

# Text recognition, event detection and some theoretical aspects of cellular wave computer architectures

Cellular wave algorithms with semantic embedding and a new graph representation of cellular binary dynamics to investigate invertibility

*Ph.D. dissertation*

**Kristóf Karacs**

*Scientific adviser:*

**Tamás Roska, D.Sc.**

ordinary member of the Hungarian Academy of Sciences

*Consultant:*

**Gábor Prószéky, D.Sc.**



Doctoral School of Multidisciplinary  
Engineering Sciences,  
Faculty of Information Technology,  
Pázmány Péter Catholic University  
(PPKE-ITK)



Analogical and Neural Computing  
Systems Laboratory,  
Computer and Automation Research Institute,  
Hungarian Academy of Sciences  
(MTA-SZTAKI)



*“Ask, and it shall be given you; seek, and ye shall find; knock, and it shall be opened unto you.”*

Mt 7:7.



## Acknowledgment

I would like to thank my scientific adviser *Prof. Tamás Roska* for continuously giving me directions throughout the years, for his unfailing enthusiasm in the pursuit of excellence and for his support of all my efforts. I would like to say thank you to *Gábor Prószéky*, who helped me to understand how to approach natural languages, and gave very useful insights regarding my work.

I would like to extend my special gratitude and thanks to my wife, *Ágnes*, and to my parents and our whole family for their continuous support, love and encouragement during my studies.

I would also like to thank *Prof. Wolfgang Porod* for the year I spent at the University of Notre Dame as a research visitor. I am very grateful to *Prof. Árpád Csurgay* and *Ildikó Csurgay* for their support and wise advices, and for the motivating discussions we have had.

I would further like to say thanks to my present and past colleagues, especially to *Gergely Tímár*, who brought my attention to the topic of handwriting recognition and with whom I started to explore the beauty of this area, to *Levente Török* and *László Orzó* for the personal discussions in the lab and their help with mathematical, programming and linguistic difficulties, to *György Cserey* for his friendly support and helpfulness, to *Dávid Bálya*, who was always willing to assist me in scientific questions, to *Mihály Szuhaj*, *Róbert Wagner* and *Anna Lázár*, with whom I have been working on the Bionic Eyeglass, to *Gábor Pohl* and *Gábor Hodász*, who gave me a helping hand when dealing with language corpora, and to *Csaba Rekeczky* and *Ákos Zarándy*, for their help and advices. Many other people contributed to the fruitful intellectual climate and the friendly atmosphere that are invaluable for research, including *Péter Szolgay*, *András Radványi*, *Tamás Szirányi*, *László Kék*, *Péter Földesy*, *István Szatmári*, *Zoltán Szlávik*, *Ahmed Ayoub*, *István Petrás*, *Viktor Gál*, *Viktor Binzberger*, *Dániel Hillier*, *Gergely Soós*, *András Mozsáry*, *Zoltán Fodróczy*, *Péter Jónás*, *Balázs Rakos*, *Zoltán Rácz*, *Alexandra Imre*, *Kristóf Iván* and *András Oláh*.

I would also like to extend thanks to my former teachers, especially to *Gábor Horváth*, who was my adviser for my M.Sc. Thesis and who taught me the basics of neural networks, to *Prof. András Recski*, whose fascinating algebra lectures I will never forget, to *Prof. Lajos Rónyai*, *Katalin Friedl* and *Iván Bach*, for introducing me to the theory of computing, and to *Antal Hirka, OSB* and *Ambrus Pintér, OSB*, my math and physics teachers in the Benedictine High School of Pannonhalma.

Special thanks are due to *Katalin Keserű*, *Gabriella Kékné Ladányi* at MTA-SZTAKI and *Anna Csókási* and *Livia Adorján* at PPKE-ITK for the kind help with all the administration I had to deal with.

# **Text recognition, event detection and some theoretical aspects of cellular wave computer architectures**

*by Kristóf Karacs*

## **Abstract**

The dissertation is connected to locally connected, cellular systems in two ways. On one hand I present detection and recognition algorithms developed for cellular wave computers, on the other hand I give a new method to analyze trajectories of one dimensional cellular automata over finite strings.

Recognition of cursive handwritten texts is a complex, in some cases unsolvable, task. One problem is that in most cases it is difficult or impossible to identify each letter, even if the words are separated. In the new method I developed the identification of letters is not needed due to the extensive and iterative use of semantic and morphological information of a given language. I use spatial feature-codes, generated by a CNN (Cellular Nonlinear Network) based cellular wave computer algorithm, and combine it with the linguistic properties of the given language. Most general purpose Handwriting Recognition Systems lack the ability to integrate linguistic background knowledge because they only use it for post processing recognition results. The high level a priori background knowledge is, however, crucial in human reading and likewise it can boost recognition rates dramatically in case of recognition systems. In the new system visual source is treated as the only input: geometric and linguistic information are given equal importance. On the geometric side word level holistic feature detection is used (without letter segmentation) by analogic CNN algorithms designed for cellular wave computers [16], [20]. The linguistic side is based on a morpho-syntactic linguistic system [48]. A statistical context selection method is also applied to further reduce the output word lists.

Vision is our most important sense, especially in case of orientation and navigation. Lacking this information source visually impaired people are very defenseless in most situations, and there are very few means to compensate their disability. A mobile navigation device would serve them well by providing some sense of safety and independence in many real-life situations. This, of course, has to incorporate some level of understanding the environments it will operate in. Semantics therefore gets an even more important role in this

task. I created a new model of multilevel and multimodal information processing with embedded semantics. Based on this model I made semantic description of specific situations in connection with route number detection and recognition on public transport vehicles and developed cellular wave algorithms for these tasks.

The Cellular Automaton (CA) is a discrete time computational model. In contrast to CNN it was not designated for practical computation, rather to understand the basis of phenomena in cellular architectures. In my work I investigated invertibility of elementary CAs by modeling their operation with directed graphs.



# Table of Contents

Acknowledgment.....	3
Abstract .....	5
1 Introduction .....	17
1.1 The Cellular Nonlinear Network model.....	18
1.2 The CNN core cell and inter-cell interactions.....	20
1.3 The CNN Universal Machine.....	21
2 Handwriting recognition.....	25
2.1 Introduction .....	25
2.1.1 Recognition and reading.....	25
2.1.2 Perception as a recognition model.....	26
2.1.3 Perception and the Analog-logic CNN Visual microprocessor.....	27
2.1.4 Language model .....	28
2.1.5 Hierarchical interpretation.....	28
2.2 System architecture .....	29
2.3 Baseline detection.....	31
2.4 Writing style parameters .....	32
2.5 Word length estimation .....	34
2.5.1 Word image width.....	34
2.5.2 Corrected word width.....	34
2.5.3 Universal Width Limits .....	35
2.5.4 Interval estimation.....	38
2.6 Baseline refinement.....	38
2.7 Feature extraction.....	41
2.7.1 Perceptual features .....	41
2.7.2 Feature classification.....	41
2.7.3 Ascenders and descenders .....	43
2.7.4 Holes.....	44

---

2.7.5	Accents and punctuation marks.....	46
2.7.6	Junction points.....	46
2.7.7	Hills and valleys .....	47
2.8	Shape codes .....	49
2.8.1	Representation of shape codes .....	49
2.8.2	Abstract shape descriptors.....	49
2.8.3	Conversion to regular expression.....	51
2.9	Generating and inverse filtering word lists .....	56
2.9.1	Morpho-lexical word list generation .....	56
2.9.2	Word length filtering.....	56
2.9.3	Filtering based on less reliable features .....	59
2.10	Statistical context selection .....	59
2.10.1	Word graph construction.....	59
2.10.2	Candidate filtering.....	60
2.10.3	Choosing the optimal path.....	60
2.11	Results and Evaluation .....	61
2.11.1	Databases used .....	61
2.11.2	Experimental results .....	61
2.11.3	Analysis of causes of errors.....	64
2.12	Conclusions .....	64
3	Detection and recognition of signs and displays in 2D visual flows.....	65
3.1	Blind Mobile Navigation.....	65
3.1.1	The “Bionic Eyeglass” framework.....	65
3.1.2	Detection and recognition of signs and displays.....	66
3.2	A novel semantic framework for multimodal information processing .....	67
3.2.1	Information flow.....	68
3.2.2	Description of abstraction levels .....	69
3.2.3	Further considerations .....	71
3.3	Sign and display localization.....	71
3.3.1	Localizing black & white signs .....	72
3.3.2	Number localization .....	74
3.3.3	Localizing displays: combining colors and morphology .....	75

---

3.3.4	Registration with previous frames.....	78
3.4	Number recognition.....	80
3.5	Experimental Results.....	82
3.5.1	Blind Acquired Visual Flow Database.....	82
3.5.2	Black & white signs .....	82
3.5.3	Color displays.....	83
3.6	Conclusions .....	84
4	Invertible Cellular Automata.....	85
4.1	Introduction .....	85
4.2	Isles of Eden .....	86
4.3	Rotation invariance.....	90
4.4	Locating points with multiple preimages .....	91
4.4.1	Graph theoretical background .....	92
4.4.2	The new construction: Isles of Eden digraph .....	94
4.4.3	Constructing the Isles of Eden digraph .....	97
4.4.4	Full Isles of Eden digraph .....	100
4.4.5	Effect of global equivalence transformations on Isles of Eden digraphs .....	101
4.5	Detecting isles of Eden with Isles of Eden digraph.....	102
4.6	Invertibility for infinitely many string lengths .....	103
4.6.1	Class “Period 2” .....	105
4.6.2	Class “Period 3” .....	109
4.7	Conclusions .....	112
5	Summary .....	113
5.1	Methods of Investigation.....	113
5.2	New Scientific Results .....	114
5.3	Application areas of the results .....	122
	References .....	123
	The Author’s Journal Papers .....	123
	The Author’s Conference Papers .....	123
	The Author’s Other Publications.....	124

---

Publications related to CNN Technology.....	125
Publications related to Handwriting Recognition .....	126
Publications related to Sign detection and recognition .....	127
Publications related to Natural Language Processing .....	128
Publications related to Cellular Automata .....	129
Other Publications .....	129
Appendix: CNN Templates .....	131
Linear, isotropic templates .....	131
Linear, non-isotropic templates.....	132

## List of Figures

Figure 1.1. A 2-dimensional CNN defined on a square grid. The $i,j$ -th cell of the array is colored green, whereas cells that fall within the sphere of influence of neighborhood radius $r = 1$ (the nearest neighbors) are colored pink.....	19
Figure 1.2. A CNN base cell corresponding to the Equations (1.1) and (1.2). The linear control and feedback terms are represented by voltage controlled current sources ( $B_{ij,kl}$ and $A_{ij,kl}$ ). .....	21
Figure 1.3. The architecture of the CNN Universal Machine. ....	22
Figure 2.1. Relation of key elements of the system. The analog-logic cellular wave computer is the architectural platform on which perception based feature extraction is implemented. Shape codes represent the interface between geometric and linguistic models. The morpho-lexical linguistic framework provides an efficient representation of linguistic knowledge. ....	27
Figure 2.2. Flow diagram of the recognition process.....	30
Figure 2.3. Computed baselines of the word ‘apparently’. Upper and lower baselines cut off ascenders and descenders (Section 2.7.3) respectively from the main word body.....	31
Figure 2.4. (a) A sample row with marked upper (2) and lower (3) baselines, top ascender line (1) and bottom descender line (4). (b) Horizontal connected components diagram of the row.....	32
Figure 2.5. Minimal and maximal UWL values divided by number of characters.....	37
Figure 2.6. Upper baselines of some words. (a)-(c) show examples when the algorithm based on local northern elements fails to detect the correct baseline. (d) is an example for a word with a highly inclined upper baseline.....	39
Figure 2.7. (a) Propagating, directed erosion type templates and (b) typical usage shown by a UMF diagram.....	42
Figure 2.8. Feature maps showing different types of ascenders and descenders. Red and purple refer to normal and narrow ascenders, blue and green refer to narrow and wide descenders, respectively. ....	43

Figure 2.9. HOLE-FILLING4 template fills holes with 4-connected outer boundaries compared to HOLE-FILLING that finds holes with 8-connected outer boundaries. ....	45
Figure 2.10. Comparison between the results of HOLE-FILLING and HOLE-FILLING4 templates. Detection wave is shown in light-blue, red patches are valid holes detected by both templates, and the green patch is an invalid hole detected only by the HOLE-FILLING template. ....	45
Figure 2.11. Classification of holes. Red refers to big holes, green to small holes, yellow to fat holes, cyan to upper holes and magenta to lower holes. ....	46
Figure 2.12. Accents and punctuation marks .....	46
Figure 2.13. Junction points .....	47
Figure 2.14. UMF diagram of hill and valley detection. Inputs labeled uline and dline refer to upper and lower baselines of the word, respectively. ....	48
Figure 2.15. Hills and valleys shown in the word image of ‘number’. Note that not only in-letter hills and valleys are detected, but intra-letter ones too, like the valley between letter ‘m’ and ‘b’ and the hill between letters ‘b’ and ‘e’ .....	49
Figure 2.16. Feature map of the word “today”. Stripes refer to character positions for 5 letters. Gray stripes refer to integer positions, white stripes refer to intra-letter positions. ....	50
Figure 2.17. Mapping of topographic features is ambiguous, the hole in letter ‘o’ can belong to two distinct positions, doubling the number of possible mappings. ....	51
Figure 2.18. Two sample rows of handwriting, containing a whole sentence .....	56
Figure 2.19. Word graph for a part (words 7-13) of the sentence in Figure 2.18. In this case there is only one path through this sub-graph and it gives the correct sentence. ....	60
Figure 2.20. Some sample lines from the LOB corpus .....	61
Figure 2.21. Classes of lengths of output word-lists. ....	63
Figure 2.22. Length of the output word-list in function of the number of characters of the word to be recognized. ....	63
Figure 3.1. Information flow in a multimodal sensory system .....	68
Figure 3.2. Block diagram of the sign detection and recognition framework. ....	72
Figure 3.3. UMF diagram of the algorithm locating signs with a white background. ....	73
Figure 3.4. Sign localization on a tram. (a) Original input frame (b) Binarized and eroded image (c) Smoothed window areas (d) Sign location .....	74

---

Figure 3.5. Flow diagram of display detection. Dotted lines refer to RGB data flows, bold lines refer to multiple binary flows and narrow lines refer to single binary flows.....	76
Figure 3.6. Image sequence showing the detection process. (a) Color corrected image. (b) Wide range filter for yellow channel. (c) Tight range filter for yellow channel. (d) and (e) Noise removed from (b) and (c) respectively (f) Figure reconstruction using (d) as input and (e) as initial state. (g) Noise removed from (f). (h) Result of dark filter on (a). (i) Closing performed on (h). (j) Number image.....	77
Figure 3.7. Enhancement of the number image by superposing the actual frame and the image memory.....	78
Figure 3.8. Feature maps of route numbers: vertical line (blue), upper hole (red), lower hole (green), triangular hole (yellow) .....	80
Figure 3.9. Sample feature maps. Right open holes and their auxiliary lines are shown in cyan, middle vertical line is shown in blue, and upper round hole is shown in red.....	82
Figure 3.10. A sample image on which parts of number images are missing, thus the recognition module fails to recognize them correctly. (a) Original image (b) Output of denoised bright filters (c) Final result of detection.....	83
Figure 4.1. A bipartite graph and its adjacency matrix .....	93
Figure 4.2. De Bruijn graph of strings of length two. ....	94
Figure 4.3. Isles of Eden digraph for local rule 154. The nodes shaded in gray are degenerate denoting that any path consisting solely of them can only generate equal strings. Horizontal gray lines partition the graph in a way that they cross only downward edges.....	95
Figure 4.4. Isles of Eden digraph for local rule 45. Cycles in it refer to different inputs that map to the same output. Horizontal gray lines partition the graph in a way that they cross only downward edges.....	98
Figure 4.5. Isles of Eden digraph of local rule 105 (left) and 150 (right). ....	110
Figure 5.1. Different classes of (a) holes, (b) ascenders and descenders .....	115
Figure 5.2. Mapping of topographic features is ambiguous, the hole in letter ‘o’ can belong to two distinct positions, doubling the number of possible mappings. ....	115
Figure 5.3. Sample word image with computed horizontal upper baseline. ....	116
Figure 5.4. Sign localization on a tram. (a) Original input frame (b) Sign location on binarized input.....	118

---

Figure 5.5. UMF diagram of the algorithm locating signs with a white background. ....	118
Figure 5.6. Sample feature maps. Right open holes and their auxiliary lines are shown in cyan, middle vertical line is shown in blue, and upper round hole is shown in red.....	119
Figure 5.7. Isles of Eden digraph for local rule 45. All cycles in it are of even length, because their containing subgraphs are bipartite. ....	121



## List of Tables

Table I. Minimal and maximal Universal Width Limit vectors and their difference.....	36
Table II. Parameters for horizontal position calculation of features. ....	50
Table III. ASD of the word is given in the first two columns, the third one gives the character position the features in the ASD belong to.....	51
Table IV. Sample extended regular expression for the word shown in Figure 2.16. ....	55
Table V. Default feature macro mappings. ....	55
Table VI. Top ten word candidates returned for the sentence according to their individual frequencies. Correct words are shown in inverse. ....	56
Table VII. Assigned width values in computation of the expected word width .....	57
Table VIII. Minimal and maximal Universal Width Limit vectors referring to average letter width multipliers for in unit character width and their difference .....	58
Table IX. Top ten word candidates after word length filtering. Correct words are shown in inverse.....	58
Table X. Comparison of lexicon reduction methods.....	62
Table XI. Typical tasks considered for the Bionic Eyeglass.....	66
Table XII. Event generation by a series of modules activated by an audio feature .....	70
Table XIII. Feature conversion table.....	81
Table XIV. Detection and recognition results.....	83
Table XV. Mapping rules of local rule 45 and 90.....	91
Table XVI. Mapping rules of local rule 45. ....	98
Table XVII. Mapping rules of local rule 154.....	99
Table XVIII. Rules that are invertible for more than one string length for $2 < I < 20$ . Rules belonging to the same equivalence class are displayed on the same line. ....	104
Table XIX. Examples of cycles of even length for Rule 45.....	107
Table XX. CA rules that are invertible for infinitely many pattern lengths.....	121



# 1 Introduction

Every seeing human takes it for granted from early childhood to be able to recognize objects and patterns around us. It is so evident for us we rarely realize how complex the visual pattern matching process of our brain is. For some time, using the advances of technology, research of human vision is gaining increasing attention among biologists, but there is still a long way to go to fully understand the processes that take place in our mind when we see people and objects and we are able to recognize them. In parallel, computer scientists have been trying to create models for visual recognition tasks, but they faced enormous difficulties, and solutions could be made only for a very limited set of problems, typically with high power dissipation. There are two main difficulties in performing such pattern recognition tasks. Both are connected to the fact that even among the most common and simplest tasks it is hard to find one in which the human brain does not make use of its huge network of highly associative semantic knowledge. One of the problems is that there are no fully adequate models representing these networks within the realms of current technological boundaries. The other one is that there are several knowledge levels in these networks and the brain uses them in a deeply parallel way, indeed sensing and processing cannot even be really distinguished. This parallel model is very far from traditional sequential models, in which different levels of information are processed in sequential blocks, so that they can be handled independently, and some feedback is added to the system to handle problems arising from the base model and to provide stability.

When I started my research I was convinced that plasticity and parallel evaluation of background knowledge have to be built into our models. The amount of incoming data in recognition tasks is huge, and extracting relevant features is the only way to tackle with it. If background knowledge is used only after the feature extraction stage, then many details might be discarded too early that makes it impossible to correctly recognize the target. During my research I was challenged with two interesting and complex pattern matching problems:

Offline recognition of handwritten cursive texts, and localization and recognition of route number signs on public transport vehicles. Both of these tasks are deeply embedded in a semantic context.

Handwriting recognition is the machine analogue of the human reading process. My goal was to design a system mimicking the human reading by letting the linguistic knowledge to act immediately on letter and word level features without their explicit recognition. Handwriting recognition is discussed in Chapter 2.

The algorithms for sign detection and recognition were motivated by a project aiming to create a portable device for blind and visually impaired people, called the “Bionic Eyeglass”, to help them in everyday problems they face due to the lack of vision. Such applications require deep understanding of the environment where the observed events occur, similarly to how reading a sequence of letters is embedded in a linguistic environment.

The ability to identify a bus or a tram is one of the most important ones among the dozen crucial functions determined by the representatives of potential users. Chapter 3 presents a semantic framework for processing information coming from sensors of different modalities, and its application for sign recognition problems, namely algorithms created by means of transforming semantic descriptions of typical signs, displays and vehicles into procedures for cellular wave computers.

Chapter 4 presents results in the area of theory of binary cellular nonlinear networks, also known as cellular automata. I gave a method that allows analyzing all orbits of one dimensional cellular automata of neighborhood size one for the existence of points where trajectories merge and proved that the nonexistence of such points is equivalent to the nonexistence of Garden of Eden<sup>1</sup> states as well as all orbits being Isles of Eden<sup>2</sup>. This topic belongs to the theoretical side of the field of locally connected cellular architectures.

## 1.1 The Cellular Nonlinear Network model

This section gives a brief introduction to the theory of Cellular Nonlinear/Neural Networks (CNN). A CNN is defined by the following principles [13], [14], [15]:

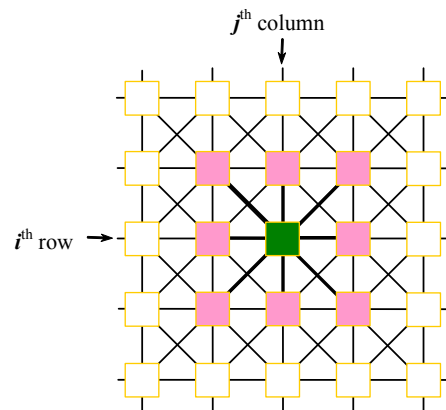
---

<sup>1</sup> A state of a CA is called Garden of Eden *iff* no other state is mapped to it, i.e. it has no predecessor under the global map.

<sup>2</sup> A state of a CA is called Isle of Eden *iff* it is uniquely mapped to itself under the  $n^{\text{th}}$  iterated global map. For the exact definition see Section 4.2.

- A spatially discrete collection of continuous nonlinear dynamical systems called cells where information can be encrypted into each cell via three independent variables called input ( $u$ ), initial state ( $x(0)$ ), and threshold ( $z$ ).
- A coupling law relating one or more relevant variables of each cell to all local neighboring cells located within a prescribed sphere of influence  $S_r(i,j)$  of radius  $r$  centered at cell  $(i,j)$ .

Figure 1.1 shows a 2D rectangular CNN composed of cells that are connected to their nearest neighbors. Due to its symmetry, regular structure and simplicity this type of arrangement (a rectangular grid) is primarily considered in all implementations.



**Figure 1.1.** A 2-dimensional CNN defined on a square grid. The  $i,j$ -th cell of the array is colored green, whereas cells that fall within the sphere of influence of neighborhood radius  $r = 1$  (the nearest neighbors) are colored pink.

The bias (also referred to as the “bias map”) of a CNN layer is a grayscale image. The bias map can be viewed as the space-variant part of the cell threshold. By using a pre-calculated bias map, “linear” spatial adaptivity can be added to a template. If the bias map is not specified, it is assumed to be zero.

The mask (also referred to as the “fixed-state map”) of a CNN layer is a binary image specifying whether the corresponding CNN cell is in active or inactive state in the actual operation. Using the binary mask is one of the simplest ways to incorporate “nonlinear” spatial adaptivity into templates. If the mask is not specified, it is assumed that all CNN cells are in active state, that is, the initial state is not fixed.

In order to specify fully the dynamics of the array, the boundary conditions have to be defined. Cells along the edges of the array may see the value of cells on the opposite side of

the array (toroid boundary), a fixed value (Dirichlet-boundary) or the value of mirrored cells (zero-flux boundary).

## 1.2 The CNN core cell and inter-cell interactions

The CNN paradigm does not specify the properties of a cell. As the basic framework throughout this dissertation, let us consider the standard, first order two dimensional ( $M \times N$ ) CNN array, in which the cell dynamics is described by the following nonlinear ordinary differential equations:

State equation:

$$\dot{x}_{ij}(t) = -\frac{1}{\tau}x_{ij}(t) + z_{ij} + \sum_{k,l \in S_r(i,j)} A_{ij,kl} \cdot y_{kl}(t) + \sum_{k,l \in S_r(i,j)} B_{ij,kl} \cdot u_{kl}(t) \quad (1.1)$$

Output equation:

$$y_{ij}(t) = f(x_{ij}(t)) = \begin{cases} 1 & \text{if } x_{ij}(t) \geq 1 \\ x_{ij}(t) & \text{if } -1 \leq x_{ij}(t) \leq 1 \\ -1 & \text{if } x_{ij}(t) \leq -1 \end{cases} \quad (1.2)$$

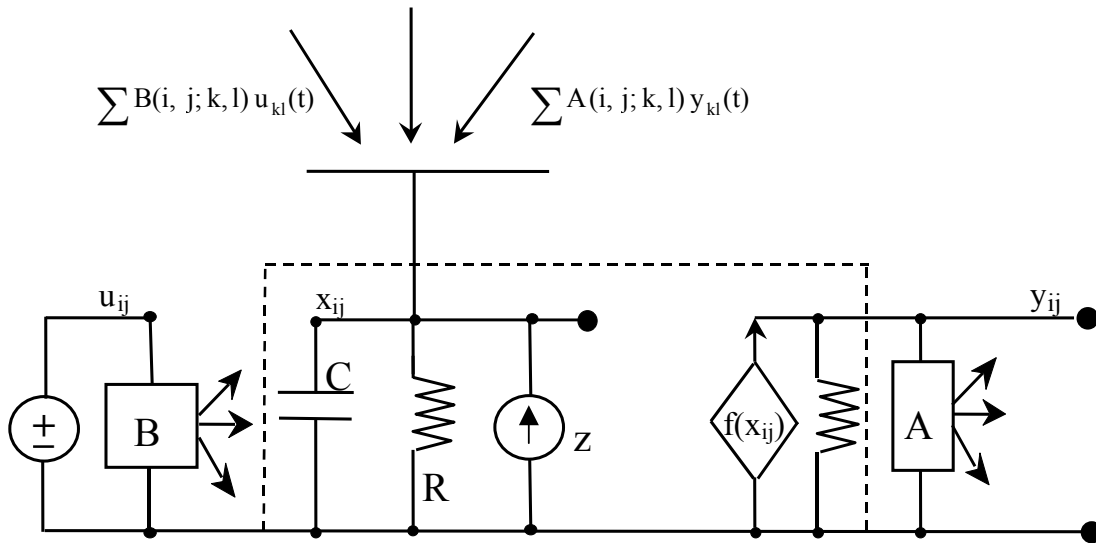
where

- $x_{ij}$ ,  $y_{ij}$ ,  $u_{ij}$  are the state, the output, and the input variables of the specified CNN cell, respectively, The state and output vary in time, the input is static (time independent),  $ij$  refers to a grid point associated with a cell on the 2D grid, and  $kl \in S_r$  is a grid point in the neighborhood within the radius  $r$  of the cell  $(i,j)$
- $z_{ij}$  is the cell threshold (also referred to as bias) which could be space and time variant
- term  $A_{ij,kl}$  represents the linear feedback,  $B_{ij,kl}$  the linear control
- $\tau$  is the cell time constant, for simplicity the default value is  $\tau = 1$
- function  $f(\cdot)$  is the output nonlinearity, in our case a unity gain sigmoid
- $t$  is the continuous time variable.

Equations (1.1) and (1.2) define a rather complex framework for computation. The first part of Equation (1.1) is called cell dynamics, whereas the additive terms following it

represent the synaptic interactions. If the threshold  $z_{ij}$  is space-invariant, then the template (space-invariant case). Equation (1.2) is the output equation.

A CNN base cell, when implemented via analog electronic circuits, is shown in Figure 1.2.



**Figure 1.2.** A CNN base cell corresponding to the Equations (1.1) and (1.2). The linear control and feedback terms are represented by voltage controlled current sources ( $B_{ij,kl}$  and  $A_{ij,kl}$ ).

The time constant of a CNN cell is determined by the linear capacitor ( $C$ ) and the linear resistor ( $R$ ) and it can be expressed as  $\tau = RC$ . A CNN cloning template, the instruction of the CNN array, is given by the linear and nonlinear terms implemented by the voltage controlled by current sources.

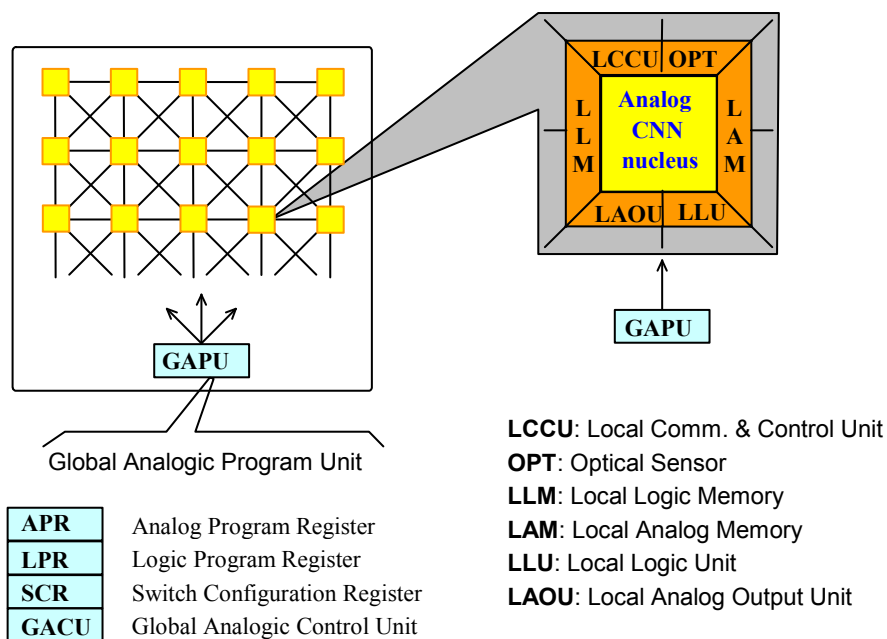
The cell model in Figure 1.2 is the so called Chua-Yang model. In some cases, from implementation point of view, the so called full-range model is more convenient [17]. In the full range model, the state and input is connected and the circuit in the dashed line area is composed of a capacitor, a nonlinear resistor and a current source.

In addition to the analog electronic circuit there are other physical implementations, e.g. optical, digital, and electromagnetic.

### 1.3 The CNN Universal Machine

All early neural network chip realizations had a common problem: they implemented a single instruction only, thus the weight matrix was fixed when processing some input. Reprogramming (i.e. changing the weight matrix) was possible for some devices but took in order of magnitudes longer time than the computation itself.

This observation motivated the design of the CNN Universal Machine (CNN-UM) [16], a stored program nonlinear array computer. This new architecture is able to combine analog array operations with local logic efficiently. Since the reprogramming time is approximately equal to the settling time of a non-propagating analog operation it is capable of executing complex analogic algorithms. To ensure programmability, a global programming unit was added to the array, and to make it possible an efficient reuse of intermediate results, each computing cell was 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. The architecture of the CNN-UM is shown in Figure 1.3.



**Figure 1.3. The architecture of the CNN Universal Machine.**

As illustrated in Figure 1.3 the CNN-UM is built around the dynamic computing core of a simple CNN. An image can be acquired through the sensory input (e.g. **OPT**: Optical Sensor). Local memories store analog (**LAM**: Local Analog Memory) and logic (**LLM**: Local Logical Memory) values in each cell. A Local Analog Output Unit (**LAOU**) and a Local Logic Unit (**LLU**) perform cell-wise analog and logic operations on the stored values. The output is always transferred to one of the local memories. The Local Communication and Control Unit (**LCCU**) provides for communication between the extended cell and the central programming unit of the machine, the Global Analogic Programming Unit (**GAPU**). The **GAPU** has four functional blocks. The Analog Program Register (**APR**) stores the analog program



---

instructions, the CNN templates. In case of linear templates, for a connectivity  $r = 1$  a set of 19 real numbers have to be stored (this is even less for both linear and nonlinear templates assuming spatial symmetry and isotropy). All other units within the GAPU are logic registers containing the control codes for operating the cell array. The Local Program Register (**LPR**) contains control sequences for the individual cell's LLU, the Switch Configuration Register (**SCR**) stores the codes to initiate the different switch configurations when accessing the different functional units (e.g. whether to run a linear or nonlinear template). The Global Analogic Control Unit (**GACU**) stores the instruction sequence of the main (analogic) program. The GACU also controls timing, sequence of instructions and data transfers on the chip and synchronizes the communication with any external controlling device. It has its own global analog and logic memories (**GAM** and **GLM**, respectively) and global Arithmetic Logic Unit (**ALU**). As a special case the GACU can be implemented by a digital signal processor (DSP) or a microcontroller.

Synthesizing an analogic algorithm running on the CNN-UM the designer should decompose the solution in a sequence of analog and logical operations. A limited number of intermediate results can be locally stored and combined. Some of these outputs can be used as a bias map (space variant current) or fixed-state map (space-variant mask) in the next operation adding spatial adaptivity to the algorithms without introducing complicated inter-cell couplings. Analog operations are defined by either linear or nonlinear templates. The output can be defined both in fixed and non-fixed state of the network (equilibrium and non-equilibrium computing) depending on the control of the transient length. It can be assumed that elementary logical (**NOT**, **AND**, **OR**, etc.) and arithmetical (**ADD**, **SUB**) operations are implemented and can be used on the cell level between LLM and LAM locations, respectively. In addition data transfer and conversion can be performed between LAMs and LLMs. All templates mentioned but not defined in the paper can be found in the Cellular Wave Computing Library [12].



## 2 Handwriting recognition

### 2.1 Introduction

#### 2.1.1 *Recognition and reading*

Reading plays a substantial role in our everyday life since major part of our knowledge is stored in written texts. Although higher and higher percentage of documents are created electronically by means of word processors, a huge amount of data gets still recorded by means of handwriting where the use of computers is not suitable or feasible. But in many cases a fundamental need is present to have handwritten information available in a digital form to allow for fast searching, editing and archiving. This need is what motivates the development of handwriting recognition systems that aim to perform the process of reading and typing a handwritten text by a machine.

There are two main branches of Cursive Script Recognition (**CSR**): Offline CSR and Online CSR. Offline recognition deals with a handwritten text sometime after the writer has created it. This means that the input of the recognition engine is a binary or grayscale image that contains handwritten text. In case of online CSR the recognition takes place ‘real-time’, during the writing process, and the input is usually provided by a device that can measure temporal information and trajectory data, such as an electronic pen or a touch-screen. Our work concentrates on the offline case where a substantial part of the data used in the online case is unavailable, since dynamic characteristics can only be extracted at the time of writing.

Offline CSR consists of many really complex subtasks. Only vertical applications, aiming to give solution for only a section of the field, have reached a level of quality required in commercial applications. Continuous advances in handwriting recognition technology in addition to the present industrial demand for CSR systems could boost a much wider interest for recognition devices in the near future. This possibility gave the main motivation for our work. If a more general solution could perform well enough to reach the critical level of

recognition that proves to be acceptable for end users, then it would open the way for a wide range of practical applications related to common personal activities thereby contributing to the improvement of availability and processing of information.

### ***2.1.2 Perception as a recognition model***

We define recognition in a general sense as a certain level of interpretation of the data acquired from sensors. Experiments show that in case of Handwriting Recognition this level should be fairly high to be able to achieve acceptable results [39]. According to [24] researchers tend to think that it seems appropriate to base an automated handwriting reading system on the human perception model, since they can read handwritten texts with apparent ease. This is the reason why [29] concludes that Human Handwriting Processing (**HHP**) systems are rather going to read than to recognize the text.

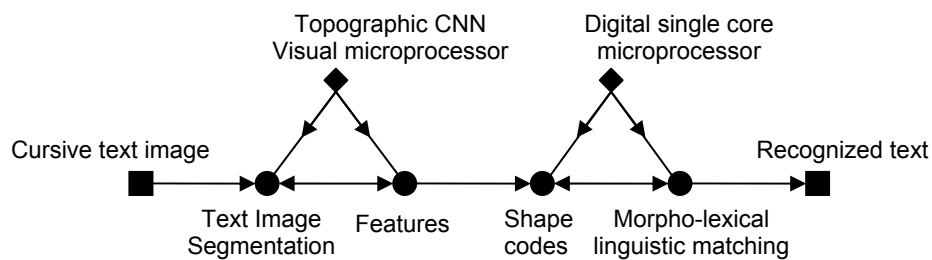
What really gives motivation to perception based methods is the Sayre paradox [34]: “To recognize a letter one must find the borders of it first. To find the borders we need to know first what letter it is.” Many approaches have been already published to overcome this crucial problem of CSR (oversegmentation to primitives, whole word recognizers, Hidden Markov Models, etc.). The answer one gives to the Sayre paradox characterizes the whole architecture of the recognizer. Considering the facts mentioned above we have chosen the holistic paradigm to enable a development track flexible enough to avoid having to make restrictions on the input.

The perception model focuses on simple features that make the shape of words different. These can be either global or local features. During the complex process of human perception, the detected shape features are immediately combined in the mind with all available background information, a huge part of which is related to language. This is the point where the process of interpretation starts.

There are three essential parts we have to model to realize a similar combinational technique:

1. adequate detection and representation of perceptual features
2. good language model
3. flexible interface between 1 and 2 that enables the recall of linguistic patterns based on geometric patterns

The last component is implemented via *shape coding*. A shape code contains the features detected in a word image assigned to possible letter positions in the word. Shape codes are represented as (extended) regular expressions, thus allowing to be used as filters in the linguistic model, essentially realizing a language reduction system. The first two components are discussed in the following subsections. Figure 2.1 shows the role of these components in our system.



**Figure 2.1. Relation of key elements of the system. The analog-logic cellular wave computer is the architectural platform on which perception based feature extraction is implemented. Shape codes represent the interface between geometric and linguistic models. The morpho-lexical linguistic framework provides an efficient representation of linguistic knowledge.**

### 2.1.3 Perception and the Analog-logic CNN Visual microprocessor

Human perception is parallel to a great extent. *Cellular Nonlinear Network* based computers perform outstandingly in global image processing tasks due to their architecture [16], [19]. The operation of the CNN Visual microprocessor is based on local interactions, making it extremely suitable for parallel processing. This is the main reason why it is very efficient in modeling the human reading process.

Almost every algorithm used for detecting obtrusive features needs complex, computationally intensive operations on the input images making reasonable the use of the fastest possible parallel hardware architecture, the CNN based wave computer, based on the Analog-logic CNN Visual microprocessor. On wave computers simple image processing tasks can be performed with a single elementary instruction. Templates that consist of matrices interaction pattern containing real values define these instructions. More complex tasks can be implemented by template-algorithms that can be described by UMF diagrams, containing several template- and logical instructions [20]. However, digital operations are still computed on a digital platform. The relation the dual hardware architecture to the main processing steps of the system is shown in Figure 2.1.

### ***2.1.4 Language model***

By analyzing the whole recognition process we have found some initial steps in case of which understanding the text or knowing the language has very small or no role. These steps belong to Text Image Segmentation (Figure 2.1). Using the above partitioning the second part includes tasks that require some sort of linguistic knowledge. Therefore the appropriate use of the context information is very important and practically inevitable. Trying to read a text in a completely unknown language humans become abecedarians too, making a lot of errors. Most recent handwriting recognition systems attempt to use a word list based language model. Lexical databases with fully inflected forms are fairly standard for recognition systems, mainly where a small closed vocabulary is used, and new, unknown or ad hoc word formations are not required [26]. This procedure is convenient in languages with very small inflectional paradigms, like English. However, agglutinative languages, such as Turkish, Finnish and Hungarian, have complex inflectional and derivational morphology with significantly more endings on all verbs, nouns, adjectives and pronouns. The number of endings increases the size of a basic vocabulary by a factor of thousands, thus a word list based approach provides no real solution. For efficient composition of inflected forms and to avoid a finite but unmanageable explosion of lexicon size morpho-lexical and syntactic models and appropriate algorithmic morphological techniques are needed [51].

### ***2.1.5 Hierarchical interpretation***

The linguistic knowledge one uses when reading a text consists of multiple levels that are tightly interwoven and have no clear interfaces. Some of the most significant categories are the following: structure (syntax), meaning (morphology and semantics), context and use (pragmatics). Other types of linguistic knowledge (e.g. phonology) can influence the process of reading too, but their effect is small.

Letter features detected in the text image collaborate with background knowledge of existing words. Word candidates form potential sentences, and knowledge of grammar helps to choose between words, indirectly influencing the perception at feature level. This information flows up and down among different levels. This flow has been modeled in [29] for bank check processing, considering three different levels: features, letters and words. In case of a coherent text at least one additional level of processing is necessary that models expression-level relations.

The higher the level of knowledge we have in the given field the more complex understanding we can obtain of the text. This statement holds also for recognition systems: increasing the number of knowledge levels can yield to more complex comprehension and thus to better recognition results.

One could argue that humans are able to read texts without any meaning. That is true, but they are much less fluent in meaningless texts. One could further add that humans can also read grammatically incorrect sentences. That is also true, but it hardens their job. Finally we have to admit that humans can even read texts consisting of nonexistent strings, but at a much lower speed than the former two cases.

The levels of linguistic knowledge refer to units of different sizes of the text, from letters to the whole text itself. When dealing with automated systems the most basic linguistic support to provide is at the level of morphology, which can be best set apart from other levels. This is partly due to the fact that the identification of word boundaries is the most separable task from the actual recognition. Therefore in word recognition tasks we can mostly make use of morphologic linguistic knowledge [10], but, when dealing with sentences, paragraphs, or longer texts, the use of higher levels becomes inevitable.

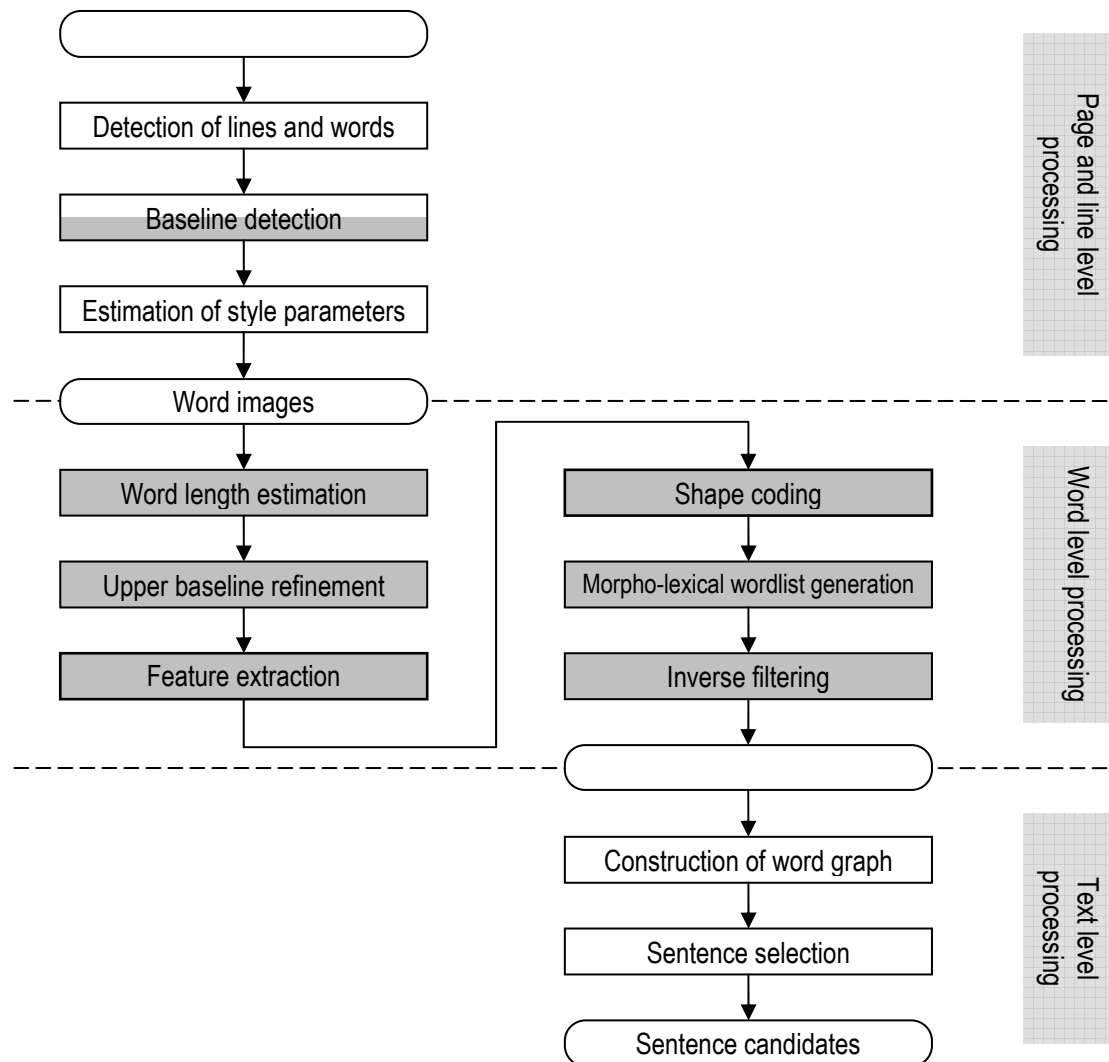
As mentioned earlier, higher level linguistic knowledge enables better recognition results, but representing high level knowledge in general requires such a complex, intelligent, adaptive and flexible framework that has not been developed yet. However, we will demonstrate that, even without understanding the meaning of the text, using only word collocation statistics, confidence values of sentence candidates can be estimated based on bigram frequencies, enabling sentence level recognition directly from the output of the lexicon reduction system.

## 2.2 System architecture

An image containing multiple lines of cursive handwritten text serves as the input of the system. In the field of Text Image Segmentation we strongly rely on results published in [3] and partly in [38], which include pre-processing, line detection, word detection, and upper and lower baseline extraction algorithms. Lines and words are detected by means of analogic algorithms based on the ones described in [3]. The process includes noise filtering, adaptive line detection, skew correction, and word position detection.

Once word candidate images are extracted, their shape morphology can be analyzed individually. Word level processing steps are shown in a gray rectangle in Figure 2.2. At first

baselines are extracted for both lines and words (Section 2.3). Parameters characteristic to the writing are also computed for every line (Section 2.4). At the word level a rough estimation is given for the number of characters of the word (Section 2.5). Based on this estimation the upper baseline can be further refined (Section 2.6).



**Figure 2.2. Flow diagram of the recognition process**

For each type of feature a feature image is generated from the word image, where features can be extracted from, as described in Section 2.7. The features are combined into a shape code in form of an Abstract Shape Descriptor (Section 2.8.2) that is then transformed into a regular expression (Section 2.8.3). This is the input of the linguistic module that searches the matching words giving a word list as an output (Section 2.9.1). The linguistic module does not simply perform a symbol-to-symbol conversion from its input. A direct conversion is insufficient, because source symbols are under-specified or incorrectly recognized. The



morpho-lexical and syntactic model used can help this process as it recognizes elements of the language, extracting meaningful passages from the input sequence. A so called inverse filtering takes place on the generated word list using global and less reliable letter features (Section 2.9.2 and 2.9.3).

Section 2.10 describes the construction of the word graph from candidate lists and shows a sample method for sentence selection using a statistical approach. Section 2.11 summarizes experimental results.

## 2.3 Baseline detection

The handwritten text image can be divided into three horizontal regions that are separated from one another by the upper and lower baselines. Baselines have key significance in feature detection and classification (see Section 2.7). Precise detection of baselines is very important because inaccuracy can lead to false features and thus to misrecognition.



*Figure 2.3. Computed baselines of the word ‘apparently’. Upper and lower baselines cut off ascenders and descenders (Section 2.7.3) respectively from the main word body.*

Baseline detection is performed at word and line level as well. This section deals with word level detection, whereas line level is mentioned in the next one. Baseline extraction is performed by fitting lines on upper and lower extreme points of the word image by using the analogic algorithm given in [3]. Here we only briefly describe the basic steps.

The method basically computes the pseudo convex hull of the word using the **HOLLOW** template, and finds the lowest and uppermost points of the pseudo convex hulls using the **LOCAL SOUTHERN ELEMENT (LSE)** and the **LOCAL NORTHERN ELEMENT (LNE)** detector templates respectively. Pixels are clustered based on their vertical distance from the center pixel that is determined by the **FINDCENTER** algorithm [12]. Pixels belonging to clusters whose center has a larger vertical distance from the center pixel than a threshold are considered outliers. The threshold value is derived from the distance of baselines computed

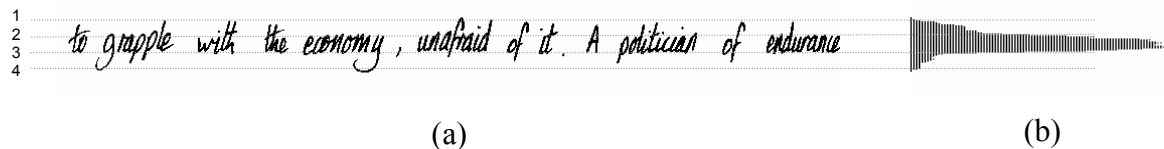
for the whole line. The baselines are fitted with linear regression on the remaining pixels, and the image is rotated so the lower baseline becomes horizontal.

It is worth to mention that theoretically the baselines are inter-pixel lines, but in practice we use one pixel wide baselines, with the upper edge of the upper baseline and the lower edge of the lower baseline referring to the theoretical baselines.

Position of the upper baseline determined in this step is refined later in Section 2.6, by means of a method based on horizontal connected components and horizontal histogram of the word image using character number estimates (computed in the Section 2.5) to determine an appropriate threshold.

## 2.4 Writing style parameters

To be able to extract valid features from the cursive text parameters characteristic to the writing style have to be determined. The parameters include baseline distance, thresholds for ascender and descender heights, average letter width and slant. Parameters are calculated for each line, because text size and style can change gradually as one writes. Nevertheless the ratio of size parameters usually still remains constant to a degree, therefore it is stored and compared through lines to enable more precise parameter estimations.



**Figure 2.4.** (a) A sample row with marked upper (2) and lower (3) baselines, top ascender line (1) and bottom descender line (4). (b) Horizontal connected components diagram of the row.

*Baseline distance* is measured between the upper and lower baselines (line 2 and 3 in Figure 2.4), that are estimated using a method described in [3]. The algorithm is based on the same idea as the one described in Section 2.3 for the detection of baselines of words, but no refinement is made and it is applied to a whole line image. *Ascender and descender height thresholds* are defined to be equal to the 30% of the distance between the top ascender line (line 1 in Figure 2.4) and the upper baseline and the bottom descender line (line 4 in Figure 2.4) and the lower baseline, respectively. These horizontal markers are determined using the horizontal connected components of the row image (Figure 2.4 (a)), detected by the

**HORIZONTAL CONNECTED COMPONENTS DETECTOR (HCCD)** template. Top ascender and bottom descender lines are defined to be at the outer edge of shoulders containing at least three components.

Let  $n$  denote the number of characters on the line. To calculate the *average letter width* in pixels at first we have to give an estimate for  $n$ . We rely on an empirical measurement performed on a training set of 500 words, in which we examined the number of horizontal components per letter. We will show that a good estimate can be given for  $n$  based on horizontal component counts. Let  $C$  be the set of horizontal component counts for the whole vertical interval of the line image and let  $E$  denote the set of those component counts that are above a certain threshold. The threshold is the maximal component count in the region between the baselines multiplied by a scalar  $\mu \in [0,1]$ .  $E$  is formally defined as

$$E = \{c \in C : c > \mu \max\{C\}\} \quad (2.1)$$

Let  $\bar{E}$  be the average over  $E$ . Our experiments show that the  $\bar{E}/n$  ratio is rather independent from the writing style. The smaller the variance of this ratio, the more exact the character number estimation. We examined the variance over the training set considering values for  $\mu$  between 0 and 1 with steps of 0.05, and  $\mu_0 = 0.7$  has been determined to minimize the variance  $\bar{E}/n$ .

With  $\mu = \mu_0$  the mean of the ratio is 1.4 with a standard deviation of 0.064. We use the (1.25;1.55) interval for the approximation, which corresponds to a 98% confidence interval supposing the distribution is normal. By calculating  $\bar{E}/1.25$  and  $\bar{E}/1.55$  we obtain an upper and a lower bound for  $n$ , and the average letter width is calculated by summing the widths of word images and dividing it by the estimated number of characters. The computed average letter width is used in Section 2.5 to estimate the number of characters in a word.

For *slant* detection an algorithm given in [38] is used. The algorithm calculates the number of overlapping pixels between the word image and a line of a known slant angle slid along the word image for every position of the line and sums the three highest values. The computation is performed for different slant angles and the one with the highest sum is chosen as the correct slant.

## 2.5 Word length estimation

### 2.5.1 *Word image width*

The aim of this block is to give a good estimate of the number of letters in the word without segmenting it. The primary goal of estimating the number of letters in the word is to enable the binding of features to letter positions. But, as described in Section 2.6, an estimate for the number of characters can be also used to allow for more precise determination of the upper baseline of the word image.

At this point of the processing the only information source to rely on to calculate an estimate is the width of the word image. Evidently the net width of the word image is correlated with the number of letters in the word, but due to variations in letter widths a given pixel width can refer to multiple word lengths. The reasons for the variations are twofold. On one hand letters have different shapes having different normal widths. Although this variance is predictable, letter level width information can be used in character recognition in case of segmentation based methods. In a holistic approach it can only be taken into account after the morphological word generation, when the word width can be compared to the expected width of word candidates. On the other hand the width of individual appearances of each letter also varies due to the nature of human handwriting. This makes it unreasonable to assign a single letter-number to a given pixel width, because such a model would lead to a very high error rate, since if word length estimation provides an incorrect length, then the correct word will not match the generated shape descriptor and morphologic filtering will exclude it from the output word list.

Therefore we use an interval technique, utilizing the flexibility of the linguistic system that allows for this level of ambiguity. This means that we estimate the minimum and maximum number of letters for every input word image, based on the corrected width of the image. Although the interval estimation leads to longer word candidate lists at the output of the morphologic search and thus to lower reduction rate respected to a single value estimation, but a much better recognition rate can be achieved with it. Furthermore, the length of the output list can be decreased later by a more a sophisticated length filtering method after the morphologic search, based on the analysis of the candidates (Section 2.9.2).

### 2.5.2 *Corrected word width*

To increase the performance of the estimator we eliminate some outliers in the word image. These include mainly horizontal leading and ending strokes of the first and the last

letter, because the width of these outliers can be significant respected to the character width, but it is independent of the number of characters in the word. The same holds for descenders (Section 2.7.3) that may reach out horizontally far beyond the borders of the word image part between the baselines. In case of some special writing styles ascenders might also have long horizontal outliers that are uncorrelated with the number of characters and therefore should be removed too. By neglecting the mentioned outliers the additive noise to the width of the word caused by them can be filtered out, which leads to a lower variance of word width, and thus allows for a better estimate of the number of characters. The mentioned outliers are erased from the word image only temporarily, after the word length estimation process they are reinserted since they may contain features that are crucial in the recognition phase.

Descenders are filtered by masking out the region below the lower baseline. The method is similar to the detection of descenders, but in this case the opposite part of the image is masked. To eliminate leading and ending strokes, we look for vertical regions on the left and the right side of the word image box, wherein only a single stroke is located. These vertical boxes have to be of a certain minimal width to be considered as containing a leading or ending stroke. To locate the potential vertical regions we rely on two auxiliary images, derived from the word image. On the vertical histogram, abscissas with a value under a certain threshold (maximal stroke-width) indicate regions that possibly contain outliers. The **VERTICAL CONNECTED COMPONENTS DETECTOR** template (VCCD) generates the other image, which enables to locate abscissas where only one vertical run is present. The widest abscissa intervals on the sides of the intersection of the two sets give the actual outlier regions.

Henceforth the term *word width* will refer to the corrected word width, unless otherwise noted.

### 2.5.3 *Universal Width Limits*

To estimate the upper and lower bounds for the number of characters in the word, we introduce width limit vectors. Limits refer to the minimal and maximal pixel widths of words containing a given number of characters, and can be measured on a training set. If we divide the limit vectors by the average letter width, then we obtain Universal Width Limit (UWL) vectors that are expressed in terms of unit character width. UWL vectors can be used in case of arbitrary cursive inputs because they are independent from the letter sizes and to a great extent from the writing style as well. The estimated interval for the number of characters in the word is calculated by comparing the width of the word in terms of unit letter width to the

UWL vectors. The UWL vectors have been determined on a set of 1000 words for widths up to 14 characters by the following method.

Let us partition the training set  $X$  into disjoint  $X_i$  subsets for which the letter size and style can be considered constant. Let  $W(x)$  denote the width of a word  $x \in X_i$  measured in pixels, and let  $C(x)$  denote the number of characters in  $x$ . The average letter width over  $X_i$  is given by

$$\overline{w}_i(X_i) = \frac{\sum_{x \in X_i} W(x)}{\sum_{x \in X_i} C(x)} \quad (2.2)$$

The width limits measured in pixels for length  $n \in \mathbb{N}^+$  are as follows:

$$\begin{aligned} w_{\min}(n, X_i) &= \min\{W(x \in X_i : C(x) = n)\} \\ w_{\max}(n, X_i) &= \max\{W(x \in X_i : C(x) = n)\} \end{aligned} \quad (2.3)$$

Dividing these limits by the average letter width and selecting the lowest and largest values we can obtain minimal and maximal UWL vectors:

$$\begin{aligned} \Omega_{\min}(n) &= \min_i \left\{ \frac{w_{\min}(n, X_i)}{\overline{w}_i(X_i)} \right\} \\ \Omega_{\max}(n) &= \max_i \left\{ \frac{w_{\max}(n, X_i)}{\overline{w}_i(X_i)} \right\} \end{aligned} \quad (2.4)$$

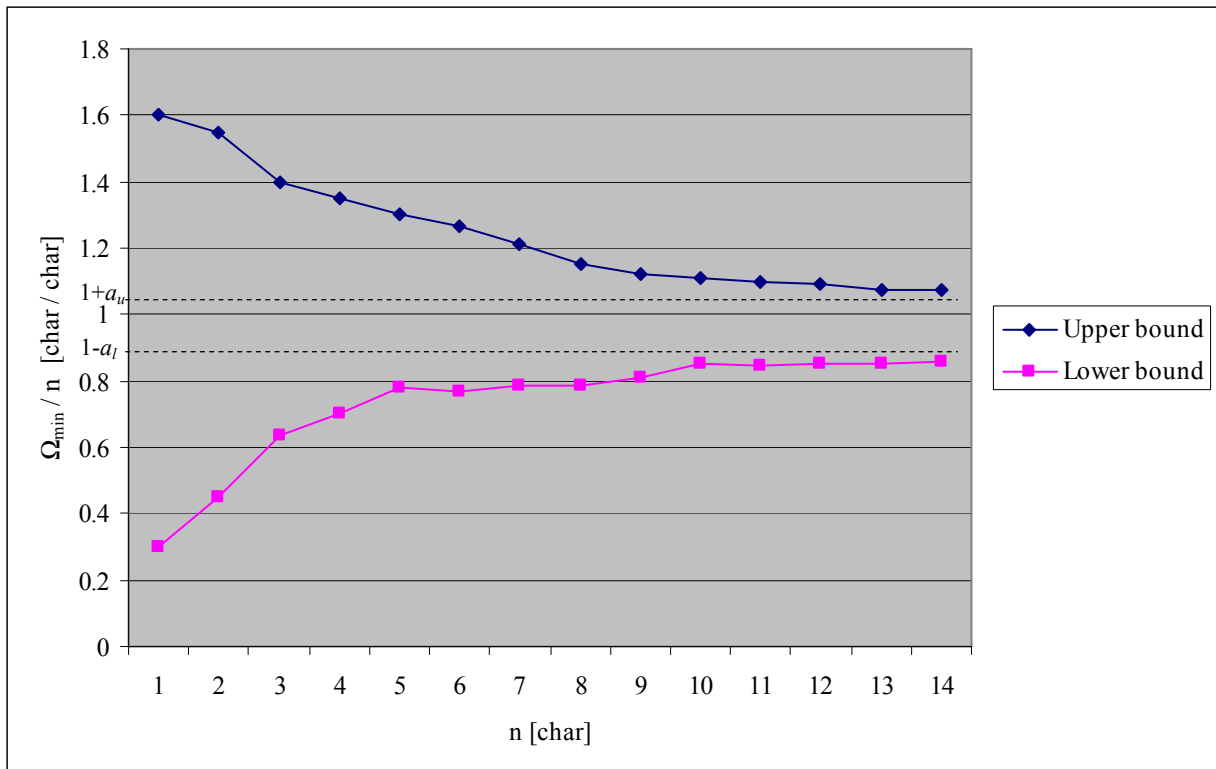
**Table I. Minimal and maximal Universal Width Limit vectors and their difference**

No. of characters	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\Omega_{\min}$	0.3	0.9	1.9	2.8	3.9	4.6	5.5	6.3	7.3	8.5	9.3	10.2	11.1	12.0
$\Omega_{\max}$	1.6	3.1	4.2	5.4	6.5	7.6	8.5	9.2	10.1	11.1	12.1	13.1	14.0	15.0
$\Omega_{\max} - \Omega_{\min}$	1.3	2.2	2.3	2.6	2.6	3.0	3.0	2.9	2.8	2.6	2.8	2.9	2.9	3.0

Measured minimal and maximal UWL vectors are shown in Table I. Note that the difference between maximal and minimal UWL values grows with the number of characters for shorter words, but it saturates for six letter words and above at around three characters.

This phenomenon can be explained by the following simple stochastic model. Let us denote the random variable for the width of a word of length  $n$  by  $Y_n$  and the random variable for the width of the letter  $n+1$  by  $L_{n+1}$ . Calculating the increment of variance in the model when increasing the length of the word by one gives

$$\text{var}(Y_{n+1}) - \text{var}(Y_n) = \text{var}(Y_n + L_{n+1}) - \text{var}(Y_n) = \text{var}(L_{n+1}) + 2 \text{cov}(Y_n, L_{n+1}) \quad (2.5)$$



**Figure 2.5. Minimal and maximal UWL values divided by number of characters.**

The existence of an upper bound for the difference of UWL values when increasing the number of characters is due to the bounded variance of the length of words, which is a result of  $Y_n$  and  $L_{n+1}$  not being independent. If they were independent, then the covariance term would be zero, and the variance should grow indefinitely since the first term is positive. The measurements show that negative covariance values asymptotically equalize the variance caused by additional characters for words of length six and more in the sense that their difference has a constant upper bound. If we divide the maximal and minimal UWL values by  $n$ , the ratios converge as  $n$  grows towards the thresholds  $1 + a_u$  and  $1 - a_l$  (Figure 2.5). Our measurements have led to  $a_u \approx 0.05$  and  $a_l \approx 0.12$ , showing an asymmetric distribution of word lengths between the limits.

### 2.5.4 Interval estimation

During the processing stage, given the  $w$  width of a word, the estimator provides an interval using the pre-computed width limit vectors. Upper and lower estimates for the number of characters are defined by the following formulas:

$$\begin{aligned}\tilde{C}_{\min}(x) &= \min \left\{ n \in \mathbb{N}^+ : \Omega_{\max}(n) > \frac{w}{\omega_{\max}(L)} \right\} \\ \tilde{C}_{\max}(x) &= \max \left\{ n \in \mathbb{N}^+ : \Omega_{\min}(n) < \frac{w}{\omega_{\min}(L)} \right\}\end{aligned}\tag{2.6}$$

where  $\omega_{\max}(L)$  and  $\omega_{\min}(L)$  are the pre-calculated upper and lower estimates for the average letter width of line  $L$ , respectively (see Section 2.4).

Due to our statistical approach the obtained width intervals are generally relatively wide, but in many cases incorrect lengths do not generate many word candidates, because they are excluded by illegitimate feature combinations when binding features to letter positions using shape codes (Section 2.8). However, combining the method described above with a segmentation technique that can validate letter segments (see algorithms in [3]), could lead to narrower interval estimates.

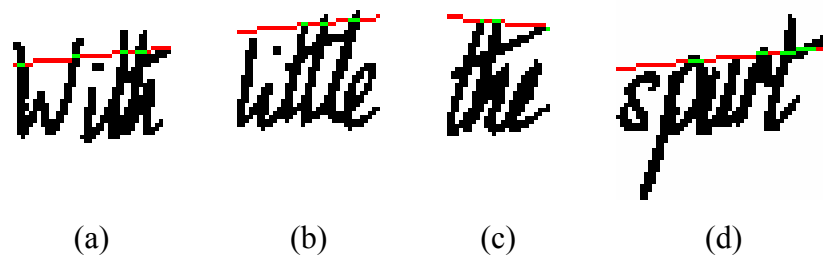
## 2.6 Baseline refinement

The role of this block is to correct potential errors in the upper baseline fitted on the local northern elements of the pseudo convex hull, as already mentioned in Section 2.3. The majority of the errors occur in case of words with a relatively high number of ascenders respected to the width of the word, the computed center pixel of the word will be closer to the top ascender line than the outlier threshold. Some of the uppermost pixels of the ascender(s) do not get classified as outliers and falsely “pull up” the regression line fitted on them (Figure 2.6. (a) – (c)). In these cases either the slope of the baseline becomes too steep, or the baseline gets positioned above or close to the top of the ascenders, disrupting the detection of ascenders.

Since the measured tangent is unreliable in many cases, especially words of length less than five characters, we use a horizontal upper baseline. Note that the word image is already skew-corrected, therefore the lower baseline is horizontal, and the upper baseline should be



close to parallel to it. Although using a horizontal estimate for the upper baseline could theoretically introduce further errors in the detection of ascenders, either false positives or false negatives, our experiments show that false negatives can be prevented by an adequate choice of the horizontal baseline, whereas false positives can be filtered by using a proper ascender height threshold. Figure 2.6 (d) shows an example of an inclined upper baseline replaced by its horizontal estimate. The decreased degree of freedom proved to be an advantage of using a horizontal baseline because its single parameter can be estimated more robustly than two free parameters.



**Figure 2.6. Upper baselines of some words. (a)-(c) show examples when the algorithm based on local northern elements fails to detect the correct baseline. (d) is an example for a word with a highly inclined upper baseline.**

The problem is addressed in two steps. In the first step the upper baseline is compared to the upper baseline of the neighbors and it is corrected if their difference is unacceptably high. The second step serves for locating the optimal ordinate for the baseline adaptively, based on horizontal pixel distribution measures.

At first, in addition to the upper baseline of the word, a common upper baseline is also computed for the word and its direct neighbors from the LNE points of the three words using the method described in Section 2.3. Horizontal baselines are derived from both the word and the common upper baseline by taking the average of the vertical components of the endpoints of the baseline segments. The ordinates of the horizontal upper baselines will be referred to as  $y_u$  and  $y_{u,c}$ , respectively, whereas  $y_l$  denotes the ordinate of the lower baseline of the word.

If the horizontal candidate is above the common upper baseline and the distance between them is higher than 50% of the distance between the common baselines, then the candidate is substituted by the line  $y_u = (3y_{u,c} - y_l) / 2$ . This conservativeness (i.e. not simply using the common baseline) is necessary because the common baseline does not definitely lead to a better result than the computed one, and it might be even below the ideal upper baseline. The significance of this step is to ensure that the upper baseline candidate is not above the mid-

ascender region, which is important for the second step to be effective. The exact value of the threshold used is not important, because adaptive adjustment takes place in the second step.

For this reason we perform a second step, in which the computed baseline is adaptively lowered, aiming to reach the ‘shoulders’ of the word. ‘Shoulders’ are meant to be interpreted on either the horizontal histogram or the horizontal connected components diagram that are combined.

In this step a series of operations are performed. All of these operations represent some kind of knowledge regarding the expected position of the upper baseline (i.e. the properties of the shoulders), thereby ensuring a robust and precise detection. The operations are as follows:

1. If the number of foreground runs that the baseline crosses is larger than the estimated maximum number of characters in the word, then the baseline is moved upwards as long as it meets the threshold.
2. If the maximal number of crosses is larger than 5 (words at least 3 characters long), then if the ratio of the horizontal histogram value the baseline refers to and the peak horizontal histogram value is above  $1/5$ , then the baseline is shifted upwards as long as it meets the threshold.
3. If the number of crossed foreground runs multiplied by 1.5 is less than the estimated lower bound for character count, then the baseline is moved downwards as long as it meets the threshold.
4. The baseline is moved downwards as long as it retains the current number of crossed foreground runs.
5. The final position is determined by minimizing the goal function:

$$y_{u,w} = \arg \min_{y_u < y < y_m} (c(y) H_h(y) - \alpha y) \quad (2.7)$$

where  $y$  is the vertical coordinate increasing downwards,  $c(y)$  refers to the number of foreground runs on the  $y' = y$  horizontal line,  $H_h(y)$  is the horizontal histogram of the word, and  $\alpha$  is a parameter whose value has been determined empirically to be  $\alpha \approx 1.5$  by minimizing the root mean square error of the upper baselines respected to manually drawn baselines on a test set consisting of words from multiple writers.

The number of foreground runs is computed by the **HCCD** template. Note that the first term of the goal function is the product of two different horizontal measures of the word thus trying to establish equilibrium between them and allow for a more robust detection.

## 2.7 Feature extraction

### 2.7.1 *Perceptual features*

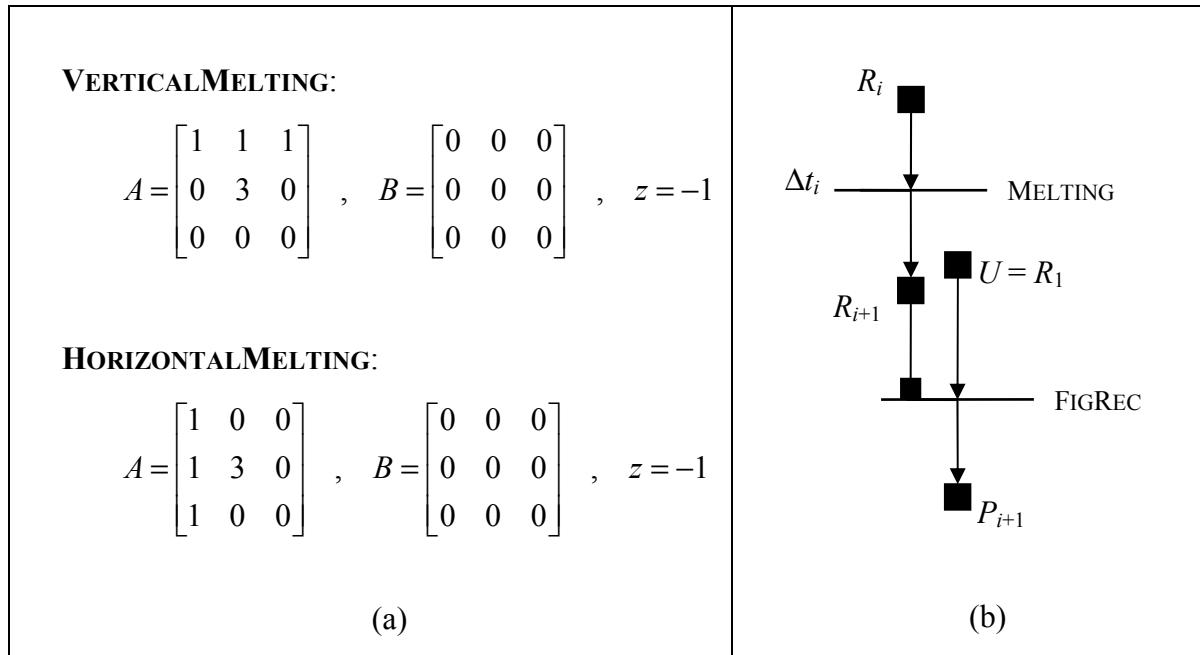
Building on the perception model we assumed that features used by humans during reading must be quite adequate for our purpose. This question is examined in [35] and it is confirmed that features used in human reading are the ones typically used by holistic systems: ascenders, descenders, holes and junction points. It is also concluded that the errors made by such “reading” systems are less counterintuitive to the user and thus they can be more easily corrected.

We have developed algorithms to detect the following perceptual features in word images:

- ascenders, descenders
- holes
- accents
- punctuation marks
- junction points
- hills, valleys

### 2.7.2 *Feature classification*

Features extracted are classified based on size and vertical position and then inserted to the shape code as components (Section 2.8). The **VERTICALMELTING** and **HORIZONTALMELTING** templates are used with different running times to create size classes. These templates erode the objects from the upper and from the left side, respectively. The number of disappearing pixels is roughly proportional to the running time except for spikes at edges that erode faster. This property is advantageous for rough size measurement, because perceptual size is better approximated by volume based measures than the distance between extrema. The running time parameter is scaled by the baseline distance.



**Figure 2.7. (a) Propagating, directed erosion type templates and (b) typical usage shown by a UMF diagram**

Reconstructed patches form a subset hierarchy in the following way. Let  $t_1, t_2, \dots, t_n$  be the required running times of one of the melting-type templates, and let us assume that  $t_i < t_{i+1}$  for all  $1 < i < n$ . The input of each run is the output of the previous run, and the  $i^{\text{th}}$  run should last for  $\Delta t_i = t_{i+1} - t_i$  time. Let us denote the output of the  $i^{\text{th}}$  run by  $R_i$ . During the melting process patches loose size along the melting direction and gradually disappear from the image. Therefore as  $i$  grows, the number of patches in  $R_i$  monotonically decreases. From each  $R_i$  a feature image  $P_i$  is reconstructed, corresponding to the  $i^{\text{th}}$  size class, using the **FIGREC** template (See Figure 2.7). The **FIGREC** template preserves the number of patches on the image, but restores their original sizes, hence as  $i$  grows, the minimum size of patches in  $P_i$  monotonically increases, and patches in  $P_i$  are also present in  $P_j$  for all  $i > j$ :

$$P_n \subseteq P_{n-1} \subseteq \dots \subseteq P_2 \subseteq P_1 \quad (2.8)$$

Therefore to obtain disjoint classes we calculate the pairwise differences  $P'_i = P_i \setminus P_{i-1}$  for all  $i > 1$  with the **LOGDIF** template. Size classification is used for all feature types, but most extensively for holes (see Figure 2.11).

Classification by vertical position is carried out in the following way: Upper and lower baselines are shadowed both up and down, using the **SHADOWDOWN** and **SHADOWUP**

templates respectively, masking necessary or unnecessary regions in the image. Target patches are obtained by means of the **LOGDIF** and the **AND** templates respectively, and reconstructed with the **FIGREC** template. This classification scheme is used for holes and accents-punctuation marks.

Great caution is needed after the classification phase due to the variance of different handwriting styles and noise present in the image: detection algorithms may miss distorted features and may detect false features due to irregular letter shapes. To take this into account we calculate the robustness of each detected feature and use only the robust ones in the first iteration of shape code generation. The characters these features belong to have been referred to as key characters in the literature [24], [40].

I have considered implementing a probabilistic model by calculating a confidence value for each feature, but the robustness measures I used did not prove to be reliable above a certain level of distortion. Therefore less reliable features are used later for inverse filtering, together with global features (Section 2.9.3).

### 2.7.3 *Ascenders and descenders*

*Ascenders* are large vertical strokes extending well above the corpus, or ‘x’ size, whereas *descenders* are large vertical strokes extending well below the lower baseline of handwriting [35]. Therefore precise ascender and descender detection heavily depends on the accuracy of baselines.

By shadowing the upper baseline upwards and the lower baseline downwards we create detection masks. Propagating waves initiated from the baselines are used for shadowing, generated by the **SHADOWUP** and **SHADOWDOWN** templates. Patches that fall under the mask range will appear on the ascender and descender feature candidate images. Feature candidates are filtered based on height using parameters determined in Section 2.4.



**Figure 2.8.** *Feature maps showing different types of ascenders and descenders. Red and purple refer to normal and narrow ascenders, blue and green refer to narrow and wide descenders, respectively.*

Ascenders and descenders are classified based on their widths into three and two classes, respectively. The most important factor in ascender and descender robustness is the height of the feature. Additionally, collocation with upper and/or lower holes (Section 2.7.4) increases feature robustness.

### 2.7.4 Holes

*Holes* (also called loops) are parts of the background encircled by the script in the foreground. These isolated parts can be located with a single CNN instruction. The **HOLE-FILLING** template is a standard template included in the Cellular Wave Computing Library [12], but it detects holes with an outer boundary that is continuous in an 8-connected manner, because it only explores the 4-connected background. As a result those parts of the background that are only connected to it in an 8-connected manner are not considered to be connected to the background, and hence they are also detected as holes. Figure 2.10 illustrates the problem arising from this behavior: down-sampling of the input image can result in narrow spaces between letters becoming only a single pixel wide, hence the **HOLE-FILLING** template detects invalid holes in the word image.

I constructed a new template called **HOLE-FILLING4** that detects only holes having a 4-connected outer boundary, thus overcoming the above problem (Figure 2.9). This has been achieved by adding non-zero values to all 8 surrounding elements of the feedback template. Similarly to **HOLE-FILLING**, the **HOLE-FILLING** template should be used with a homogeneous black initial state and a gray boundary condition. The gray boundary condition initiates a white wave front that propagates through the whole image in an 8-connected manner, crossing the barriers where neighboring foreground strokes have pixels being 8-connected, but not 4-connected neighbors, and whitening all background pixels not belonging to a hole. An earlier, slower solution for this problem [9] led to the same result by more template instructions. The foreground pixels are removed from the detected holes with the **LOGDIF** template.

Holes detected in the word image can be classified into classes by size and location. To remove small patches due to aliasing, **SMALLKILLER** template is run prior to classification.

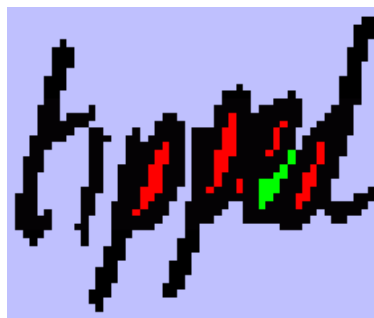
**HOLE-FILLING4:**

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 3 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad z = -1.5$$

**HOLE-FILLING:**

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad z = -1$$

*Figure 2.9. HOLE-FILLING4 template fills holes with 4-connected outer boundaries compared to HOLE-FILLING that finds holes with 8-connected outer boundaries.*



*Figure 2.10. Comparison between the results of HOLE-FILLING and HOLE-FILLING4 templates. Detection wave is shown in light-blue, red patches are valid holes detected by both templates, and the green patch is an invalid hole detected only by the HOLE-FILLING template.*

Holes are first classified based on their vertical location into upper and lower and middle holes. The middle hole class is not final; patches in it are further classified by height and width. Patches disappearing in the first run of using the **VERTICALMELTING** operator belong to the small hole class, whereas the remaining ones become big hole candidates. Patches that disappear on a consecutive **HORIZONTALMELTING** are considered high holes, whereas fat holes refer to those that do not totally disappear after another **VERTICALMELTING**. Remaining big hole candidates constitute the big hole class.

Robustness of holes mostly depends on their size and collocation with other holes. Small holes that are only separated from other small holes by a single stroke are not considered to be robust.



*Figure 2.11. Classification of holes. Red refers to big holes, green to small holes, yellow to fat holes, cyan to upper holes and magenta to lower holes.*

### 2.7.5 Accents and punctuation marks

We use these feature names in their common sense, except that apostrophes, and acute and grave type accents will all be referred to as primes. These features are usually quite distinct on the word image, thus making their detection straightforward. The algorithm supposes that accents and punctuation marks do not touch the text, i.e. they are not connected to the “main” part of the writing. The center line (mean of upper and lower baselines) and the **FIGREC** template is used to identify the pixels belonging to the main image, and applying the **LOGDIF** template to result image we can subtract it from the original word image, leaving the accents and punctuations marks only.



*Figure 2.12. Accents and punctuation marks*

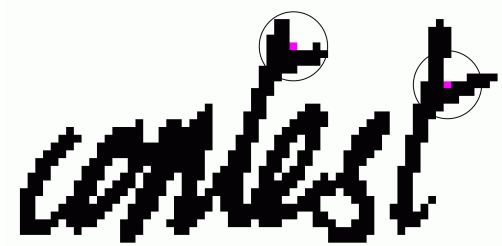
Accents are distinguished from punctuation marks by position, and then both features are classified based on vertical expansion into two classes: dots, primes, and periods, commas. The expansion is measured by the **VERTICALMELTING** template.

### 2.7.6 Junction points

A junction point is a point on the word image where more than two strokes meet. This includes crossing of strokes like in letter ‘t’ or ‘x’ and meeting strokes like in letter ‘d’, ‘g’ and ‘p’. The **JUNCTION** template in the template library is an obvious choice to detect such



points, but with the limited neighborhood available on present hardware robust detection is not possible.



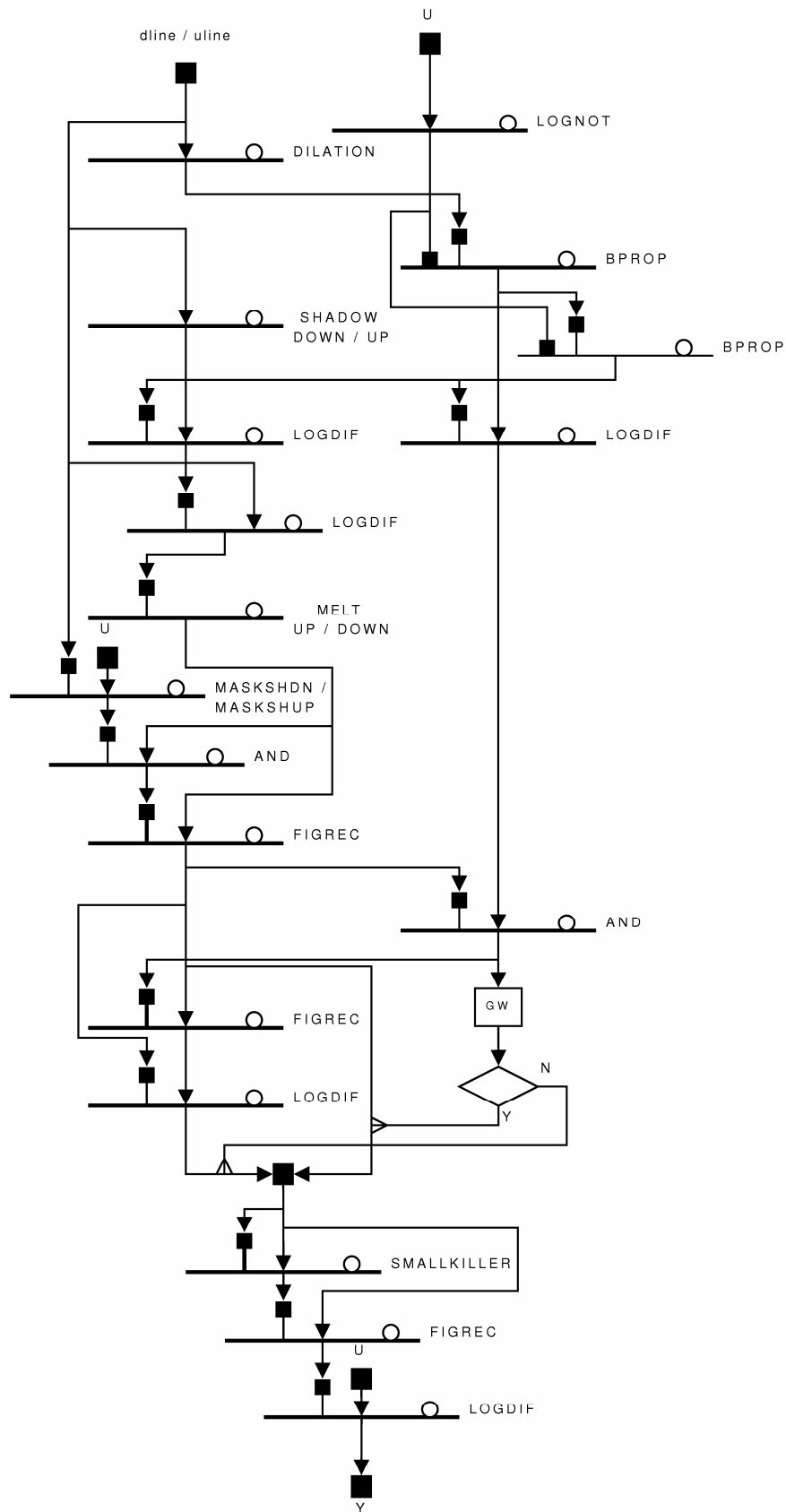
*Figure 2.13. Junction points*

Therefore we use another approach by concentrating on the ascender region of the image and applying the `CONCAVEARCFILLER45` template that detects concave arcs. Our preliminary intent was to detect letter ‘t’, but experiments showed that the feature class detected by the template may appear in other letter instances as well, which contain concave arcs in the ascender area, including ‘B’, ‘M’ and ‘R’. This poses no problem, since all these letters have been also included in the default feature mapping for the junction feature. Due to the fact that these capitals occur much less frequently than letter ‘t’ (especially when considering non-initial letters of the word), the effectiveness of the original idea is preserved.

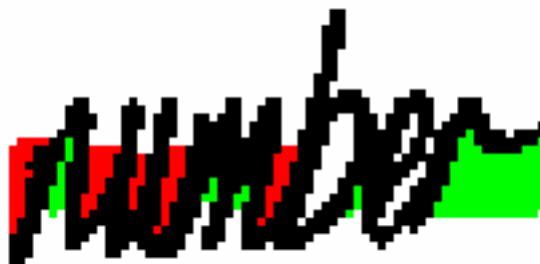
### **2.7.7 Hills and valleys**

Letters containing holes, ascenders and descenders are relatively easily identifiable. But quite many words do not contain any of these features. Typical letters containing hills are ‘m’ and ‘n’ whereas ‘u’, ‘v’, and ‘w’ are ones that contain valleys. Since many hills and valleys appear in inter-letter positions they are not robust enough to be included in shape code descriptions, they are used in inverse filtering instead (Section 2.9.3).

Detection of hills and valleys is quite complex respected to the algorithms discussed in previous subsections. Its UMF diagram is shown in Figure 2.14. Upper and lower baselines are dilated and propagating waves are initiated from them on the background of the word image in the region between the baselines. Propagation time is proportional to the distance of the baselines. Small patches are removed as noise, as well as patches belonging to transients that did not settle.



**Figure 2.14.** UMF diagram of hill and valley detection. Inputs labeled *uline* and *dline* refer to upper and lower baselines of the word, respectively.



*Figure 2.15. Hills and valleys shown in the word image of ‘number’. Note that not only in-letter hills and valleys are detected, but intra-letter ones too, like the valley between letter ‘m’ and ‘b’ and the hill between letters ‘b’ and ‘e’.*

## 2.8 Shape codes

### 2.8.1 Representation of shape codes

The interface between the perception and the linguistic system aims to realize a language reduction system. This means that shape codes serve for constructing a filter for the linguistic system. The main difference between our system and previously published lexicon reduction systems (e.g. [40]) is that we use a language model, not a fixed lexicon. The role of shape coding is to combine geometric information of features with their position information.

Shape codes have two different representations in our system: *Abstract Shape Descriptors (ASD)* and *extended regular expressions (ERE)*. An ASD consists of the features instances detected and the corresponding horizontal locations, given as a ratio of the image width. In ASDs the number of letters of the word is not specified, and therefore they contain no information on which character do the individual features belong to and neither on whether two neighboring features are part of the same letter or of different ones. EREs are regular expressions, but macros are allowed in them that refer to character sets, this way they allow for a much more compact description. The character sets used typically share a common feature; therefore macros are named after the corresponding feature. In EREs the possible number of letters is already determined, and features are assigned to character positions. An ERE, however, can contain multiple combinations of assignments, and these may refer to different character numbers.

### 2.8.2 Abstract shape descriptors

To calculate the horizontal position of the features at first we project them vertically onto the horizontal axis using the **SHADOWDOWN** template and find the minimal and maximal

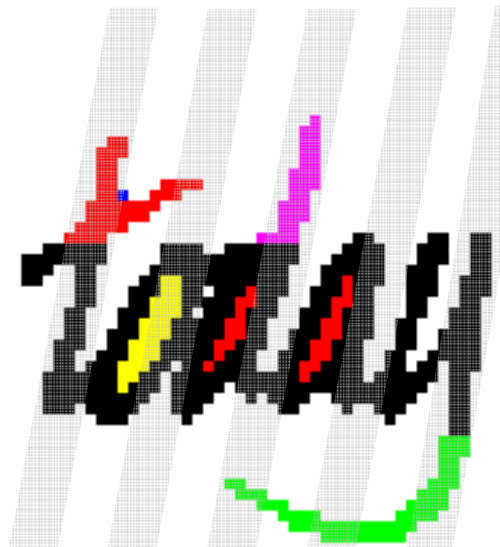
abscissas  $x_{\min}$  and  $x_{\max}$  on the projected line. The horizontal position  $p$  is a normalized value that is calculated by the following equation:

$$p = \alpha \frac{x_{\min}}{w} + (1 - \alpha) \frac{x_{\max}}{w} - \beta \Delta \tan \varphi \quad (2.9)$$

where  $w$  is width of the word image,  $\Delta$  is the distance between the upper and lower baseline,  $\varphi$  is the estimated slant respected to the vertical axis and  $\alpha$  and  $\beta$  are parameters depending on the type of the feature defined in Table II. The homotopy parameter  $\alpha$  weights the abscissas of left and right edges of the feature patch, whereas the term containing  $\beta$  accounts for the slant of the word and  $\beta$  itself refers to the vertical distance of a feature type from the line at the bottom of descenders (horizontal line no. 4 in Figure 2.4 (a)). Table III shows the ASD of the word in Figure 2.16.

**Table II. Parameters for horizontal position calculation of features.**

Feature type	$\alpha$	$\beta$
Ascenders, accents	0.9	2.5
Descenders, punctuation	0.1	0.5
Other	0.5	1.5



**Figure 2.16. Feature map of the word “today”. Stripes refer to character positions for 5 letters. Gray stripes refer to integer positions, white stripes refer to intra-letter positions.**

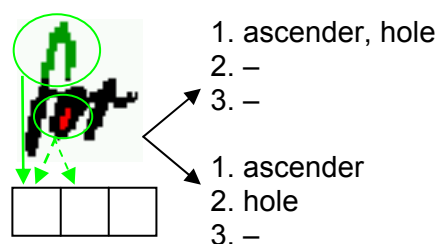
**Table III.** *ASD of the word is given in the first two columns, the third one gives the character position the features in the ASD belong to.*

Feature name	Horizontal position	Estimated character position
Ascender	5.6%	1
Junction	8.3%	1
Fat hole	20.7%	1.5
Big hole	39.3%	2.5
Narrow ascender	46.3%	3
Big hole	61.2%	3.5
Wide descender	90.2%	5

### 2.8.3 Conversion to regular expression

To be able to convert an ASD into a regular expression we have to assign the features to letters, more precisely character positions. Character positions can be thought of as empty boxes and each feature is placed into one of these boxes. Each box (character position) refers to one letter and the features in it are part of this letter.

The ambiguity in this task is twofold: on one hand the number of letters is not known exactly, we only have an estimated interval for it; on the other hand the width of the letters varies considerably, therefore even if we assume the number of letters to be fixed, a given horizontal coordinate can refer to different character positions depending on the type and distribution of letters in the word.



**Figure 2.17.** *Mapping of topographic features is ambiguous, the hole in letter ‘o’ can belong to two distinct positions, doubling the number of possible mappings.*

To overcome this problem calculation of fuzzy functions has been proposed in [8] to match the features at the presumed positions against all words in a small sized dictionary. Since the matching function differs for every word this method would not perform well in case of large dictionary tasks, which is especially true in case of our general approach. Therefore we use a simpler assignment that still conserves the vagueness of the position.

Our basic assumption is that characters are approximately evenly distributed in width. Based on this we divide the image into vertical blocks of equal widths. This process is performed separately for every word length in the estimated interval, i.e. for every  $\hat{C}_{\min} \leq n \leq \hat{C}_{\max}$  (see Section 2.5.4). To handle variance of letter widths we use overlapping blocks that refer to character positions, so that the width of blocks at intermediate positions is  $3/(2n-1)$ , whereas the leftmost and the rightmost blocks are of  $2/(2n-1)$  width, both values normalized with the total width of the word image. Let us denote the horizontal intervals belonging to the blocks by  $B(i,n) = [b_{\text{left}}(i,n), b_{\text{right}}(i,n))$  where  $1 \leq i \leq n$ . Boundaries of the blocks are defined as

$$\begin{aligned} b_{\text{left}}(i,n) &= \begin{cases} 0, & \text{if } i=1 \\ \frac{2i-3}{2n-1}, & \text{if } 2 \leq i \leq n \end{cases} \\ b_{\text{right}}(i,n) &= \begin{cases} \frac{2i}{2n-1}, & \text{if } 1 \leq i \leq n-1 \\ 1, & \text{if } i=n \end{cases} \end{aligned} \quad (2.10)$$

This partitioning results in a horizontal sequence of disjoint rectangles of the same width, every other being the intersection of two neighboring blocks defined above. When mapping features to character positions these  $2n-1$  disjoint blocks of normalized width of  $1/(2n-1)$  are used. For every  $1 \leq j \leq 2n-1$  the corresponding intervals are:

$$\tilde{B}(j,n) = \left[ \frac{j-1}{2n-1}, \frac{j}{2n-1} \right) \quad (2.11)$$

We will refer to these intervals as estimated character positions. Estimated character positions are numbered from 1 to  $n$  with a step of 0.5. Thus the interval belonging to character position  $i$  for estimated word length  $n$  is  $\tilde{B}(2i-1, n)$ . Estimated character positions integer numbers refer to a single final character position, whereas the rest of them refer to two final character positions, namely their neighbors.

A feature belongs to the character position  $i$  for an estimated word length  $n$  if its horizontal position falls into the interval  $\tilde{B}(2i-1, n)$ . Let us denote the set of features belonging to these intervals by  $F^n(j)$ . To resolve the ambiguity in this description, features belonging to intra

letter character positions (those that are in sets of even indices) need to be moved to neighboring positions in every possible combination. Let  $m(n)$  be the number of features in these sets. Then there are  $2^{m(n)}$  distinct descriptions generated for word length  $n$ . Let us denote the set of the  $k^{\text{th}}$  description at character position  $j$  for word length  $n$  by  $F_k^n(j)$ . The algorithm to generate these sets is as follows:

```

Cycle n :=  $\hat{C}_{\min}$  to  $\hat{C}_{\max}$  // word lengths
  d := 1; //description index
  Cycle j := 1 to 2n-1 // character positions
    If j mod 2 = 0 then
      Cycle f through all the features in  $F_1^n(j)$ 
        Cycle k := 1 to d // descriptions
          Create a new set series  $F_{d+k}^n$ ;
          Cycle i := 1 to 2n-1 // character positions
            Copy all features from  $F_k^n(j)$  to  $F_{d+k}^n(j)$ ;
          End cycle
          Move feature f from  $F_k^n(j)$  to  $F_k^n(j-1)$ ;
          Move feature f from  $F_{d+k}^n(j)$  to  $F_{d+k}^n(j+1)$ ;
          d := 2*d;
        End cycle
      End cycle
    End if
  End cycle
End cycle

```

After this algorithm is run, all  $F_k^n$  features set series refer to the features of the input word and have all features mapped to integer estimated character positions, that is sets  $F_k^n(j)$  are empty for every even  $j$ . At this point a check is all  $F_k^n$  features set series are validated if the features in them have preserved their relative positions after the disambiguation algorithm. Those feature set series that have features in a wrong horizontal order are dropped.

For every  $1 \leq i \leq n$  feature sets  $F_k^n(2i-1)$  are converted to new feature sets  $\tilde{F}_k^n(i)$  whose index refer to final character positions (integers only) and whose elements are not feature instances, but feature types. If there is only one feature of a type in a given  $F_k^n(2i-1)$ , then it

is simply copied to  $\tilde{F}_k^n(i)$ . If there are more, then they are converted to a multi-feature. Theoretically every multiplicity of every feature constitutes a multi-feature, but very few of them occur in practice. In our experiments it happened in only a single case, when two ascenders were mapped to one character position.

Extended regular expressions are generated from descriptions  $\tilde{F}_k^n(i)$  by the following steps:

1. At each character position  $i$ , in all descriptions  $k$ , and for all word lengths  $n$  in the estimated interval if at least one feature is mapped to position  $i$ , then take the conjunction of macros referring to multi-features in  $\tilde{F}_k^n(i)$ , otherwise, if no features are mapped here, then mark it as an empty set:

$$\varphi_k^n(i) \triangleq \begin{cases} \bigwedge_{f_i \in \tilde{F}_k^n(i)} M(f_i), & \text{if } |\tilde{F}_k^n(i)| > 0 \\ M(\emptyset) & , \text{if } |\tilde{F}_k^n(i)| = 0 \end{cases} \quad (2.12)$$

where  $M(f)$  is the macro name of feature  $f$ , defined in Table V.

2. Concatenate the conjuncts  $\varphi_k^n(j)$  of character positions 1 to  $n$  one after another in all descriptions  $k$  for all word lengths  $n$  in the estimated interval:

$$\varphi_k^n \triangleq \varphi_k^n(1) \circ \varphi_k^n(2) \circ \dots \circ \varphi_k^n(n) \quad (2.13)$$

3. Take the disjunction of the expressions  $\varphi_k^n$  over all  $2^{m(n)}$  descriptions for word length  $n$ :

$$\varphi^n \triangleq \bigvee_{k=1}^{2^{m(n)}} \varphi_k^n \quad (2.14)$$

4. Take the disjunction of the expressions  $\varphi^n$  over all estimated word lengths:

$$\varphi \triangleq \bigvee_{n=\tilde{C}_{\min}}^{\tilde{C}_{\max}} \varphi^n \quad (2.15)$$



The ERE of the previous example is shown in Table IV. Regular expressions are obtained from the EREs by substituting the character sets into feature macros according to their definitions. We have defined a mapping (Table V) from the recognized features to all orthographic characters or character sequences that contain it. The latter case occurs rarely, only if two narrow and neighboring characters are merged and share all their features (e.g. ‘ff’). The basis of the mapping was created manually based on expected letter shapes and feature definitions and tuned based on results on the training set.

**Table IV. Sample extended regular expression for the word shown in Figure 2.16.**

```
<fh>&<j>&<da><bh><bh>&<na><@><wd> |
  <fh>&<j>&<as><bh>&<as><bh>&<na><@><wd> |
  <fh>&<j>&<da><bh><na><bh><wd> |
  <fh>&<j>&<as><bh>&<a><na><bh><wd> |
  <fh>&<j>&<da><@><bh>&<na><bh><wd> |
  <j>&<da><fh><bh>&<na><bh><wd> |
  <fh>&<j>&<a><a><bh>&<na><bh><wd> |
  <j>&<a><fh>&<a><bh>&<na><bh><wd>
```

**Table V. Default feature macro mappings.**

Feature name	Macro name	Definition
ascender	<a>	A B C D G I J L O P Q R S f ff h k l t
narrow ascender	<na>	I L b d h k l t
wide ascender	<wa>	B E F I M T Z ff k t
double ascender	<da>	H K M N U V W X Y t T
wide descender	<wd>	f ff g j y
narrow descender	<nd>	f p q
small hole	<sh>	a e k o s
high hole	<hh>	a d e g o
big hole	<bh>	a b d e g k o p q s t
fat hole	<fh>	B a b d g o p q t
upper hole	<uh>	f ff h k l A B D O P Q R
lower hole	<lh>	f ff g j y
dot	<d>	i j
prime	<p>	'
comma	<c>	,
period	<.>	. ? !
junction	<j>	t B M R
no feature	<@>	c i m n r s u v w x z e

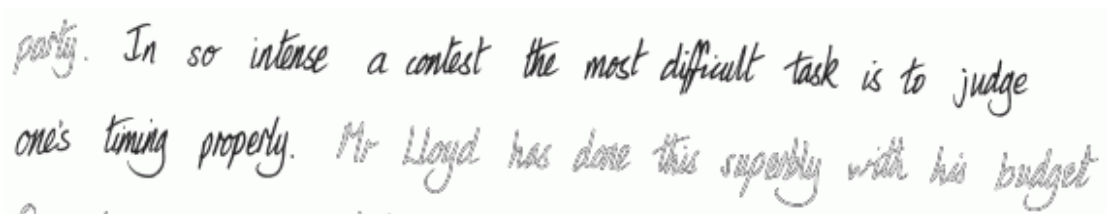
## 2.9 Generating and inverse filtering word lists

### 2.9.1 Morpho-lexical word list generation

The regular expression generated for every word summarizes the extracted shape information, and can be used as a mask by the morphologic word generator to provide all possible words the given language model can generate that match the criteria.

We use a linguistic framework described in [48]. The regular expression is converted and stored as a trie. The linguistic framework then searches for valid orthographic sequences using built-in orthographic lexicons. This search can be performed very efficiently; the framework operates in linear time in function of the length of the input regular expression.

An example for a sample sentence and the returned word lists are shown in Figure 2.18.



**Figure 2.18.** Two sample rows of handwriting, containing a whole sentence

**Table VI.** Top ten word candidates returned for the sentence according to their individual frequencies. Correct words are shown in inverse.

In	was.	mistake	a	market	the	most	difficult	take	is	to	judge	and	's	Mining	property
Is	can.	sixteen	on	market?	of	must	definite	told	di	Be		are	'm	timing	property
Be	so	intense	Be	contact	to	want	deficits	talk	jo	Re		but	'e	Mixing	opaquely
Its	no	sisters	be	context	Is	went		task	pi	MA		one	'i	tiring	puppetry
Me	up	sixties	as	content	Be	next		tend	si	Bo		two	'n	Miming	
Mr	its	intake	or	defeat	be	sort		tank	ti	Rd		out	'c	tieing	
Few.	now.	intends	an	ticket	he	west		tied	id	td		did		Miring	
Men.	man.	virtues	so	contest	As	cost		tale	iq	Bd		see			
Mm	end.	winters	no	outset	as	east		tube	bi			get			
Ten	ca	integer	up	bucket	at	rest		tide	gi			put			

### 2.9.2 Word length filtering

As mentioned in Section 2.5.1, variance in width of letters due to their shape cannot be taken into account in word length estimation. But at this point when word candidates are

already generated we can use character level information by calculating an expected normalized word width for each candidate, and comparing it to the actual word image width.

Expected normalized word width is expressed in unit character width by summing the expected normalized widths for each character of the candidate. The unit character width is theoretical measure and it refers to the standard width of letter ‘o’. The normalized width of a character is its width in pixels divided by this ideal width.

Since individual styles may exhibit different relative letter sizes, thus estimating the expected pixel widths of letters or bigrams would require a style-specific character segmented training set with the corresponding characters. It is generally not feasible to expect such training even for non auto-adapting systems. Expected pixel widths could be also estimated by probabilistic optimization if we treat individual letter widths as random variables and estimate their distribution functions. This method would still require style-specific training that we want to avoid. Therefore we use a simpler approach with three width classes and hard-coded letter widths based on their ideal shape. The unit width (1.0) is assigned to most lower case letters and 1.5 to most capitals, with a few exceptions shown in Table VII.

**Table VII. Assigned width values in computation of the expected word width**

Letters		Expected normalized width [unit character width]
<i>i, j, l, t</i>		0.5
<i>M, w</i>		1.5
<i>T, I</i>		1
Others	Lower case	1
	Capitals	1.5

Not being an integer, the calculated *expected normalized word width*  $\Theta$  has a better resolution than the number of characters, and thus a higher correlation with the expected pixel width. Therefore new normalized UWL (NUWL) vectors are needed to tighten the estimated interval width, and to realize the filtering of the candidate list. The NUWL vectors were determined in a similar way as in Section 2.5.4 on the same training set, and they are shown in Table VIII. We use the method described there to determine the pixel width limits for the candidate.

**Table VIII. Minimal and maximal Universal Width Limit vectors referring to average letter width multipliers for in unit character width and their difference**

$\Theta$	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7
$\Omega'_{\min}$	0.3	0.6	0.9	1.5	2	2.2	2.4	2.7	3.5	4	4.5	5.3	5.8
$\Omega'_{\max}$	1.1	1.7	2.3	3	3.6	4.1	4.8	5.4	6.1	6.8	7.7	8	8.2
$\Omega'_{\max} - \Omega'_{\min}$	0.8	1.1	1.4	1.5	1.6	1.9	2.4	2.7	2.6	2.8	3.2	2.7	2.4

$\Theta$	7.5	8	8.5	9	9.5	10	10.5	11	11.5	12	12.5	13	13.5
$\Omega'_{\min}$	6.2	6.7	7.1	7.7	8.2	8.7	9.3	9.8	10.4	10.9	11.3	11.6	12
$\Omega'_{\max}$	8.4	9.3	9.8	10.5	11.1	11.5	11.9	12.5	13	13.4	13.7	14.1	14.5
$\Omega'_{\max} - \Omega'_{\min}$	2.3	2.6	2.7	2.8	2.9	2.8	2.6	2.7	2.6	2.5	2.4	2.5	2.5

Let  $\Omega'_{\min}$  and  $\Omega'_{\max}$  be the NUWL values corresponding to  $\Theta$ . For the pixel width of the word to be validated, it has to meet the following condition:

$$\Omega'_{\min}(\Theta)\omega_{\min}(L) < w < \Omega'_{\max}(\Theta)\omega_{\max}(L) \quad (2.16)$$

where  $w$  is the word width in pixels,  $\omega_{\max}(L)$  and  $\omega_{\min}(L)$  are the pre-calculated upper and lower estimates for the average letter width of line  $L$ , respectively (see Section 2.4). Table IX shows the filtered output for the word lists in Table VI.

**Table IX. Top ten word candidates after word length filtering. Correct words are shown in inverse.**

In	so	mistake	a	market	The	most	difficult	take	is	to	judge	and	's	Mining	property
Is	no	sixteen	b	contact	Of	must	definite	told	di	td		are	'm	timing	properly
Be	up	intense	p	context	Is	want	deficits	talk	jo			but	'e	Mixing	opaquely
its	its	sisters	d	weight?	Be	went		task	pi			one	'i	tiring	puppetry
Me	ca	sixties	tv	content	Be	next		tend	si			two	'n	Miring	
Mr	wo	intake	o	defeat	He	sort		tank	ti			out	'c		
Mm	etc	intends	g	contest	As	west		tied	id			did			
Ten	via	virtues	di	outset	As	cost		tale	iq			see			
ten	co	winters	pi	bucket	Do	east		tube	bi			get			
Tv	rid	integer	ti	naught?	Do	rest		tide	gi			put			

### ***2.9.3 Filtering based on less reliable features***

Hills, valleys, and the features that are not used in the morpho-lexical matching, are used to filter the output word list in a second run. Since the number of words to be checked is much smaller respected to the first run (we have actually obtained a small size dictionary), we can match these features against word candidates.

For every feature considered an estimated character position is assigned by the method described in Section 2.8.2 and 2.8.3, and then we try to match them against generated feature descriptions for each remaining candidate. Feature descriptions are derived from word candidates based on the feature mapping table. In case of hills and valleys separate tables are used for single letters and for those letter pairs, for which the intra-letter space establishes a hill or a valley. Even if not all features in the description generated are detected, all features have to match the feature description at correct positions. Therefore we look for a matching for every feature, and if no match can be found, then the candidate is eliminated.

## **2.10 Statistical context selection**

### ***2.10.1 Word graph construction***

In this system we have chosen a simple model to demonstrate the interaction between the recognition-associative and the linguistic system. We integrate the contextual knowledge by means of a statistical approach: we try to order these lists and choose the correct word according the relative frequencies of all bigrams in the sentence. Bigram frequencies are gained from the BNC (British National Corpus) [52], which is a general topic corpus. These relative frequencies can serve as an estimate for the probability of the occurrence of the bigrams in the given text. A more sophisticated post-processing filter is under development by applying the HumorESK parser module [49].

After word candidate lists are generated for every word image being processed, we look for relative frequencies of bigrams formed by all candidates for all the word in inner positions (neither first, nor last) and word candidates of the previous and the next positions. Once bigram frequencies are obtained for both neighbors, those with a zero frequency are dropped.

The bigram frequencies are used to generate a word graph (a directed acyclic graph) from the remaining word lists. In this graph each level refers to a position in the sentence, and nodes on that level are the candidates for the given position. If candidates have confidence

values they can be used as weights for nodes, whereas edges are assigned weights of the relative frequencies the nodes they connect.

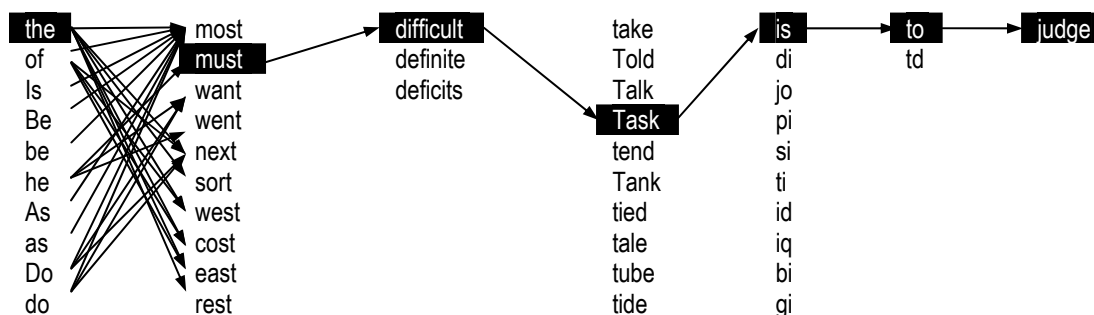
### 2.10.2 Candidate filtering

Due to the bigram model and the technique we use word graphs are generally dense enough in the following sense: a node (word candidate) on level  $i$  having at least one outbound edge to level  $i+1$  and another node on level  $i+2$  with at least one inbound edge from level  $i+1$  are connected (through a path of length two). As a consequence we can use a simple filtering algorithm before to radically decrease the nodes in the graph, which reflects the local influence area property of natural languages. Obviously natural languages cannot be generated by a regular grammar, but generally there is a remarkable part of the grammar that reflects local generation of words, e.g. expressions, phrasal verbs, etc.

Considering a position  $i$  in a sentence we will refer to the ordered pair of candidates of position  $(i-1$  and  $i)$  as left bigram, supposing  $i$  is not the first position, and of position  $(i$  and  $i+1)$  as right bigram, supposing  $i$  is not the last position, if the nodes referring to the candidates are connected by an edge. In the list of left bigrams a pair will be validated if its second word can be found as a first word of at least one bigram in the right list, and vice versa. These are called matching bigrams. Following the validation we filter out invalid entries from all bigram lists and the corresponding nodes from the graph. Remaining entries will provide us with possible sentence candidates.

### 2.10.3 Choosing the optimal path

The longest path in the graph gives the most probable sentence and the  $n$  longest paths correspond to the  $n$  most probable sentences. These paths can be selected using the Viterbi algorithm. A sample graph is shown in Figure 2.19.



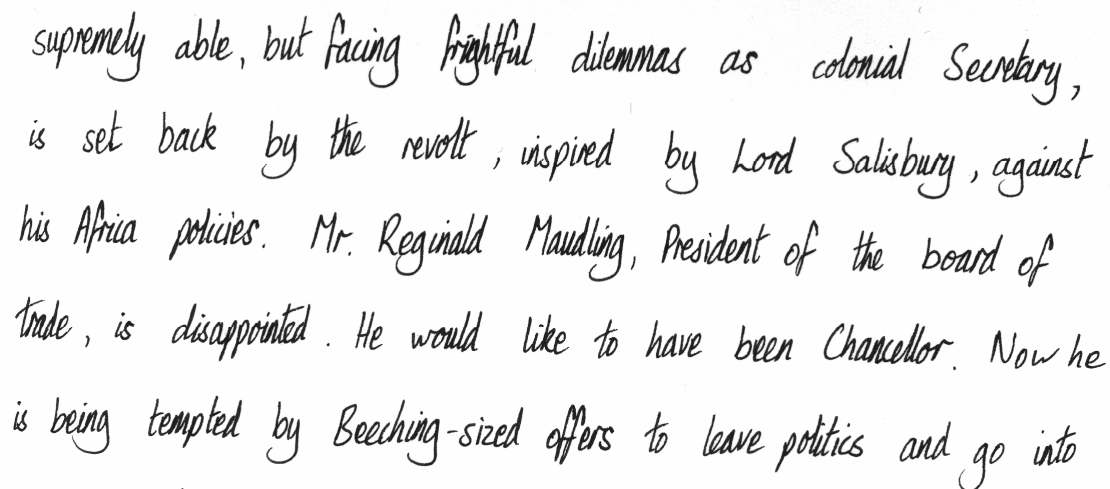
**Figure 2.19.** Word graph for a part (words 7-13) of the sentence in Figure 2.18. In this case there is only one path through this sub-graph and it gives the correct sentence.

## 2.11 Results and Evaluation

### 2.11.1 Databases used

The system was tested on a database collected by Senior and Robinson [36]. The database contains 25 handwritten pages and 7000 words from the LOB corpus (Lancaster-Oslo / Bergen) that were written by a single writer. Some sample lines are shown in Figure 2.20.

I have also created a small test set to enable testing adaptation to writing style parameters. This consists of five handwritten pages, each written by a different person.

A photograph of a handwritten document in cursive script. The text is written in dark ink on a light-colored background. The handwriting is fluid and somewhat slanted. The text describes a colonial secretary facing a revolt, inspired by Lord Salisbury, and a disappointed Mr. Reginald Maudling who is being tempted by Beeching-sized offers to leave politics.

*Figure 2.20. Some sample lines from the LOB corpus*

### 2.11.2 Experimental results

To evaluate the results we use two concepts. The *coverage* of the lexicon reduction system refers to the capacity of the reduced lexicon to include the right answer [28]. By *reduction rate* we mean the reduction of the lexicon size, i.e. the number of eliminated word forms divided by all word forms in the lexicon. In our system the size of the lexicon cannot be exactly determined, since the linguistic system can generate many forms from the entries in its lexicon. We estimated the number of included word forms to be over 200 000 by parsing the word list of the BNC [52] with the linguistic system and counting the correctly recognized forms.

In the literature reduction rate usually refers to the average reduction rate, the minimal (worst case) reduction rate is hardly ever reported. This might be due to the fact that it is impossible to give a theoretical minimum for the reduction rate, but the minimum over the test set can be easily determined.

To generate a test set from the corpus on which word level tests can be performed the position of words in the scanned images need to be assigned. This process should either be done manually, or if automated, then the results must be verified by a human. This requirement makes the preparation of tests time consuming. Lexicon reduction systems are typically tested on several hundred word images in the literature ([28],[30],[31],[40]), which is accepted to give a reliable measure of system performance.

Our word filtering algorithm was tested on 400 English words, and in 80.5% of the cases the correct word was returned in the output list. The average length of the output list is 115, meaning an average reduction rate over 99.95%. The longest word list returned had 2342 words giving a minimal reduction rate of 98.8%. At first glance this coverage ratio might seem too low for a lexicon reduction system, because one would expect that only surely incorrect words should be filtered. But on one hand an analysis of the errors shows that the reasons the problems originate in are not deeply inherent to the architecture and therefore can be improved later, on the other hand for the ultra-high reduction rate of our system this result is still better than the ones published earlier (Table X). Comparing our system to ones mentioned in the literature one can observe that we have achieved a much higher reduction rate than usual, and still maintained better coverage than the system with the highest reduction rate. Even if we approximate our lexicon size to be 20 000, we still get an average reduction rate of 99.45%, which is still above the reference values.

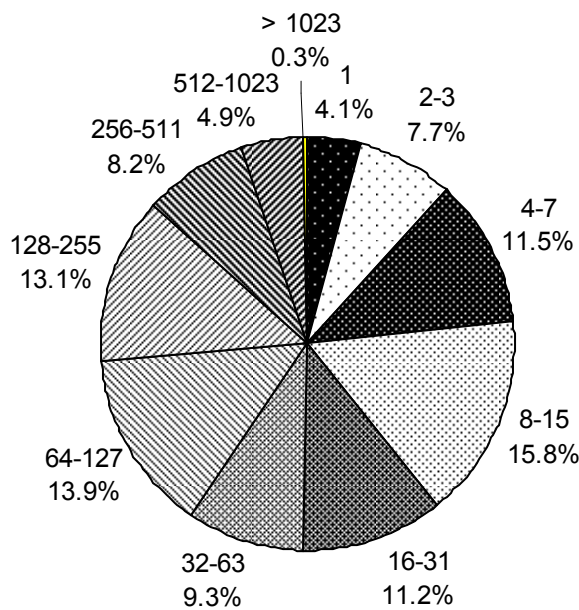
**Table X. Comparison of lexicon reduction methods**

Method	Lexicon size	Test set	Average reduction rate	Coverage
Presented	> 200 000	400	> 99.95%	80.5%
M. Zimmermann et al [40]	20–1000	811	72.9%	98.6%
S. Madhvanath et al [30]	20 000	825	50%	> 98%
S. Madhvanath et al [30]	20 000	825	99%	74%
S. Madhvanath et al [31]	21 000	825	95.2%	95%
S. Madhvanath et al [31]	21 000	825	99%	80%
Guillevic et al [27]	3 000	500	3.5%	95%

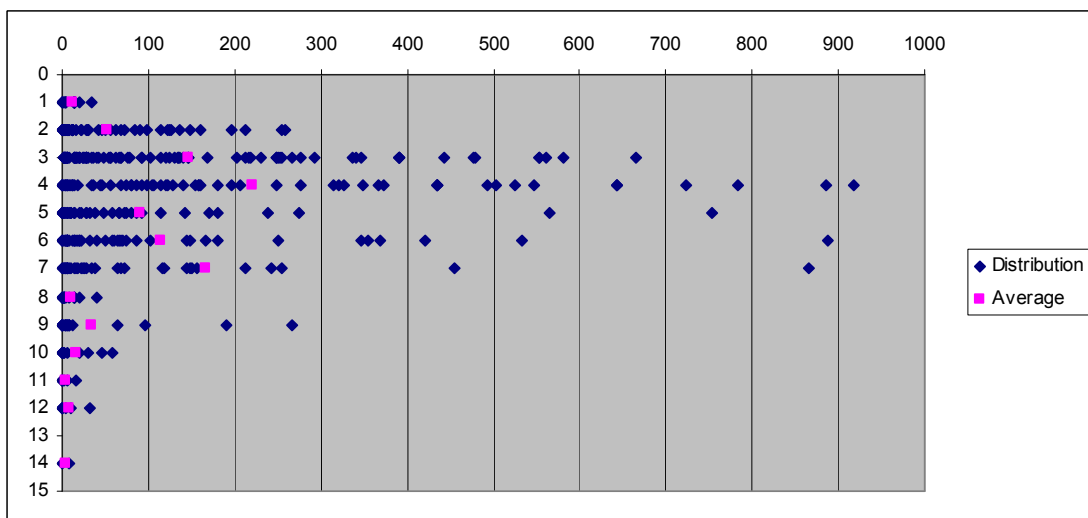
In [40] the same corpus was used for testing, in [30] and [31] the tests were carried out on U.S. city names, whereas Guillevic et al [27] considered only handwritten uppercase words. Comparison of the results would be more reasonable if the evaluation had taken place on the same test set for each of the compared methods, however, there are only a few similar systems developed yet for offline handwriting recognition and in many cases proprietary databases are used for testing.



Figure 2.21 shows the distribution of the length of output word lists after inverse filtering. The importance of inverse filtering is shown by the achieved 30% decrease of the average word list length from 156 to 115. It is also worth to note that in half of the cases the final word list contains less than 30 words. One can observe that the curve has a quasi-exponentially decreasing characteristic. In Figure 2.22 the connection between the length of the input word and the length of output word list is shown. The curve of average word list lengths resembles a Poisson distribution with  $\lambda \approx 4$ .



**Figure 2.21. Classes of lengths of output word-lists.**



**Figure 2.22. Length of the output word-list in function of the number of characters of the word to be recognized.**

### ***2.11.3 Analysis of causes of errors***

Wrong length estimation is the most important cause of error, it accounts for about 30%. Invalid features cause another 30% of the errors, the half of which is due to irregular letter shapes, whereas the other half is due to aliasing on the low resolution binary image. Out of vocabulary words and missing inflection forms from the linguistic system account for about 15% of the errors, whereas wrong baseline calculation and mispositioning of features each cause about 10%.

## **2.12 Conclusions**

I introduced a framework that integrates topographic, linguistic and statistical aspects in the handwriting recognition process at a low level. Dual use of cellular visual microprocessor and standard digital microprocessor enabled efficient processing in all aspects. The integration was motivated by the human reading process based on perceptual features and linguistic background knowledge.

A holistic approach is used to extract word level topographic features with no letter detection. I introduced the use of shape codes to handle the ambiguity of letter positions of the features. An extension has been developed to regular expressions to establish a proper interface between the recognition and the linguistic module. This provides under-specified information for the linguistic filtering, where morpho-lexical and syntactic models validate (either accept or reject) different orthographical candidates derived from a single recognized symbol sequence. I also introduced an inverse filtering technique based on global and unreliable local features, where the feature description of word candidates is matched against the detected geometry.

I have shown the great importance of linguistic knowledge in recognition systems, similarly to human reading. Implementation aspects of different levels of linguistic knowledge influencing the recognition process have been analyzed. I have implemented a statistical syntactic knowledgebase, integrated it into a recognition system and showed that even by using simple models of high level information, the number of possible sentence candidates can be decreased by a high degree meanwhile the recognition rate for words increases.

## 3 Detection and recognition of signs and displays in 2D visual flows

### 3.1 Blind Mobile Navigation

Visually impaired people not only have to miss all visual information that surrounds us, but also all the self-confidence that it provides. In our visually orientated world they are very defenseless in most situations, and there are very few means to compensate their disability. In spite of the impressive advances related to retinal prostheses, there is no imminent promise to make them soon available with a realistic performance to help navigating blind and visually impaired persons. A mobile navigation device would serve them well by providing some sense of safety and independence in many real-life situations. Although such a device has many possible applications, this chapter concentrates only on some well defined, simple scenarios.

#### 3.1.1 The “*Bionic Eyeglass*” framework

The *Bionic Eyeglass* is a wearable device (with a power consumption under 300mW) under development with TeraOps visual computing power and with audio input-output to guide visually impaired people in their daily life. It provides a broad functionality for typical situations where visual information is crucial. These situations can be categorized into three classes based on the location where they typically occur. Each situation defines one or more tasks that the device can perform for the user. Situations are summarized in Table XI.

**Table XI. Typical tasks considered for the Bionic Eyeglass**

Place	Home	Street	Office
Lightning	Controlled	Uncontrolled	Controlled
Events	Both emergency and conscious		
User-initiated	Color and pattern recognition of clothes	Recognition of marked and unmarked crosswalks	Recognition of control signs and displays in elevators
	Bank note recognition	Escalator direction recognition	Support in navigation in public offices and restrooms
		Public transport sign recognition	Identification of restroom signs
		Bus and tram stop identification	Recognition of signs on walkways
		Recognition of client displays (e.g. in banks)	
		Recognition of messages on ATMs	
Autonomous warnings	Light left switched on	Obstacles at head and chest level (branches, signs, devices attached to the wall, etc.)	
	Gas oven left turned on		

### 3.1.2 Detection and recognition of signs and displays

Detection and recognition of signs and displays in real, noisy environments is a key element in many functions of the Bionic Eyeglass. Taking photographs is not a real option for blind or visually impaired people, for the following reasons:

- it is really difficult for them to point the camera towards a sign that they do not know the position of
- the sign can be in motion
- other people might hide the sign temporarily
- in many cases there is a time constraint, i.e. the user has to make a decision relatively fast

Due to the reasons mentioned above it is much easier for a visually impaired person to pan towards potential directions with the camera while it continuously processes the input and it can also give real-time positive feedback if the target is in the scene, thus helping the user to locate it. Video input is especially crucial in situations where all of the conditions above are true, e.g. when identifying the route number of a public transport vehicle arriving to a stop. Supposing a video-rate of 15 frames/sec is realistic for consumer level cameras, whereas the user can take at best 1-2 photographs per second. This means that a video input can give at least an order of magnitude more input respected to taking steady images.

In most previous works in the field (e.g. [45],[47]) good quality and relatively high resolution inputs were used. Assuming a good quality input video flow is not realistic, because the cameras to be used must be small enough and must have small power consumption to be worn head-mounted or fit in a light, mobile device that a user can always carry with him- or herself.

The spatial-temporal analogic cellular algorithms I developed are able to localize signs and displays in low-resolution 2D visual flows recorded by mobile devices, and to recognize numbers they contain.

### **3.2 A novel semantic framework for multimodal information processing**

When I started to work on detection and recognition algorithms for unconstrained real-world video flows, I soon realized that without modeling semantics it is impossible to achieve reasonable results. I created this semantic model of multimodal sensory information processing to give a framework for related research. The key concept of this model is to embed a priori semantic information into the process of detection and recognition.

The information flow in our model is based on a probabilistic approach which actually maps a probabilistic association map whose weights can be continuously modified. Initial probability values are determined by a priori knowledge or set to a uniform distribution. Techniques for handling the ambiguities include probability mass functions, fuzzy sets and rules, and abductive inference (best explanation reasoning).

### 3.2.1 Information flow

The sensors of multiple modalities collect information on their environment. The information is processed through multiple levels in the system. The level of abstractness increases along the processing.

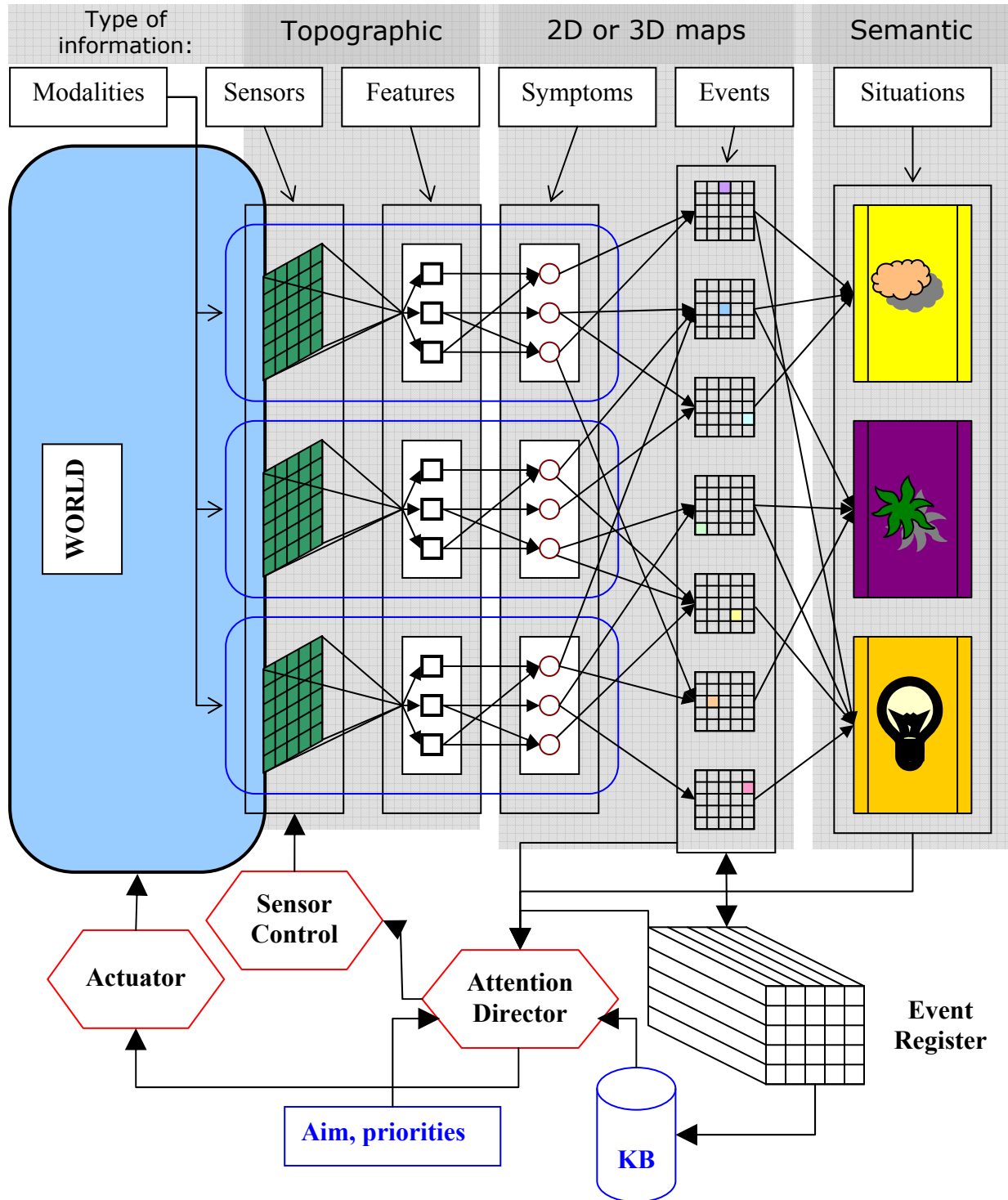


Figure 3.1. Information flow in a multimodal sensory system

Information of different modalities is continuously and separately processed by analogic algorithms to obtain *features*. These are combined into simple *symptoms* in each modality channel. Certain symptoms induce an event, which usually has influence on multiple modalities. Symptoms can be excitatory or inhibitory, and they can be either required or optional in the formation of an *event*. Furthermore, other events can also serve as an – excitatory or inhibitory – input of the event through the Event Register (ER). The highest level in information processing is the identification of situations. These are characterized by events that must have happened in order to allow the situation to be considered active. Additionally, there can be certain conditions on the order of the events.

The most crucial module in the system is the Attention Director (AD). This determines the operation and parameters of the actuators and indirectly the sensors too, while it receives triggers that arise from combination of the active situation and certain events either directly or indirectly through a knowledge base that contains basic information and general rules about the physical environment in a logic form. The AD is also influenced by information explicitly provided by the user on the aim of the system and tasks to perform.

The actuators perform any operation concluded to be necessary in the previous level. Sensor Control interprets the instructions from the AD in order to properly adjust sensors.

### ***3.2.2 Description of abstraction levels***

Abstraction levels form a hierarchy that processes information in a bottom-up manner. In the following we describe the meaning and role of each level. Table XII lists active modules for a sample situation, when the slamming of a door is identified through the audio modality.

*Sensors* for different modalities have to be able to identify certain features. Modalities can be categorized into two classes:

- Array sensors: audio, visual, infra, tactile
- Miscellaneous sensors (scalar, binary): temperature, pressure, various switches

*Features* describe simple observable properties of a single modality of the environment, extracted from the sensory information. Topographic features always have a spatial location. We require that features have a confidence level, which can be different for each symptom they are bound to. A visual feature may be the color or the shape of an object, a tactile feature may be the rudeness or the temperature of a surface.

**Table XII. Event generation by a series of modules activated by an audio feature**

	<b>Module</b>	<b>Example</b>
	Active situation	sitting in an office
	Knowledgebase record	state of the door: open (visual)
$t_0$	Feature	clashing sound from the direction of the door
	Symptom	slamming of a door (audio)
	Attention director	visual observation of the door
	Symptom	door is closed (visual)
	Event generated	door has been closed
$t$	New Knowledgebase record	state of the door: closed (visual and audio)

*Symptoms* are still monomodal, but more abstract than features. A symptom is an event-like entity, but it is limited to a single modality. A visual symptom can be the visual presence or movement of an object or a person, an audio symptom can be the roar of the sea or the buzz of a neon light.

An *event* is an action or any kind of change in the environment. An event is most characterized by its result. Since events happen at a certain point of time, but can have an effect that lasts longer, we need a memory that registers the occurrence of every event and remembers them as long as they might have an effect on the given environment. This task is realized by the Event Register. An event can be someone entering a room or the arrival of a bus to the stop.

*Situations* refer to the environment of the observer, like traveling on a train, visiting a museum or paying in a supermarket. A situation influences the attention direction, but only if it is active. However the determination of the active situation is not unequivocal due to two reasons. On one hand the symptoms might not be restrictive enough to determine a single situation, on the other hand two or more previously described (known) situations can be relevant and active in a given case. Parallel situations may be independent or somewhat related.

Therefore situations are weighted, and the more weight a situation is assigned, the bigger the influence it will have on decisions and attention direction. The weight of a situation is determined by a priori information and by events identified during operation.

The influences described above serve for weighing the significance of the events and situations represented in the system. The exact mathematical functions are not yet determined.



In the simplest case they are linear weights, but use of nonlinearities will be inevitable in order to achieve good results. Exponential weighing seems the most promising way to introduce nonlinearities, though implementation on present architectures would suggest a nonlinearity after the linear combination of the weights. The use of bias is important and introducing exponential decays for memories or weights could also bring significant advantages.

### **3.2.3 Further considerations**

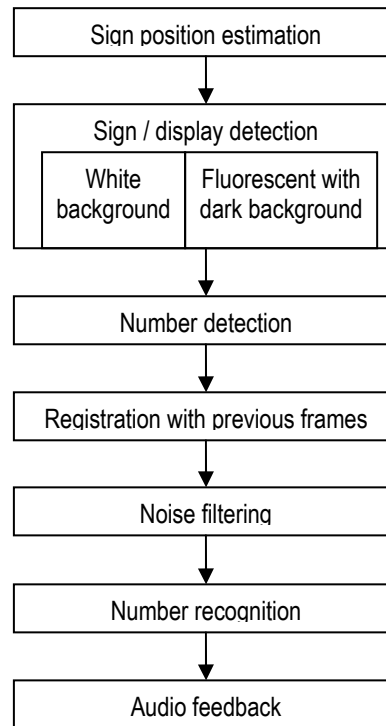
Self learning and adaptivity is very important to be able to recognize, learn and handle unknown symptoms, events and situations. This could be implemented through a memory that registers feature combinations occurring. The Knowledgebase should be able to create new symptoms based on frequent combination of features, new events from frequent combination of symptoms and new situations from frequent combination of events. Frequency of co-occurring symptoms, events and situations should be also assigned to newly created entities.

Expectations can also heavily influence the actual performance of the system, so these should be included in the priorities provided to the system. The *Attention Director* module should also incorporate some randomness and it should be very closely related to the actuators.

## **3.3 Sign and display localization**

After the general framework presented in the previous section, let us narrow our attention to the specific semantic situation described in Section 3.1.2, namely detection and recognition of signs and displays on public transport vehicles.

Determining whether a sign or a display is present on an image and localizing it is a very non-trivial task, especially on low-resolution, noisy images. The purpose of the sign localization step is to find the text or the numbers to be recognized. But the intuitive way to tell the location of the sign is by looking for letters and/or numbers on the image. In other words the presence of a sign is only justified by the existence of information on it.



**Figure 3.2. Block diagram of the sign detection and recognition framework**

Detection of the signs is complicated due to several reasons:

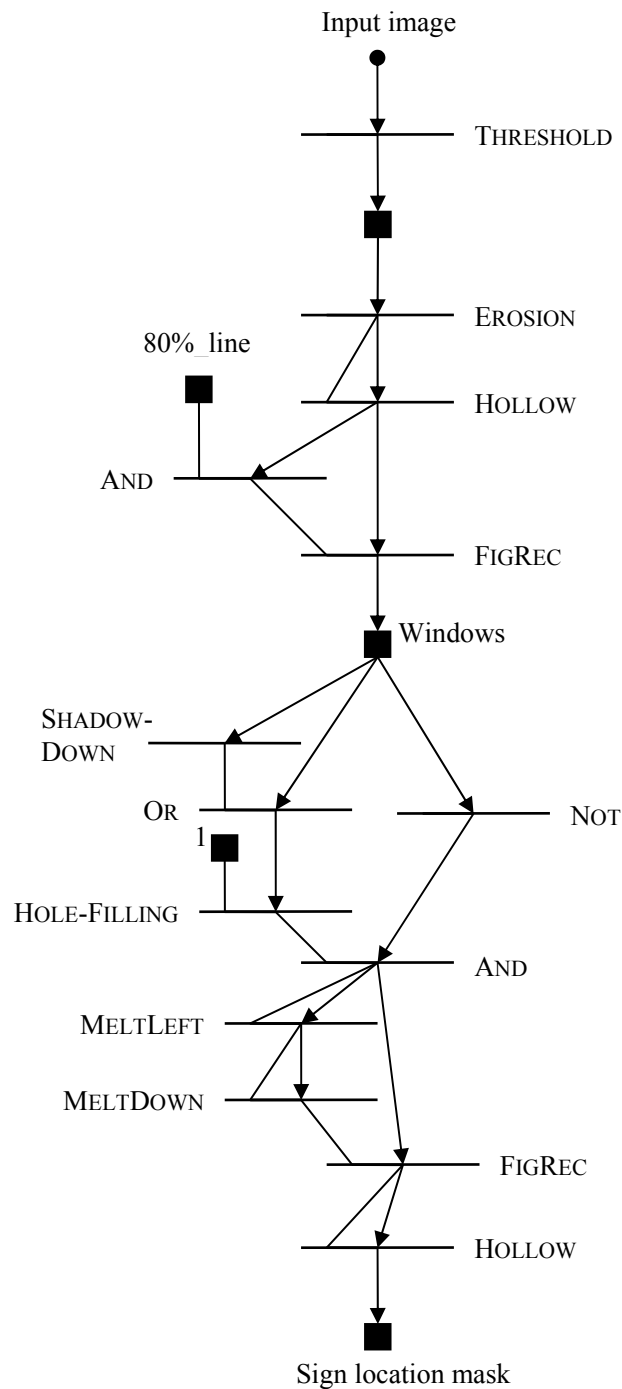
- great variety in types of displays and in places they are used,
- various lighting conditions,
- small field of view of the sensor,
- the user typically has no knowledge in which direction the sign is located,
- speed of the feedback to help the user in finding the right direction is restricted.

Different types of signs and displays can be detected in different ways. I have developed algorithms to detect signs with a white background and ones that have fluorescent numbers with a dark background. The steps of the processing are shown in a block diagram in Figure 3.2.

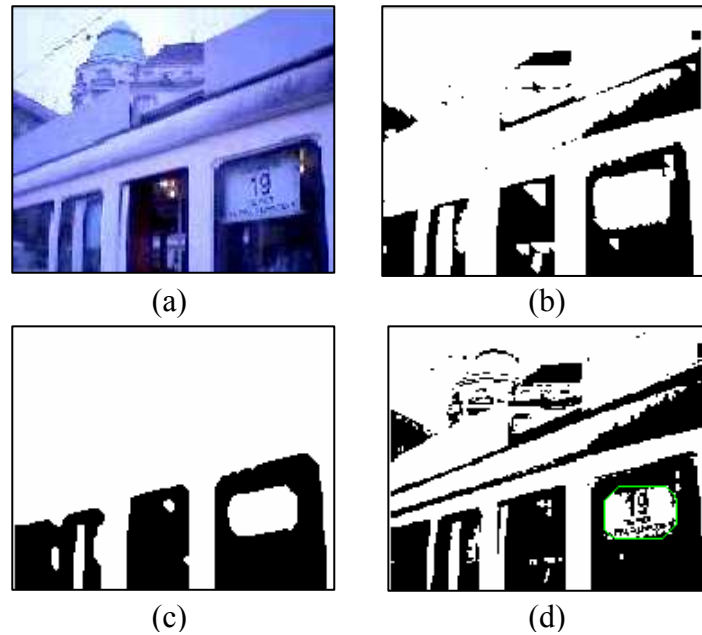
### **3.3.1 Localizing black & white signs**

The idea behind the algorithm is as follows. After thresholding the image with an adaptive threshold computed by minimizing intra-class variance of grayscale values in the black and the white regions, dark window areas are detected first. In the next step the algorithm looks

for almost white holes in the window areas with certain size constraints. Basic steps are shown in Figure 3.3, in the form of a Universal Machine on Flows (UMF) diagram. The result of the algorithm for a sample frame is shown in Figure 3.4.



**Figure 3.3.** UMF diagram of the algorithm locating signs with a white background.



**Figure 3.4.** *Sign localization on a tram. (a) Original input frame (b) Binarized and eroded image (c) Smoothed window areas (d) Sign location*

The sign location is tracked through the frames with a simple linear kinematical model. The model gives the probable location of the sign on the next frame based on the location in the previous two frames. This makes the localization of the sign faster since the target area is much smaller. If the sign cannot be located in the estimated area, then the track is assumed to be lost and the algorithm is rerun for the whole frame.

### 3.3.2 Number localization

Once the sign area is located a new threshold is computed for the sign area using the same method. Actual numbers can be extracted by getting rid of other text and noise present in the sign area. However, it is not critical to remove every noise at this point, because a final noise removal step takes place before the actual recognition.

Noise removal is realized in two parallel ways to make the extraction more robust. On one hand the frame is removed together with patches lower than a certain value, using the fact that the numbers are printed with the largest font size. The threshold was determined to be  $1/40^{\text{th}}$  of the vertical resolution of the camera by taking into account possible distances from the vehicle and the typical size of figures of the route number signs. On the other hand we also use the a priori information on the number location based on previous frames (they are usually in the center).

### 3.3.3 *Localizing displays: combining colors and morphology*

In contrast to printed, black and white sign detection, the perceptual localization of displays is mainly driven by the color of the display. The background itself cannot be located, its perceptual function is only to create high contrast with respect to the foreground.

The particular difficulty in detecting displays is that in general they cannot be identified by some distinctive features. Human observers are able to identify them by recognizing the signs on them. But this poses a paradox for a machine vision system: the aim of locating the display is to recognize the sign on it thereafter, but to locate them we need to recognize their content first. To overcome this paradox we defined some simple, but characteristic properties of displays:

- bright and large figures are displayed on them,
- typical colors are yellow, green and red,
- they have high contrast for good legibility,
- form of figures is not patch like, rather stroke type.

The first three properties refer to color and luminance, whereas the last one is a morphological property that, despite being ambiguous, enables basic differentiation from other formations with similar color and luminance patterns.

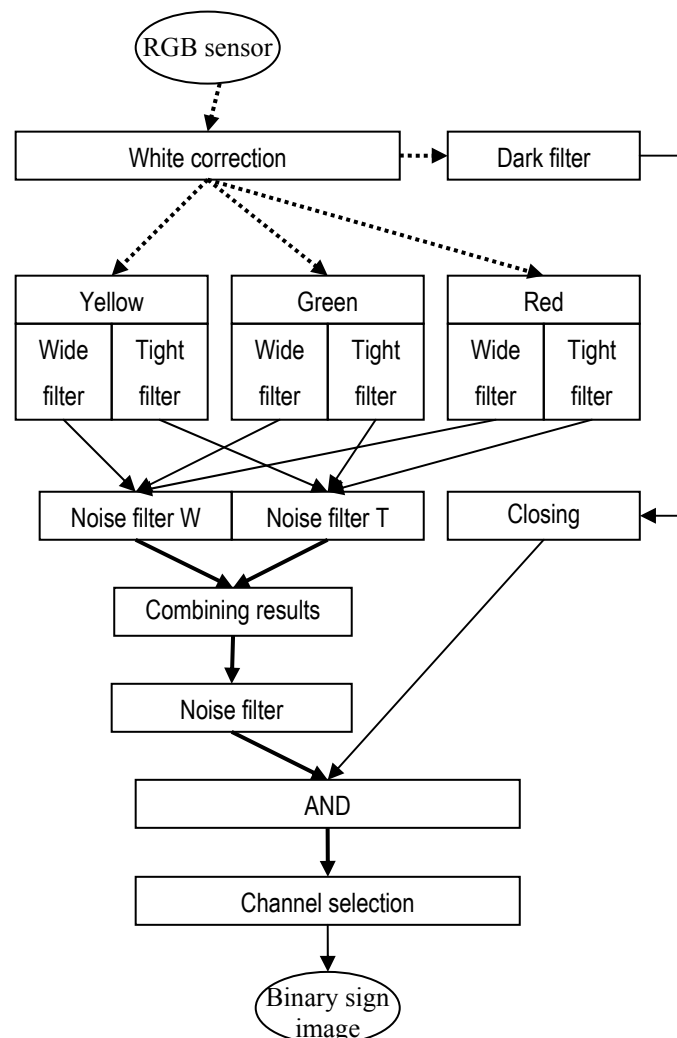
However, to better cope with the great variety of displays, I incorporated the possibility to give specific properties of displays. These may include the font(s) used, exact foreground and background color ranges, size of figures respected to the display background, as well as other information on neighboring objects. This allows the system to perform with greater robustness in case of frequent display models and typical scenarios.

Figure 3.5 shows the process of localization of color displays. Due to lack of availability of a locally adaptive sensor, to deal with different lighting conditions and retain color constancy, automatic color correction is performed on the input as a first step, using a simple method that aims to compensate for described in [46]. This consists of luminance adaptation, performed in the Luv color space by low-pass filtering the luminance, and chromatic compensation, which is based on the assumption that the mean of each of the RGB channels should be close to the half of the maximum.

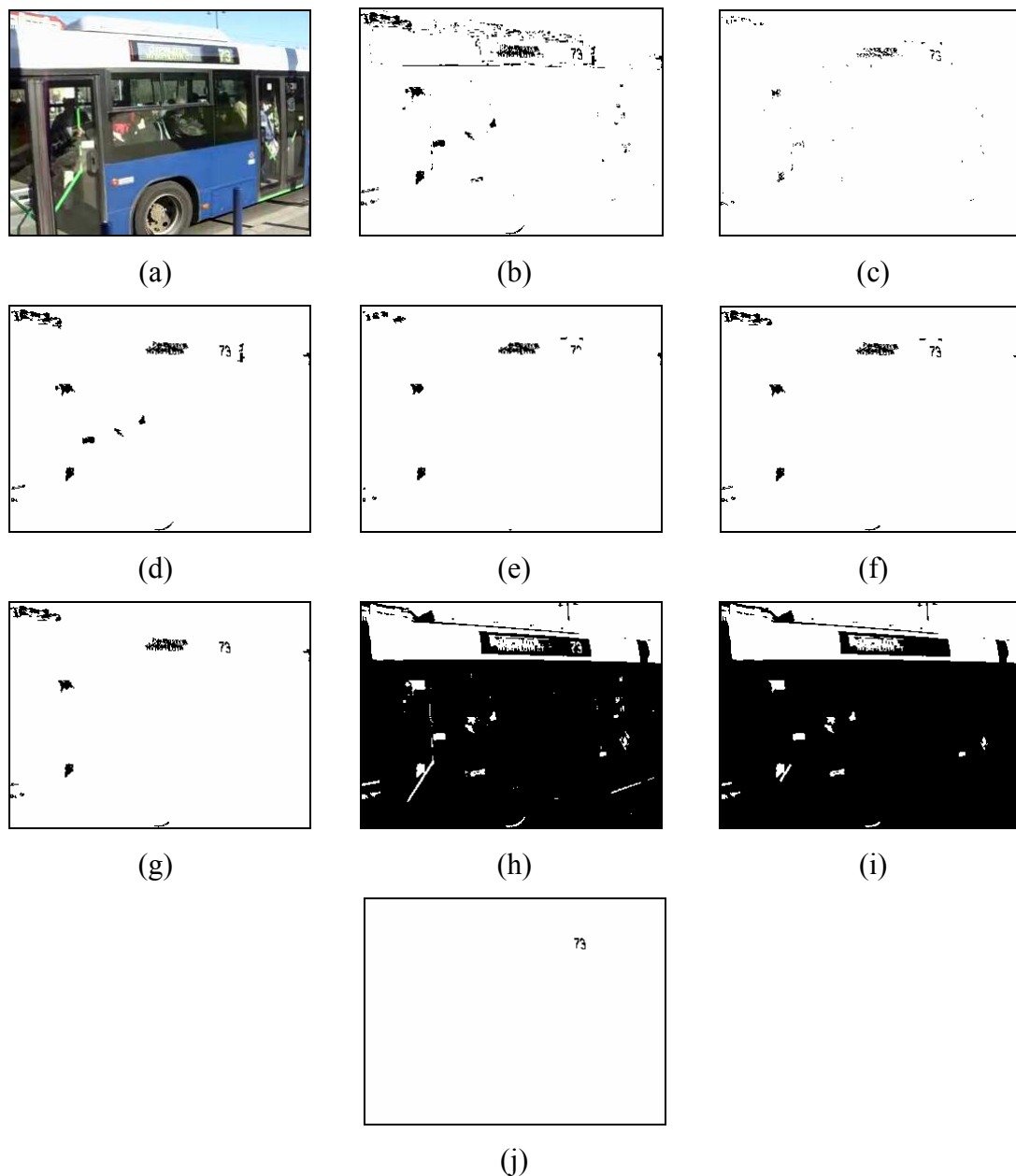
Then three different bright color range filters and a background (dark) color range filter are applied. Bright color range detectors have both a wide and a tight detection range. Wide range

filters detect the foreground pixels of the display of the given color robustly, but cover many other irrelevant objects too, whereas narrow range filters are much less sensitive to noise, but may miss some foreground pixels (see Figure 3.6 (b) and (c)). To determine proper wide and tight color ranges for specific display types we used sample training videos taken at different hours of the day and under different light conditions, and analyzed the colors manually. Ranges were determined for three typical display colors: yellow, green and red.

Pixel noise is removed from the output of both the wide and the tight color filter by using the **MELTDOWN**, **SMALLKILLER** and **FIGREC** templates respectively. We use the output of the denoised tight range filter to reconstruct the other denoised image and noise is removed from this one too with the same method (Figure 3.6. (f) and (g)).



**Figure 3.5. Flow diagram of display detection. Dotted lines refer to RGB data flows, bold lines refer to multiple binary flows and narrow lines refer to single binary flows.**



**Figure 3.6.** Image sequence showing the detection process. (a) Color corrected image. (b) Wide range filter for yellow channel. (c) Tight range filter for yellow channel. (d) and (e) Noise removed from (b) and (c) respectively (f) Figure reconstruction using (d) as input and (e) as initial state. (g) Noise removed from (f). (h) Result of dark filter on (a). (i) Closing performed on (h). (j) Number image

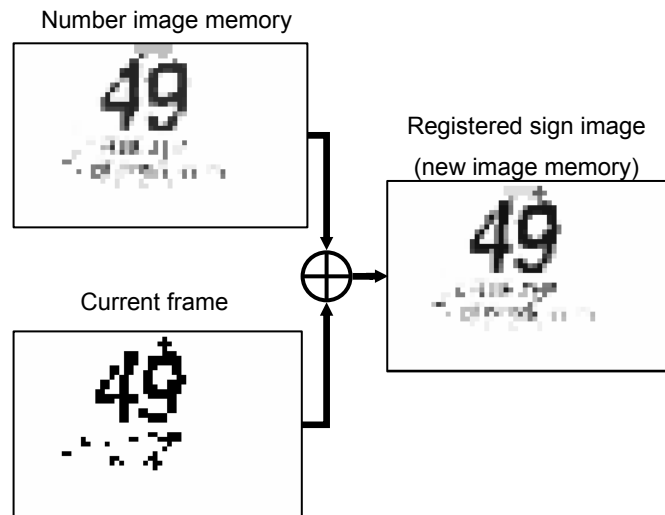
The use of the dark filter is motivated by trying to locate those bright formations where they do not form patches; that is every pixel has dark pixels in a close neighborhood, where the diameter of the neighborhood is the stroke width. A closing operation is performed on the dark filtered image to achieve this goal, and the result is used as a mask to find the right formations on the combined image from the denoised bright filters. This is carried out by an

**AND** template. Closing is the key operation to distinguish strokes from patches and the length of its operation depends on the stroke width. All these operations are performed in parallel for all three colors.

Finally the channels are verified if they contain patches having the correct ratio of width and height, and if yes their output is sent to the recognition module.

### 3.3.4 Registration with previous frames

Due to the low resolution of the images and the high level of noise present on them a number of semi-dark pixels belonging to the number to be recognized are below the threshold value. Therefore the binary number images become vague. To overcome this problem we make use of the a priori knowledge that the signs normally do not change, which means we can superpose subsequent sign images to achieve better image quality. (See Figure 3.7)



**Figure 3.7.** Enhancement of the number image by superposing the actual frame and the image memory.

As a first step of this process the images need to be registered, because detected borders of the sign can differ due to noise and changes in light conditions. Since rotation in the plane of the image is negligible (the user is expected to and can generally easily avoid twisting wrist moves), registration can be done by shifting. Optimal shifting parameters are calculated via cross-correlation:

$$c_n(u, v) = \sum_{x, y} f_n(x - u, y - v) f_{n-1}(x, y), \quad (3.1)$$



where  $f_i$  denotes the sign image of the  $i^{\text{th}}$  frame,  $u$  and  $v$  are the shifting parameters and  $c_i(u, v)$  is the cross correlation matrix at the  $i^{\text{th}}$  frame ( $i \geq 1$ ). The values of  $u$  and  $v$  are determined by maximizing the cross-correlation:

$$(u, v) = \arg \max_{u, v} c(u, v) \quad (3.2)$$

The size of the correlation window has been determined – based on experimental data – to be  $\pm 10\%$  (i.e. 20%) of the image size, both vertically and horizontally.

The shift operation accounts for translation in the plane, but it cannot handle the remaining three degrees of freedom (d.o.f.):

- translation towards the camera (zooming – 1 d.o.f.),
- rotation out of the plane (changing perspective – 2 d.o.f.).

According to our experiments these two changes can be neglected through 2-3 frames (at 15 frames/sec), but not longer. Therefore we maintain a gradually fading memory of the number image and we use the weighted sum of the memory and the actual frame to determine the new “aggregated” number image.

Let  $a_i$  denote the aggregate memorized number image in step  $i$ . At the beginning of a new sign track the number memory is initialized with  $a_0 = f_l$ . Registration works the same way as described above except that one writes  $a_{n-1}$  instead of  $f_{n-1}$ . The new number image is given by the following homotopy:

$$a_n = \alpha f_n + (1 - \alpha) a_{n-1}, \quad (3.3)$$

where  $\alpha$  is a function of normalized correlation. The optimal function depends on usage habits, assumptions made on the input image flow, and the threshold value used for binarization (recognition uses a binary number image as input). We used a threshold value of 0.1 and the following piecewise linear function:

$$\alpha = \left\{ \begin{array}{ll} 0.8, & \text{if } \hat{c}_{\max} < a \\ 0.8 + 0.3 \frac{a - \hat{c}_{\max}}{b - a}, & \text{if } a \leq \hat{c}_{\max} \leq b \\ 0.5, & \text{if } \hat{c}_{\max} > b \end{array} \right\}, \quad (3.4)$$

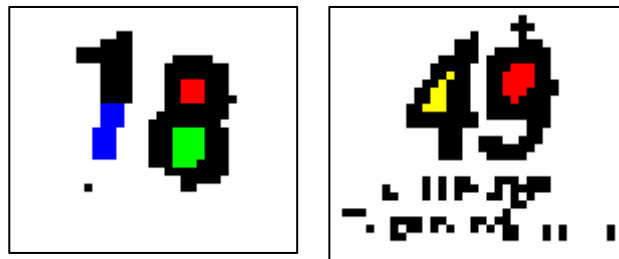
where  $\hat{c}_{\max}$  is the maximum normalized correlation,  $a$  is the average normalized correlation of some random sign images and  $b$  is an adaptively corrected maximum value of the normalized correlation. These were determined based on the following considerations:

- image memory should be preserved as long as the number features are detectable from the aggregated image, to compensate for binarization noise.
- in case of fast camera moves or a relatively high speed vehicle, binary number images may change in size or orientation in under 3 frames too much for the correctly super-positioned numbers to remain recognizable

Parameters  $a$  and  $b$  were tuned on 5 sample video flows. If the sign track gets lost, then the image memory is cleared.

### 3.4 Number recognition

For number recognition we use topographic shape features that can be extracted by cellular wave algorithms. In the first step the number of figures in the number is determined by counting connected objects on the image that are bigger than a threshold. This threshold can be higher than the one used in Section 3.3.2, because at this stage we assume the figures are already fully connected. In a second step feature maps are generated (see Figure 3.8).



**Figure 3.8.** Feature maps of route numbers: vertical line (blue), upper hole (red), lower hole (green), triangular hole (yellow)

Features used include holes and lines. Holes are classified based on shape, size and position, whereas lines are classified according to orientation (horizontal or vertical) and position. The method is based on the algorithms used for handwritten word recognition, described in detail in Section 2.7.

Holes are defined the same way as in Section 2.7.4 and the same **HOLE-FILLING4** template is used to detect them. Lines are detected as narrow but long rectangles, with threshold parameters based on height of figures, with two different combinations for vertical and horizontal lines.

Size classification parameters are also derived from figure height. The figure the feature belongs to and horizontal position is given by vertical projection, whereas horizontal projection enables to calculate vertical position.

Shape classification is needed to differentiate between round and triangular holes. Classification is based on vertical and horizontal histograms. Histogram value increases downwards and to the right in case of triangular holes (“4” is the only number having this feature).

**Table XIII. Feature conversion table**

Fig.	Hole			Line	
	<i>Big</i>	<i>Round</i>	<i>Triang.</i>	<i>Horiz.</i>	<i>Vert.</i>
<b>0</b>	+				
<b>1</b>	LTO				M
<b>2</b>	LO		DRO	D	
<b>3</b>	LO				
<b>4</b>			U	M	
<b>5</b>		URO, DLO		U	
<b>6</b>		URO, D			
<b>7</b>				U	
<b>8</b>		U, D			
<b>9</b>		U, DLO			

+: Present    D: Down    U: Up    M: Middle  
L: Left    R: Right    O: Open    T: Tight

Features are converted to numbers based on a feature allocation table (see Table XIII). The feature detection method has been extended by checking for open holes in figures to make the recognition more robust and make it able to distinguish figures without holes (especially ‘3’). This is carried out by drawing side bars on the figure image, and checking for holes on these

modified images. These holes are classified in the same manner as normal holes. Table XIII shows the new feature conversion table, Figure 3.9 shows sample feature maps.



**Figure 3.9.** *Sample feature maps. Right open holes and their auxiliary lines are shown in cyan, middle vertical line is shown in blue, and upper round hole is shown in red.*

## 3.5 Experimental Results

### 3.5.1 Blind Acquired Visual Flow Database

We have recorded more than 100 video flows of lengths between 15 and 90 seconds in bus and tram stops of arriving and departing vehicles. Stops serving several lines were selected (each one at least five) to let the video database contain various types of vehicles and route number signs. The video flows were recorded by a blind person in daylight, but under different light conditions.

We used commercial cellular phones and digital cameras to record videos. The resolution of the videos is either QCIF or QVGA. Phones appropriate for this task must have a camera capable of video recording with at least QCIF resolution, and there must be a hard-button by which recording can be started and stopped (soft buttons on a touch-screen are too vague for a visually impaired user). Digital camera recordings were included in the database as well, because they the methods used for compressing the video flows are less lossy and allow for better quality.

### 3.5.2 Black & white signs

Black and white signs can be found on 25 videos. We categorized them into four classes based on recognizability. 7 records belong to the first class, on these not even a thorough human analysis can make out the signs. The second class (6 videos) contains records on which the signs are blurred and can only be seen for a very short time (3-4 frames), but smart humans can recognize them if looking at them frame by frame. The third class consists of 4 videos that have only a few frames with the sign being readable on them, but most humans can still recognize them. On the remaining 8 videos the numbers are clear and shown for more

seconds. These records form the fourth class. Detection and recognition results are shown in Table XIV.

**Table XIV. Detection and recognition results**

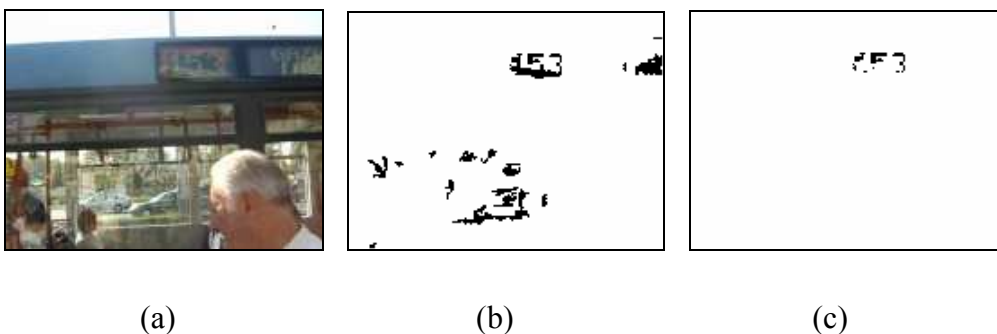
Class	Number of videos in class	Sign correctly located	Number correctly located	Number correctly recognized (without line info)	Number correctly recognized (with line info)
1	7	1	0	0	0
2	6	4	3	1	1
3	4	3	3	2	3
4	8	7	7	6	7

Recognition of videos in classes 1 and 2 are not critical, because users can be easily taught where to stand in the stop and how to direct the camera during the arrival of the vehicle in order to make a good quality record of the passing sign.

For classes 3 and 4 we have achieved a recognition rate of 83% and a rejection rate (when the sign could not be located) of 17%, with the recognizer knowing the bus or tram lines touching the given stop.

### 3.5.3 Color displays

The measurements were carried out on 17 video sequences that contain buses or trams with color displays. The color display detection algorithm could correctly locate the display on 14 videos, even if not for all frames. Very bright sunlight and glancing cover caused or weak lighting conditions in evening shots caused the failure in the remaining three cases. In case of four videos out of the 14, more sign candidates were detected. The correct patch could be selected out of these by tracking the movement of patches on an optical flow map.



**Figure 3.10. A sample image on which parts of number images are missing, thus the recognition module fails to recognize them correctly. (a) Original image (b) Output of denoised bright filters (c) Final result of detection**

The recognition algorithm could correctly recognize 11 out of the 14 localized signs. Clear number forms were always correctly recognized, but in cases when numbers touched each other or part of them was distorted, then false or missing features lead to wrong results. An example is shown in Figure 3.10

### **3.6 Conclusions**

I presented a semantic framework for multimodal information processing. I showed how to use semantic embedding in algorithm design by giving algorithms, providing an audio guide for blind or visually impaired people to help them to identify and signs and displays and read information on them. The algorithms are based on the cellular wave computing paradigm and they were designed to operate robustly on noisy, low resolution video flows provided by mobile cameras. For algorithm design I used semantic descriptions of the visual scenarios that occur in tram and bus stops.

Our results show that we need to give some basic instructions to the blind or visually impaired user in order to enable him/her to use the device in accordance with some basic assumptions that greatly increase the probability of a correct recognition.

## 4 Invertible Cellular Automata

### 4.1 Introduction

The concept of Cellular Automata (CA) was introduced by John von Neumann [54] and Stanislaw Ulam [53]. Originally they were defined on a two-dimensional lattice, every cell of which has a state out of a finite set  $\Sigma$ , called cell state. The evolution of the automaton is driven by a *local mapping function*  $N: B(\Sigma) \rightarrow \Sigma$ , that maps the states of a local neighborhood of each cell to a new cell state. In our case we restrict ourselves to elementary CAs, heavily studied by Wolfram [55]. An *elementary CA* is one dimensional, the neighborhood size is one and the set of cell states has only two elements, i.e. the cell states are binary. Thus the global state of the lattice can be represented by a binary string. Let  $\Sigma^L$  be the set of  $L$ -bit binary strings. The local map can be expressed as a truth table with three input bits and one output bit:  $N: \Sigma^3 \rightarrow \Sigma$ . A standard notation introduced by Wolfram is to refer to local rules as the decimal form of the 8-bit output column of the truth table, with the first bit (corresponding to the input (0,0,0)) being the least significant bit (LSB) and the last bit being the most significant bit (MSB).

Naturally, the local mapping function defines a *global map*  $T_{\boxed{N}}: \Sigma^L \rightarrow \Sigma^L$  from the set of binary strings of a length  $L$  to itself. By iterating the global map we gain trajectories over the set of global states. Let  $T_{\boxed{N}}^r: \Sigma^L \rightarrow \Sigma^L$  be the  $r$ -times iterated mapping function of rule  $\boxed{N}$ .  $T_{\boxed{N}}$  is not necessarily invertible, indeed, in most of the cases it is not, because  $\Sigma^L$  contains two global states that are mapped to the same global state under  $T_{\boxed{N}}$ .

In this chapter I define the Isles of Eden digraph and I show that it can be used to determine the invertibility of elementary CAs. Then I apply it to all local rules that are not trivially invertible. For rule 45 and 154 (and all rules in their equivalence classes) I prove that

every orbit over  $\Sigma^L$  is an *isle of Eden* if and only if  $L$  is odd. For rule 105 and 150 I prove that every orbit over  $\Sigma^L$  is an *isle of Eden* if and only if  $L$  is not a multiple of 3. The statement is proved by applying Theorem 4.1 to these rules in Theorem 4.2 and 4.3. The main result is the proof of the backward direction of Theorem 4.1.

## 4.2 Isles of Eden

In this section, after defining isles of Eden, we will prove the equivalence of the following statements for a local rule  $\boxed{N}$  and a given string length  $L$ :

1. Every orbit is an isle of Eden
2. There are no Garden of Eden states ( $T_{\boxed{N}}$  is surjective)
3. There are no merging points in the trajectory, ( $T_{\boxed{N}}$  is injective)
4. Every global state has a unique preimage under  $T_{\boxed{N}}$
5. The cellular automaton is globally invertible

Equivalence of statement 3 and 4 is trivial, since they express the same thing in forward and inverse direction under the global map  $T_{\boxed{N}}$ . The fourth statement is used as the definition of global invertibility for elementary cellular automata (statement 5). Lemma 4.1 and 4.2 prove the equivalence between statement 1 and 4, whereas Lemma 4.3 proves the equivalence of statement 2 and 3.

First we introduce the mapping  $\Phi$  of binary strings to the unit interval  $[0,1]$  of real numbers [59]:

$$\Phi(\mathbf{x}) \triangleq \sum_{i=0}^{L-1} 2^{-(i+1)} x_i \quad (4.1)$$

where  $\mathbf{x} = (x_0 \ x_1 \ x_2 \ \dots \ x_{L-1})$ . Using this mapping the global map  $T_{\boxed{N}}: \Sigma \rightarrow \Sigma$  induces an equivalent map  $\chi_{\boxed{N}}: \mathbb{R} [0,1] \rightarrow \mathbb{R} [0,1]$ , called the *CA characteristic function*. Let us now formally define the concept of isle of Eden [60].



**Definition 4.1. Isle of Eden**

A bit string

$$\mathbf{x} = (x_0 \ x_1 \ x_2 \ \dots \ x_{L-1})$$

is said to be a *period- $n$  isle of Eden* of a local rule  $\boxed{N}$  iff its preimage under  $\chi_{\boxed{N}}^n$  is itself, where  $\chi_{\boxed{N}}^n$  is the *time- $n$  characteristic function* of  $\boxed{N}$ .

More precisely,  $\mathbf{x}$  is a *period- $n$  isle of Eden* of a local rule  $\boxed{N}$  iff

$$\chi_{\boxed{N}}^n(\mathbf{x}) = \mathbf{x}$$

**Lemma 4.1.**

An  $L$ -bit binary string  $\mathbf{x} = (x_0 \ x_1 \ x_2 \ \dots \ x_{L-1})$  is a *period- $n$  isle of Eden* of a local rule  $\boxed{N}$  iff  $\mathbf{x}$  has a unique preimage under  $T_{\boxed{N}}^n$ :

$$\mathbf{x}_n \triangleq \left(T_{\boxed{N}}^n\right)^{-1}(\mathbf{x}) = \mathbf{x} \quad (4.2)$$

*Proof*

The definition of isle of Eden implies that  $\mathbf{x}$  is an isle of Eden iff the point  $\Phi(\mathbf{x})$  referring to it is an isle of Eden.

$\Rightarrow \Phi(\mathbf{x})$  being an isle of Eden can be formally written as

$$\left(\chi_{\boxed{N}}^n\right)^{-1}(\Phi(\mathbf{x})) = \Phi(\mathbf{x}) \quad (4.3)$$

Applying  $\chi_{\boxed{N}}^n$  to both sides we obtain

$$\Phi(\mathbf{x}) = \chi_{\boxed{N}}^n(\Phi(\mathbf{x})) \quad (4.4)$$

According to the definition of  $\Phi$  the following equation holds:

$$\Phi \circ T_{\boxed{N}} = \chi_{\boxed{N}} \circ \Phi \quad (4.5)$$

using which we can write the right-hand side as

$$\begin{aligned}
\chi_{\boxed{N}}^n(\Phi(\mathbf{x})) &= \underbrace{\chi_{\boxed{N}} \circ \chi_{\boxed{N}} \circ \dots \circ \chi_{\boxed{N}}}_{n \text{ times}} \circ \Phi(\mathbf{x}) = \\
&= \Phi \circ \underbrace{T_{\boxed{N}} \circ T_{\boxed{N}} \circ \dots \circ T_{\boxed{N}}}_{n \text{ times}}(\mathbf{x}) = \Phi(T_{\boxed{N}}^n(\mathbf{x}))
\end{aligned} \tag{4.6}$$

By substituting Equation (4.5) into Equation (4.3) and applying  $(T_{\boxed{N}}^n)^{-1} \circ \Phi^{-1}$  to both sides we obtain

$$(T_{\boxed{N}}^n)^{-1}(\mathbf{x}) = \mathbf{x} \tag{4.7}$$

which is exactly what the lemma states.

$\Leftarrow$  Starting from Equation (4.6), applying  $\Phi \circ T_{\boxed{N}}^n$  to both sides and substituting Equation (4.5) into the result we can obtain Equation (4.3). By applying  $(\chi_{\boxed{N}}^n)^{-1}$  to both sides we obtain Equation (4.2), from which it follows that  $\mathbf{x}$  is an isle of Eden.  $\square$

### **Lemma 4.2.**

Every orbit of a local rule  $N$  over  $\Sigma^L$  is an isle of Eden *iff* every  $\mathbf{x} \in \Sigma^L$  has a unique preimage under  $T_{\boxed{N}}$ .

#### *Proof*

$\Rightarrow$  Let us suppose indirectly that  $\mathbf{x}_0 \in \Sigma^L$  is an isle of Eden, but it either does not have a preimage under  $T_{\boxed{N}}$ , or its preimage is not unique. Using Lemma 4.1 we know that  $\mathbf{x}_0$  has a unique preimage under  $T_{\boxed{N}}^n$  (namely itself).

It is trivial that if a binary string does not have a preimage under  $T_{\boxed{N}}$ , then it cannot have a preimage under  $T_{\boxed{N}}^r$  for any  $r > 1$  either. Thus  $\mathbf{x}_0$  does not have a preimage under  $T_{\boxed{N}}^n$ , which is a contradiction, so  $\mathbf{x}_0$  must have a preimage under  $T_{\boxed{N}}$ .

If the preimage under  $T_{\boxed{N}}$  is not unique, then its preimage under  $T_{\boxed{N}}^n$  cannot be unique either, thus we arrived to a contradiction once again, so  $\mathbf{x}_0$  must have a preimage under  $T_{\boxed{N}}$  in this case too.

$\Leftarrow$  If every  $\mathbf{x} \in \Sigma^L$  has a unique preimage  $\mathbf{x}_1 \triangleq T_{\overline{N}}^{-1}(\mathbf{x})$  under  $T_{\overline{N}}$ , then their preimages, being members of  $\Sigma^L$ , also have unique preimages under  $T_{\overline{N}}$ , and this holds recursively for every preimage. Therefore every  $\mathbf{x} \in \Sigma^L$  has a unique preimage  $\mathbf{x}_r \triangleq (T_{\overline{N}}^r)^{-1}(\mathbf{x})$  under  $T_{\overline{N}}^r$  for every  $r > 1$ .

Since there are  $2^L$  binary strings of length  $L$ , every  $\mathbf{x} \in \Sigma^L$  has to map to itself under the inverse map  $(T_{\overline{N}}^{r(\mathbf{x})})^{-1}$  for some  $r(\mathbf{x}) \leq 2^L$  and thus it is, by definition, a period- $r(\mathbf{x})$  isle of Eden of local rule  $N$ .  $\square$

**Lemma 4.3.**

Every  $\mathbf{x} \in \Sigma^L$  has a preimage under  $T_{\overline{N}}$  iff every  $\mathbf{y} \in \Sigma^L$  has at most one preimage under  $T_{\overline{N}}$ , or equivalently  $T_{\overline{N}}: \Sigma^L \rightarrow \Sigma^L$  is surjective iff it is injective.

*Proof*

Let  $T_{\overline{N}}(\Sigma^L)$  denote the image of  $\Sigma^L$  under  $T_{\overline{N}}$ : 
$$T_{\overline{N}}(\Sigma^L) = \bigcup_{\mathbf{x} \in \Sigma^L} T_{\overline{N}}(\mathbf{x})$$

$\Rightarrow$  Let us suppose indirectly that every  $\mathbf{x} \in \Sigma^L$  has a preimage under  $T_{\overline{N}}$  for a local rule  $N$ , and there are  $\mathbf{x}_1 \in \Sigma^L$  and  $\mathbf{x}_2 \in \Sigma^L$  for which  $\mathbf{x}_1 \neq \mathbf{x}_2$  and  $T_{\overline{N}}(\mathbf{x}_1) = T_{\overline{N}}(\mathbf{x}_2)$ . Then the size of  $T_{\overline{N}}(\Sigma^L)$  must be less than the size of  $\Sigma^L$ , because  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are mapped to the same string and no  $\mathbf{x} \in \Sigma^L$  can be mapped to more than one string. Thus there is a string  $\mathbf{z} \in \Sigma^L$  that is not in  $T_{\overline{N}}(\Sigma^L)$ , which means, by definition, that there is no  $\mathbf{x} \in \Sigma^L$  that maps to  $\mathbf{z}$ , so  $\mathbf{z}$  does not have a preimage under  $T_{\overline{N}}$ , which contradicts our initial assumption.

$\Leftarrow$  Let us suppose indirectly that every  $\mathbf{y} \in \Sigma^L$  has at most one preimage under  $T_{\overline{N}}$ , but there is an  $\mathbf{x}_0 \in \Sigma^L$  that does not have a preimage under  $T_{\overline{N}}$ . Since there is no  $\mathbf{x} \in \Sigma^L$  for which  $T_{\overline{N}}(\mathbf{x}) = \mathbf{x}_0$ , thus  $\mathbf{x}_0 \notin T_{\overline{N}}(\Sigma^L)$ , and therefore the number of elements of  $T_{\overline{N}}(\Sigma^L)$  is less

than the number of elements of  $\Sigma^L$ . It follows that there must exist a  $\mathbf{y}_0 \in T_{\lfloor N \rfloor}(\Sigma^L)$  and an  $\mathbf{x}_1 \in \Sigma^L$  and an  $\mathbf{x}_2 \in \Sigma^L$  for which  $\mathbf{x}_1 \neq \mathbf{x}_2$  and  $T_{\lfloor N \rfloor}(\mathbf{x}_1) = \mathbf{y}_0$  and  $T_{\lfloor N \rfloor}(\mathbf{x}_2) = \mathbf{y}_0$ . This, however, contradicts our initial assumption.  $\square$

### 4.3 Rotation invariance

The following lemma expresses the rotation invariance property of local cellular automaton rules. We will use it when proving Theorem 4.1 in Section 4.5.

**Lemma 4.4.**

Let  $R^{(n)}(\mathbf{u})$  denote the vector obtained by circularly shifting the bits of an arbitrary vector  $\mathbf{u}$  by  $n$  bits, i.e.  $R^{(n)}(\mathbf{u}) = \{u_n, u_{(n+1) \bmod L}, u_{(n+2) \bmod L}, \dots, u_{(n-1) \bmod L}\}$ .

Let  $\mathbf{x} = (x_0 x_1 x_2 \dots x_{L-1})$  be an  $L$ -bit binary string and let  $n \in \mathbb{Z}$ .

The shifting operator  $R^{(n)}$  and the mapping function of a local rule  $N$  can be exchanged, that is  $T_{\lfloor N \rfloor}(R^{(n)}(\mathbf{x})) = R^{(n)}(T_{\lfloor N \rfloor}(\mathbf{x}))$ .

*Proof*

The  $i^{\text{th}}$  bit of  $R^{(n)}(\mathbf{y})$  is defined by

$$\mathbf{y}_{(n+i) \bmod L} = f_{\lfloor N \rfloor}(\mathbf{x}_{(n+i-1) \bmod L}, \mathbf{x}_{(n+i) \bmod L}, \mathbf{x}_{(n+i+1) \bmod L}) \quad (4.8)$$

for all  $0 \leq i \leq L-1$ , where  $f_{\lfloor N \rfloor} : I^3 \rightarrow I$  is the local mapping function of rule  $N$ , with  $I = \{0,1\}$ .

These set of equations are equivalent to those that define the  $j^{\text{th}}$  bit of  $\mathbf{y}$ , where  $j = ((i+n) \bmod L)$ . Thus the stated equation holds for every position.  $\square$

**Example 4.3.1.**

Let us consider rule 45, and let  $\mathbf{x} = 0110100$ . The output string  $T_{\lfloor 45 \rfloor}(\mathbf{x})$  is 0101101. If we circularly shift  $\mathbf{x}$  by three characters ( $n = 3$ ), we obtain  $R^{(3)}(\mathbf{x}) = 0100011$ . The output after

the shifting can be calculated by circularly shifting  $T_{\boxed{45}}(\mathbf{x})$  too by three characters, i.e.

$$T_{\boxed{45}}(R^{(3)}(\mathbf{x})) = R^{(3)}(T_{\boxed{45}}(\mathbf{x})) = 1101010.$$

**Example 4.3.2.**

Now let us consider rule 90, and let  $\mathbf{x} = 00001111$ . Thus  $T_{\boxed{90}}(\mathbf{x}) = 10011001$ . Let us circularly shift  $\mathbf{x}$  by  $n = -7$  characters:  $R^{(-7)}(\mathbf{x}) = 00011110$ . The output after the shifting operator can be obtained by applying circular shifting by  $-7$  characters to  $T_{\boxed{90}}(\mathbf{x})$  as well:

$$T_{\boxed{90}}(R^{(-7)}(\mathbf{x})) = R^{(-7)}(T_{\boxed{90}}(\mathbf{x})) = 00110011.$$

**Table XV. Mapping rules of local rule 45 and 90.**

Input pattern	Output bit	
	r. 45	r. 90
000	1	0
001	0	1
010	1	0
011	1	1
100	0	1
101	1	0
110	0	1
111	0	0

**4.4 Locating points with multiple preimages**

In this section we define the Isles of Eden digraph (and give a method to create it) that can generate the bits of two binary strings of any length  $L$  in parallel in every possible way so that their image under  $T_{\boxed{N}}$  is equal. We use the digraph in a theorem that allows one to decide if it is possible to generate different bit strings of a length  $L$  (inputs) that have the same image under  $T_{\boxed{N}}$ , by simply examining the cycles of the Isles of Eden digraph.

The idea behind the construction is that every move along an edge in the graph adds one bit to the length of input strings as well as to the output string, so  $L$  moves will generate strings of length  $L$ . The edges are labeled with these three bits. The nodes are labeled with the previous two bits of  $\mathbf{x}$  and  $\mathbf{y}$ , thus making it possible to compute the output based on the present node

and on the edge through which we leave the node. The label of the node where the edge leads to is composed of the second bits of the previous node and the corresponding bits of the edge. Actually, the nodes of the graph refer to a sliding double window of width 2 over  $\mathbf{x}$  and  $\mathbf{y}$ , and each move along an edge moves the windows to the right by one bit.

To compute the first move we need an initial condition of two bits that, due to the cyclic boundary condition, are also used at the end of the generation. This means that to meet the required boundary condition we need to return to the starting node. Let us give a brief overview on basic concepts in graph theory that we will use.

#### 4.4.1 Graph theoretical background

##### **Definition 4.2. Directed graph**

A finite directed graph (digraph)  $G$  is given by an ordered pair  $(V, E)$  and two functions  $i, t : E(G) \rightarrow V(G)$ , where

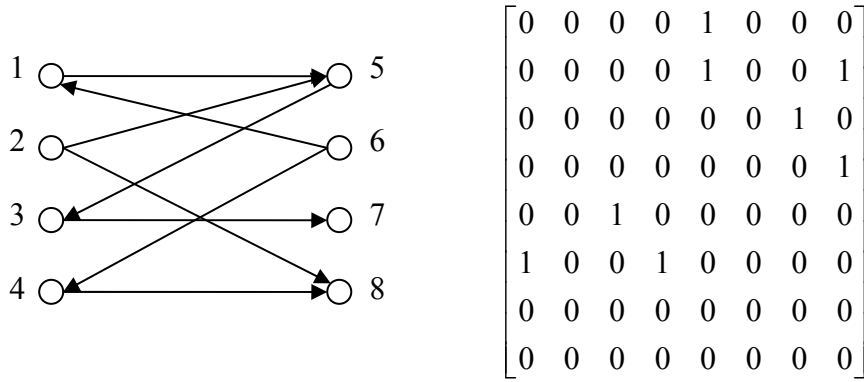
- $V(G)$  is a finite set of *vertices* (also called nodes),
- $E(G)$  is a finite set of *edges*,
- $i(e)$  and  $t(e)$  are the initial and terminal vertex of edge  $e \in E(G)$ , respectively.

##### **Definition 4.3. Walks and cycles**

A walk of length  $n$  in a directed graph is a finite sequence  $\pi = e_1 e_2 \dots e_n$  of edges such that  $t(e_k) = i(e_{k+1})$  for  $k = 1, 2, \dots, n - 1$ . A vertex can be included more times in the sequence. The vertex  $i(e_1)$  is called the *start vertex* and the vertex  $t(e_n)$  is called the *end vertex*. A walk with the same starting and ending vertex, i. e. where  $i(e_1) = t(e_n)$ , is called a *closed walk* or a *cycle*.

##### **Definition 4.4. Bipartite graph**

A graph whose vertices can be decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent is called a bipartite graph. A sample bipartite graph is shown in Figure 4.1.



**Figure 4.1.** A bipartite graph and its adjacency matrix

**Definition 4.5. Adjacency matrix of a directed graph**

The adjacency matrix of a finite directed graph  $G$  on  $n$  vertices is the  $n \times n$  matrix  $A$  where  $A(i, j)$  is the number of edges from vertex  $i$  to vertex  $j$ .

The powers of  $A$  refer to the number of walks in  $G$ , that is  $A^n(i, j)$  is the number of walks of length  $n$  from vertex  $i$  to vertex  $j$ .

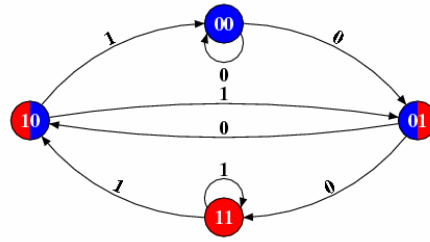
Let us now introduce the concept of de Bruijn graph [56] that is strongly related to the Isles of Eden digraphs. The vertices of a de Bruijn graph represent overlaps between sequences of symbols from an alphabet.

**Definition 4.6. De Bruijn graph**

Given an alphabet  $\Sigma = \{s_1, \dots, s_m\}$  an  $(m, n)$ -de Bruijn graph is a directed graph that has  $m^n$  vertices corresponding to all possible sequences  $x_1x_2\dots x_n$  of  $n$  symbols, where  $x_i \in \Sigma$ , for  $i \in \{1, \dots, n\}$ . The set of vertices is thus given by:

$$V = \left\{ \underbrace{(s_1, \dots, s_1, s_1)}_{n \text{ times}}, \underbrace{(s_1, \dots, s_1, s_2)}_{n \text{ times}}, \dots, \underbrace{(s_m, \dots, s_m, s_m)}_{n \text{ times}} \right\}. \tag{4.9}$$

There is an edge from a vertex  $x_1x_2\dots x_n$  to all vertices of the form  $x_2x_3\dots x_n\alpha$ ,  $\alpha \in \Sigma$ . There are  $k$  such nodes, thus each vertex has  $k$  incoming and  $k$  outgoing edges.



**Figure 4.2.** De Bruijn graph of strings of length two.

#### 4.4.2 The new construction: Isles of Eden digraph

##### **Definition 4.7.** Isles of Eden digraph (Double preimage locator)

Let us consider a directed graph  $G_N$  with nodes referring to pairs of binary strings of length two. There is an edge from node  $u$ , labeled  $(\{u_{1,x}, u_{2,x}\}, \{u_{1,y}, u_{2,y}\})$ , to node  $v$ , labeled  $(\{v_{1,x}, v_{2,x}\}, \{v_{1,y}, v_{2,y}\})$ , if  $v_{1,y} = u_{2,y}$ ,  $v_{2,y} = u_{1,y}$  and local rule  $N$  maps both  $\mathbf{x}^* = \{u_{1,x}, u_{2,x}, v_{2,x}\}$  and  $\mathbf{y}^* = \{u_{1,y}, u_{2,y}, v_{1,y}\}$  to the same output bit  $z$ , and the edge is labeled  $(v_{2,x}, v_{1,y}, z)$ . Let us call such a graph the *Isles of Eden digraph* of local rule  $N$ .

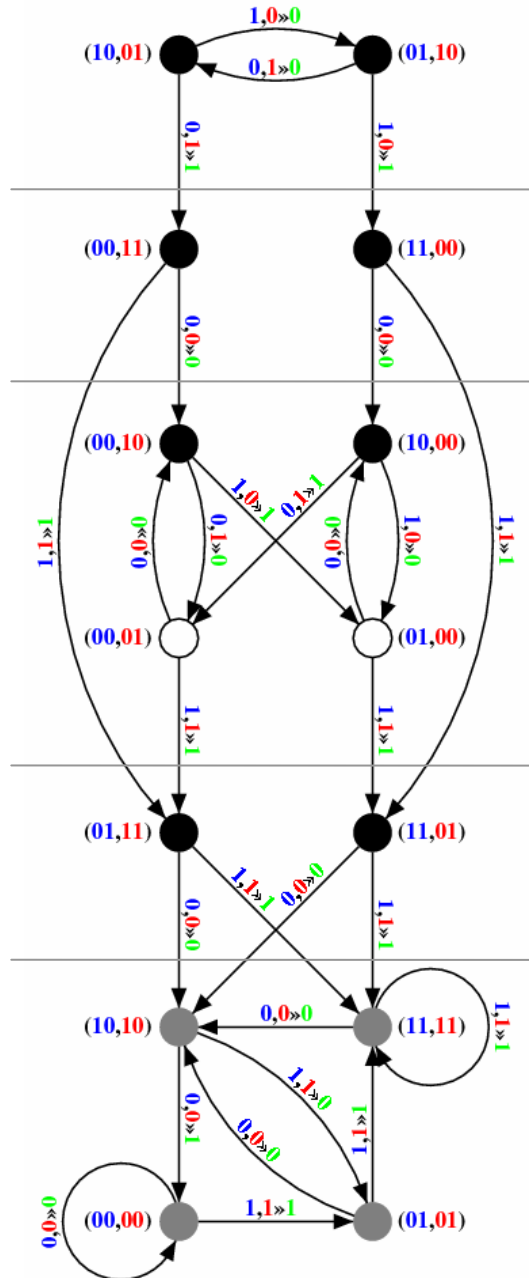
Normally it is enough to include nodes for which  $u_{1,x} \neq u_{1,y}$  (generator nodes) and those that can be reached from them. If all 16 possible nodes are included we call it the full Isles of Eden digraph (Section 4.4.4).

It is also called double preimage locator, because it is only possible to generate two strings that are mapped to the same output if there is string that has two distinct preimages, that is when there is a fork in the inverse map and therefore the map is not invertible.

As an example, Figure 4.3 shows the Isles of Eden digraph of rule 154. Colors of nodes refer to different properties: black nodes are generators, gray nodes are degenerate, and every other node is white. Bits of the two generated input strings are colored blue and red, respectively, whereas the bits of their image, the output string, are green.

The Isles of Eden digraph is basically the de Bruijn graph of the strings of length  $L$  with the nodes being the Cartesian product of the set of two bit strings.





**Figure 4.3.** *Isles of Eden digraph for local rule 154. The nodes shaded in gray are degenerate denoting that any path consisting solely of them can only generate equal strings. Horizontal gray lines partition the graph in a way that they cross only downward edges.*

A walk  $e_0, e_1, \dots, e_{n-1}$  in  $G_N$ ,  $e_i$  labeled  $(e_{i,x}, e_{i,y}, e_{i,z})$ , starting from a node  $u$ , labeled  $(\{u_{1,x}, u_{2,x}\}, \{u_{1,y}, u_{2,y}\})$  generate two input strings  $\mathbf{x}'$  and  $\mathbf{y}'$  and one output string  $\mathbf{z}'$ :

$$\begin{aligned}
\mathbf{x}' &= \{u_{1,x}, u_{2,x}, e_{0,x}, e_{1,x}, \dots, e_{n-1,x}\}, \\
\mathbf{y}' &= \{u_{1,y}, u_{2,y}, e_{0,y}, e_{1,y}, \dots, e_{n-1,y}\}, \\
\mathbf{z} &= \{e_{0,z}, e_{1,z}, \dots, e_{n-1,z}\},
\end{aligned} \tag{4.10}$$

for which local rule  $N$  maps both  $\{e_{i-2,x}, e_{i-1,x}, e_{i,x}\}$  and  $\{e_{i-2,y}, e_{i-1,y}, e_{i,y}\}$  to  $e_{i,z}$ . for all  $0 \leq i < n$ , where  $e_{-2,x}$ ,  $e_{-1,x}$ ,  $e_{-2,y}$ , and  $e_{-1,y}$  refer to  $u_{1,x}$ ,  $u_{2,x}$ ,  $u_{1,y}$ , and  $u_{2,y}$  respectively.

The cyclic boundary condition of the local cellular automaton rules makes interesting those walks that are closed (cycles), which implies  $n = L$ . The cyclic boundary condition can be formalized by the following equations:

$$\begin{aligned}
e_{-2,x} &\equiv u_{1,x} = e_{n-2,x}, \\
e_{-1,x} &\equiv u_{2,x} = e_{n-1,x}, \\
e_{-2,y} &\equiv u_{1,y} = e_{n-2,y}, \\
e_{-1,y} &\equiv u_{2,y} = e_{n-1,y},
\end{aligned} \tag{4.11}$$

and thus  $T_{\overline{N}}(\mathbf{x}) = \mathbf{z}$  and  $T_{\overline{N}}(\mathbf{y}) = \mathbf{z}$ , where

$$\begin{aligned}
\mathbf{x} &= \{e_{L-1,x}, e_{0,x}, e_{1,x}, \dots, e_{L-2,x}\}, \\
\mathbf{y} &= \{e_{L-1,y}, e_{0,y}, e_{1,y}, \dots, e_{L-2,y}\}, \\
\mathbf{z} &= \{e_{0,z}, e_{1,z}, \dots, e_{L-1,z}\}.
\end{aligned} \tag{4.12}$$

It follows that the initial node is labeled  $(\{e_{L-2,x}, e_{L-1,x}\}, \{e_{L-2,y}, e_{L-1,y}\})$ .

We can assume  $e_{L-2,x} \neq e_{L-2,y}$  because from Lemma 4.4 we know that if it is the  $n^{\text{th}}$  bit where  $\mathbf{x}$  and  $\mathbf{y}$  differ, then we can take  $\mathbf{x}^{(L-n)}$  and  $\mathbf{y}^{(L-n)}$  instead of  $\mathbf{x}$  and  $\mathbf{y}$ . Thus the first generated output bit is  $z_0$  and the last is  $z_{L-1}$ .

#### **Definition 4.8. Degenerate node and cycle**

A node  $u$  of the Isles of Eden digraph, labeled  $(\mathbf{u}_x, \mathbf{u}_y)$ , is called *degenerate* iff  $\mathbf{u}_x = \mathbf{u}_y$ . A cycle in the Isles of Eden digraph is called degenerate if all the nodes in it are degenerate.

### 4.4.3 *Constructing the Isles of Eden digraph*

We give a constructive method to create the Isles of Eden digraph for a given local rule

$N$ .

#### *Algorithm to construct the Isles of Eden digraph*

1. Add a node for every possible pair of 2-bit binary strings  $(x_{L-2}, x_{L-1}) - (y_{L-2}, y_{L-1})$  so that  $x_{L-2} \neq y_{L-2}$ , and label them accordingly. (There are 8 such nodes). These are the generator nodes, marked with black circles in the graph.

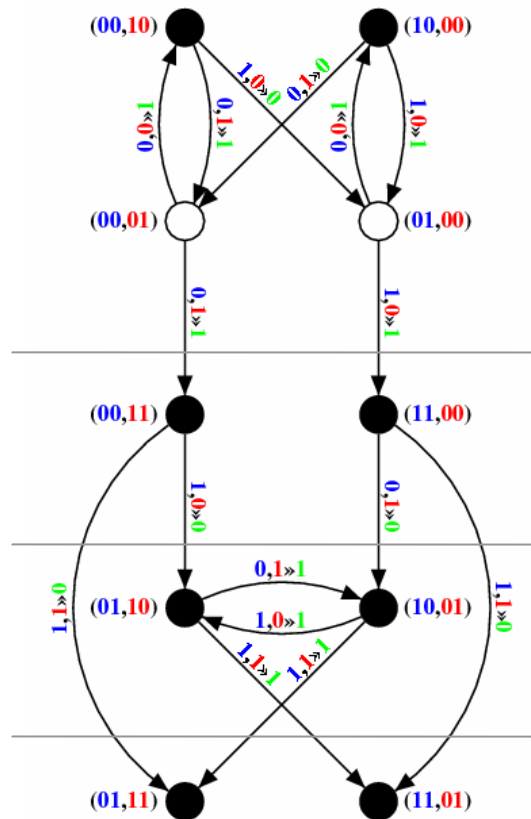
2. Now take a node, labeled  $(x_{i-1}, x_i) - (y_{i-1}, y_i)$ , and examine it to see if an additional pair of bits  $(x_{i+1}$  and  $y_{i+1})$  can be added so that  $(x_{i-1}, x_i, x_{i+1})$  and  $(y_{i-1}, y_i, y_{i+1})$  generate the same output. (We suppose that all previous output bits are equal, thus in this case  $\mathbf{x}$  and  $\mathbf{y}$  will also generate the same output, and to calculate the upcoming output bit the three input bits are available.)

For every possible pair add an edge with a label containing  $x_{i+1}$  and  $y_{i+1}$  as the first two bits, and the generated bit  $N(x_{i-1}, x_i, x_{i+1}) = N(y_{i-1}, y_i, y_{i+1})$  as the third bit. (Nodes with no appropriate continuation will have no outgoing edges, and thus will be dead ends.)

3. If there is a node labeled  $(x_i, x_{i+1}) - (y_i, y_{i+1})$ , then connect the given edge to it, if not, then create a new node for each such edge, and connect the edge to this new node.

4. If all nodes have been examined for possible continuations, then quit, otherwise pick a node and go to step 2.

**Example 4.4.1. Constructing the Isles of Eden digraph for Rule 45.**



**Figure 4.4. Isles of Eden digraph for local rule 45. Cycles in it refer to different inputs that map to the same output. Horizontal gray lines partition the graph in a way that they cross only downward edges.**

**Table XVI. Mapping rules of local rule 45.**

Input pattern	Output bit
000	1
001	0
010	1
011	1
100	0
101	1
110	0
111	0

Initial nodes to be added are (00,10), (00,11), (01,10), (01,11), and their mirrors: (10,00), (11,00), (10,01) and (11,01). Let us first examine node (00,10). According to the construction the label of a node contains the first two bits of the three bit strings for the next mapping. Thus 00 refers to the first two rows of Table 1, and can generate a 1 and a 0 with

continuations of 0 and 1, respectively. The substring 10 refers to rows 5 and 6 and can generate a 0 and a 1 with continuations of 0 and 1, respectively. Therefore we have to add two edges from this node referring to the two possible outputs.

One will be labeled (0,1,1), which means we concatenate a 0 to 00 and a 1 to 10 obtaining 000 and 101 respectively. Both generate an output 1 that is the third digit on the label of the edge. This edge will point to the node (00,01) that is obtained by taking the last two digits of 000 and 101, respectively. The other outgoing edge will be labeled (1,0,0). The first digit is concatenated to 00, giving 001; the second digit is concatenated to 10, giving 100. Both strings generate 0, the third digit of the label. The edge will point to node (01,00) that refers two the last two digits of 001 and 100, respectively.

The nodes (00,01) and (01,00) have to be added to the graph, because they are not present yet. This process has to be done for the newly created nodes, and repeated until every node added to the graph was examined.

The nodes (01,11) and its mirror (11,01) are interesting, because they have no possible continuation to generate the same output: rows 3 and 4 (having the prefix 01) both generate 1, whereas rows 7 and 8 (having the prefix 11) both generate 0.

***Example 4.4.2. Constructing the Isles of Eden digraph for Rule 154.***

This example is a bit more complicated since it includes degenerate nodes as well. We used the Isles of Eden digraph for Rule 154 as the sample digraph at the definition, shown in Figure 4.3.

***Table XVII. Mapping rules of local rule 154.***

<b>Input pattern</b>	<b>Output bit</b>
000	0
001	1
010	0
011	1
100	1
101	0
110	0
111	1

Initial nodes to be added are (00,10), (00,11), (01,10), (01,11), and their mirrors: (10,00), (11,00), (10,01) and (11,01). Let us first examine node (10,01). According to the construction the label of a node contains the first two bits of the three bit strings for the next mapping.

Thus the substring 10 refers to the rows 5 and 6 of Table 2, and can generate a 1 and a 0 with continuations of 0 and 1, respectively. The substring 01 refers to rows 3 and 4 and can generate a 0 and a 1 with continuations of 0 and 1, respectively. Therefore we have to add two edges from this node referring to the two possible outputs.

One will be labeled (0,1,1), which means we concatenate a 0 to 10 and a 1 to 01 obtaining 100 and 011 respectively. Both generate an output 1 that is the third digit on the label of the edge. This edge will point to the node (00,11) that is obtained by taking the last two digits of 100 and 011, respectively. The other outgoing edge will be labeled (1,0,0). The first digit is concatenated to 10, giving 101; the second digit is concatenated to 01, giving 010. Both strings generate 0, the third digit of the label. The edge will point to node (01,10) that refers to the last two digits of 001 and 100, respectively.

The nodes (00,11) and (01,10) have to be added to the graph, because they are not present yet. This process has to be done for the newly created nodes, and repeated until every node added to the graph was examined.

### Some comments

Note that both graphs are structured to have the mirrored states next to each other, except for the degenerate (self symmetric) ones. Also note that the statement of Lemma 4.4 means that a closed walk on a cycle can be started from any of its nodes, and the corresponding input and output strings can be transformed to each other by circular shifting of the same length.

#### 4.4.4 Full Isles of Eden digraph

The number of nodes in the Isles of Eden digraph is between 8 and 16. The lower limit is due to the fact that 8 nodes are always added in the first step of the algorithm. The upper bound is given by the number of possible pairs of bit-strings of length 2 that is equal to  $(2^2)^2 = 16$ . The reason for not including some nodes for certain local rules is that they have no incoming edges and thus they cannot be part of a cycle in the Isles of Eden digraph. Nevertheless, we may include all 16 nodes in any Isles of Eden digraph, the number of non-degenerate cycles will not change. This digraph is called the full Isles of Eden digraph  $\mathcal{G}_{IE}^F(\boxed{N})$  of a given local rule  $\boxed{N}$ . It can be constructed using the following algorithm:

***Algorithm to construct the full Isles of Eden digraph***

1. Add a node for every possible pair of 2-bit binary strings and label them accordingly. (There are 16 such nodes.)
  
2. Now take a node, labeled  $(x_{i-1}, x_i) - (y_{i-1}, y_i)$ , and examine it to see if an additional pair of bits  $(x_{i+1}$  and  $y_{i+1})$  can be added so that  $(x_{i-1}, x_i, x_{i+1}) - (y_{i-1}, y_i, y_{i+1})$  generate the same output. (We suppose that all previous output bits are equal, thus in this case  $\mathbf{x}$  and  $\mathbf{y}$  will also generate the same output, and to calculate the upcoming output bit the three input bits are available.) For every possible pair add an edge from this node to the node labeled  $(x_i, x_{i+1}) - (y_i, y_{i+1})$ , with a label containing  $x_{i+1}$  and  $y_{i+1}$  as the first two bits, and the generated bit  $N(x_{i-1}, x_i, x_{i+1}) = N(y_{i-1}, y_i, y_{i+1})$  as the third bit. (Nodes with no appropriate continuation will have no outgoing edges, and thus will be dead ends.)
  
3. If all nodes have been examined for possible continuations, then quit, otherwise pick a node and go to step 2.

Since the full Isles of Eden digraph contains all 16 possible nodes, it also contains all 4 degenerate ones. Note that the degenerate subgraph always contains the same edges, since an edge from a degenerate node to a (not necessarily different) degenerate node refers to the same three bit pattern for both  $\mathbf{x}$  and  $\mathbf{y}$ , and therefore to the same output bit. These edges are invariant for the full Isles of Eden digraphs of all 256 local rules. There are 8 of them, 2 per each degenerate node.

***4.4.5 Effect of global equivalence transformations on Isles of Eden digraphs***

Based on the three global equivalence transformations, appropriate transformations can be established for the Isles of Eden digraphs of globally equivalent rules:

1. *left-right transformation*:  $\mathcal{G}_{IE}^\dagger(\boxed{N}) = \mathcal{G}_{IE}(T^\dagger(\boxed{N}))$
2. *global complementation*:  $\overline{\mathcal{G}_{IE}}(\boxed{N}) = \mathcal{G}_{IE}(\overline{T}(\boxed{N}))$
3. *left-right complementation*:  $\mathcal{G}_{IE}^*(\boxed{N}) = \mathcal{G}_{IE}(T^*(\boxed{N}))$
4. *alternating transformation*:  $\tilde{\mathcal{G}}_{IE}(\boxed{N}) = \mathcal{G}_{IE}(\tilde{T}(\boxed{N}))$

The global complementation on  $\mathcal{G}_{IE}(\boxed{N})$  can be easily characterized for any local rule  $\boxed{N}$ .  $\overline{\mathcal{G}_{IE}}(\boxed{N})$  can be generated from  $\mathcal{G}_{IE}(\boxed{N})$  by inverting all bits in the labels of the nodes and the edges. Naturally,  $\mathcal{G}_{IE}^*(\boxed{N})$  can be generated from  $\mathcal{G}_{IE}^\dagger(\boxed{N})$  in the same way. The alternating transformation is even simpler: to compute  $\tilde{\mathcal{G}}_{IE}(\boxed{N})$  from  $\mathcal{G}_{IE}(\boxed{N})$ , only the output bits have to be inverted.

Characterization of the left-right transformation is more complicated. 20 out of the 64 possible edges are not affected by it, whereas changes regarding the rest of the edges depend on the given rule.<sup>3</sup>

#### 4.5 Detecting isles of Eden with Isles of Eden digraph

As we have seen, in order to generate an output with correct cyclic boundary conditions we need to return to the starting node. It follows that for every input pair  $(\mathbf{x}, \mathbf{y}) \in \Sigma^L$  that can generate the same output string  $\mathbf{z} \in \Sigma^L$  and for which  $\mathbf{x} \neq \mathbf{y}$ , there is a cycle of length  $L$  in the Isles of Eden digraph, and every cycle of length  $L$  in the graph represents such an input pair  $(\mathbf{x}, \mathbf{y}) \in \Sigma^L$ . Thus the length of the input strings equals to the length of the cycles, and all lengths for which there is no cycle in the graph will give a contradiction. The following theorem is a joint result of this observation and Lemma 4.2:

---

<sup>3</sup> 8 out of the 20 are the invariant edges in the degenerate subgraph. The remaining 12 edges start from the 12 non-degenerate nodes of  $\mathcal{G}_{IE}^F(\boxed{N})$ .



**Theorem 4.1.**

Every orbit of a local rule  $\boxed{N}$  over  $\Sigma^L$  is an *isle of Eden* if and only if  $\mathcal{G}_{IE}(\boxed{N})$  has no non-degenerate cycle of length  $L$ .

*Proof*

$\Rightarrow$  Let us suppose indirectly that every orbit of a local rule  $\boxed{N}$  over  $\Sigma^L$  is an isle of Eden, but there is a non-degenerate cycle of length  $L$  in  $\mathcal{G}_{IE}(\boxed{N})$ . The input strings  $\mathbf{x} \in \Sigma^L$  and  $\mathbf{y} \in \Sigma^L$  generated by this cycle have the same image  $\mathbf{z} \in \Sigma^L$  under  $T_{\boxed{N}}$ , but they are different because it has a node labeled  $(\mathbf{u}_x, \mathbf{u}_y)$  with  $\mathbf{u}_x$  and  $\mathbf{u}_y$  being different substrings of  $\mathbf{x}$  and  $\mathbf{y}$ , respectively, starting at the same index. Therefore  $\mathbf{z}$  has two different preimages under  $T_{\boxed{N}}$ . This contradicts the fact that every  $\mathbf{z} \in \Sigma^L$  has a unique preimage under  $T_{\boxed{N}}$  that follows from Lemma 4.2, since we assumed that every orbit of a local rule  $\boxed{N}$  over  $\Sigma^L$  is an *isle of Eden*.

$\Leftarrow$  Since there is no non-degenerate cycle of length  $L$  in  $\mathcal{G}_{IE}(\boxed{N})$ , thus there exist no two different strings that are mapped to the same  $\mathbf{z} \in \Sigma^L$ . Therefore every  $\mathbf{z} \in \Sigma^L$  has at most one preimage under  $T_{\boxed{N}}$ , and using Lemma 4.3 and Lemma 4.2 respectively, it follows that every orbit of a local rule  $\boxed{N}$  over  $\Sigma^L$  is an *Isle of Eden*. ■

## 4.6 Invertibility for infinitely many string lengths

By calculating the images for all rules and for bit-lengths  $2 < L < 20$  I have numerically determined all cases where the rule is invertible, i.e. all preimages are unique. From the table one could easily see that out of the 256 rules there are only 16 rules in case of which there are more than one string lengths for which all preimages are unique. String lengths for which these rules are not invertible either do not exist (the property holds for all string lengths of a certain rule), or occur periodically with a period of 2 or 3. The rules are shown in Table XVIII. Some of them belong to same equivalence classes [58], which are grouped into lines, i.e. every row refers to a single equivalence class. Those rules for which non-invertible string lengths do not exist are shown with an infinite period. It is worth to mention that the local

mapping function of these rules only depend on one of their inputs. Therefore they can be treated as trivially invertible cases.

Of course, except for the trivially invertible cases, the periodicity for this finite interval does not imply that it holds for all integers. To prove this we have to examine the Isles of Eden digraphs of these rules. The following sections contain the graphs and the attached proofs for the non-trivially invertible classes. The proofs are based on showing that the graphs do not contain non-degenerate cycles of lengths non-divisible by the period.

**Table XVIII. Rules that are invertible for more than one string length for  $2 < I < 20$ . Rules belonging to the same equivalence class are displayed on the same line.**

Period of non-invertible lengths	Rule numbers
2	45, 75, 89, 101 154, 166, 180, 210
3	105 150
$\infty$	15, 85 51 170, 240 204

Finding cycles in directed graphs is generally a difficult task, but by topologically ordering the nodes it can be made much easier, if most edges move to the same direction. We organized the Isles of Eden digraphs in Figure 4.3 and Figure 4.4 so that most of the edges point downwards. It is trivial that a cycle has to contain edges of other directions (horizontal or upward) too. Some cuts are drawn on the graphs that have only downward crossing edges, therefore no cycles can contain nodes from the two sides of any such a cut. Therefore these cuts partition the graph to subgraphs, and we only need to examine these subgraphs for cycles. We will use this observation to find all cycles of a Isles of Eden digraph.

It is enough to look for simple cycles (all nodes are different except the starting and ending node), because other cycles can be decomposed to single cycles, therefore their lengths are the sums of the simple cycles they are composed of.

### 4.6.1 Class “Period 2”

The 8 rules in this class belong to two global equivalence classes, so it is enough to consider one from each. Theorems 4.2 and 4.3 deal with rule 45 and 154, respectively.

#### **Theorem 4.2.**

Every orbit of Rule 45 over  $\Sigma^L$  is an isle of Eden if and only if  $L$  is an odd number.

#### *Proof*

Figure 4.4 shows the Isles of Eden digraph for Rule 45. We will see that for any positive even number there is a cycle in  $\mathcal{G}_{IE}(\boxed{45})$  with this length. From this, using Theorem 4.1, it follows that if  $L$  is an even number, then there is a cycle of length  $L$  in the graph and thus there is an orbit of Rule 45 over  $\Sigma^L$  that is not an Isle of Eden. Then we will prove that all cycles in the graph are of even length, therefore if  $L$  is an odd number, then there are no cycles of length  $L$ , and using Theorem 4.1 we can conclude that every orbit of Rule 45 over  $\Sigma^L$  is an Isle of Eden.

$\Rightarrow$  For every length  $L = 2n$  there is a cycle in  $\mathcal{G}_{IE}(\boxed{45})$ , e.g. starting from (01,10) and taking the cycle of length 2 through (10,01) back to (01,10)  $n$  times. These cycles refer to the strings  $(01)^n$  and  $(10)^n$  and both map to  $(11)^n$ .

$\Leftarrow$  The upper four nodes form a bipartite subgraph, within which there are only cycles of even length. If the starting state is not in the upper four nodes, then the only possibility for a cycle is between node (01,10) and (10,01), because the rest of the subgraphs are not even connected. This is also a bipartite subgraph, so this allows only for cycles of even length too. ■

#### *Another proof*

If we determine the adjacency matrix  $A_{45}$  of the graph in Figure 4.4, the number of cycles of length  $n$  can be easily determined. There are no degenerate nodes in  $\mathcal{G}_{IE}(\boxed{45})$ , so all cycles are non-degenerate. Since the number of cycles of length  $n$  starting at node  $i$  is  $A_{45}^n(i, i)$ , thus the total number of cycles of length  $n$  is  $\text{Tr}(A_{45}^n)$ , the trace of  $A_{45}^n$ .  $\text{Tr}(A_{45}^n)$  is equal to the sum of the eigenvalues of  $A_{45}^n$ , which allows us to compute it for any  $n$  without actually computing  $A_{45}^n$ . The adjacency matrix for rule 45 is the following:

$$A_{45} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The eigenvalues of  $A_{45}$  can be computed from its characteristic polynomial:

$$x^6(x^4 - 3x^2 + 2) = 0 \quad (4.13)$$

The nonzero eigenvalues are the following:  $\lambda_{1,2} = \pm\sqrt{2}$ ,  $\lambda_{3,4} = \pm 1$ . Note that the eigenvalues of  $A_{45}$  are real and sign symmetric, and hence their sum is zero. Since the eigenvalues of  $A_{45}^n$  are  $(\lambda_i)^n$ , they are also sign symmetric for odd  $n$ , whereas they are positive integers for even  $n$ . Therefore

$$\begin{aligned} \text{Tr}(A) &= \sum (\lambda_i)^n > 0, \text{ for } n = 2k \\ \text{Tr}(A) &= \sum (\lambda_i)^n = 0, \text{ for } n = 2k + 1 \end{aligned} \quad (4.14)$$

where  $k \in \mathbb{N}$ . This is what we wanted to prove. ■

**Table XIX. Examples of cycles of even length for Rule 45.**

<b>L</b>	<b>Route</b>	<b>Input strings</b>	<b>Output string</b>
4	(01,10)- (10,01)- (01,10)- (10,01)- (01,10)	0101 1010	1111
4	(00,01)- (00,10)- (01,00)- (10,00)- (00,01)	0100 0001	0101
6	(01,10)- (10,01)- (01,10)- (10,01)- (01,10)- (10,01)- (01,10)	010101 101010	111111
6	(00,10)- (00,01)- (00,10)- (00,01)- (00,10)- (00,01)- (00,10)	000000 101010	111111
8	(01,10)- (10,01)- (01,10)- (10,01)- (01,10)- (10,01)- (01,10)- (10,01)- (01,10)	01010101 10101010	11111111

**Theorem 4.3.**

Every orbit of Rule 154 over  $\Sigma^L$  is an isle of Eden if and only if  $L$  is an odd number.

*Proof*

Figure 4.3 shows the Isles of Eden digraph for Rule 154. Similarly to the proof of Theorem 4.2 we only have to show that for any positive even number there is a cycle in the Isles of Eden digraph of this length touching a non-degenerate node and that all cycles in the graph touching a non-degenerate node are of even length.

$\Rightarrow$  For every length  $L = 2n$  there is a cycle in  $\mathcal{G}_{IE}(\boxed{154})$ , e.g. starting from  $(01,10)$  and taking the cycle of length 2 through  $(10,01)$  back to  $(01,10)$   $n$  times. None of these nodes are degenerate. These cycles refer to the strings  $(01)^n$  and  $(10)^n$  and both map to  $(00)^n$ .

$\Leftarrow$  Let us take the indicated subgraphs of  $\mathcal{G}_{IE}(\boxed{154})$  in a top-down order. The upper two nodes form a bipartite subgraph, within which there are only cycles of even length. The next subgraph (nodes  $(00,11)$  and  $(11,00)$ ) is not connected. The next subgraph of four nodes is also a bipartite subgraph, thus it contains only cycles of even length. The next subgraph (nodes  $(01,11)$  and  $(11,01)$ ) is not connected again. Although the bottommost subgraph contains cycles of odd length, these cycles include only degenerate nodes. If the starting node is in this subgraph, then the two input strings will be equal since the pairs of these four nodes contain equal strings, and also, all edges have equal digits for the two input strings. Therefore it does not contain any cycles corresponding to different input strings. ■

*Another proof*

Since  $\mathcal{G}_{IE}(\boxed{154})$  contains degenerate nodes, the number of total cycles is higher than the number of non-degenerate cycles. Thus we cannot use the trace of its adjacency matrix  $A_{154}$ . But since degenerate nodes in  $\mathcal{G}_{IE}(\boxed{154})$  do not have outgoing edges going to a non-degenerate node, therefore no non-degenerate cycles can include degenerate nodes, and we

only need to consider the adjacency matrix of the non-degenerate subgraph of  $\mathcal{G}_{AE}(\boxed{154})$ .<sup>4</sup>

Let  $\hat{A}_{154}$  denote the adjacency matrix of the non-degenerate subgraph of  $A_{154}$ . Computing  $\hat{A}_{154}$  gives

$$\hat{A}_{154} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Although the non-degenerate subgraph of  $\mathcal{G}_{AE}(\boxed{154})$  is not isomorphic to  $\mathcal{G}_{AE}(\boxed{45})$ , but their characteristic polynomials are the same. Therefore their spectra are equal, and  $\hat{A}_{154}$  has the same nonzero eigenvalues as  $A_{45}$ . Therefore the rest of the proof is the same as rule 45. ■

### 4.6.2 Class “Period 3”

#### Theorem 4.4

Every bit string of rules  $\boxed{150}$  and  $\boxed{105}$  is an *Isle of Eden* if, and only if,  $\frac{L}{3}$  is not an integer.

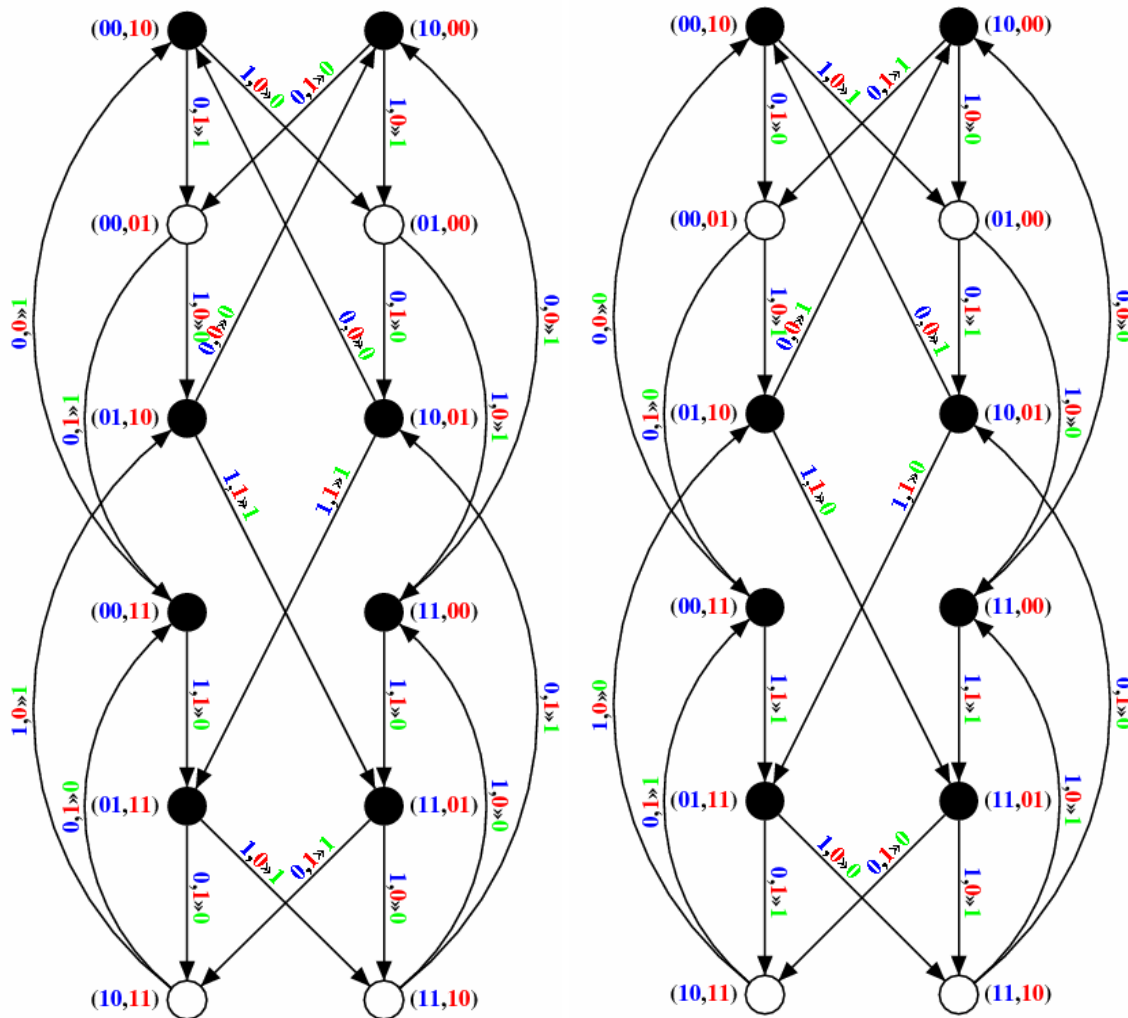
*Proof*<sup>5</sup>

At first we construct the Isles of Eden digraphs for rules  $\boxed{150}$  and  $\boxed{105}$ . These are shown in Figure 4.5.

---

<sup>4</sup> The same reasoning also works if the degenerate subgraph is connected to the non-degenerate subgraph only by outgoing edges.

<sup>5</sup> This theorem is proven in [60] using circular matrices.



*Figure 4.5. Isles of Eden digraph of local rule 105 (left) and 150 (right).*

Since rule  $\boxed{105}$  and  $\boxed{150}$  are the alternate transforms from each other, therefore  $\mathcal{G}_{IE}(\boxed{105})$  and  $\mathcal{G}_{IE}(\boxed{150})$  are very similar. Observe that only the green output bits are inverted. Since the output bit does not affect the topology of the digraph, their adjacency matrixes are equal:



$$A_{105} = A_{150} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The characteristic polynomial of the matrix is

$$x^9(x^3 - 8) = 0 \quad (4.15)$$

The nonzero eigenvalues are the following:  $\lambda_1 = 2$ ,  $\lambda_{2,3} = -1 \pm i\sqrt{3}$ . Note that the nonzero eigenvalues are the complex cubic roots of 8, that is vectors of length 2 and argument 0,  $\frac{2\pi}{3}$ , and  $\frac{4\pi}{3}$ , respectively. Therefore the following statements hold for the complex eigenvalues and  $k \in \mathbb{N}$ :

$$(\lambda_2)^{3k+1} = \frac{1}{2}(\lambda_3)^{3k+2} = 8^k \lambda_2 \quad (4.16)$$

and

$$(\lambda_3)^{3k+1} = \frac{1}{2}(\lambda_2)^{3k+2} = 8^k \lambda_3 \quad (4.17)$$

The sum of the eigenvalues is zero, and due to Equations (4.15) and (4.16) the sum of powers of the eigenvalues is also zero for every integer not divisible by 3, whereas they are positive integers for  $n = 3k$ , because  $(\lambda_i)^{3k} = 8^k$  for  $i = 1, 2, 3$ . Therefore

$$\begin{aligned} \text{Tr}(A_{105}) &= \sum (\lambda_i)^n > 0, \text{ for } n = 3k \\ \text{Tr}(A_{105}) &= \sum (\lambda_i)^n = 0, \text{ for } n = 3k + 1 \text{ and } n = 3k + 2 \end{aligned} \quad (4.18)$$

where  $k \in \mathbb{N}$ . Since  $A_{105} = A_{150}$ , it follows that all cycles in  $\mathcal{G}_{IE}(\boxed{105})$  and  $\mathcal{G}_{IE}(\boxed{150})$  are of lengths divisible by 3. The statement of the theorem now directly follows from Theorem 4.1.

■

## 4.7 Conclusions

I dealt with the invertibility of elementary CAs, and I showed that invertibility for infinitely many string lengths occurs also for many non-trivial cases. I defined the Isles of Eden digraph that allows one to quickly check if the given CA is invertible for a particular length and proved the equivalence of the non-existence of a non-degenerate cycle of length  $n$  and the invertibility of the CA for length  $n$ . Using graph cycle counting algorithms, and bipartiteness of subgraphs, as well as spectral analysis of graphs I proved that all rules that exhibit periodic invertibility for small string lengths retain this property for every string length.

Analyzing invertibility of cellular architectures is an emerging research field due to its connection to computational models and simulating interactions of many particle physical systems. Indeed, it is proven that there exist computationally universal invertible cellular automata [61]. There are physical systems that are in fact can modeled by invertible cellular automata, including lattice-gas model of fluid dynamics [62] and quantum computing [63].

## 5 Summary

### 5.1 Methods of Investigation

Most of my research is directly connected to the Cellular Nonlinear/Neural Network (CNN), the CNN Universal Machine (CNN-UM) and the Cellular Wave Computing paradigm. The CNN templates developed were designed using analytical methods published in the literature ([18],[19]). I relied on the concept of the CNN-UM being a Universal Machine on Flows (UMF) [20] for the design of cellular wave algorithms. Besides the templates designed by myself I utilized many standard template classes [12], including ones that implement morphological and other image processing operators.

A key method I have developed is semantic embedding. I applied it to in spatial and multimodal spatial-temporal detection tasks to find topographic features by giving semantic description of sample input images and video flows based on structural scene analysis. I validated the feature detection algorithms on standard and self made test sets.

I used standard document scanners to acquire handwritten texts. Experiments regarding pattern detection in 2D video flows were performed on a Blind Acquired Visual Flow Database containing recordings taken with commercial cell phones with built-in cameras and compact digital cameras by blind persons in real world situations.

For algorithm development I used the software package named Aladdin developed by Analogic Computers Ltd [21], the ACE-16k cellular visual microprocessor [22] and the Matlab software environment [23] with the MatCNN toolbox [64].

For expression level language modeling I used bigrams, as a simple, non-semantic model, and I relied on statistical methods for probability estimations.

In the field of cellular automata I relied on results in theory of nonlinear dynamics, graph theory, symbolic dynamics, and especially on new results of L. O. Chua in qualitative theory of binary Cellular Nonlinear Networks [60].

## 5.2 New Scientific Results

### *Thesis I. Analogic algorithms with spatial semantic embedding for handwritten text recognition*

Great variety of handwriting styles makes it hard to recognize handwritten texts by machines in general. I created a handwritten text recognition system that mimics the human reading process by incorporating cellular wave algorithms as a model of perception and integrating the use of linguistic knowledge into the recognition process. The system realizes lexicon reduction with a very high reduction rate (>99.9%) and with a coverage over 80%, which is comparable to previous results in the literature.

#### **I.1. I developed a method called shape coding to embed the recognition of 2D morphological shape features into a semantic environment.**

Making use of linguistic information can greatly improve the performance of recognition systems, but in the traditional way of using it for post-processing satisfactory results could not be achieved. Shape coding enables embedding linguistic knowledge into the recognition process without actually recognizing the letters or the word.

Advantages of this approach are twofold. On one hand it overcomes the problem of the mutual dependence of segmentation and recognition of letters (Sayre paradox [34]). On the other hand linguistic knowledge can influence the relevancy and importance of geometric features.

#### **I.2. I determined six holistic features detectable on cellular wave computers and implemented several feature extraction and feature classification analogic algorithms**

Holistic features are primitives of letters and they are detected on the word image without the need to segment it into letters. The six features are as follows: holes, ascenders, descenders, junction points, hills and valleys. Some features are classified into several feature classes based on their size, shape and/or position. Some characteristic feature maps are shown in Figure 5.1.



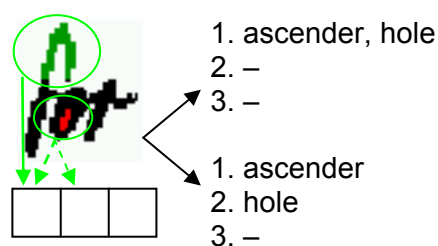
**Figure 5.1. Different classes of (a) holes, (b) ascenders and descenders**

**I.3. I determined general parameters characteristic to the writing style and I gave methods to adaptively compute them**

Identified parameters include distance of baselines of the writing, thresholds for minimum ascender and descender height, size intervals for classification of holes, and average letter width. The methods developed are based on my experiments and are realized using cellular wave algorithms.

**I.4. I created a method to map topographic features detected at the word level to possible letter positions**

A topographic feature detected can belong to multiple positions in a word, therefore the mapping is ambiguous. Figure 5.2 shows a simple mapping problem. The method gives all possible mappings obeying the following geometric constraints: slant of the writing, horizontal order of features and vertical coupling.



**Figure 5.2. Mapping of topographic features is ambiguous, the hole in letter ‘o’ can belong to two distinct positions, doubling the number of possible mappings.**

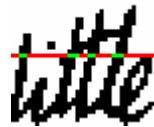
**I.5. I created a method to estimate the number of characters in a handwritten word image using horizontal connected components as a quasi-independent measure**

I experimentally proved that, considering a handwritten cursive word image, the number of horizontal connected components and the number of letters of the word has a strong correlation and their ratio is independent of the writing style. The new method can estimate

the number of letters in a word more accurately than an estimation based on the pixel length of the word, meanwhile the input it relies on can be computed with a single cellular wave instruction, the **HCCD** template.

**I.6. I developed a novel upper baseline detection algorithm that gives robust results even for words with a high number of ascenders**

Detecting the upper baseline properly is very important to enable accurate feature detection. I gave a cellular wave algorithm for upper baseline detection that uses connected component detector to locate a horizontal baseline on a skew-corrected word image. Detection result is shown in Figure 5.3 for a sample word. I showed that restricting the upper baseline to be horizontal does not decrease the accuracy of feature detection, and that one free parameter can be determined more robustly than two.



*Figure 5.3. Sample word image with computed horizontal upper baseline.*

## ***Thesis II. Detection of dynamic events and specific 2D patterns in saccadic and noisy visual flows recorded by a moving, blind platform***

Extracting and reading signs from video flows recorded by a moving camera in real-world situations is a very complex task motivated by the way humans find and read signs and use it for orientation. I gave algorithms to perform sign detection in specific situations that can be described by pre-defined semantics and to recognize numbers in them. The algorithms have been tested and validated on recordings from the Blind Acquired Visual Flow Database, containing more than an hour of recordings and over 100 dynamic events.

### **II.1. I defined a general semantic framework for hierarchical processing of multimodal sensory information with embedded semantics**

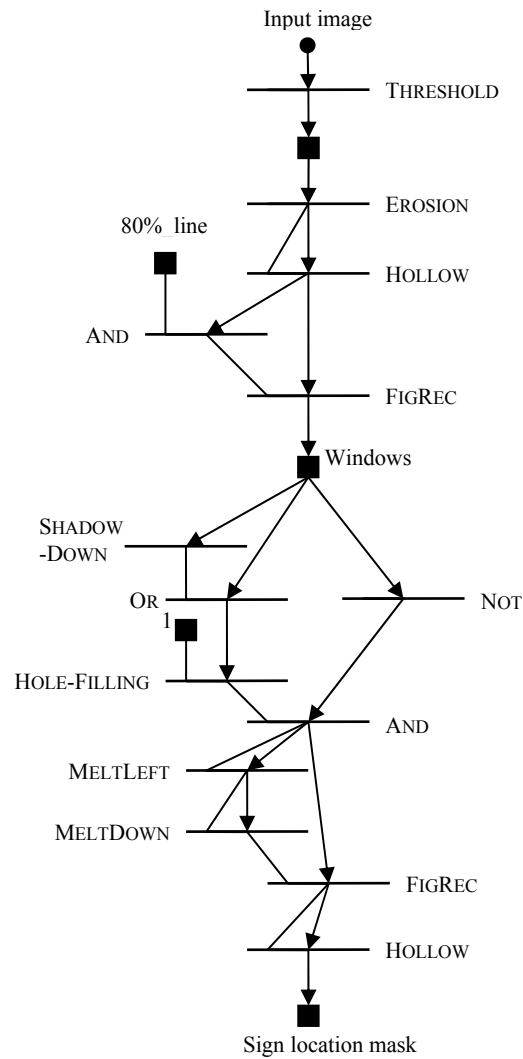
I created a framework for hierarchical processing of information coming from multimodal, topographic sensors and integrating semantics to control sensing-processing. I defined the following abstraction levels in the hierarchy: features, symptoms, events, situations. Events are recorded in an Event Register that, together with active situations and a priori goals, influences the attention mechanism. Attention Director controls actuators to interact with the environment and sensors to facilitate better sensing with regard to current situations and events.

### **II.2. I gave analogic cellular wave algorithms to detect and validate route number signs on public transport vehicles in a 2D visual flow**

The video flows recorded by mobile devices are of low-quality and some control is needed to direct the camera towards the sign. The algorithm I created detects possible sign candidates and validates them by checking if they contain large figures. UMF diagram of the detection algorithm is shown in Figure 5.5. Detection is shown in Figure 5.4 for a sample frame. Valid signs are tracked through frames, and an enhanced image is produced by superposing sign images extracted from consecutive frames on one another to allow for better recognition results.



**Figure 5.4. Sign localization on a tram. (a) Original input frame (b) Sign location on binarized input**



**Figure 5.5. UMF diagram of the algorithm locating signs with a white background.**



---

### II.3. I gave a topographic feature detection based method to recognize numbers in signs extracted from 2D visual flows

Recognition is based on topographic feature detection by cellular wave algorithms. I use holes and straight lines as features and a box model is used in addition to allow the detection of open semi-holes in the shape of numbers. Sample topographic feature maps are shown in Figure 5.6.



*Figure 5.6. Sample feature maps. Right open holes and their auxiliary lines are shown in cyan, middle vertical line is shown in blue, and upper round hole is shown in red.*

### ***Thesis III. Invertibility of one dimensional cellular wave computers***

Analysis of invertibility of dynamical systems is of great importance. Based on the concept of isle of Eden, introduced by L. O. Chua, I developed a new construction for detecting merging points in binary cellular nonlinear networks (cellular automata) trajectories using a graph-theoretical approach. *Isles of Eden* are orbits with every state of it being the unique preimage of itself under the  $n^{\text{th}}$  iterated global map. I proved a general theorem connecting a well-defined set of cycles of the digraph to isles of Eden, and analyzed all 256 elementary cellular automata for non-trivial invertibility.

**III.1. I defined a digraph for analyzing pairs of input patterns in elementary CAs, called the Isles of Eden digraph, and I gave an algorithm to construct it for any elementary CA. I proved a general theorem stating that there are no degenerate cycles of length  $L$  in the Isles of Eden digraph of a given CA if and only if all of its orbits are isles of Eden, which is equivalent to the invertibility of the CA.**

The Isles of Eden digraph is a de Bruijn graph of pairs of two bit binary patterns, and its cycles of length  $L$  correspond to two binary strings of length  $L$  that are mapped to the same output. If the strings are equal the cycle is called degenerate. Non-degenerate cycles refer to different strings, referring to a merging point in the trajectory. To prove the theorem I proved through some auxiliary lemmas that for a given CA and a given pattern length the following statements are equivalent:

1. All cycles of this length are degenerate
2. Global mapping is surjective (there is no Garden of Eden)
3. Global mapping is injective (there is no merging point in the trajectory)
4. All orbits are isles of Eden
5. The cellular automaton is invertible

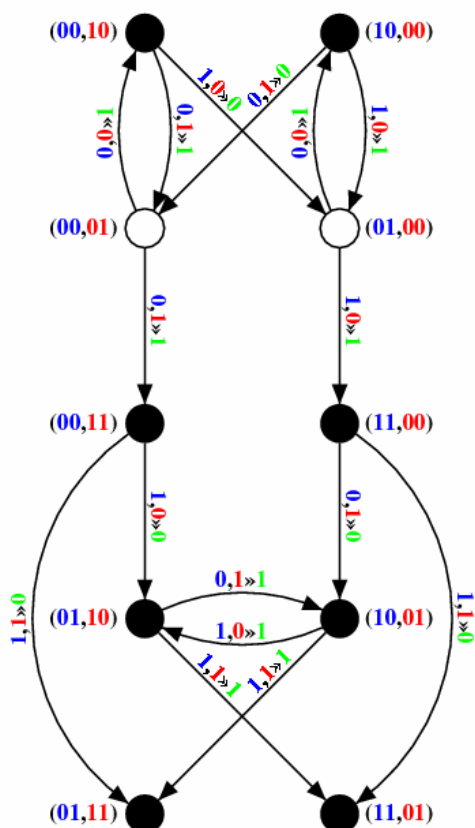
**III.2. I determined all elementary cellular automata that are invertible for an infinite number of pattern lengths, and proved that non-invertible pattern lengths occur periodically.**

I performed numerical simulations on the space of elementary cellular automata to analyze the surjectivity of the global map for all pattern lengths less than 20. Based on the results I determined candidates that might be invertible for infinitely many pattern lengths. I

analyzed their Isles of Eden digraphs, and proved that non-invertible pattern lengths occur periodically for all of them (Table XX). Figure 5.7 shows a sample Isles of Eden digraph for rule 45.

**Table XX.** CA rules that are invertible for infinitely many pattern lengths.

Period of non-invertible states	Rule numbers
2	45, 75, 89, 101 154, 166, 180, 210
3	105 150
$\infty$	15, 85 51 170, 240 204



**Figure 5.7.** Isles of Eden digraph for local rule 45. All cycles in it are of even length, because their containing subgraphs are bipartite.

### 5.3 Application areas of the results

Most of the research I did is directly motivated by applications. The advantage of the methods I developed for handwriting recognition is that they are general purpose algorithms, and they are not restricted to small dictionaries. Possible applications include processing of personal notes and official forms.

The semantic embedding framework can be used in a wide area, for any multimodal, multi-sensor information processing task. Of course, the general framework should be adapted and sophisticated according to specific aspects of the given problem.

Route number localization and recognition algorithms can be utilized in a portable device like the Bionic Eyeglass at bus and tram stops to provide help in finding the right vehicle for a visually impaired person. They can also serve as a basis for a more general Blind Mobile Navigation framework capable of high-level object detection and recognition. Such a framework can not only be used by a visually impaired person, but also in autonomous robots designed to operate in hazardous environments to facilitate their automatic navigation.

Results on the invertibility of elementary cellular automata can be used for further theoretical investigations in the field, and they can also serve as a basis of a more general research on new computational models using quantum computers and the relation of physics and computation. Moreover it can be used to investigate methods for detecting real world patterns that are easy to generate but difficult to detect.

## References

### *The Author's Journal Papers*

- [1] **K. Karacs**, G. Prószéky, and T. Roska, “CNN algorithms with spatial semantic embedding for handwritten text recognition,” *International Journal of Circuit Theory and Applications*, to be published
- [2] L. O. Chua, **K. Karacs**, V. I. Sbitnev, J. Guan, and J. Shin, “A Nonlinear Dynamics Perspective of Wolfram’s New Kind Of Science. Part VIII: More Isles of Eden,” *International Journal of Bifurcation and Chaos*, to be published
- [3] G. Tímár, **K. Karacs**, and Cs. Rekeczky, “Analogic Preprocessing and Segmentation Algorithms for Off-line Handwriting Recognition,” *Journal of Circuits, Systems and Computers*, vol. 12, no. 6, pp. 783–804, Dec. 2003.

### *The Author's Conference Papers*

- [4] **K. Karacs** and T. Roska, “Locating and Reading Color Displays with the Bionic Eyeglass,” in *Proc. of the 18th European Conference on Circuit Theory and Design (ECCTD 2007)*, Seville, Spain, Aug. 2007. pp. 515–518.
- [5] G. E. Paziienza and **K. Karacs**, “An Automatic Tool to Design CNN-UM Programs,” in *Proc. of the 18th European Conference on Circuit Theory and Design (ECCTD 2007)*, Seville, Spain, Aug. 2007. pp. 492–495.
- [6] **K. Karacs**, A. Lázár, R. Wagner, D. Bálya, T. Roska, and M. Szuhaj, “Bionic Eyeglass: an Audio Guide for Visually Impaired,” in *Proc. of the First IEEE Biomedical Circuits and Systems Conference (BIOCAS 2006)*, London, UK, Dec. 2006, pp. 190–193.

- 
- [7] **K. Karacs** and T. Roska, “Route Number Recognition of Public Transport Vehicles via the Bionic Eyeglass,” in *Proc. of the 10th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA 2006)*, Istanbul, Turkey, Aug. 2006, pp. 79–84.
- [8] T. Roska, D. Bálya, A. Lázár, **K. Karacs**, R. Wagner, and M. Szuhaj, “System aspects of a bionic eyeglass,” in *Proc. of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, Island of Kos, Greece, May 21–24, 2006, pp. 161–164.
- [9] **K. Karacs** and T. Roska, “Holistic Feature Extraction from Handwritten Words on Wave Computers,” in *Proc. of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA 2004)*, Budapest, Hungary, July 2004, pp. 364–369.
- [10] **K. Karacs**, G. Prószéky, and T. Roska, “Intimate Integration of Shape Codes and Linguistic Framework in Handwriting Recognition via Wave Computers,” in *Proc. of the 16th European Conference on Circuits Theory and Design (ECCTD 2003)*, Krakow, Poland, Sept. 2003, pp. 409–412.
- [11] G. Tímár, **K. Karacs**, and Cs. Rekeczky, “Analogic Preprocessing and Segmentation Algorithms for Off-Line Handwriting Recognition,” in *Proc. of the 7th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA 2002)*, Frankfurt, Germany, July 2002, pp. 407–414.

### ***The Author’s Other Publications***

- [12] L. Kék, **K. Karacs**, and T. Roska, Eds., *Cellular Wave Computing Library (Templates, Algorithms and Programs) V. 2.1*. MTA-SZTAKI, Budapest, Hungary, 2007.

---

### ***Publications related to CNN Technology***

- [13] L. O. Chua and L. Yang, “Cellular Neural Networks: Theory,” *IEEE Trans. on Circuits and Systems*, vol. 35, pp. 1257–1272, Oct. 1988.
- [14] L. O. Chua and L. Yang, “Cellular Neural Networks: Applications,” *IEEE Trans. on Circuits and Systems*, vol. 35, pp. 1273–1290, Oct. 1988.
- [15] L. O. Chua and T. Roska, “The CNN Paradigm,” *IEEE Trans. on Circuits and Systems*, vol. 40, pp. 147–156, Mar. 1993.
- [16] T. Roska and L. O. Chua, “The CNN Universal Machine: An Analogic Array Computer,” *IEEE Trans. on Circuits and Systems – II: Analog and Digital Signal Processing*, vol. 40, pp. 163–173, 1993.
- [17] A. Rodríguez-Vázquez, S. Espejo, R. Domínguez-Castron, J. L. Huertas, and E. Sanchez-Sinencio, “Current mode techniques for the implementation of continuous- and discrete-time cellular neural networks,” *IEEE Trans. on Circuits and Systems – II*, vol. 40, no.3, pp. 132–146, Mar. 1993.
- [18] Á. Zarándy, “The Art of CNN Template Design,” *International Journal of Circuit Theory and Applications*, vol. 27, pp. 5–23, 1999.
- [19] L. O. Chua and T. Roska, *Cellular neural networks and visual computing, Foundations and applications*. Cambridge, UK & New York: Cambridge University Press, 2002.
- [20] T. Roska, “Computational and Computer Complexity of Analogic Cellular Wave Computers,” *Journal of Circuits, Systems, and Computers*, vol. 12, pp. 539–562, 2003.
- [21] Analogic Computers Ltd, *Aladdin Professional*, <http://www.analogic-computers.com>
- [22] A. Rodríguez-Vázquez, G. Liñán, L. Carranza, E. Roca, R. Carmona, F. Jiménez, R. Domínguez-Castro, and S. Espejo, “ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Toward VSoCs,” *IEEE Trans. on Circuits and Systems*, vol. 51, no. 5, pp. 851–863, 2004.
- [23] MTA-SZTAKI, *MatCNN*, <http://lab.analogic.sztaki.hu/Candy/matcnn.html>

---

***Publications related to Handwriting Recognition***

- [24] M. Côté, E. Lecolinet, M. Cheriet, and C. Y. Suen, "Automatic reading of cursive scripts using human knowledge," in *Proc. of the International Conference on Document Analysis and Recognition*, 1997, pp. 107–111.
- [25] S. Edelman, T. Flash, and S. Ullman, "Reading cursive handwriting by alignment of letter prototypes," *International Journal of Computer Vision*, vol. 5, pp. 303–331, 1990.
- [26] D. Gibbon, R. Moore, and R. Winski, Eds., *Handbook of Standards and Resources for Spoken Language Systems*. Berlin & New York: Walter de Gruyter Publishers, 1997.
- [27] D. Guillevic, D. Nishiwaki, and K. Yamada, "Word lexicon reduction by character spotting," in *Proc. of the 7th International Workshop on Frontiers in Handwriting Recognition*, Amsterdam, Netherlands, 2000, pp. 373–382.
- [28] A. L. Koerich, R. Sabourin, and C. Y. Suen, "Large vocabulary off-line handwriting recognition: A survey," *Pattern Analysis & Applications*, vol. 6, no. 2, pp. 97–121, 2003.
- [29] G. Lorette, "Handwriting recognition or reading? What is the situation at the dawn of the 3rd millennium?" *International Journal on Document Analysis and Recognition*, vol. 2, pp. 2–12, 1999.
- [30] S. Madhvanath and V. Govindaraju, "Holistic lexicon reduction for handwritten word recognition," in *Proc. of the 8th SPIE International Symposium on Electronic Imaging: Science and Technology*, vol. 2660, San Jose, Mar. 1996, pp. 224–234.
- [31] S. Madhvanath and V. Krpāsundar, "Pruning large lexicons using generalized word shape descriptors," in *Proc. of the International Conference on Document Analysis and Recognition (ICDAR 1997)*, Ulm, Germany, Aug. 1997, pp. 552–555.
- [32] S. Madhvanath and V. Govindaraju, "The role of holistic paradigms in handwritten word recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 149–164, 2001.
- [33] G. Qian, "An Engine for Cursive Handwriting Interpretation," in *Proc. of the 11th IEEE International Conference on Tools with Artificial Intelligence*, 1999, pp. 271–278.
- [34] K. M. Sayre, "Machine recognition of handwritten words: A project report," *Pattern Recognition*, vol. 5, pp. 213–228, 1973.



- 
- [35] L. Schomaker and E. Segers, "Finding features used in the human reading of cursive handwriting," *International Journal on Document Analysis and Recognition*, vol. 2, pp. 13–18, 1999.
- [36] W. Senior and A. J. Robinson, "An Off-line Cursive Handwriting Recognition System," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 309–321, 1998.
- [37] T. Steinherz, E. Rivlin, and N. Intrator, "Offline cursive script recognition – a survey," *International Journal on Document Analysis and Recognition*, vol. 2, pp. 90–110, 1999.
- [38] G. Tímár, "Applied High-Speed Analogic Algorithms for Multitarget Tracking and Offline Handwriting Segmentation," PhD. dissertation, Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, Hungary, 2006.
- [39] A. Vinciarelli, "A survey on Off-Line Cursive Script Recognition," *Pattern Recognition*, vol. 35, no. 7, pp. 1433–1446, 2002.
- [40] M. Zimmermann and J. Mao, "Lexicon Reduction Using Key Characters in Cursive Handwritten Words," *Pattern Recognition Letters*, vol. 20, pp. 1297–1304, 1999.

### ***Publications related to Sign detection and recognition***

- [41] A. Amedi, F. Bèrmppohl, J. Camprodon, L. Merabet, P. Meijer, and A Pascual-Leone, "LO is a meta-modal operator for shape: an fMRI study using auditory-to-visual sensory substitution," *12th Annual Meeting of the Organization for Human Brain Mapping (HBM 2006)*, Florence, Italy, June 11–15, 2006.
- [42] C.-H. Cheng, C.-Y. Wu, B. Sheu, L.-J. Lin, K.-H. Huang, H.-C. Jiang, W.-C. Yen, and C.-W. Hsiao, "In the blink of a silicon eye," *IEEE Circuits and Devices Magazine*, vol. 17, no. 3, pp. 20–32, May 2001.
- [43] A. Dollberg, H. G. Graf, B. Höfflinger, W. Nisch, J. D. S. Spuentrup, and K. Schumacher, "A Fully Testable Retinal Implant," in *Proc. of International Conference on Biomedical Engineering (BioMED)*, Salzburg, Austria, 2003, pp. 255–260.

- [44] P. B. L. Meijer, "An Experimental System for Auditory Image Representations," *IEEE Trans. on Biomedical Engineering*, vol. 39, no. 2, pp. 112–121, Feb. 1992. (Reprinted in the 1993 IMIA Yearbook of Medical Informatics, pp. 291–300.)
- [45] P. Silapachote, J. Weinman, A. Hanson, R. Weiss, and M. Mattar, "Automatic Sign Detection and Recognition in Natural Scenes," in *Proc. of IEEE Workshop on Computer Vision Applications for the Visually Impaired (in conjunction with CVPR)*, San Diego, California, June 2005.
- [46] R. Wagner and M. Szuhaj, "Color Processing in Wearable Bionic Eyeglass," in *Proc. of 10th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA 2006)*, Istanbul, Turkey, Aug. 28-30, 2006, pp. 85–90.
- [47] T. Yamaguchi, Y. Nakano, M. Maruyama, H. Miyao, and T. Hananoi, "Digit Classification on Signboards for Telephone Number Recognition," in *Proc. of 7th International Conference on Document Analysis and Recognition*, Edinburgh, UK, Aug. 2003, pp. 359–363.

### ***Publications related to Natural Language Processing***

- [48] G. Prószték, M. Naszódi, and B. Kis, "Recognition Assistance," in *Proc. of COLING-2002*, vol. II, Taipei, Taiwan, 2002, pp. 1263–1267.
- [49] G. Prószték, "Humor: a Morphological System for Corpus Analysis," in *Proc. of First TELRI Seminar on Language Resources and Language Technology*, Tihany, Hungary, 1996, pp. 149–158.
- [50] G. Prószték, "Syntax As Meta-morphology," in *Proc. of COLING-96*, vol. 2, Copenhagen, Denmark, 1996, pp. 1123–1126.
- [51] G. Prószték and B. Kis, "Agglutinative and Other (Highly) Inflectional Languages," in *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland, USA, 1996, pp. 261–268.
- [52] *The British National Corpus*, version 2 (BNC World). Distributed by Oxford University Computing Services on behalf of the BNC Consortium. URL: <http://www.natcorp.ox.ac.uk/>, 2001.

---

***Publications related to Cellular Automata***

- [53] S. Ulam, “Processes and Transformations,” in *Proc. of International Congress on Mathematics*, 1952, vol. 2, pp. 264–265.
- [54] J. von Neumann (A. Burks, Ed.), *The Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, USA & London, UK, 1966.
- [55] S. Wolfram, *A New Kind of Science*. Wolfram Media, Champaign, IL, USA, 2002.
- [56] N. G. de Bruijn, “A Combinatorial Problem,” *Koninklijke Nederlandse Akademie v. Wetenschappen*, vol. 49, pp. 758–764, 1946.
- [57] D. Lind and B. Marcus, *An introduction to symbolic dynamics and coding*. Cambridge, UK & New York: Cambridge University Press, 2002.
- [58] L. O. Chua, V. I. Sbitnev, and S. Yoon, “A Nonlinear Dynamics Perspective of Wolfram’s New Kind of Science. Part III: Predicting the unpredictable,” *International Journal of Bifurcation and Chaos*, vol. 14, no. 11, pp. 3689–3820, 2004.
- [59] L. O. Chua, V. I. Sbitnev, and S. Yoon, “A Nonlinear Dynamics Perspective of Wolfram’s New Kind of Science. Part IV: From Bernoulli shift to  $1/f$  spectrum,” *International Journal of Bifurcation and Chaos*, vol. 15, no. 4, pp. 1045–1183, 2005.
- [60] L. O. Chua, J. Guan, V. I. Sbitnev, and J. Shin, “A Nonlinear Dynamics Perspective of Wolfram’s New Kind of Science. Part VII: Isles of Eden,” *International Journal of Bifurcation and Chaos*, to be published
- [61] T. Toffoli, “Computation and Construction Universality of Reversible Cellular Automata,” *Journal of Computer and System Sciences*, vol. 15, pp. 213–231, 1977.
- [62] U. Frisch, B. Hasslacher, and Y. Pomeau, “Lattice-Gas Automata for the Navier-Stokes Equations,” *Physical Review Letters*, vol. 56, no. 14, pp. 1505–1508, Apr. 1986.
- [63] R. P. Feynman, “Simulating Physics with Computers,” *International Journal of Theoretical Physics*, vol. 21, no. 6-7, pp. 467–488, June 1982.

***Other Publications***

- [64] Mathworks, *MATLAB*, <http://www.mathworks.com>



## Appendix: CNN Templates

*Linear, isotropic templates*

$$A = \begin{bmatrix} a_2 & a_1 & a_2 \\ a_1 & a_0 & a_1 \\ a_2 & a_1 & a_2 \end{bmatrix}, \quad B = \begin{bmatrix} b_2 & b_1 & b_2 \\ b_1 & b_0 & b_1 \\ b_2 & b_1 & b_2 \end{bmatrix}, \quad z$$

Template	Feedback matrix (A)			Control matrix (B)			Threshold	Boundary condition
	$a_0$	$a_1$	$a_2$	$b_0$	$b_1$	$b_2$	$z$	
<b>AND</b>	2	0	0	1	0	0	-1	X
<b>BPROP</b>	3	0.25	0.25	0	0	0	3.75	-1
<b>FIGREC</b>	4	0.5	0.5	4	0	0	3	0
<b>HOLE-FILLING</b>	3	1	0	4	0	0	-1	0
<b>HOLE-FILLING4</b>	3	0.5	0.5	4	0	0	-1.5	0
<b>HOLLOW</b>	3	0.25	0.25	0	0	0	2.25	-1
<b>JUNCTION</b>	1	0	0	6	1	1	-3	-1
<b>LOGDIF</b>	2	0	0	-1	0	0	-1	X
<b>NOT</b>	1	0	0	-2	0	0	0	X
<b>OR</b>	2	0	0	1	0	0	1	X
<b>SMALLKILLER</b>	2	1	1	0	0	0	0	0

X: "don't care"

*Linear, non-isotropic templates*

<b>CONCAVEARCFILLER45</b>	$A = \begin{bmatrix} 0.5 & 0 & 0 \\ 0.5 & 2 & 0 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = -1$
<b>DILATION</b>	$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	,	$z = 2$
<b>EROSION</b>	$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = -2$
<b>HORIZONTAL CONNECTED COMPONENTS DETECTOR (HCCD)</b>	$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = 0$
<b>HORIZONTAL MELTING (MELTRIGHT)</b>	$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 3 & 0 \\ 1 & 0 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = -1$
<b>LOCAL NORTHERN ELEMENT (LNE)</b>	$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = -3$
<b>LOCAL SOUTHERN ELEMENT (LSE)</b>	$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	,	$z = -3$
<b>SHADOWDOWN</b>	$A = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = 0$
<b>SHADOWUP</b>	$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = 0$
<b>VERTICAL CONNECTED COMPONENTS DETECTOR (VCCD)</b>	$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = 0$
<b>VERTICAL MELTING (MELTDOWN)</b>	$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	,	$z = -1$