell is extended by local memories. In addition to local storage, every cell might be equipped with local sensors and additional circuitry to perform cell-wise analog and logical operations.

Using the CNN-UM we are able to design and run analog and logic CNN wave algorithms. It is known that CNN-UM is universal as a Turing Machine[40] and as a nonlinear operator. Therefore many problems can be solved by this machine.

Its structure suggests using it for image processing in numerous applications. Beyond the classical image processing there are a lots of new methods of solving problems based on partial differential equations which need huge computational power. Most of these kind of problems can be transformed into CNN algorithm too.

Another important scope is the biological modeling. The researchers found in early times that CNN can be used for modeling some parts of the human visual system, mainly the outer retina. Recently, a multilayer, multichannel retina model has been developed [41]. Because of the simple structure of CNN, it is realizable in real hardware. Nowadays implementations run 64×64 Ace4k or 128×128 Ace16k chips.

A.1 Standard CNN Dynamics

The cellular nonlinear network (CNN) is a locally connected, analog processor array which has two or more dimensions. A standard CNN architecture consists of an $M \times N$ rectangular array of cells $C(i,j)$ with Cartesian coordinate $(i,j) i = 1..M, j = 1..N$ (Figure A.1)

The sphere of influence, $S_r(i,j)$, of radius of r of cell $C(i,j)$ is defined to be the set of all neighboring cells satisfying the following property:

$$S_r(i,j) = \left\{ C(k.l) | \max_{1 \leq k \leq M, 1 \leq l \leq N} \{ |k-i|, |l-j| \} \leq r \right\} \tag{A.1}$$

where r is a positive integer. The structure of an elementary cell can be seen on Figure A.2.

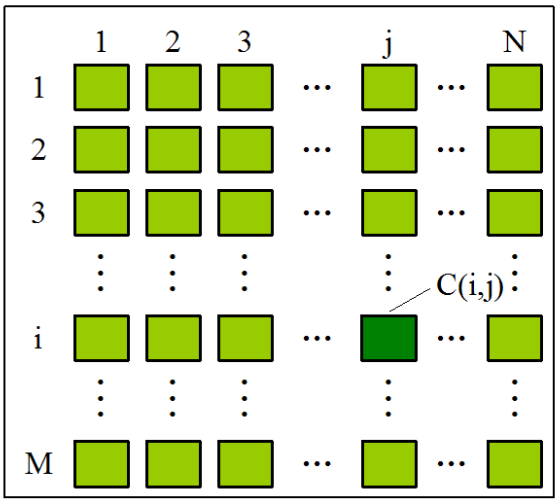


Figure A.1: MxN representation of CNN structure

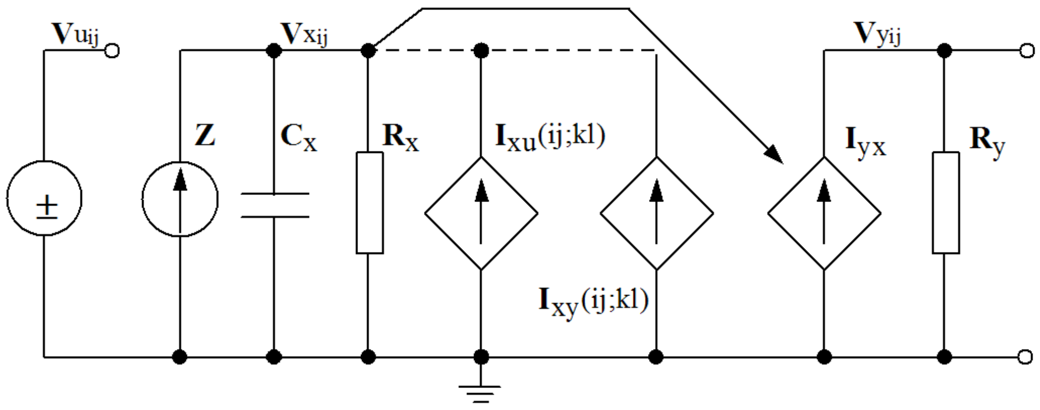


Figure A.2: The build-up of a CNN cell

$$I_{xu}(ij,kl) = B_{ij,kl}v_{u_{kl}};\quad I_{xy}(ij,kl) = A_{ij,kl}v_{y_{kl}};$$
$$I_{yx} = \frac{1}{2R_y}\left(\left|v_{x_{ij}}+1\right| - \left|v_{x_{ij}}-1\right|\right) \tag{A.2}$$

A.2 CNN Templates

The state of a cell depends on interconnection weights between the cell and its neighbors. These parameters are expressed in the form of the template:

$$A = \begin{bmatrix} a_{i-1j-1} & a_{i-1j} & a_{i-1j+1} \\ a_{ij-1} & a_{i1j} & a_{ij+1} \\ a_{i+1j-1} & a_{i+1j} & a_{i+1j+1} \end{bmatrix} B = \begin{bmatrix} b_{i-1j-1} & b_{i-1j} & b_{i-1j+1} \\ b_{ij-1} & b_{i1j} & b_{ij+1} \\ b_{i+1j-1} & b_{i+1j} & b_{i+1j+1} \end{bmatrix} z = z_{ij} \tag{A.3}$$

A template has two main parts, a feedforward and feedback matrixes. These parts are called A and B templates. The z on Equation (A.3) is the offset (bias) term. In the simplest case the template is given by 19 numbers, 9 feedback, 9 feedforward and one bias terms. This 19 number template is an elementary operation of CNN-UM and codes a complex spatial-temporal dynamics. An analogical algorithm might contain some templates and logical operations. The following differential equation system describes the dynamics of the network:

$$\begin{aligned} C_x\frac{dv_{x_{ij}}(t)}{dt} &= -\frac{1}{R_x}v_{x_{ij}}(t) + \sum_{C(k,l)\in S_r(i,j)} A_{ij;kl}v_{y_{kl}}(t) + \\ &\quad \sum_{C(k,l)\in S_r(i,j)} B_{ij;kl}v_{u_{kl}}(t) + z_{ij} \\ v_{y_{ij}}(t) &= f(v_{x_{ij}}(t)) = 1/2(\left|v_{x_{ij}}(t)+1\right| - \left|v_{x_{ij}}(t)-1\right|), \\ &\quad i=\overline{1,M}; j=\overline{1,N} \end{aligned} \tag{A.4}$$

The figure of the given function can be seen on Figure A.3. This is called standard nonlinearity.

In the case where the values of $A_{ij;kl}$; $B_{ij;kl}$ do not depend on i and j, the template is space invariant. In most cases the value of the offset current does not depends on space $z_{ij} = z$. Because of the regular 2D shape of the CNN, the value of a cell can be represent by a pixel of a picture. This gray-scale value can

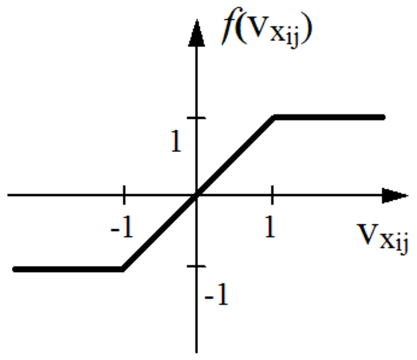


Figure A.3: The output characteristic function of a CNN cell.

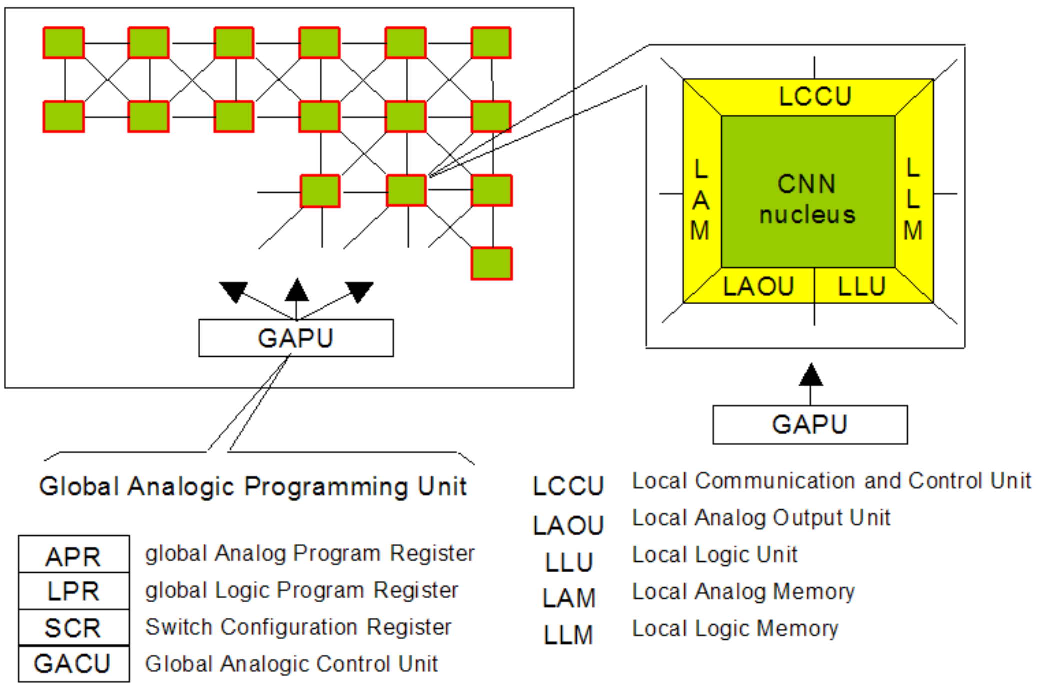
be between white (-1) to black (1). Sometimes we use fixed state mask whose values allow or permit the change of the values of their cells. 3D CNN networks can connect like layers and this gives multi-layer CNN networks. Its differential equation is similar to Eq. (A.4):

$$C_{xm}\frac{dv_{x_{mij}}(t)}{dt} = -\frac{1}{R_{xm}}v_{x_{mij}}(t)+ \sum_{n=1}^{P}\Big(\sum_{C(k,l)\in S_r(i,j)} A_{mn;ij;kl}v_{y_{nkl}}(t) + \sum_{C(k,l)\in S_r(i,j)} B_{mn;ij;kl}v_{u_{nkl}}(t)\Big) + z_{mij} \tag{A.5}$$

where p is the number of layers, m is the current layer, and A_{mn} and B_{mn} give the connection between n and m layers. For the solution of a given example, we have to give the input U, $x(0)$ initial state and the templates with the algorithm. The result is Y after running the transient. In most cases we can work with predefined templates that can be found in the software library [42].

A.3 CNN Universal Machine

The CNN Universal Machine (CNN-UM) is based on a CNN (Figure A.4). This is the first programmable analog processor array computer with own language and operation system whose VLSI implementation has same computing power as a supercomputer in image processing applications [20]. The extended universal cells of CNN-UM are controlled by global analogic programming unit (GAPU), which has analog and logic parts: global analog program register, global logic program register, switch configuration register and global analogic control unit. Every cell has analog and logical memory.



GAPU
LCCU
L A M
CNN nucleus
L L M
LAOU
LLU
GAPU
Global Analogic Programming Unit
APR global Analog Program Register
LPR global Logic Program Register
SCR Switch Configuration Register
GACU Global Analogic Control Unit
LCCU Local Communication and Control Unit
LAOU Local Analog Output Unit
LLU Local Logic Unit
LAM Local Analog Memory
LLM Local Logic Memory

Figure A.4: The architecture of CNN Universal Machine

Appendix B

Universal Machine on Flows (UMF)

This appendix introduces a new computational model[4] for exploring the algorithmic and computational complexity of the CNN-UM operating on image flows. The continuous nature of the computations performed on the CNN-UM is captured by a subset of the recently defined Universal Machine on Flows (UMF). The new, purely continuous computational model can imitate the complete CNN-UM processing. Hopefully, using this model, new insights can be gained into the computational capabilities of the CNN-UM and the algorithms developed for the CNN-UM can be more easily described from a computational complexity standpoint.

The first section describes the Continuous Machine on Flows (CMF) model with mathematical precision, specifying its mode of computation, inputs, outputs and other details. Furthermore, constructive algorithm is given, which prove that the CNN-UM can be translated into a CMF.

B.1 The Continuous Machine on Flows

The formal definition of the new computational model provides the basis on which to build the computational complexity analysis in later sections. Let us first start with the definition of a **flow**. Φ is a flow iff:

$\phi(t) \in [-B, B] \in \Re; t \in [0, T] \in \Re$ and differentiable except in countable number of points and $\phi(0) \in \Re$ is a well-defined known value; notation: $<\phi>= T$, i.e. the length of the flow.

You can intuitively think of a flow as a continuously changing function in the time interval $t \in [0, T]$. Our proposed computing model operates exclusively on such flows.

The continuous machine on flows (CMF) as a mathematical object is given by an 8-tuple, $CMF(\Phi, A, B, \Gamma, \diamond, ^i\Phi, ^o\Phi, ^o\phi)$, where:

- The $\diamond$ is a special symbol that signals the non-existence of a flow, all functions on $\diamond = \diamond$

- $\Phi = \{\phi_i, i \in [1, I] \cap Z\}$ is a set of a countable number of flows defined in a finite time-window. Outside of the time-window: $\phi_i(t) = \diamond$, notation: $|\Phi| = I$ i.e. number of flows

- $^i\Phi$ is the set of explicit input flows: $^i\Phi \subset \Phi$. These flows have predefined dynamics in a time-window or as a default: $^i\Phi(t) = {}^i\Phi(0)$, if $t > 0$

- $^o\Phi$ is the set of result (or output) flows $^o\Phi \subset \Phi$ and $^o\phi \notin {}^o\Phi$

- $^o\phi$ is a specially designated single output flow, serving as an indicator showing when the machine has stopped. This is analogous to the acceptance state of a Turing machine.

- $A = \{\alpha_i\}, \alpha_i(\Phi_k(t))$ is a special function: $\Re \cup \{\diamond\} \rightarrow \Re \cup \{\diamond\}$ otherwise if $\alpha(\diamond)$ is undefined then $\alpha(\diamond) = \diamond$

- $B = \{\beta_i\}, \beta_i$ is a multi-variable function of $^{\gamma\beta}\Phi_i \subseteq \Phi : \Re^{\left|^{\gamma\beta}\Phi_i\right|} \rightarrow [-K, +K] \in \Re$. It is bounded, continuous except in countable number of points.

- $\Gamma = \{\gamma_i\}, \gamma_i$ is a multi-variable function of ${}^{\gamma\beta}\Phi_i \subseteq \Phi : \Re^{\left|{}^{\gamma\beta}\Phi_i\right|} \rightarrow [-L, +L] \in \Re$. It is bounded, differentiable and not continuous only in countable number of points.

- ${}^{\gamma\beta}\Phi_i$ is the set of input variables in the β_i or γ_i functions of ϕ_i , thus ϕ_i is an $\left|{}^{\gamma\beta}\Phi_i\right|$-port [9]

- to ensure locality: $3/4\left|{}^{\gamma\beta}\Phi_i\right|^2 + 1/4\left|{}^{\gamma\beta}\Phi_i\right| > \sum_j \left|{}^{\gamma\beta}\Phi_i \cap {}^{\gamma\beta}\Phi_j\right|$ should be satisfied

This 8-tuple completely specifies the machine for computation. The dynamics of the i^{th} flow: $\phi_i(t) \in \Phi/{}^i\Phi$ is governed by these rules using the flows $\left\{{}^{\gamma\beta}\Phi_i \cup \phi_k\right\}; i, k < |\Phi|$:

if $\alpha_i\left(\phi_k(t)\right) = \diamond$ or $\exists\phi \in^{\gamma\beta} \Phi_i(t), \phi = \diamond$ then $\phi_i(t) = \diamond$

else if $\alpha_i(\phi_k(t)) = 0$ then $\phi_i(t) = \gamma_i({}^{\gamma\beta}\Phi_i(t))$

else $\frac{d\phi_i(t)}{dt} = \beta_i({}^{\gamma\beta}\Phi_i(t)))$

From the above equations, it follows that the CMF defines a map: $\Re^{\left|{}^i\Phi\right|}(t) \rightarrow \Re^{|{}^o\Phi|}(t)$ The operation of the CMF can be described as follows: it starts at $t = 0$, setting $\Phi(0) = 0$ and executing the dynamics specified by A, B and Γ. It stops if ${}^o\phi(t) = \diamond$ and ${}^o\Phi(t)$ is the output. Note that this stopping condition is similar to the way the end of computation is defined on a Turing machine.

Remarks: The CMF can be used as a **decision machine**:

- Accept: if ${}^o\phi(t) = \diamond\ \forall\phi_i(t) \in {}^o\Phi_i : \phi_i(t) \neq \diamond$

- Decline: if ${}^o\phi(t) = \diamond\ \exists\phi_i(t) \in {}^o\Phi_i : \phi_i(t) = \diamond$

- The partial recursive functions are equivalent to $\exists^i\Phi$ or $\Phi(0)$ input: $\forall t > 0$ ${}^o\phi(t) \neq \diamond$

Simplification: $\phi(0) = 0$ can be pre-defined for all $\phi \in \Phi/{}^i\Phi$

${}^i\Phi_2 = {}^i\Phi \cup \Phi_0, |\Phi_0| = |\Phi/{}^i\Phi|$ to specify the initial states and $\Phi = \Phi \cup \{s_0\}$

for $s_0 : \alpha = 1, \beta = 1$

for all other $\phi_i : \alpha' = s_0\alpha$ and $\gamma' = \delta(s_0)\Phi_0^{(i)} + (1 - \delta(s_0))\gamma$, where $\delta(x) =$ if $(x = 0)$ 1 else 0

B.1.1 The flow graph of CMF

Let us define the flow graph of CMF, which tries to capture the internal interdependencies of the computation. Let the nodes be the flows, with the directed links representing the connections; formally: $G_{CMF}(\Phi, \{\forall i^{\gamma\beta}\Phi_i \to \phi_i\})$. The processing structure of the CMF at a given time-instant can be described by a sub-graph, which contains only those links, where the source node is in the actual/valid function of the destination flow. This sub-graph can be decomposed to α_0 and α_1 sub-graphs. The α_0 **sub-graph** contains only those nodes, whose α value is equal to zero and only those links, where the source node is a flow in its γ function. Similarly, the α_1 **sub-graph** contains only those nodes, whose $\alpha \neq 0$ and only those links, where the source node is a flow in its β function.

The locality of the CMF network is guaranteed if the clustering coefficient is less than 3/4. The clustering coefficient is the division of the total number of sub-graph edges of the nodes in the immediate neighborhood of the central node by all the possible edges in this sub-graph.

Consider the following simple flow-structure: $\phi_i(0) = 1, \alpha = 0, \gamma = \phi_2$ $\phi_2(0) = 2, \alpha = 0, \gamma = \phi_1$ This situation is called **state-anomaly**. This anomaly means that the computation can be in a deadlock, whence the algorithm is unusable. Keeping any of the following constraints (which are not necessary but enough), the CMF avoids the state-anomaly.

(1) All α functions must have only a countable number of zero crossings.

(2) The α_0 sub-graph at each time-instance must not contain a directed circle.

B.1.2 Implementing the CNN-UM on the CMF

Consider the following CNN template $\{A, B, z\}$ execution on an $M \times N$ grid running until T

$\dot{x} = -x + A * y + B * u + z$, $y = f(x)$

The CMF equivalent:

- ${}^i\Phi = \{u_{11}, \ldots, u_{MN}, z_{11} \ldots z_{MN}\}$; ${}^o\Phi = \{y_1, \ldots, y_{MN}\}$; $\Phi = {}^i\Phi \cup {}^o\Phi \cup \{x_{11}, \ldots, x_{MN}, {}^o\phi\}$

- for $\forall ij, x_{ij}(0)$ is given; $\alpha_{ij} = 1, \gamma_{ij} = x_{ij}, \beta_{ij} = A * y + B * u + z - x$
- for $\forall ij, y_i(0) := f(x_i)$; $\alpha_{ij} = 0, \gamma_{ij} = f(x_{ij}), \beta_{ij} = 0$
- ${}^{o}\phi(t) :=$ if $(o < t < T) :$ 1 else $\diamond$

Boundary conditions can be handled in the following way:

- fixed: coded into flow with $\alpha = 0$
- periodic or zeroflux: modifying the function β of the boundary cells/flows

Extension for using a **fixed state mask**:

- ${}^{i}\Phi_2 = {}^{i}\Phi \cup \{m_{11}, \ldots, m_{MN}\}$
- change $x_{ij} : \alpha'_{ij} = m_{ij}$; if $m_{ij} = 0$ freezes else activate the cell

A **timer** or clock can be established with the following construction:

- $\phi(0), \alpha = \phi - T, \beta = 1$, where T is the period/trigger-time
  - to stop computation: $\gamma = \diamond$
  - to restart computation: $\gamma = 0$

Time-constant of a layer defined by the layer's state resistance and capacity: R and C

- change $x_{ij} : \beta'_{ij} = (RC)^{-1}\beta_{ij}$; the RC is the time-constant: τ

Arithmetic operation, e.g.,x **ADD** y: $\alpha = 0; \gamma(x, y) = x + y$ Logic operation, e.g.,x **NAND** y:

- if 0 = false, 1 = true: $\alpha = 0$; $\gamma(x, y) = 1 - xy$
- if -1 = false, 1 = true: $\alpha = 0$; $\gamma(x, y) = (1 - x - y - xy)/2$

GW: Global white (white defined as -1) $\alpha = 0, \gamma = \sum(\phi_i + 1)$
Higher-order cells: connect flows in β-functions
Memory to store one time-instant of a flow: $\Phi_{memory} : \beta = 0, \gamma = \phi_{input}, \alpha = 0(write), 1(read)$

Appendix C

Level-set Method

The level-set method is popular in the numerical implementation of curve evolution and boundary localization. One of its attractive ability is to handle topological changes automatically. Curve evolution methods have received lots of attention and found applications in various areas.

Consider a moving boundary in normal direction with a known speed function F. The goal is to track the curve evolution and stop the curve at interesting object boundaries. To achieve this goal, the speed needs to be properly designed. The speed function F can be written as $F = F(L, G, I)$ where L are local properties determined by curvature, normal direction or any local geometric information; G are global properties of the front determined by the shape and position of the front; I are independent properties of the shape of the front. Among different methods, the variational approach is the most popular, where the curve evolution speed is derived from the gradient of the energy function.

There are two ways to describe interface motion: with the boundary value and initial value partial differential equations. In the first case, $\frac{dC}{dt} = F\vec{N}$, where C is the curve, F is the speed function and $\vec{N}$ is the outward normal of curve C (Figure C.1a). In the second case, $\frac{d\Phi}{dt} + F\,|\nabla\Phi| = 0$, where Φ is a higher-dimensional time dependent level-set function and F is the speed function. The curve C is represented inplicitly as the zero level set of a function Φ as shown in Figure C.1b. In principle, the level set function Φ can be arbitrary as long as its

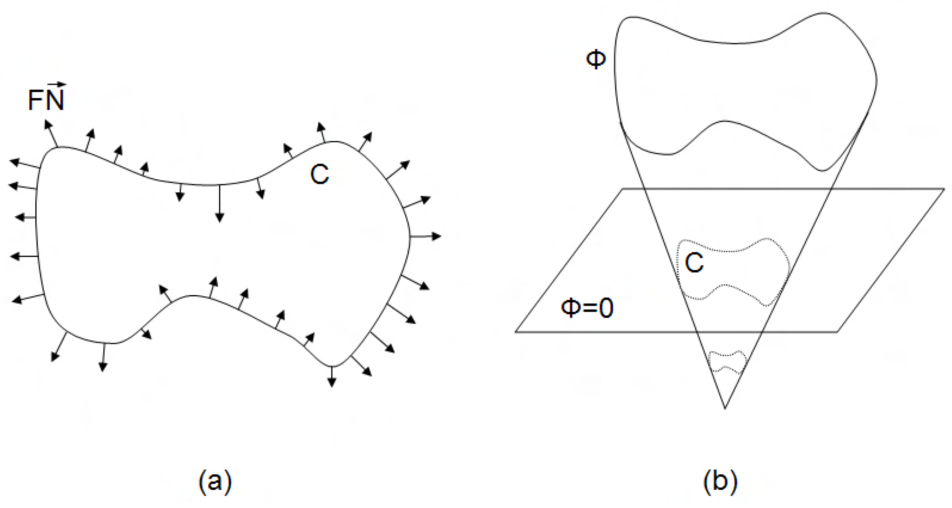


Figure C.1: The boundary value (a) and initial value (b) partial differential equations based methods.

zero level set equals C. For numerical stability reasons, a popular choice for Φ in numerical practice is the signed distance function defined as:

$$\Phi(x) = \begin{cases} min_{y\in C}\,\|x-y\| & \text{if } x \text{ is outside } C; \\ 0, & \text{if } x \in C; \\ -min_{y\in C}\,\|x-y\| & \text{if } x \text{ is inside } C; \end{cases} \qquad \text{(C.1)}$$

where $\|x-y\|$ denotes the Euclidean distance.

There are certain advantages associated with these two perspectives on propagating interfaces [43]:

- both are unchanged in higher dimension.
- topological changes in the evolving front are handled naturally.
- both rely on viscosity solutions of the associated partial differenctial equations on order to guarantee that the unique, entropy-satisfying weak solution is be obtained.
- both are accurately approximated by computational schemes which exploit techniques boorwed from the numerical solutions of hyperbolic conversation laws.

- intrinsic geometric properties of the front are easily determined in both formulations.
- both methods are made efficient through the use of adaptive computational strategies.

Nevertheless, there are significant differences between the two approaches[43]. The most obvious difference is that the initial value level set formulation allows for both negative and positive speed function F. The front may move backward and forward as it evolves. The boundary value perspective is restricted to fronts that always move in the same direction.

Boundary value formulation leads to the Fast Marching Method[43], initial value formulation leads to the Narrow Band Level-Set Method[43, 44, 45]. These PDE based methods with additional formulations provide solutions a wide range of problems and applications, such areas as geometry, grid generation, seismic analysis, computational geometry, computer vision, fluid mechanics and material sciences.

Appendix D

Morphological processing – Examples

D.1 Simulation and chip experiment results – Closing and opening

As test images both natural or artificial images are chosen. The examples here demonstrate the advantages of my complex histogram modification algorithm: simultaneous contrast enhancement, noise filtering and shape enhancement.

Simulation results and chip experiments of the extended histogram modification algorithm with embedded morphologic closing and opening. The input of the algorithm is the original image shown at the top. One dimension of the montage represents the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0,1,2,3,(4,5,6)). The other dimension reflects the number of equalization levels (top-down: 2,4,8,16,32,(64,128)).

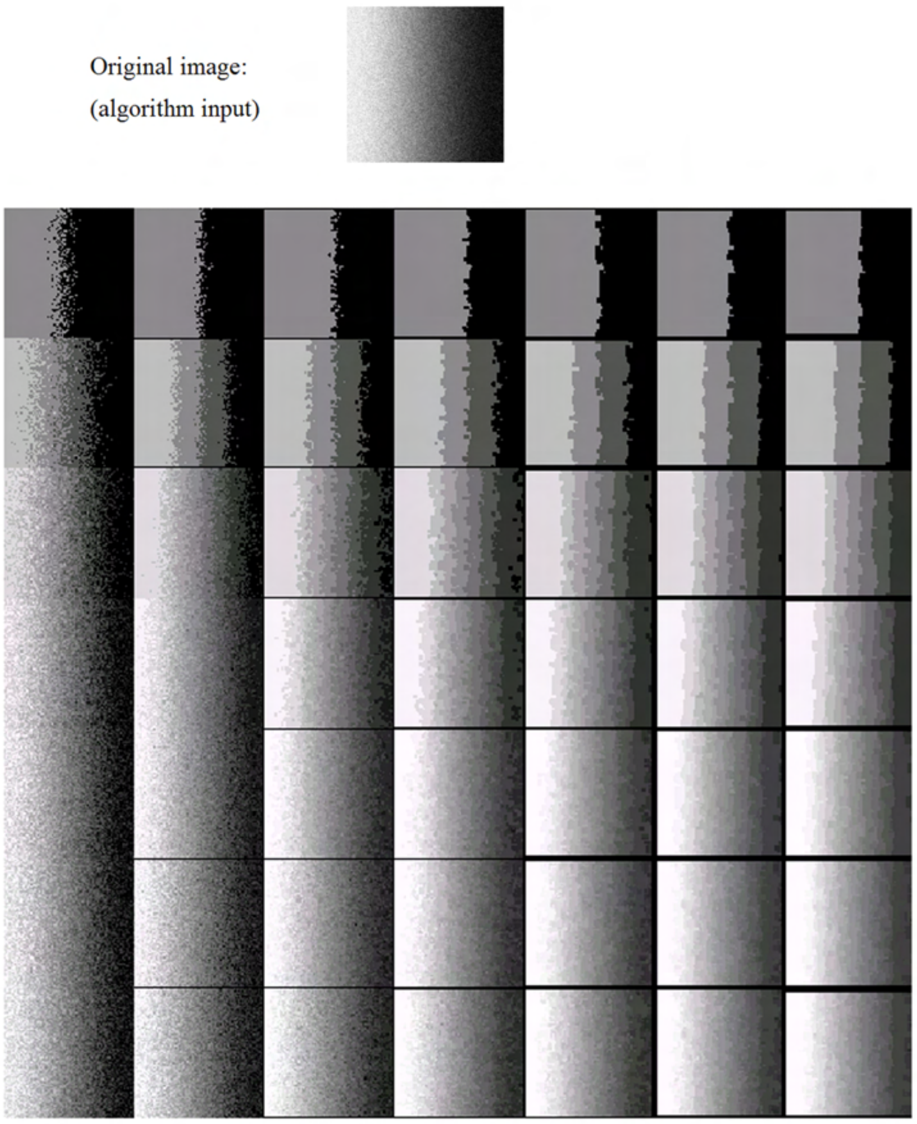


Figure D.1: Simulation results of the extended histogram modification algorithm with embedded morphologic closing. The input of the algorithm is the original image shown at the top. The columns of the montage stand for the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0-6). The rows reflect the number of equalization levels (top-down: 2,4,8,16,32,64,128).

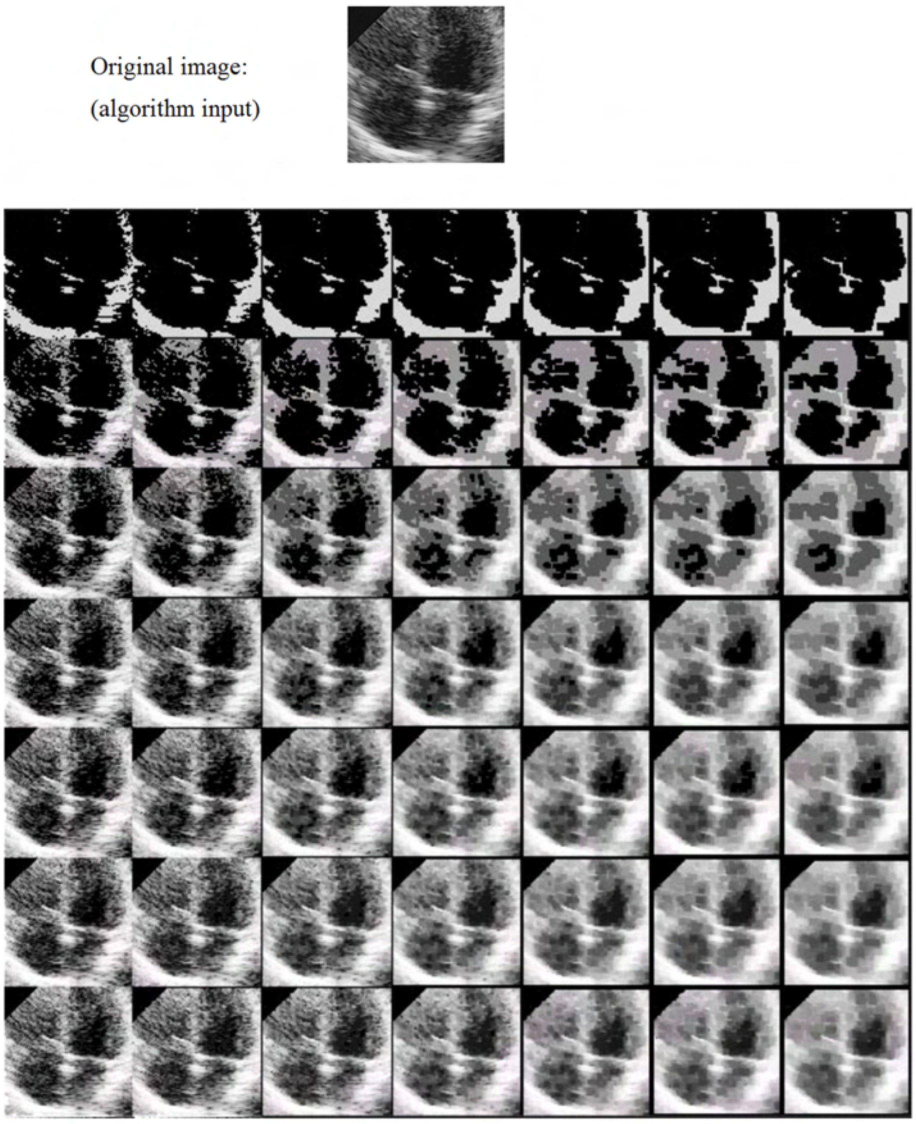


Figure D.2: Simulation results of the extended histogram modification algorithm with embedded morphologic closing. The input of the algorithm is the original image shown at the top. The columns of the montage stand for the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0-6). The rows reflect the number of equalization levels (top-down: 2,4,8,16,32,64,128).

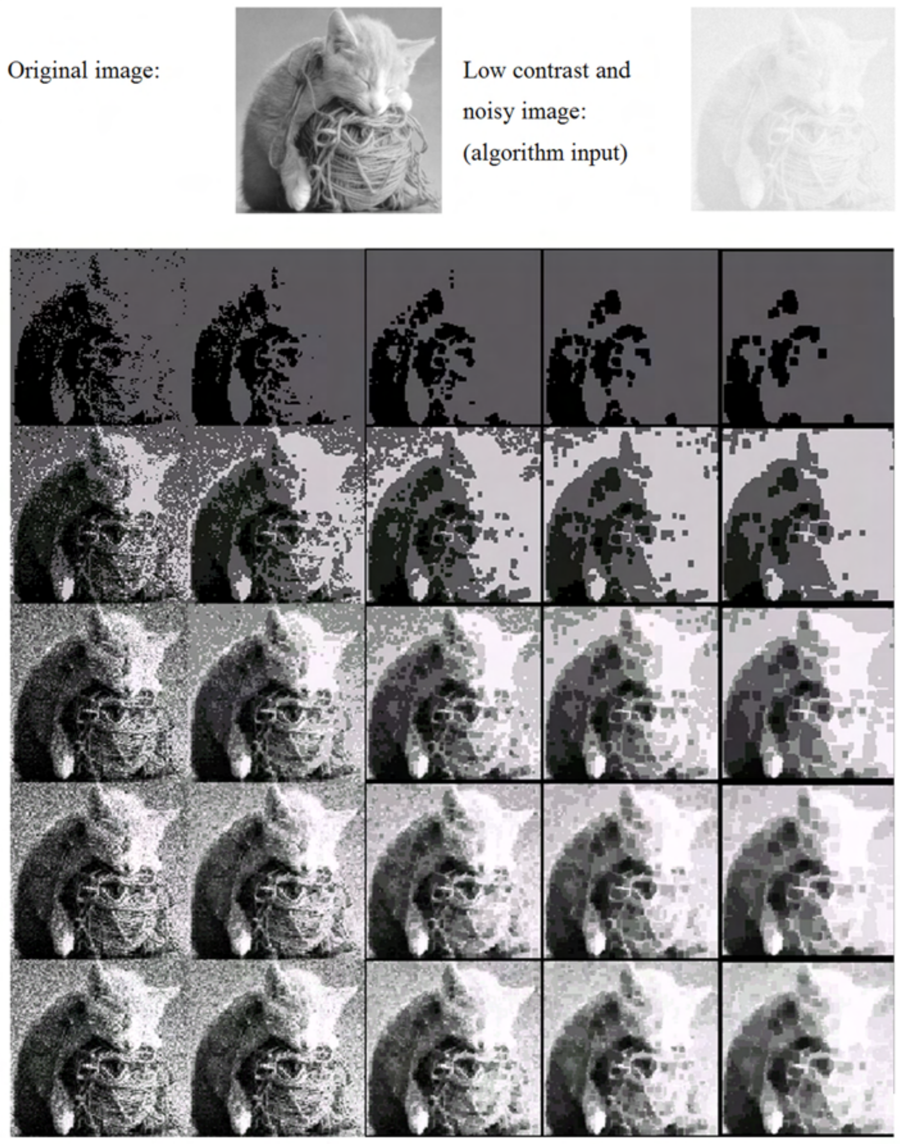


Figure D.3: Simulation results of the extended histogram modification algorithm with embedded morphologic closing. The input of the algorithm is a contrast suppressed noisy version of the original image shown at the top. The columns of the montage stand for the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0-4). The rows reflect the number of equalization levels (top-down: 2,4,8,16,32).

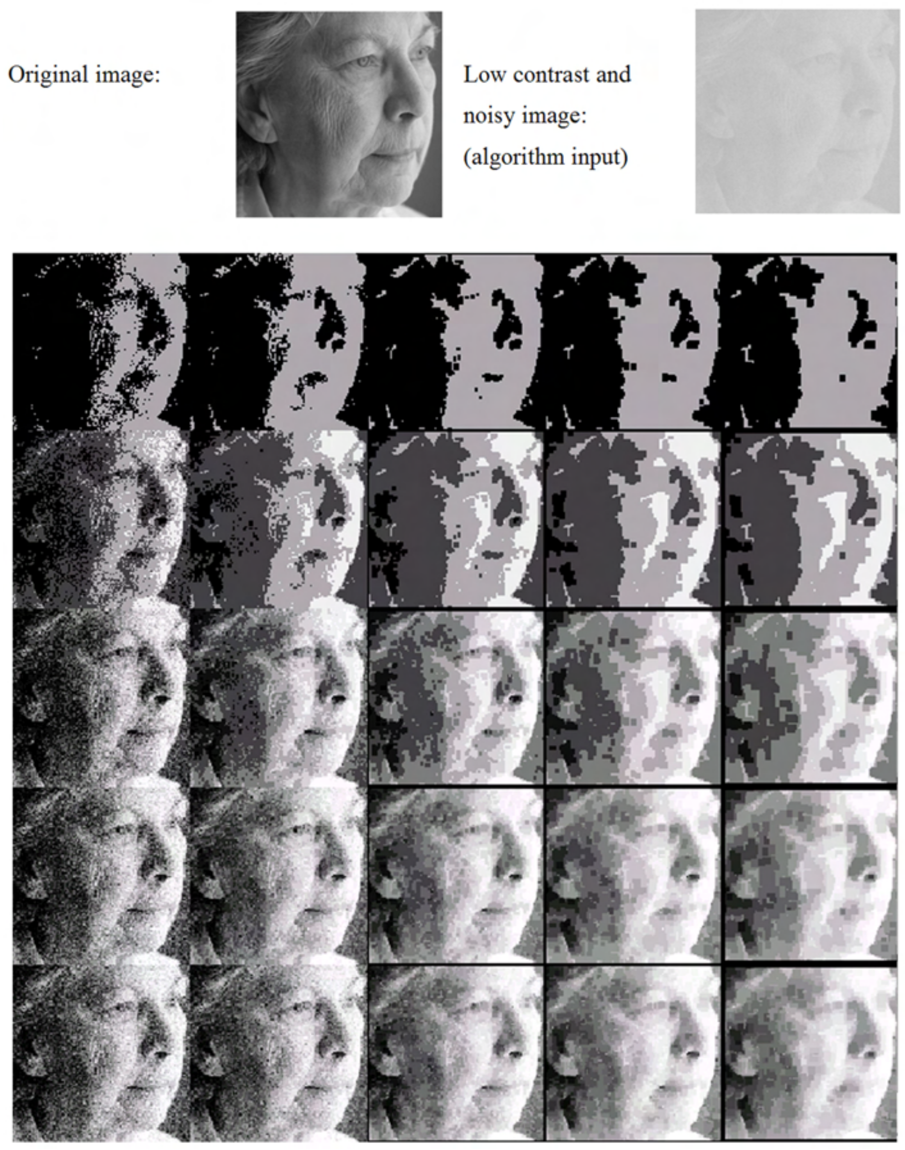


Figure D.4: Simulation results of the extended histogram modification algorithm with embedded morphologic closing. The input of the algorithm is a contrast suppressed noisy version of the original image shown at the top. The columns of the montage stand for the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0-4). The rows reflect the number of equalization levels (top-down: 2,4,8,16,32).

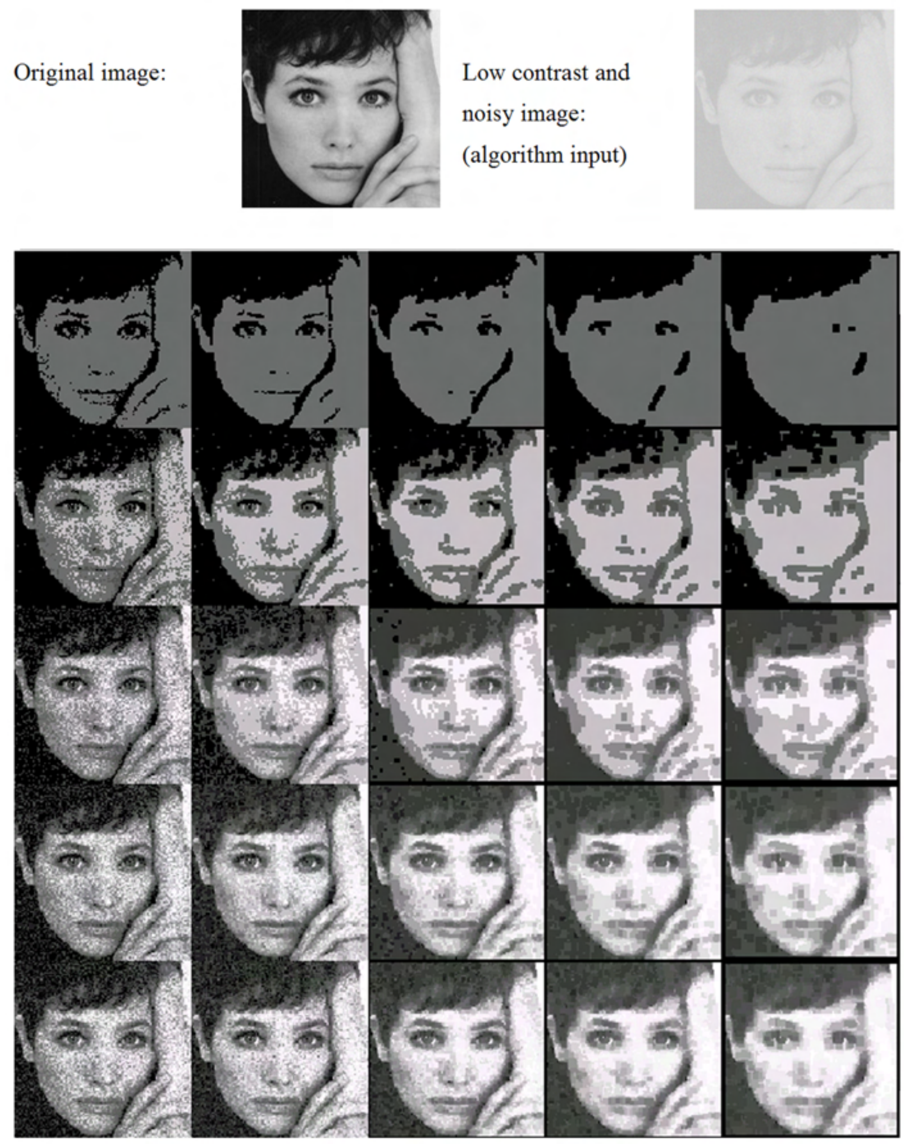


Figure D.5: Simulation results of the extended histogram modification algorithm with embedded morphologic closing. The input of the algorithm is a contrast suppressed noisy version of the original image shown at the top. The columns of the montage stand for the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0-4). The rows reflect the number of equalization levels (top-down: 2,4,8,16,32).

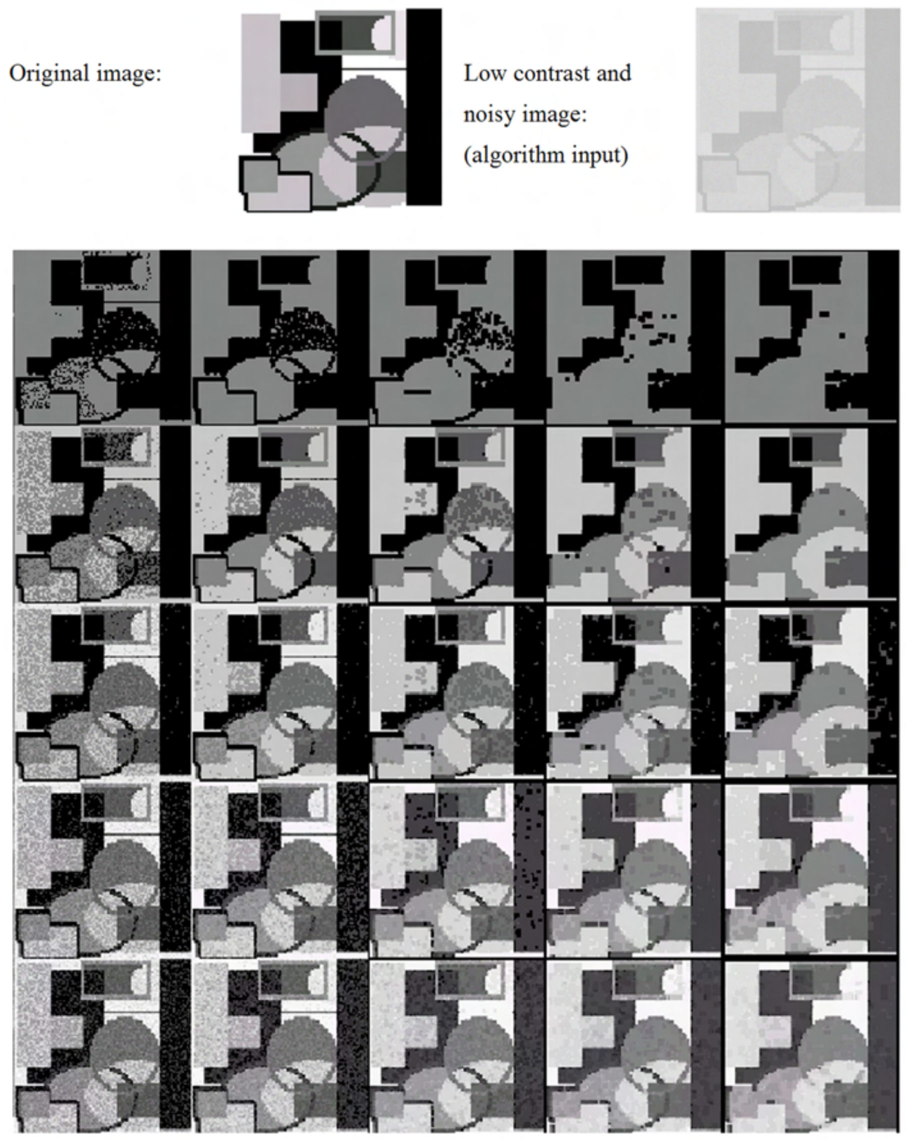


Figure D.6: Simulation results of the extended histogram modification algorithm with embedded morphologic closing. The input of the algorithm is a contrast suppressed noisy version of the original image shown at the top. The columns of the montage stand for the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0-4). The rows reflect the number of equalization levels (top-down: 2,4,8,16,32).

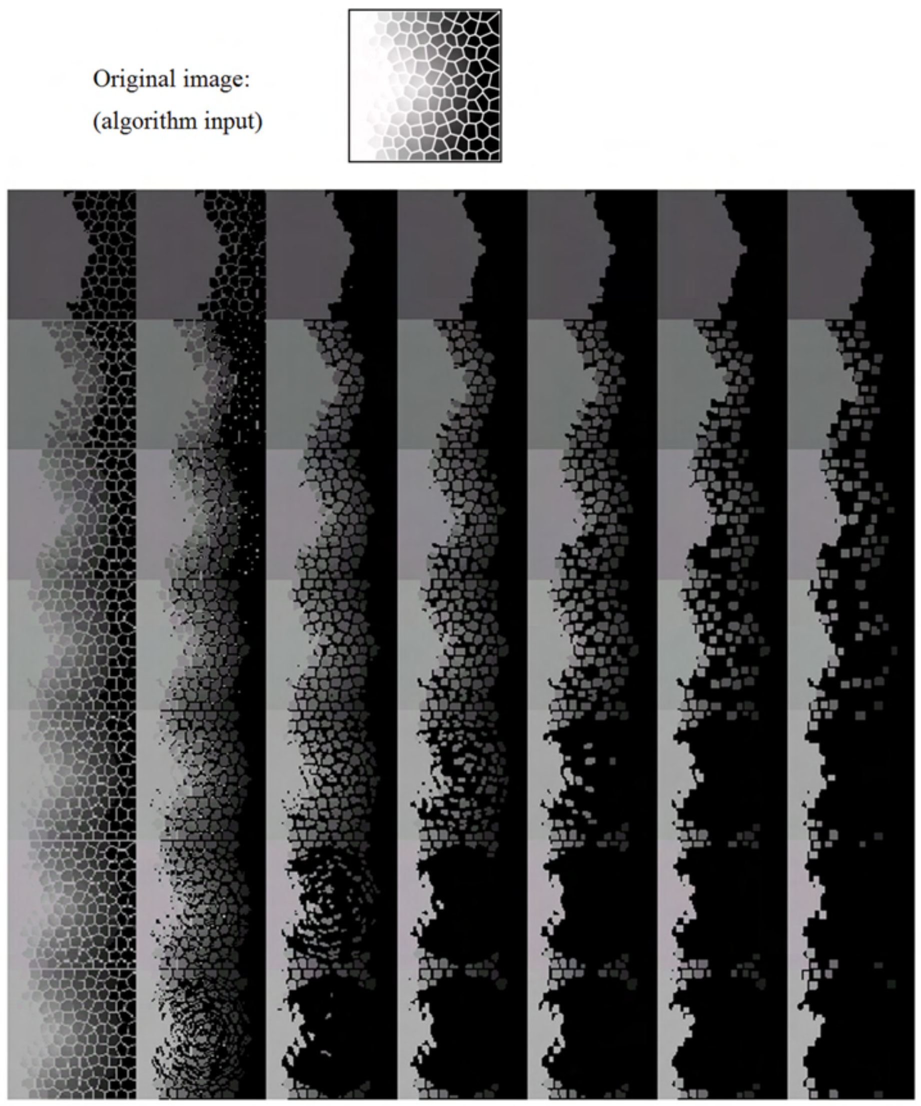


Figure D.7: Simulation results of the extended histogram modification algorithm with embedded morphologic opening. The input of the algorithm is the original image shown at the top. The columns of the montage stand for the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0-6). The rows reflect the number of equalization levels (top-down: 2,4,8,16,32,64,128).

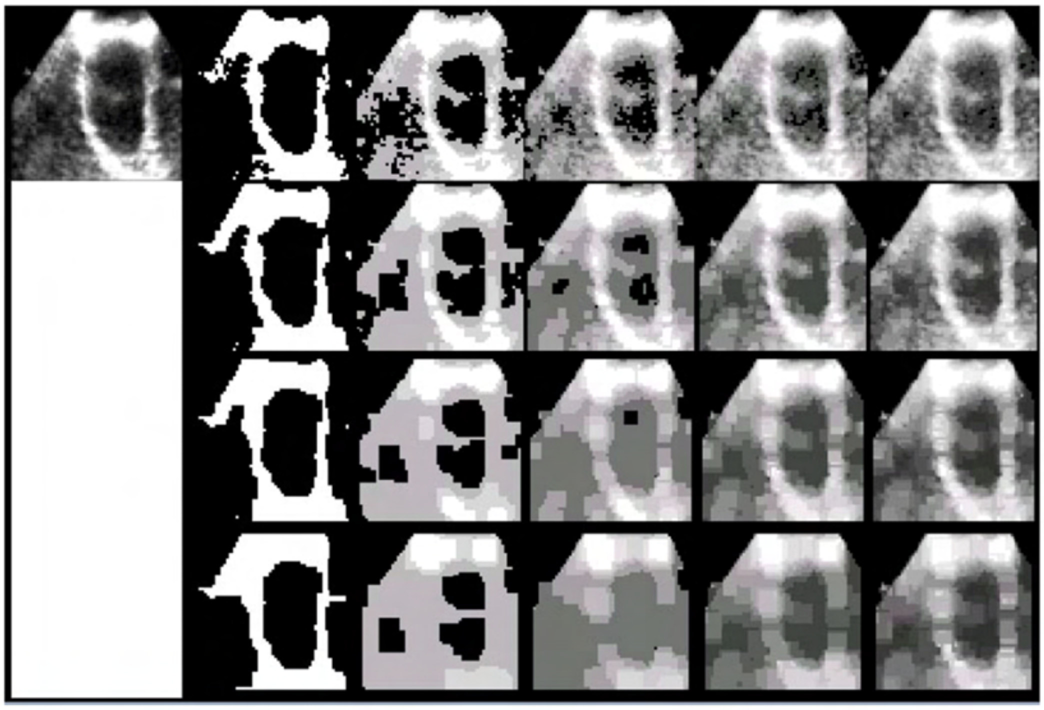


Figure D.8: Chip experiments of the extended histogram modification algorithm with embedded morphologic closing. The input of the algorithm is the original image shown in the top left corner. The columns of the montage reflect the number of equalization levels (left-right: 2,4,8,16,32). The rows stand for the morphological dimension, the number of morphological operations performed at each equalization level (top-down: 0-3).

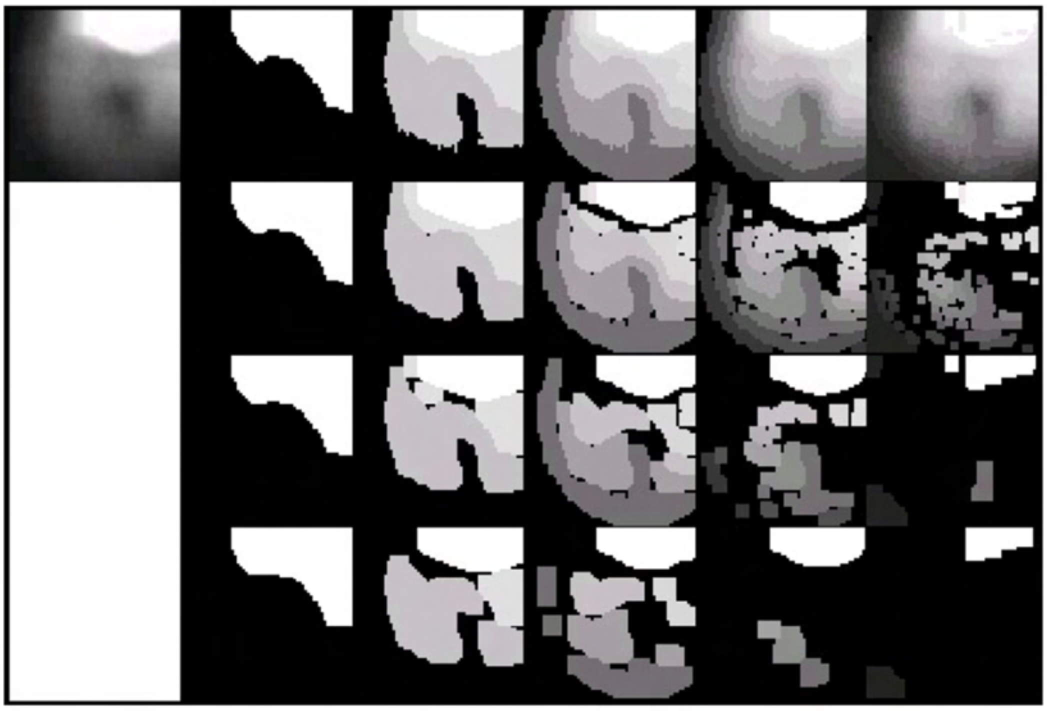


Figure D.9: Chip experiments of the extended histogram modification algorithm with embedded morphologic opening. The input of the algorithm is the original image shown in the top left corner. The columns of the montage reflect the number of equalization levels (left-right: 2,4,8,16,32). The rows stand for the morphological dimension, the number of morphological operations performed at each equalization level (top-down: 0-3).

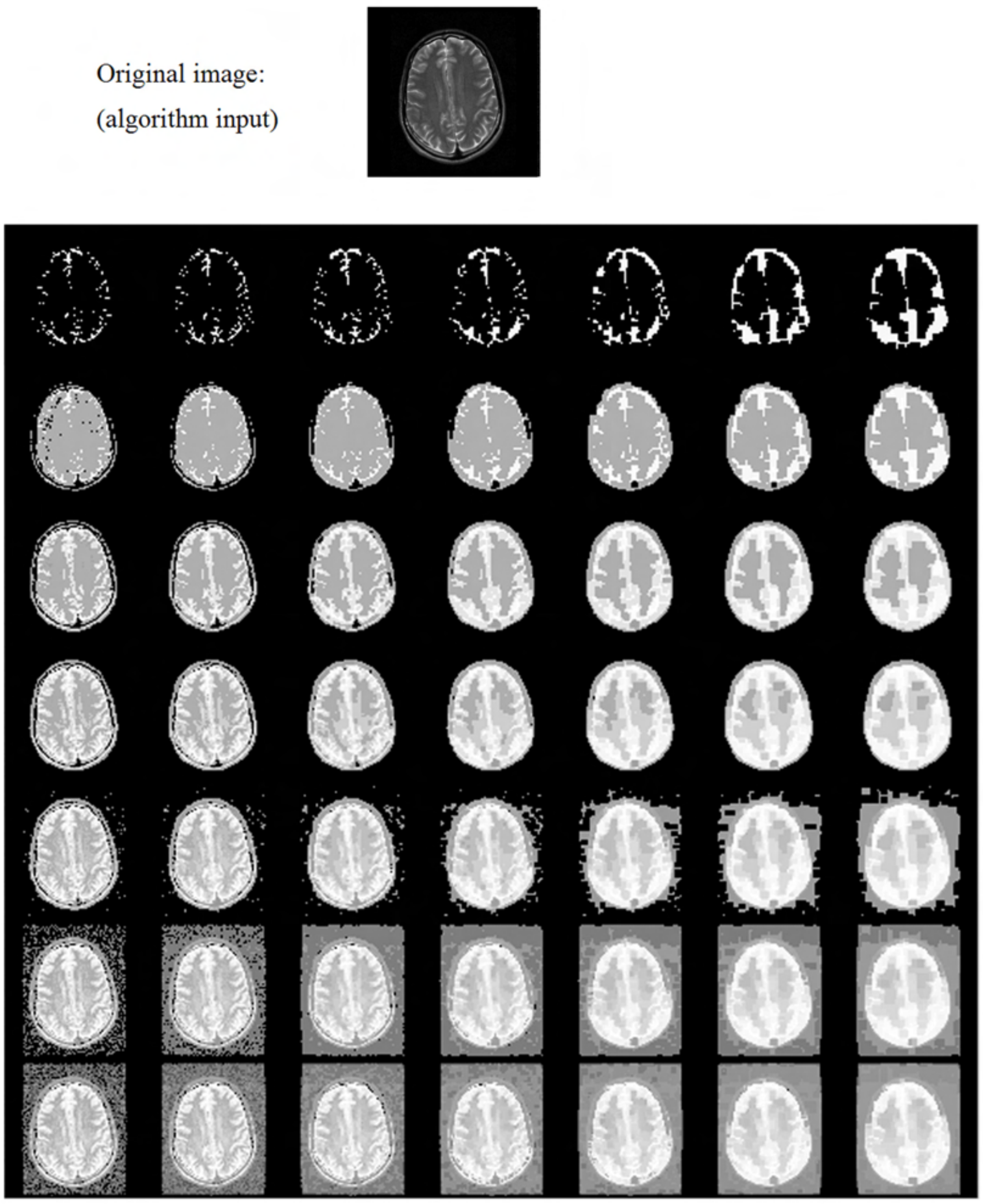


Figure D.10: Simulation results of the extended histogram modification algorithm with embedded morphologic closing. The input of the algorithm is the original image shown at the top. The columns of the montage stand for the morphological dimension, the number of morphological operations performed at each equalization level (left-right: 0-6). The rows reflect the number of equalization levels (top-down: 2,4,8,16,32,64,128).

Appendix E

The Immune System – Notions and Dynamics

The cells and molecules responsible for defending our body first detect the intruders, their "targets", then they attack and destroy them. Our immune system produces antibodies against bacteria, meanwhile activating the complement system; lymphocytes destruct cells with virus infections because they recognize little pieces (peptides) of the virus on the surface of the infected cell. Cancer is also eliminated by this defense mechanism [46].

Humans have a well-organized protection called immune response against foreign substances. (i.e. antigens: all materials, that can be specifically recognized as defensive (pathogen) or non-defensive (non-pathogen) by the immune system with specific reaction [46].) The most important members of the immune response are B- and T-lymphocytes (Lymphocytes mediate the adaptive immune response and are responsible for the recognition and elimination of the pathogens), cytokines (small molecules that signal between cells, inducing growth, differentiation, activation and regulation of immunity) and antibodies (soluble protein molecules created and secreted by B cells and plasma cells). The antibodies are also used as antigen-receptors on the surface of B-lymphocytes. The immune response is a gently controlled process with all kinds of gain and attenuate mechanisms.

Table E.1: The comparison of innate and adaptive immunity.

	Innate	Adaptive
Characteristics		
Specificity	-	+
Amplify	Linear	exponential
Memory	None	Yes
Latency	-	+
Reacting Components		
Blood proteins	Complement system	Antibodies
Cells Phagocytes,	Dendr.cells	B,T Lymphocytes

The immune system can respond to more than ten million antigens of different kinds, this response is more or less specific. The system also has a memory which can remember former antigens; therefore, the recognition is more successful.

Innate and Adaptive Immunity

Innate immunity consists of mechanisms that are capable of rapid responses to microbes, react in essentially the same way to repeated infections and may not distinguish fine differences between foreign substances.

In contrast to innate immunity, more highly evolved defense mechanisms are stimulated by exposure to infectious agents. This form of immunity develops as a response to antigens (e.g., infection) and it is called adaptive immunity [47]. This immunity is mediated by T- and B-lymphocytes, with antigen-specific receptors on the surface.

Intracellular microbes, such as viruses survive and proliferate inside host cells, where they are inaccessible to circulating antibodies. Defense against such infections is the main function of activated macrophages (innate immunity) and T cells (adaptive immunity).

Clonal Selection Hypothesis

Two theories have been developed for describing the process of antigen-detection: the instruction and the selection theories. According to the instruction theory, the antigens shape the cells (or cellular antigen receptors), which will detect them. The selection theory supposes a vast number of prefabricated immune cells and from this set the most suitable immune cells detect the antigens. The detection of an antigen causes cell division and increases the number of effective immune cells. According to the present state of science, the selection theory corresponds to the facts.

The immune cells are generated in the primary immune organs (bone marrow and thymus). The random gene rearrangement in the bone marrow proves that the antigen detecting receptors of the lymphocytes are not equal to the other cells, considering the genes at DNA level of these receptors. An adult person is exposed to approximately 10 million antigens in his life, to which he should create the distinct lymphocytes of the immune response out of 80,000-100,000 genes, which is nonsense, since we have all together 35-40,000 genes. The molecular essence of specificity can be searched in a very small part of the antigen-detecting molecule (i.e. T-cell receptors and antibodies on B cells), whose pocket consists of three parts, which can combine freely, so the immune system can choose from at least 100 billion possibilities.

The T-cells emerging from the bone marrow have to pass a two-level selection in the thymus. The T-cell only recognizes the morsels of antigens if they are accompanied by the adequate HLA-molecules.

A segment of the cell-molecules are typical for the individual and are inheritable. One of these molecule families on the surface of the cells is the so-called “histocompatibility” (tissue-tolerance) group of antigens. These are identified

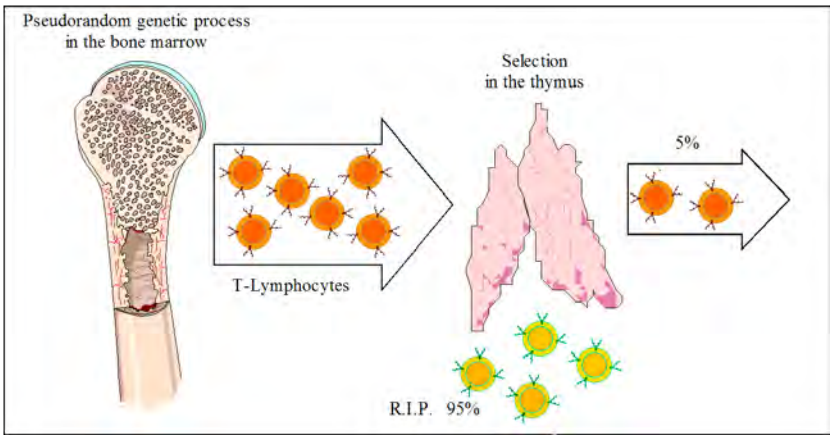


Figure E.1: After a pseudorandom genetic process in the bone marrow, less than 5% of the lymphocytes will survive the selection in the thymus.

by the genes of the "Main Histocompatibility Complex" (MHC). In the human organism, the designation of MHC genes is HLA.

Molecules of class MHC-I can be revealed in all of our cells bearing a cell nucleus/core, while molecules of class MHC-II can only be found in a small number of our cells (e.g., B-lymphocytes, dendritic cells and macrophages).

HLA is an "identification number"; it is a fact the surface of our cells is full of MHC-I and MHC-II molecules peculiar to our parents. In the course of the immune system's operation, MHC molecules have tasks in two processes. One is the selection in the thymus and the other is the "organization" of the procedure of the antigen's presenting. There are pockets on the MHC molecules, where the cell can put the morsels of antigen on display for the T-cell.

If a T-cell is unable to detect its own HLA-molecules, it dies; similarly, if it identifies the belonging "self" antigens, then it exposes the organism to danger, so it has to perish. Approximate 95% of the living T-cells that come out of bone marrow die.

The selection of the antibody-producing cells (B and plasma cells) is carried out as follows: during the immune response, the less effective (i.e. fabricating antibodies that react with weaker binding energy) B-cell will fail gradually, because it has no access to its "life-saving" antigen stimulus and dies.

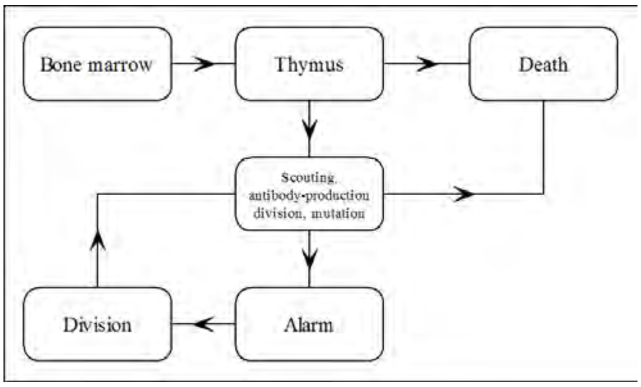


Figure E.2: The life of a lymphocyte.

The immune system continuously maintains itself in good condition; it only keeps the really useful T-lymphocytes and the best B cells in stock.

While the organism is disturbed by the stimulus of antigen, the specific fast-operating ('effector') lymphocyte recognizes exactly what kind of antigen will multiply and the immune response is prepared. The antigen-specific and also extremely long-life immune memory cells are formulated as well. Therefore, the next appearance of the newly arrived, but already detected antigen will result in a more effective immune response. This is the secondary immune response.

The Life of a Lymphocyte

The vigilant patrols of the immune system consist of constantly circulating lymphocyte cells in the circulatory system and in the lymphatic gland. Occasionally, they enter the immune organs, where the actual "job" (division, antibody-production, differentiation) is done.

The entering and exiting movements of the lymphocytes along the blood and lymphatic paths are helped by the cell-surface (adhesive) lymphocytes. These help the journey of the lymphocyte. The adhesive proteins/lymphocytes also help activation. The immune system, as a kind of police, functions perfectly, its stock is relatively big (division), its observation potential (diversity) is large-scale; they are well-trained and tested (selection) and extremely mobile (travels).

Cytokines

Cytokines are the messenger molecules of the various physiological systems including the immune system. They are highly important molecules with a complicated network among them, which transmits signs of the immune system. The crowd of chemical signs coming from the outside world (e.g., cytokines) is translated to the language of the sign transmission – here we have more options -, and during the input impacts, the final translation is the DNA, the language of gene coordination, (adapted from the signs of the outside world). The more the significance of the biological signs grows, the closer we advance to the molecular control of cell division.

Pathogen or Non-Pathogen?

The greatest secret of immunological mechanisms is how it can "separate" the self and non-self materials on the cellular level. This is an active immune response fully recognizing and refusing its own antigens.

Immune tolerance derives from two sources. First, the negative selection of the thymus filters dangerous T-cells, which is not enough in the lack of the presence of all the possible peptides of all the auto-antigens. The second step is in the periphery, where the belonging reactive T-lymphocytes receive strengthening signals inadequate in quality and quantity; therefore they become lame.

After a while, the T and B-lymphocytes decline in the periphery, if they do not get the strongest positive sign of the immune system, the antigen, associated with proper helping effects.

Our own most important substances are insured twice. The first defensive line is the immune tolerance, which applies to all our own substances. The other – a small, harmless immune response – is the presence of natural substances that hides its own components (immunological homunculus) from the real, dangerous immune response.

Immunological ignorance is a highly important, and not yet comprehended function: in a way, the immune system attempts to deal merely with the essential things; to have a neutral impact upon the organism, it does not answer.

A well-functioning system not only focuses on the appropriate parts of the antigen, but also mirrors where the peptide (pointing towards the T-cells) comes from.

Besides the selection and processing of the antigen and activation of a lymphocyte bearing a specific antigen receptor, it controls the choice of the effector system to which the procession of the antigen is left. We are far from understanding why the immune system behaves almost as a sense organ, or how it chooses from many possibilities to serves our interests in the best way.

During the immune response, the organism reaches a series of typical decisions in order to get through (in the most advantageous way) to the tolerance of self-antigens, the protection of the significant antigens until the removal of the dangerous, foreign/stranger antigens and until the ignorance of the neural foreign antigens.

Complement System

By the avalanche-like activation of the complement system (consisting of protein-degrading enzymes), we can prevent the majority of bacterium infections from attacking us. The complement system can be activated in three ways: by antigen-antibody reaction, and by two not antigen-specific ways. The activation makes a final mark/signal on the cells to be phagocytated. The cast event of the activation is the formulation of the cell-solution-inducing complex. The process is hindered and slowed down by brakes in many stages, e.g., when self-cells are dissolved.

References

The author's journal publications

[1] **Gy. Cserey**, A. Falus, and T. Roska, "Immune response inspired spatial-temporal target detection algorithms with CNN-UM," *International Journal of Circuit Theory and Applications*, vol. 34, pp. 21–47, 2006.

[2] **Gy. Cserey**, Cs. Rekeczky, and P. Földesy, "PDE based histogram modification with embedded morphological processing of the level-sets," *Journal of Circuits, System and Computers*, vol. 12, no. 4, pp. 519–538, 2003.

[3] Zs. Czeilinger, **Gy. Cserey**, L. Környei, and Cs. Rekeczky, "Objektumdetekción alapuló 3D echokardiográfia alkalmazása a gyermekkardiológiában," *Informatika és Menedzsment az Egészségügyben*, vol. 2, no. 3, pp. 37–41, 2003. www.imeonline.hu.

The author's international conference publications

[4] D. Bálya, G. Tímár, **Gy. Cserey**, and T. Roska, "A new computational model for CNN-UMs and its computational complexity," in *Proceedings of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2004*, (Budapest, Hungary), pp. 100–105, July 2004.

[5] A. Tar, J. Veres, and **Gy. Cserey**, "Design and realization of a biped robot using stepper motor driven joints," in *Proceedings of IEEE International*

Conference on Mechatronics, ICM 2006, (Budapest, Hungary), July 2006. Accepted.

[6] J. Stolte and **Gy. Cserey**, "Artificial immune systems based sound event detection with CNN-UM," in *Proceedings of European Conference on Circuit Theory and Design, ECCTD '05*, vol. 3, (Cork, Ireland), pp. 11–14, Sept. 2005.

[7] **Gy. Cserey**, A. Falus, W. Porod, and T. Roska, "An artificial immune system for visual applications with CNN-UM," in *Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS 2004*, vol. 3, (Vancouver, Canada), pp. 17–20, May 2004.

[8] **Gy. Cserey**, A. Falus, W. Porod, and T. Roska, "Feature extraction CNN algorithms for artificial immune systems," in *Proceedings of International Joint Conference on Neural Networks, IJCNN 2004*, vol. 1, (Budapest, Hungary), pp. 147–152, July 2004.

[9] **Gy. Cserey**, W. Porod, and T. Roska, "An artificial immune system based visual analysis model and its real-time terrain surveillance application," in *Proceedings of International Conference on Artificial Immune Systems, ICARIS 2004*, (Catania, Italy), pp. 250–262, Springer-Verlag, Inc, Sept. 2004. Lecture Notes in Computer Science 3239.

[10] J. McRaven, M. Scheutz, **Gy. Cserey**, V. Andronache, and W. Porod, "Fast detection and tracking of faces in uncontrolled environments for autonomous robots using the CNN-UM," in *Proceedings of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2004*, (Budapest, Hungary), pp. 196–201, July 2004.

[11] M. Scheutz, J. McRaven, and **Gy. Cserey**, "Fast, reliable, adaptive, bimodal people tracking for indoor environments," in *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2004*, vol. 2, (Sendai, Japan), sep 2004.

[12] Cs. Rekeczky, V. Binzberger, D. Hillier, Zs. Czeilinger, G. Soós, **Gy. Cserey**, L. Kék, and D. L. Vilarino, "A diagnostic echocardiography system boosted by CNN technology," in *Proceedings of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2004*, (Budapest, Hungary), pp. 243–248, July 2004.

[13] **Gy. Cserey**, A. Falus, and T. Roska, "Immune response inspired CNN algorithms for many-target detection," in *Proceedings of European Conference on Circuit Theory and Design, ECCTD '03*, vol. 1, (Krakow, Poland), pp. 405–408, Sept. 2003.

[14] **Gy. Cserey**, Cs. Rekeczky, and P. Földesy, "PDE based histogram modification with embedded morphological processing of the level-sets," in *Proceedings of the 7th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2002*, (Frankfurt, Germany), pp. 315–322, July 2002.

[15] Cs. Rekeczky, G. Tímár, and **Gy. Cserey**, "Multi target tracking with stored program adaptive CNN universal machines," in *Proceedings of the 7th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2002*, (Frankfurt, Germany), pp. 299–306, July 2002.

[16] Cs. Rekeczky, Zs. Czeilinger, **Gy. Cserey**, L. Kék, I. Szatmári, and T. Roska, "3D echocardiography powered by CNN technology," in *Proceedings of the 15th European Conference on Circuit Theory and Design, ECCTD '01*, vol. 3, (Espoo, Finland), pp. 289–292, Aug. 2001.

The author's other publications

[17] **Gy. Cserey**, A. Falus, and T. Roska, "Immune response inspired spatial-temporal target detection algorithms," tech. rep., Ányos Jedlik Laboratory, Péter Pázmány Catholic University, 2005.

[18] Cs. Rekeczky, V. Binzberger, D. Hillier, Zs. Czeilinger, G. Soós, **Gy. Cserey**, L. Kék, and D. L. Vilarino, "Analogic diagnostic system for echocardiography," Tech. Rep. DNS-10, Analogical and Neural Computing Laboratory, Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SzTAKI), 2004.

[19] **Gy. Cserey**, Cs. Rekeczky, and P. Földesy, "PDE based histogram modification with embedded morphological processing: CNN-UM chip experiments," Tech. Rep. DNS-2, Analogical and Neural Computing Laboratory, Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SzTAKI), 2002.

Publications connected to the dissertation

[20] T. Roska and L. O. Chua, "The CNN Universal Machine," *IEEE Transactions on Circuits and Systems*, vol. 40, pp. 163–173, 1993.

[21] M. Csapody and T. Roska, "Adaptive histogram equalization with cellular neural networks," in *Proceedings of the 7th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 1996*, (Seville, Spain), pp. 81–86, June 1996.

[22] W. K. Pratt, *Digital Image Processing.* John Wiley And Sons, 1991.

[23] G. Sapiro and V. Caselles, "Histogram modifications via partial differential equations," *Journal of Differential Equations*, vol. 135, pp. 238–268, 1997.

[24] G. Sapiro and V. Caselles, "Contrast enhancement via image evolution flows," *Graphical Models Image Processing*, vol. 59, pp. 407–416, 1997.

[25] V. Caselles, J. L. Lisani, and G. Sapiro, "Shape preserving local histogram modification," *IEEE Transactions on Image Processing*, vol. 8, pp. 220–230, 1999.

[26] S. Espejo, R. Domínguez-Castro, G. Liñán, and A. Rodríguez-Vázquez, "A 64×64 CNN universal chip with analog and digital I/O," in *IEEE International Conférence Electronics Systems, ICECS '98*, (Lisbon, Portugal), pp. 203–206, Sept. 1998.

[27] G. Liñán, S. Espejo, R. Domínguez-Castro, and Rodríguez-Vázquez, "ACE4k: an analog I/O 64×64 visual microprocessor chip with 7-bit analog accuracy," *International Journal of Circuit Theory and Applications*, vol. 30, no. 2-3, pp. 89–116, 2002.

[28] E. R. Daugherty, *An Introduction to Morphological Image Processing.* SPIE Optical Engineering Press, 1992.

[29] T. Roska, L. Kék, and ed., "Analogic CNN program library, version 2.0," tech. rep., Analogical and Neural Computing Laboratory, Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA Sz-TAKI), 2001.

[30] Cs. Rekeczky, T. Roska, and A. Ushida, "CNN-based difference-controlled adaptive nonlinear image filters," *International Journal of Circuit Theory and Applications*, vol. 26, no. 4, pp. 375–423, 1998.

[31] K. R. Crounse and L. A. Chua, "Methods for image processing in Cellular Neural Networks: A tutorial," *IEEE Transactions on Circuits and Systems*, vol. 42, no. 10, pp. 583–601, 1995.

[32] Cs. Rekeczky and L. O. Chua, "Computing with front propagation: Active contour and skeleton models in continuous-time CNN," *Journal of VLSI Signal Processing*, vol. 23, no. 2, pp. 373–402, 1999.

[33] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, and S. Meana, "ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Transactions on Circuits and Systems Part I: Regular Papers*, vol. 51, no. 5, pp. 851–863, 2004.

[34] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory and applications," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1257–1290, 1988.

[35] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Transactions on Circuits and Systems*, vol. 40, pp. 147–156, 1993.

[36] S. Espejo, R. Carmona, R. Domínguez-Castro, and A. Rodríguez-Vázquez, "CNN universal chip in CMOS technology," *International Journal of Circuit Theory and Applications*, vol. 24, pp. 93–111, 1996.

[37] A. Paasio, A. Dawidziuk, K. Halonen, and V. Porra, "Minimum size 0.5 micron CMOS programmable 48 by 48 CNN test chip," in *Proceedings of the 1997 European Conference on Circuit Theory and Design, ECCTD '97*, (Budapest, Hungary), pp. 154–156, 1997.

[38] A. Zarándy, Cs. Rekeczky, I. Szatmári, and P. Földesy, "The new framework of applications: The aladdin system," *IEEE Journal on Circuits, Systems and Computers*, vol. 12, no. 6, pp. 764–781, 2003.

[39] L. Kék and A. Zarándy, "Implementation of large-neighborhood nonlinear templates on the CNN Universal Machine," *International Journal of Circuit Theory and Applications*, vol. 26, no. 6, pp. 551–566, 1998.

[40] L. O. Chua, T. Roska, and P. L. Venetianer, "The CNN is universal as the Turing Machine," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 40, no. 3, pp. 289–291, 1993.

[41] D. Bálya, B. Roska, T. Roska, and F. S. Werblin, "A CNN framework for modeling parallel processing in a mammalian retina," *International Journal of Circuit Theory and Applications*, vol. 30, no. 2-3, pp. 363–393, 2002.

[42] T. Roska, L. Kék, L. Nemes, A. Zarándy, M. Brendel, and P. Szolgay, "CNN software library (templates and algorithms), version 7.2.," Tech. Rep. DNS-CADET-15, Analogical and Neural Computing Laboratory, Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SzTAKI), 1998.

[43] J. A. Sethian, *Level Set Methods and Fast Marching Methods.* Cambridge University Press, 1996.

[44] G. Sapiro, *Geometric Partial Differential Equations and Image Analysis.* Cambridge University Press, 2001.

[45] S. J. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces.* Springer, 2002.

[46] A. Falus, *Physiological and Molecular Principles of Immunology.* Semmelweis Press, Budapest, 1998. in Hungarian.

[47] A. K. Abbas and W. Schmitt, *Cellular and Molecular Immunology.* Saunders, 2000. 4th edition.

[48] T. Roska, "Computational and computer complexity of analogic cellular wave computers," in *Proceedings of the 7th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2002*, (Frankfurt, Germany), pp. 323–335, July 2002.

[49] D. Dasgupta, *Artificial Immune Systems and Their Applications.* Springer-Verlag, 1999.

[50] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach.* Springer, 2002.

[51] S. Stepney, R. E. Smith, J. Timmis, M. J. Tyrell, A. M. Neal, and A. N. W. None, "Conceptual frameworks for artificial immune systems," *International Journal of Unconventional Computing*, vol. 1, no. 3, pp. 315–338, 2005.

[52] S. A. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evolutionary Computation*, vol. 8, no. 4, pp. 443–473, 2000.

[53] L. O. Chua and T. Roska, *Cellular neural networks and visual computing, Foundations and applications.* Cambridge University Press, 2002.

[54] L. Alvarez, F. Guichard, P. L. Lions, and J. Morel, "Axioms and fundamental equations of image processing," *Arch. Rational Mech. Anal.*, vol. 16, pp. 200–257, 1993.

[55] J. Weickert, "A review of nonlinear diffusion filtering," in *SCALE-SPACE '97: Proceedings of the First International Conference on Scale-Space Theory in Computer Vision*, (London, UK), pp. 3–28, Springer-Verlag, 1997.

[56] S. Singh and M. Markou, "An approach to novelty detection applied to the classification of image regions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 396–407, 2004.

[57] M. Markos and S. Sameer, "Novelty detection: a review part 1: statistical approaches," *Signal Processing*, vol. 83, pp. 2481–2497, 2004.

[58] M. Markos and S. Sameer, "Novelty detection: a review part 2: neural network based approaches," *Signal Processing*, vol. 83, pp. 2499–2521, 2004.

[59] A. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. 42, no. 2, 1936.

[60] L. O. Chua and T. Roska, *Cellular Neural Networks and Visual Computing.* Cambridge University Press, 2001.

[61] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and Real Computation.* Springer-Verlag, 1997.

[62] L. Blum, M. Shub, and S. Smale, "On a theory of computation over the real numbers; np completeness, recursive functions and universal machines," *Bulletin of the American Mathematical Society*, vol. 21, no. 1, pp. 1–46, 1989.

[63] T. Roska, "Computational and computer complexity of analogic cellular wave computers," *IEEE Journal on Circuits, Systems and Computers*, vol. 12, no. 4, pp. 539–562, 2003.

[64] G. Chaitin, “Information theoretic computational complexity,” *IEEE Transactions on Information Theory*, vol. IT-20, pp. 10–15, 1974.

[65] S. Wolfram, *A New Kind of Science.* Wolfram Media, 2002.

[66] L. O. Chua, “Passivity and complexity,” *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 46, no. 1, pp. 71–82, 1999.

[67] T. Roska, “Cellular wave computers for brain-like spatial-temporal sensory computing,” *IEEE Circuits and Systems Magazine*, vol. 5, no. 2, pp. 5–19, 2005.

[68] A. Zarándy and C. Rekeczky, “Bi-i: a standalone ultra high speed cellular vision system,” *IEEE Circuits and Systems Magazine*, vol. 5, no. 2, pp. 36–45, 2005.

[69] P. Petrona and J. Malik, “Scale-space and edge-detection,” *IEEE Transactions on Pattern Analsis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[70] G. Gerig, O. Kübler, R. Kikinis, and F. Jolesz, “Nonlinear anisotropic filtering of mri data,” *IEEE Transactions on Medical Imaging*, vol. 11, no. 2, pp. 221–232, 1992.

[71] J. Weickert and C. Schnörr, “Pde-based preprocessing of medical images,” Tech. Rep. 8, Computer Vision, Graphics and Pattern Recognition Group, University of Mannheim, 2000.

[72] A. Sarti, K. Mikula, and F. Sgallari, “Nonlinear multiscale analysis of 3d echocardiographic sequences,” *IEEE Transactions on Medical Imaging*, vol. 18, pp. 453–466, 1999.

[73] J. S. Suri, D. Wu, J. Gao, S. Singh, and S. Laxminarayan, “A comparison of state-of-the-art diffusion imaging techniques for smoothing medical/non-medical image data,” vol. 1, (Quebec, Canada), Aug. 2002.

[74] Z. Yu and C. Bajaj, "A fast and adaptive method for image contrast enhancement," in *2004 International Conference on Image Processing, ICIP '04*, vol. 2, (Singapor), pp. 1001–1004, Oct. 2004.

[75] S. Tőkés, L. Orzó, and T. Roska, "Programmable optical CNN implementation based on the template pixels' angular coding," in *Proceedings of the 7th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2002*, (Frankfurt, Germany), pp. 148–155, July 2002.

[76] D. L. Chao and S. Forrest, "Information immune systems," in *Proceedings of the First International Conference on Artificial Immune Systems, ICARIS 2002*, (Canterbury, UK), pp. 132–140, Sept. 2002.

[77] J. H. Carter, "The immune system as a model for pattern recognition and classification," *Journal of the American Medical Informatics Association*, vol. 7, no. 1, pp. 28–41, 2000.

[78] D. R. Carvalho and A. A. Freitas, "An immunological algorithm for discovering small-disjunct rules in data mining," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2001*, (San Francisco, USA), pp. 401–404, July 2001.

[79] T. Stibor, P. Mohr, J. Timmis, and C. Eckert, "Is negative selection appropriate for anomaly detection?," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2005*, (Washington D.C., USA), pp. 321–328, June 2005.

[80] T. Stibor, J. Timmis, and C. Eckert, "On the appropriatness of negative selection defined over the hamming shape space as a network intrusion detection system," in *IEEE Congress on Evolutionary Computation, CEC 2005*, (Edinghburg, UK), 2005.

[81] S. Chang, E. Jungert, R. Green, D. Gray, and E. Powers, *Symbolic Projection for Image Information Retrieval and Spatial Reasoning.* Academic Press, 1996.

[82] D. Dasgupta and S. Balachandran, "Artificial immune systems: A bibliography," Tech. Rep. CS-04-003, Computer Science Division, University of Memphis, 2006.

[83] V. Neto and H. Nehmzow, "Visual novelty detection for inspection tasks using mobile robots," in *In Proceedings of the 8th Brazilian Symposium on Neural Networks (SBRN 2004)*, (Sao Lus, Brazil), 2004.

[84] S. Marsland, U. Nehmzow, and J. Shapiro, "On-line novelty detection for autonomous mobile robots," *Journal of Robotics and Autonomous Systems*, vol. 51, pp. 191–206, 2005.

[85] C. Diehl, "Real-time object classification and novelty detection for collaborative video surveillance," in *In Proceedings of the IEEE IJCNN Conference, IJCNN 2002*, (Hawaii, USA), 2002.

[86] D. Dasgupta and S. Forrest, "Novelty-detection in time series data using ideas from immunology," in *In Proceedings of International Conference on Intelligent Systems*, (Reno, USA), 1996.