DOI:10.15774/PPKE.ITK.2014.007

Spatial–Temporal Event Recognition and Metastable Oscillations in CNN Wave Computer

Miklós Koller

A thesis submitted for the degree of
Doctor of Philosophy

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

Supervisor:
György Cserey, Ph.D.

Faculty of Information Technology and Bionics
Pázmány Péter Catholic University
Budapest, 2014

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

Acknowledgements

I have to say "thank you". I could not have realized my studies during the years without the encouragement and help from others.

First of all I would like to thank *Professor Tamás Roska*, for his guidance and support during my studies, who quickened me with his honest enthusiasm hearing the news about the working circuit from Siena. I am thankful for the emphasizing of the intense work's necessity. He encouraged us to devote time countlessly to the research.

I am thankful to the Faculty and the Doctoral School for providing tools and caring environment for work, especially personally *Prodean Judit Nyékyné Gaizler* and *Dean Péter Szolgay*.

The administrative and financial staff of the Faculty dealt with my problems helpfully and flexibly. I am really thankful to *Mrs. Harasztiné*, *Mrs. Mikesyné*, *Mrs. Mihálffy*, *Mrs. Vida*, *Mrs. Szalay*, *Ms. Adorján*, *Mrs. Babiczné*, *Mrs. Bayerné*, *Mrs. Aliné*, *Ms. Sifter* and the rest of administrative and financial personnel, whose everytime helpful work made fluent the formal processes.

Since the first summer robotic camp, my supervisor *György Cserey* helps me with his constant support. I am thankful for the guidance even at my first LEGO-project, for the persistent help in the last hours editing my TDK-manuscript, for the belief in my work and for the encouragement in the seemingly hopeless situations. During my undergraduate studies, he and *Ákos Tar* were my supervisors; the together built infrasensor array finished the first class of the primary school since that time. I am thankful, Ákos, for the direct help during the designing, soldering and debugging, which gave me the initiating push. A significant part of my doctoral work is around the study of metastable oscillations, which was supervised by *Professor Barnabás Garay*. I am thankful for letting me a prosector by this interesting elementary dynamics; I am grateful for every discussions about loosely-coupled topics, too. I will be glad, if at the end of my life I can consider myself as the part of the creative minority.

During the third year, I spent four months at the circuit-theory group of the University of Siena, where the circuit measuring metastable oscillations was prepared, and I was absorbed in the details of the equilibria's computation. I am

DOI:10.15774/PPKE.ITK.2014.007

grateful to *Professor Mauro Forti*, *Assistant Professors Mauro Di Marco*, *Massimo Grazzini* and *Luca Pancioni*.

Special thanks go to my colleagues at the Doctoral School and the Robotic Laboratory. Both the technical and the informal conversations made this period matured and loose. Thank You, *Dóra Bihary*, *Bence Borbély*, *Sándor Földi*, *Tamás Fülöp*, *László Füredi*, *András Gelencsér*, *Zsolt Gelencsér*, *Domonkos Gergelyi*, *Antal Hiba*, *András Horváth*, *Anna Horváth*, *Balázs Jákli*, *Mátyás Jani*, *Csaba Józsa*, *Imre Juhász*, *András Laki*, *László Laki*, *Endre László*, *Balázs Ligeti*, *Csaba Nemes*, *Tamás Pilissy*, *Mihály Radványi*, *Ádám Rák*, *István Reguly*, *János Rudan*, *Norbert Sárkány*, *Borbála Siklósi*, *Attila Stubendek*, *Dávid Tisza*, *Gábor Tornai*, *Kálmán Tornai*, *Zoltán Tuza*, *József Veres* and *Tamás Zsedrovits*.

I am thankful for the acceptance by the christian community at the Faculty, for growth in faith and in experiences. Special thanks go to *Daniella Török* and *Ft. Kálmán Nyéky*, whose tireless work started everything.

Mrs. Andrea Hegedűsné Láng, my math and physics teacher at the highschool taught me a lot, whose work undoubtedly helped me during the preparation to the university.

Special thanks goes to *Zsófia Cserey* who helped a lot in the English revision of the text.

And last, but in the first place I would like to thank my parents and my wife their help and encouragement. I got every opportunity and support to realize my dreams. Thank you *Mom*, *Dad* and dear *Csenge* giving me freedom at all points and the overflowing love.

This research project was supported by the University of National Excellence Program and the Research Faculty grant awarded to the Pázmány Péter Catholic University, Faculty of Information Technology and Bionics. This work has been supported by the European Union and Hungary and co-financed by the European Social Fund through the project TÁMOP-4.2.2.C-11/1/KONV-2012-0004 - National Research Center for Development and Market Introduction of Advanced Information and Communication Technologies.

DOI:10.15774/PPKE.ITK.2014.007

Abstract

This dissertation (i-i) describes frameless computing in the case of a CNN Wave Computer extended with an infrared active sensor array, where the continuous processing makes possible the detection of a spatial–temporal feature, (ii-i) uncovers a locally adaptive, iterative algorithm to tune the depth measurement range of a depth measuring sensor array, (ii-ii) presents two standard CNN algorithms to detect different kinds of spatial movements in simulation; (iii-i) outlines numerical argumentations regarding the existence and strength of metastable oscillations observed in one dimensional CNN arrays with periodic boundary condition, (iii-ii) describes circuit measurements reproducing these oscillations on a laboratory prototype circuit.

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

Contents

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

List of Figures

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

List of Tables

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

Chapter 1

Introduction

In our everyday observations, a lot of recognition tasks are connected to the identification of spatial–temporal dynamics. Often it is not a characteristic figure that we consider familiar, but a specific movement, series of actions or views. Tennis game enthusiasts recognize their favourite player on the court from the backhand movement, horse fans recognize their favourite breed of horse from the posture and from the trot. Dance is also such an activity, whose type cannot be identified from one or more still picture; however, if we consider the whole transient of the movement, we can recognize without doubt which kind of folk or classical dance we see.

By signal processing tasks, the audio signal processing is a good example of one–dimensional temporal analysis. It can be either some kind of noise filtering, or the cut of some frequency range, or other signal processing issues. In the case of spatial (more precisely "planar") analysis we can think of the simplest image processing tasks, like a two–dimensional image from which we want to extract some features, like edges, corners, specific kind of surfaces, big or small patches, or something else. If we take into consideration the combination of spatial and temporal analysis, we reach the classical frame–by–frame processing of video analysis. The "frameless" processing philosophy of Cellular Neural Networks (CNN) breaks with this approach: not only the real hardware which represents the time and the signal as continuous (not discrete) quantities, but the whole input flow processing is continuous, too. Although the consecutive, frame–by–frame (classical) processing of the input pictures gives answer to every question, this needs appropriately built processing unit and organized memory. In contrast to this, if some characteristics of the evolving dynamics (for example, an output pattern temporarily becoming stable) indicates some event taking place in time, then no further architecture–extension is necessary. The time evolution of the dynamics provides itself alone

DOI:10.15774/PPKE.ITK.2014.007

the recognition, which is significantly "cheaper" in computation complexity.

In one part of my doctoral work I dealt with the recognition of spatial–temporal dynamics. **I was looking for the answer to the following question: what kind of recognition issues exist and how can they be solved in those cases, when the continuity of the input flow is the quality difference (compared to the separate processing of the individual still pictures) that makes solvable a recognition problem.**

In the case of engineering problems, stable solutions are welcome, which can be robust in spite of the small environmental effects / changes. It can be a stable constant value, or a regularly repeating change, a periodic oscillation. In general it is a must to be reliable, robustly resisting measurement- and environmental noise. Interesting and special case is the seemingly stable regular change in time (metastable periodic oscillation). For a long time it seems ordinary periodic oscillation, but after a point, through a short intermediate interval, it converges to a stable point, settling down to a constant value. Often this transient happens without any priori sign, just the last few cycles imply change in the behavior. As an example, specific waves in the brain activity show similar phenomenon. According to certain theories, these metastable periodic oscillations have functional role (for example, recognition, understanding, association).

In the other part of my doctoral work I dealt with metastable periodic oscillations. I have analyzed the phenomenon under specific architectural constraints (one–dimensional CNN array with periodic boundary condition). **I was looking for the answers to the following questions: what kind of oscillations exist in which region of some specific parameters? How do these parameters influence the length of the oscillation (strength of metastability)? If there exist more different waveforms, what kind of connections are there between them?**

Both of my research fields can be informally interpreted as analog computations (in contrast to the digital, discretized ones). As a specific example to analog computation, let me introduce a system, where the interaction of three different dynamics realizes the computation itself. This example will be described in details in Chapter 2, here I would like only to emphasize the joint evolution of the different continuous flows. Let me take a cellular array of computing elements (analog processors), where every computing element has an optical input from the environment. In addition to this, there are activation light sources located next to every light-sensing element. These light sources are responsible for the lighting of

DOI:10.15774/PPKE.ITK.2014.007

the environment (or the lighting of the measured object), while the optical sensors measure the reflected light from the environment. In this way we have a locally connected array where in every cell we have a processor with optical input, and an activating light source. The dynamics of the processor array can be characterized by the continuous state-equation of Cellular Neural Networks as the self-dynamics of this system. We can introduce other dynamics to the system through the light sources, which directly influences the perception of the environment, in this case the input flow. The interaction of these dynamics realizes the main computation itself, which can be quite efficient in some cases (eg. spatial-temporal event detection).

The dissertation is organized as follows. Chapter 2 uncovers the details of my frameless detection example. Not only the experiment, but a detailed, deeper analysis is presented to support the significance of this processing mode. In Chapter 3 I describe further algorithms with the model of the infrared sensor array, simulation results are also presented. Throughout Chapter 4 and 5 metastable oscillations are analyzed. In the former one I describe these oscillations with simulation, and present some numerical argumentation about the robustness of the phenomenon; while in the latter one I demonstrate the suspected robustness of these long transient metastable oscillations with a paradigmatic circuit, made by means of discrete components. Chapter 6 summarizes the main results and shows directions of application, where the results of this dissertation could be utilized. The author's publications as well as other publications connected to this dissertation can be found at the end of this document.

1.1 Methods used in the experiments

The considered objectives are strongly connected to the CNN Wave Computer's computing model. My results are based on simulations, but some of them are relying on real measurements, where the input flow is served by an infrared sensor array. This array contains 8×8 distance measuring LED–phototransistor pairs, where the sensors measure the LEDs' reflected light from the environment / measured object. Every LED can be controlled separately, and the readout of the phototransistors can be in arbitrary order. There is a controlling and readout circuit connected to the sensor array, in this way we can realize high level communication (via serial line) with the simulator of the Wave Computer.

DOI:10.15774/PPKE.ITK.2014.007

I have utilized morphological– and wave–operations in the frameless computation model. I have analyzed the effects of different wave propagating templates on the continuously changing input flow. Throughout the analysis I leaned on an earlier published template class, with fine–tuning of the elements I tried to achieve the appropriate behavior. This template class is sign–antisymmetric, coupled, and contains a few non–zero elements. Earlier it was considered due to its wave processing behavior and due to the emerging state–transitions (stable equilibrium point – periodic orbit – chaotic behavior). In the detailed analysis of the measurements I examined the behavior of the system's equilibrium points, the dependency between specific parameters and the evolving state, and the robustness of the whole phenomenon.

These simulators were realized both in C++ language and in MATLAB environment. It is easier to implement evaluating softwares in MATLAB environment; however, in some cases I used a function library which is available only in C++ language. The microcontroller of the sensor array was programmed in C language. The real measurement scenes were partially realized with an xyz table capable of moving with $10\mu m$ precision.

By the metastable periodic oscillations, firstly I examined the details of the phenomenon with the simulation of the dynamics; naively tried to discover the characteristics of the transients and their parameter–dependencies. With numerical evaluation of the data I predicted the results of the analytical eigenvalue–analysis made by *Professor Barnabás Garay*, where the latter one is the rigorous mathematical proof of the metastability in our example.

The simulations were realized both in C++ language and in MATLAB environment. The built-in solver *ode45* of MATLAB has stabilized unexpectedly the solution in some cases (symmetrical initial conditions); but the self-made solvers (MATLAB: Explicit Euler (EE) method; C++: EE, RK45) have shown the convergence to the stable point in every case. Some of the parameter–changes and the evoked bifurcations were analyzed with the AUTO bifurcation–analysis software.

Under the guidance of the italian cooperating research group (*Mauro Forti*, *Luca Pancioni*, *Mauro Di Marco*, *Massimo Grazzini*) I have prepared a one–dimensional, periodic CNN array from discrete components, on which platform I experimentally examined the existence of metastable periodic oscillations.

DOI:10.15774/PPKE.ITK.2014.007

Chapter 2

Frameless detection with an infrared sensor array

In this chapter I present the frameless detection method in the case of a CNN Wave Computer interfaced to an infrared sensor array. The measurement system, the computational environment and the key example demonstrating the advantage of frameless processing are presented as well as those scenes and parameter regions, where the frameless processing mode is unable to outperform the frame–by–frame processing mode.

2.1 Review of the related literature

According to my best knowledge, detection on the basis of frameless processing is a relatively new phenomenon. A slightly different version of it was introduced in [14], where the authors uncover the details of the Proactive Adaptive Cellular Sensory-Computer Architecture via extending the CNN Universal Machine. In this case the evolving computation is influenced by proactive and adaptive methods. A more recent, strongly coupled work is [15]. But before that work, I would like to refer to wave computing in general, which is the base computing system in the architecture I have used.

Inspecting wave-propagation, a distinction has to be made between classical waves (propagating in conservative systems, behaving as a closed system) and nonlinear (active) waves. In the case of active waves it has a great importance that the propagation of the wave is supported by the media, from an energetic point of view. Subsequently, the propagating wave does not decay nor the waveform is distorted during the propagation. These waves cannot be reflected from the boundaries, nor can they be interfered, but they can be diffracted. When they

DOI:10.15774/PPKE.ITK.2014.007

are colliding with each other, they annihilate each other (except a special kind of them, the trigger wave: after some definition they can merge at collision).

Although in general the classical wave is associated with the word "wave", there are many well-described phenomena of active wave propagation. In chemistry, reaction-diffusion systems can produce waves [41]. Spiral waves are universal form of patterns arising in dissipative media of oscillatory nature [42]. As examples from biology, nerve impulse propagation shows a wave-process [43] and the communication method of the amoeba [44], [45] is based on waves, too. Studying the examples of nature, a unified paradigm has been developed by Chua [46], [47], [48], [49], [50], [51]. Generally speaking, solving some of the computational problems with a system utilizing active waves could be much more obvious [52].

As a strongly related work, I would like to refer to the doctoral theses of István Petrás [17]. He analyzed in details a specific class of wave-propagating templates. He extensively investigated the different parameter regions, the different forms of the evolving patterns and oscillations. He processed only static input pictures, in this way the most significat difference between his work and my work is in the input of the systems (static input versus continuous flow). More details on his work and the specific comparison with my work will be described in Section 2.3.

Compared to the conventional wave processing, the phenomenon of frameless detection involves the detection of an appropriate spatial-temporal event by the dynamics itself, evolving on the computing array. In [15] the authors realize frameless detection of spatial-temporal events with the use of delayed CNN template. In the case of frameless processing, the detection is done by using continuous dynamics, without cutting the input flow into frames.

In my dissertation, the frameless processing mode is connected to a depth measuring sensor array, in this way it is necessary to review some relating depth-sensing systems and sensor array constructions. The simplest case of depth sensing, if we combine the pictures of more cameras (for example, three cameras in different locations), like in the case of the off the shelf Leonar3Do system [53]. Another input device is the LEAP Motion controller [54], which contains three IR LED and two IR camera (unfortunately in this case the detailed working method is not specified by the factory). The third solution on the market, with a bit different method is the Kinect movement detector input device [55], belonging to Microsoft's XBox system. Here the depth sensor consists of an infrared laser projector and a monochrome CMOS sensor. The laser projects a specific pattern to the environment/object, the sensor records the image with the spatially distorted IR pattern, and an algorithm

DOI:10.15774/PPKE.ITK.2014.007

computes the body model behind the distortion. The fourth kind of solution can be the time-of-flight computation, like in the case of the Canesta 101 chip [56]. In this case, the CMOS chip records the reflected signal-fragments of a blinking IR LED (blinking with 44 MHz), and on the basis of the time-of-flight, it computes the distance of the observed object/environment.

Reviewing sensor array construtions, numerous publications deal with the sensor model and measurement characteristics of IR emitter – diode pairs (in the context of mobile robotics), without being exhaustive [57], [58], [59]. As very similar constructions, I would like to highlight the doctoral work of Ákos Tar in [60] and the paper [61]. In the former one, the author has an array with 8 IR LEDs and 8 photodiodes alternately ordered in one row, meaning 15 measurement points in front of the array. The drawback of this construction is that the scanning of a two-dimensional surface needs movement during the measurement. On the other hand, the measurement accuracy and the measurement speed is far more better, than in the case of the array used by me. In [61], the authors describe an array of size 3×3, where every cell of the regular grid contains an LED – PSD (position sensitive detector) pair. Here the advantage is the fast and accurate measurement in a wide depth range (between 20 and 150 cm), but the drawbacks are the limited number of cells and the low spatial density of the sensing points (compared to my array).

2.2 The measurement system setup

The measurement device contains 8×8 infrared distance measuring LED – phototransistor pairs (TCRT1000), whose packages are organized in a regular grid. The complete architecture (schematic) can be seen in Figure 2.1. An 8–bit microcontroller (PIC 18f2321) controls the system on this peripherial board. There is a serial link (MAX232) to the simulator running on the PC. The LED–phototransistor pairs are arranged in row-wise order, each block contains eight elements. Every block gets six input/control lines (signal, clock, clear, and three for address), and has an output line. These output lines are collected in a multiplexer (4051D), resulting in the use of one input analog channel of the microcontroller. In this way, the hardware has serial execution, where the proper ordering and timing is defined by the clock and the address lines.

DOI:10.15774/PPKE.ITK.2014.007

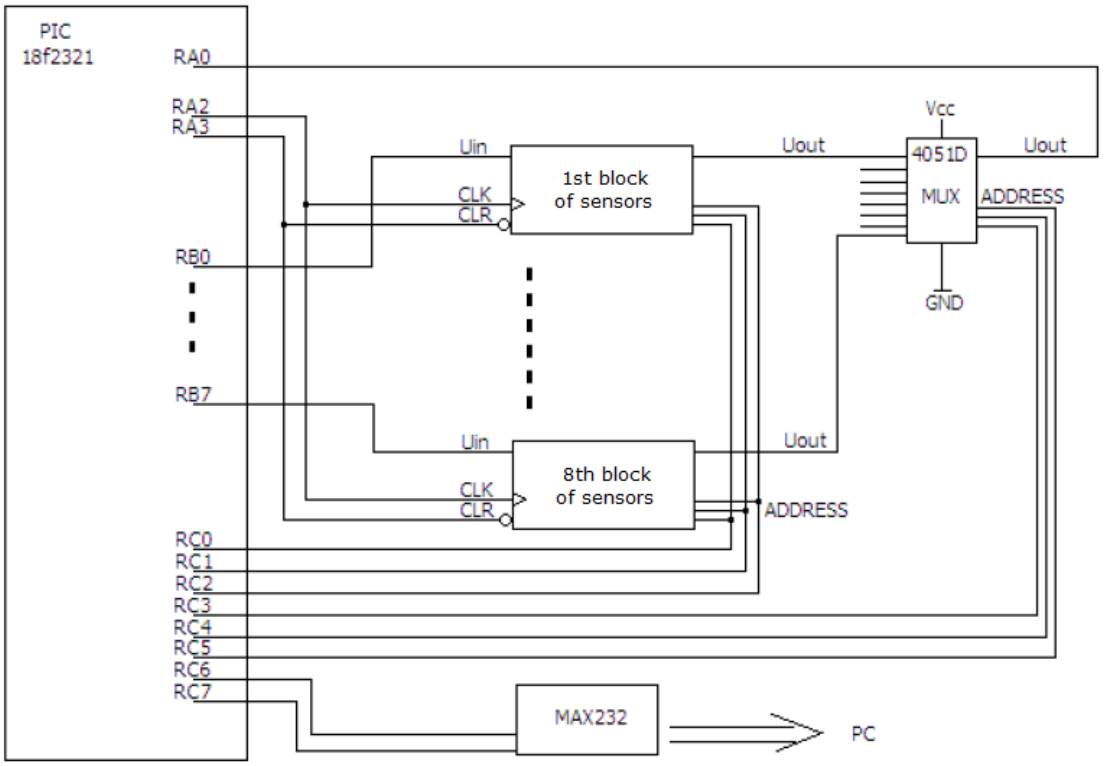

Figure 2.1: The architecture of the sensor array

Figure 2.2 depicts the schematic drawing of the architecture inside one block. There is a shift register (74ACT164) which controls the light sources of the block, on the basis of the stored pattern. Every anode is connected to the positive terminal of the power supply, while every cathode is connected to the ground through a Darlington array (ULN2803). The gates of the Darlington array is controlled by the stored pattern of the shift register. The collector–emitter line of the photosensitive phototransistor is serially connected to a resistor, between the positive terminal of the power supply and the ground. In this way, the reflected light causing the opening of the gate becomes a part of a voltage divider, meaning that, the potential of the collector indicates the amount of the incoming / reflected light. The output of every phototransistor is collected by a multiplexer (4051D), resulting in one output line.

DOI:10.15774/PPKE.ITK.2014.007

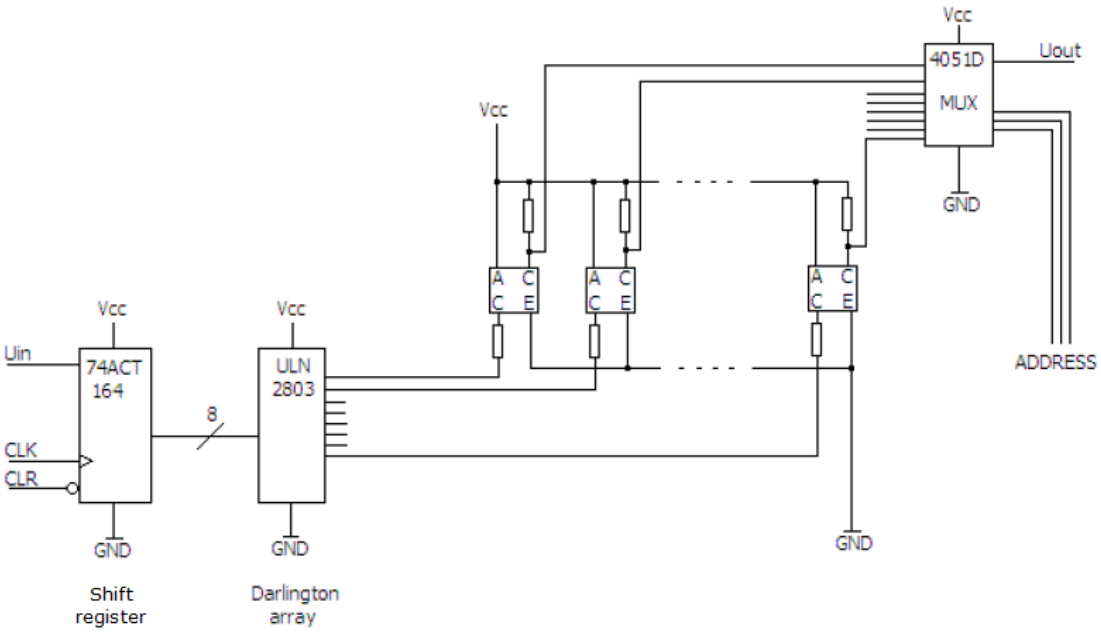

Figure 2.2: The architecture inside a block (row) of the array

Although I can implement a simpler but autonomous measuring and computing system on this microcontroller, due to the limitations of the inner memory and the low processing speed it is more advantageous to run the complex computations on a PC, leading to the utilization of the microcontroller as a low level controller (direct control of the lightsources and readout of the sensors' signal levels) and communicator. I followed this hierarchical scheme in my measurements: the simulator of the Cellular Wave Computer runs on the PC, only the raw measurement data are served by the sensor array.

2.3 Computational environment

The simulator on the PC realizes a standard, space-invariant CNN computational model (Chua-Yang model) with one layer, as the state-equation of every cell is shown in Equation 2.1. Here $x_{ij} \in \mathbb{R}$, $y_{kl} \in \mathbb{R}$, $u_{kl} \in \mathbb{R}$, $z \in \mathbb{R}$ are called the state, the output, the input and the bias of cell (i, j) respectively. A and B are the feedback and input synaptic operators. $rd \in \mathbb{N}$ is the radius of the sphere of influence of cell (i, j). The dependence of the output on the state is characterized by the well-known piecewise-linear function (Equation 2.2).

$$\dot{x}_{ij} = -x_{ij}(t) + \sum_{|k-i|\leq rd} \sum_{|l-j|\leq rd} A(i-k, j-l) y_{kl}(t) + \sum_{|k-i|\leq rd} \sum_{|l-j|\leq rd} B(i-k, j-l) u_{kl}(t) + z \quad (2.1)$$

DOI:10.15774/PPKE.ITK.2014.007

$$y = f(x) = f_{pwl}(x) = \frac{1}{2}(|x+1| - |x-1|) \tag{2.2}$$

The 19 numbers of the templates (A, B, z) determines the special task/function that the CNN Wave Computer executes (for an extensive collection of these functions the Reader is kindly referred to the Template Library [34]). The applied template class was originally published in [17], an asymmetric template with few non-zero elements. The template is given by Equations 2.3-2.4.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ s & p & q \\ 0 & r & 0 \end{bmatrix},\ B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix},\ z = z \tag{2.3}$$

$$s = 1.1,\ p = 1.0,\ q = -1.1,\ r = \pm 0.7,\ b = 1.0,\ z = 0.0 \tag{2.4}$$

Originally, this template needs the Full Signal Range (FSR) CNN model [38], zero-flux type boundary condition, and the processed static, binary image should be loaded to the input terminal ($U(t)$) and in the initial state ($X(0)$) as well. In [17] the author has examined the behavior of the different parameter-regions of this template: in some cases we can observe a stable pattern on the output, while in other cases we can get periodic, or what is more: chaotic dynamics at the output. One crucial thing is the sign of the template-element r: the positive values result in a solid, homogenous black patch with oscillating right border, while negative values result in a texture-like oscillating pattern. These outputs are easily observable in the case of bigger arrays (several hundred computing cells). In [17] the author used the size of 41×23 and 64×64 cells (simulations with the Full Signal Range model and ACE4K chip measurements, respectively), however we used the size of 8×8 cells (both the sensor array and the simulated computing array). For comparison, the behavior-difference caused by the sign-difference of r can be seen in Figure 2.3 and Figure 2.4. These figures represent different outputs (comparing to the outputs of [17]): the main reasons are the small size of the computing array and the different computation-model (Chua-Yang model, instead of the Full Signal Range model). The actual values of the template-parameters were set according to Equation 2.4.

DOI:10.15774/PPKE.ITK.2014.007

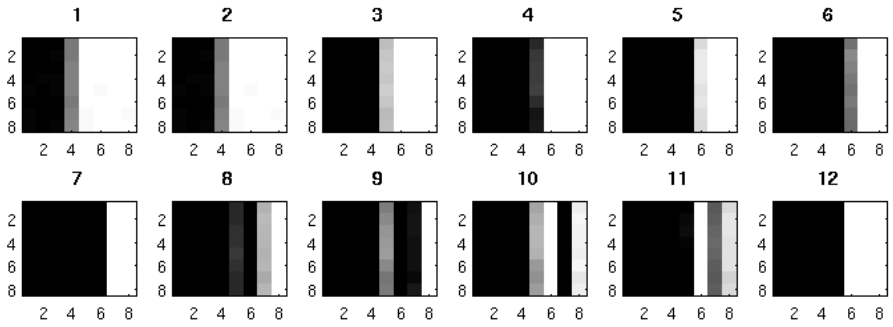

Figure 2.3: Snapshots of the array's output: the spatial-temporal evolution of an oscillating pattern, caused by **the positive sign of the template parameter r**. With greater sizes of the computing array and with the Full Signal Range model (not illustrated here, for details see [17]), the initial, thin rectangular form transforms itself to an elongated patch into the North-East direction of the array. The inner part of the new object is solid and homogenous, while the bordering-region behaves as a waving contour. With this small size of the computing array and with the Chua-Yang model (as depicted in this image-sequence), instead of the slant waving contour we can observe detached parallel lines. The time resolution of this image sequence (the elapsed time between the snapshots) is 1τ, where τ means the cells' time constant.

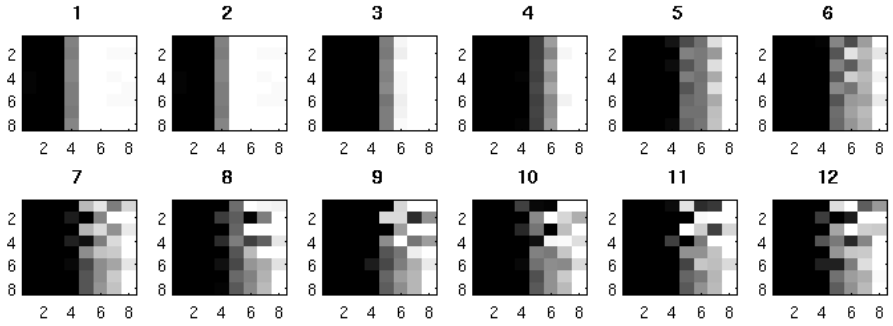

Figure 2.4: Snapshots of the array's output: the spatial-temporal evolution of an oscillating pattern, caused by **the negative sign of the template parameter r**. With greater sizes of the computing array and with the Full Signal Range model (not illustrated here, for details see [17]), the initial, thin rectangular form extends itself to a texture-like oscillating pattern, either in still or as a constantly moving texture from left to right. With this small size of the computing array and with the Chua-Yang model (as depicted in this image-sequence), we can observe traveling wavefronts from up to down. The time resolution of this image sequence (the elapsed time between the snapshots) is 1τ, where τ means the cells' time constant.

DOI:10.15774/PPKE.ITK.2014.007

2.3.1 General frames for multiple–wave computing

There are two (or three) dynamic waves combined:

- the dynamics of the spatial-temporal input flow (u – two dimensional),
- the self-dynamics of the computing cellular array (x (defined by F) – two-dimensional, the inner state of the cells),
- the dynamics of the activation strength of the light-sources (v (defined by G_1 or G_2) – two dimensional)

We are interested in their interaction in two cases:

- **independent activation** case, as described by Equation 2.5, Equation 2.6:

$$\dot{x} = F(x, f_1(x), u) \tag{2.5}$$

$$v = \text{const.} \quad \text{or} \quad \dot{v} = G_1(v, f_2(v)) \tag{2.6}$$

- **adaptive activation** case, as described by Equation 2.7 and Equation 2.8:

$$\dot{x} = F(x, f_1(x), u) \tag{2.7}$$

$$\dot{v} = G_2(v, f_2(v), f_1(x)) \tag{2.8}$$

In Equation 2.5-2.8, f_1 usually means f_{pwl} (Equation 2.2) or a kind of a smooth sigmoid function, while f_2 can denote a kind of step function (e.g. a shifted Heaviside) referring to the two main working modes of our light sources (ON or OFF). The self-dynamics is described in the same way in both cases (Equation 2.5 and Equation 2.7), but the activation-dynamics of the latter case (Equation 2.8) covers a wider set regarding the behavior of the system, than in the former case (Equation 2.6). According to Equation 2.6, the case of **independent activation** can generate only either constant or autonomous activation patterns on the light-sources (like constant strength activation light, or some stroboscopic or periodic pattern). On the contrary, as Equation 2.8 shows we are able to generate activation patterns depending also on the inner state of the computing system. This means that the activation somehow follows—or hopefully facilitates—the evolution and processing of the input flow. For a recent result on **adaptive activation**, the Reader is kindly referred to [10].

DOI:10.15774/PPKE.ITK.2014.007

The key example presented in this dissertation (as well as the different forms of the deeper analysis) belongs to the **independent activation** case, since the activation at every cell is the constant, maximal light strength.

In [1] we have changed the operation-condition of this class of templates: instead of static input pictures, we "fed" the system with a continuous flow (image sequence). We have analyzed the behavior of the output in different parameter-regions, with different input-scenes. Our main aim was at the distinction of some kind of spatial-temporal sign: which emerges only in the case of flow-processing, and which is unrevealable during the individual processing of still input-pictures. In the next section (Section 2.4) I present my main example, which is applicable to identify a terrain bump. Then, in Section 2.5, I analyze this phenomenon in a deeper context. In Section 2.6 I review those scenes and parameter-regions, which led me to the "final" template and the key example; however, in those scenes with the vast majority of those parameter values the frameless processing cannot outperform the frame–by–frame processing mode.

DOI:10.15774/PPKE.ITK.2014.007

2.4 The key example

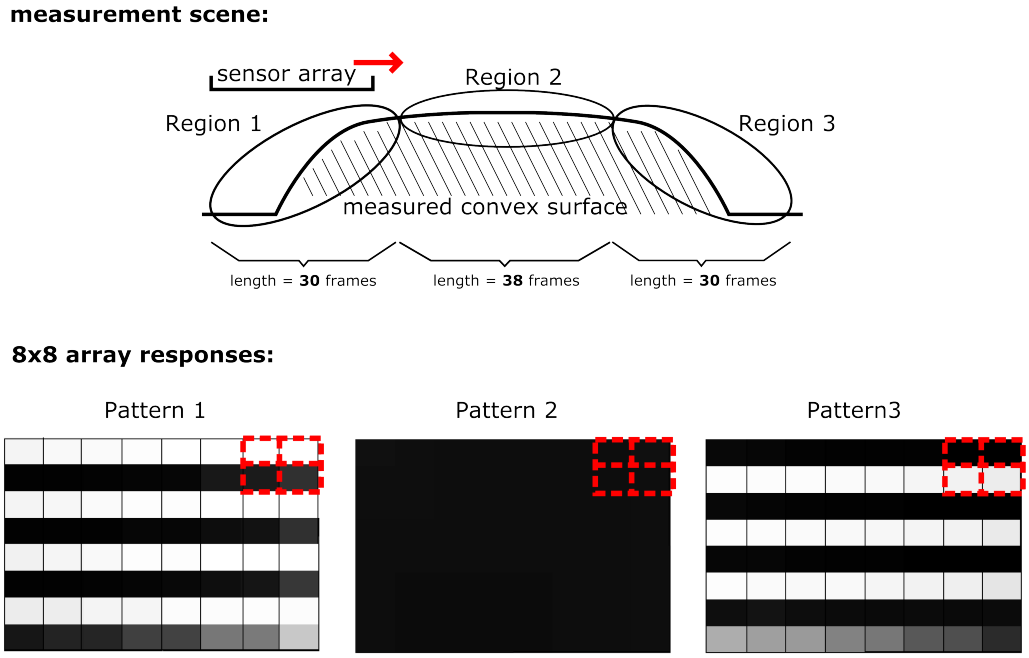

Figure 2.5: The upper part of the figure depicts the schematic drawing of the key-measurement example. The sensor array passes over a terrain bump (horizontally from left to right, as the arrow shows on the picture). The terrain bump consists of three different regions: the uphill-, the plateau- and the downhill-regions (Region1, Region2 and Region3, respectively). During the uphill- and the downhill-regions, 30-30 consecutive pictures were shot, while in the region of the plateau 38 pictures. The bottom part of the figure shows three different output patterns (array responses) of the system, each of them having the four top-right pixels marked with dashed boundary. (The details of the measurement and the appropriate relations between regions and output patterns are described in the main text.)

Figure 2.5 depicts the details of the measurement-setup. The size of the terrain bump was approximately 3.5 times bigger, than the size of the scanning sensor array itself. The sensor array passed over the bump from left to right, scanning the underlying object. The total number of the input frames was 98, meaning that, the one column wide shift of the sensor array was done during 3.5 input frame updates. Both the uphill and the downhill regions were scanned with approximately 30 frames. During these computations, the values of Equation 2.4 were used with $r = -0.7$. The boundary condition was zero-flux, the initial state of the computing array was identical with the first input-frame.

DOI:10.15774/PPKE.ITK.2014.007

Different input pictures can result in different array responses: either stable ones (like Pattern1 and Pattern3 in Figure 2.5), or oscillating ones (like the snapshot-series shows in Figure 2.4). Even if we are inside the same region of the bump, we can observe different outputs. The reason is that the qualitatively similar inputs can result in qualitatively different outputs, depending on which basin of attraction was reached during the evolution of the computation. This means we are not able to identify a region of the bump only from individual snapshots.

The main achievement here, if the whole input flow is processed (they are separate pictures, but the flow is handled as continuous in time at the input terminal), the different regions exhibit different dominant patterns at the output, from which one can identify the underlying regions. In the key example, Pattern1 belongs to Region1, Pattern2 belongs to Region2 and Pattern3 belongs to Region3. Reading out only the four marked pixels in the top-right corner (marked with dashed red squares in Figure 2.5) one can unambiguously identify which part of the terrain bump is scanned currently.

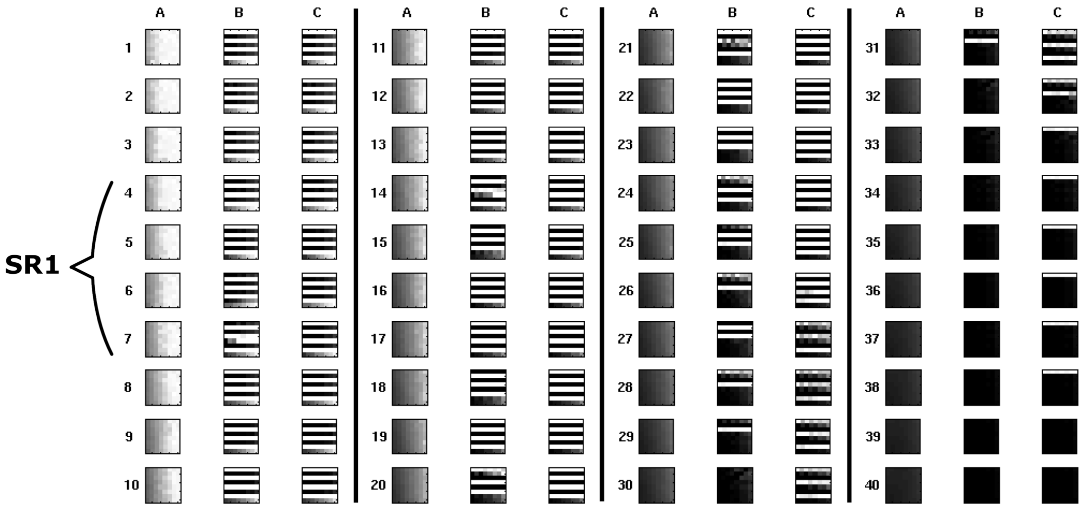

Figure 2.6: Comparison of the array responses of the frame–by–frame and frameless case. **Column A** depicts the **input** pictures. **Column B** shows snapshots of the outputs of the **frame–by–frame** computational mode. **Column C** shows the snapshots of the outputs of the **frameless** computational mode. The numbers before every row are the serial numbers of the appropriate input pictures in the input flow. The four rows indicated with a brace are marked: this sub-region will be analyzed with the derivatives of the state-variable; and also the selected frames (6th and 7th) with deeper analysis in Section 2.5 are from this sub-region.

In Figure 2.6 we can see a summary made from snapshots of the different compu-

DOI:10.15774/PPKE.ITK.2014.007

tational modes. The main advantage of the frameless computation can be clearly seen in the measurement-and-processing results of the first part of this terrain-bump. This figure reviews the first forty input pictures, and the accompanying output snapshots from the frame–by–frame and frameless computing modes. While the frame–by–frame mode presents more different output-patterns, the frameless mode remained in the same stable pattern. This is the same as Pattern1 in Figure 2.5. If we compute later phases of the input flow, we can observe Pattern2 during the plateau (Region2) and Pattern3 during the downhill region (Region3). The 4th, 5th, 6th and 7th input frames and computing cases are marked with SR1: this interval will be analyzed from the viewpoints of the different derivatives.

Analyzing the underlying dynamical system, we can derive the first and second derivatives at every cell of the array. In this way not only the visual characterization (image-flow at the output terminal, the response of the array) is available, but with the help of the derivatives we can describe the attractive behavior of the dominant patterns ("speed" and "smoothness" of the attraction). For the sake of simplicity, I suppose f_1 is the piecewise-linear output function ($f = f_{pwl}$) in Equation 2.5-2.8 (as it was $f = f_{pwl}$ in the original example as well), meaning, the derivative of it equals zero except region $[-1, 1]$, where the value of the derivative equals constant one.

$$\begin{aligned}\dot{x}_{ij} = &-x_{ij}(t) + \sum_{|k-i|\le rd}\sum_{|l-j|\le rd} A(i-k, j-l) f(x_{kl}(t)) \\ &+ \sum_{|k-i|\le rd}\sum_{|l-j|\le rd} B(i-k, j-l) u_{kl}(t) + z\end{aligned} \tag{2.9}$$

$$\frac{d\dot{x}_{ij}}{dx_{ij}} = -1 + A(0,0) f'(x_{ij}(t)) \tag{2.10}$$

$$\begin{aligned}\ddot{x}_{ij} = \frac{d\dot{x}_{ij}}{dt} = \frac{d\dot{x}_{ij}}{dx_{ij}}\frac{dx_{ij}}{dt} = -\dot{x}_{ij} + A(0,0) f'(x_{ij}(t))\dot{x}_{ij} = \\ = x_{ij} - \sum_{|k-i|\le rd}\sum_{|l-j|\le rd} A(i-k, j-l) f(x_{kl}(t)) \\ - \sum_{|k-i|\le rd}\sum_{|l-j|\le rd} B(i-k, j-l) u_{kl}(t) - z \\ + A(0,0) f'(x_{ij}(t)) \Bigg(-x_{ij}(t) + \sum_{|k-i|\le rd}\sum_{|l-j|\le rd} A(i-k, j-l) f(x_{kl}(t)) \\ + \sum_{|k-i|\le rd}\sum_{|l-j|\le rd} B(i-k, j-l) u_{kl}(t) + z \Bigg)\end{aligned} \tag{2.11}$$

DOI:10.15774/PPKE.ITK.2014.007

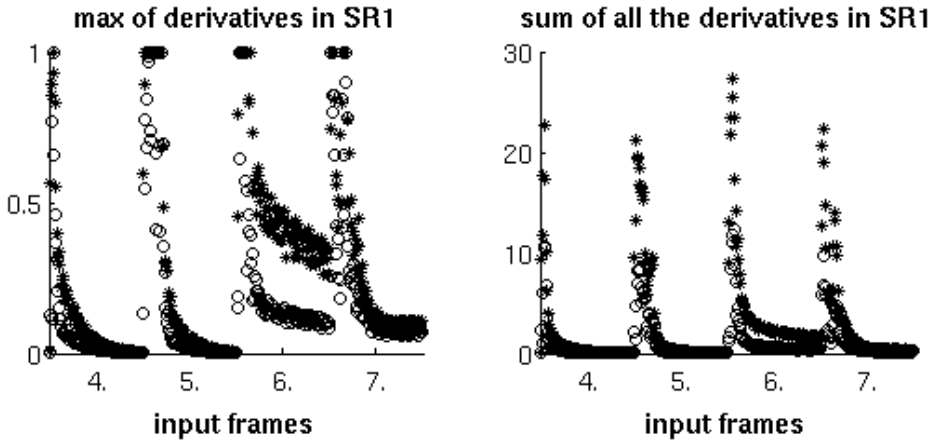

Figure 2.7: **Frame–by–frame case:** when we process the recorded input pictures in individual computations. The first (star, "*") and second (circle, "o") derivatives of the state-variable are presented. The sub-region name SR1 refers to the 4th, 5th, 6th and 7th input frames, which can be found during the uphill-region of the bump (the uphill region is called Region1). The subfigure on the left depicts the maximum value of the derivatives in every time step; while the subfigure on the right depicts the sum of all the derivatives throughout the whole array at every time step.

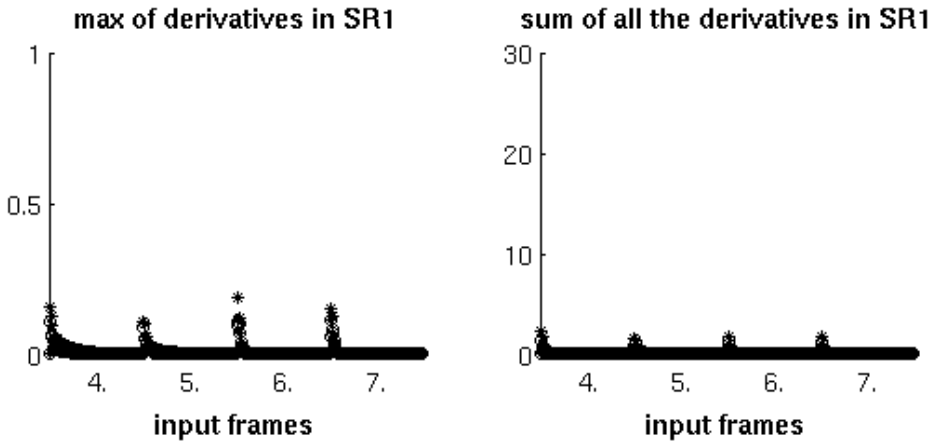

Figure 2.8: **Frameless case:** when we process the recorded input pictures in a computation continuous in time. The first (star, "*") and second (circle, "o") derivatives of the state-variable are presented. The sub-region name SR1 refers to the 4th, 5th, 6th and 7th input frames, which can be found during the uphill-region of the bump (the uphill region is called Region1). The subfigure on the left depicts the maximum value of the derivatives in every time step; while the subfigure on the right depicts the sum of all the derivatives throughout the whole array at every time step.

DOI:10.15774/PPKE.ITK.2014.007

In Figure 2.7 and Figure 2.8 we can see summing diagrams about the derivatives (Figure 2.7 is related to frame-by-frame processing, while Figure 2.8 is related to frameless processing). The first derivatives are computed according to Equation 2.9, while the second derivatives are computed according to Equation 2.11. In both of the figures, the first derivatives are drawn with stars ("*"), while the second derivatives with circles ("o"). The time-scale on axis x represents the interval when the 4th, 5th, 6th and 7th input pictures are fed with the system (let me call this sub-region1 (SR1) which is inside the uphill-region of the bump (Region1)). SR1 was chosen to contain at least two different stable output patterns in the case of the frame–by–frame processing mode. On the basis of Figure 2.7 I can say that the repeated restart of the computational system causes every time a smaller noise, or temporary fluctuation in the computing array's inner state. Especially in those cases, when the system can have possibly more stable patterns at the output. On the contrary, based on Figure 2.8, this kind of a continuous computation can remain almost still, if the consecutive inputs are qualitatively from the same group.

2.5 Analysis of the key example

In the case of frame–by–frame processing, the suspected reason for the seemingly random evolution of the output-patterns is the occasional occurrence of a different stable equilibrium point, which naturally results in the rearrangement of the basins of attraction. In this subsection, I will review the equilibrium points of the whole system as well as the equilibrium points of a reduced system. With a two-dimensional perturbation I will analyze in details the birth / death of equilibrium points simultaneously checking their validity. With a series of random-noise simulations I will show an approximation on the size of the basins of attraction regarding the different equilibrium points.

On the basis of these mentioned investigations, I hope that the surprising behavior of the frameless computing array will be more unambiguous.

2.5.1 Equilibrium points in the 64-dimensional system

In Table 2.1 we can see the number of equilibrium points, computed in the cases of the first 20 frames of the input flow. Figure 2.9 shows the small pictures of the stable equilibrium points for the first 40 frames.

DOI:10.15774/PPKE.ITK.2014.007

Table 2.1: In this table we can see the number of the equilibrium points at the specified input frames. These frames are the first 20 frames of the input flow, meaning, all of them belong to Region1 in Figure 2.5. Stable equilibrium points are at most two, while saddles are born and die a lot of time.

Frame index	Number of stable eq. points	Number of saddles	Number of unstable eq. points
1	1	126	0
2	1	16	0
3	1	1172	0
4	1	150	0
5	1	108	0
6	2	6739	0
7	1	2838	0
8	1	7410	0
9	1	9942	0
10	1	508	0
11	1	7738	0
12	1	14368	0
13	1	3732	0
14	1	3998	0
15	2	16603	0
16	1	1780	0
17	1	1504	0
18	2	16301	0
19	1	1480	0
20	2	12778	0

DOI:10.15774/PPKE.ITK.2014.007

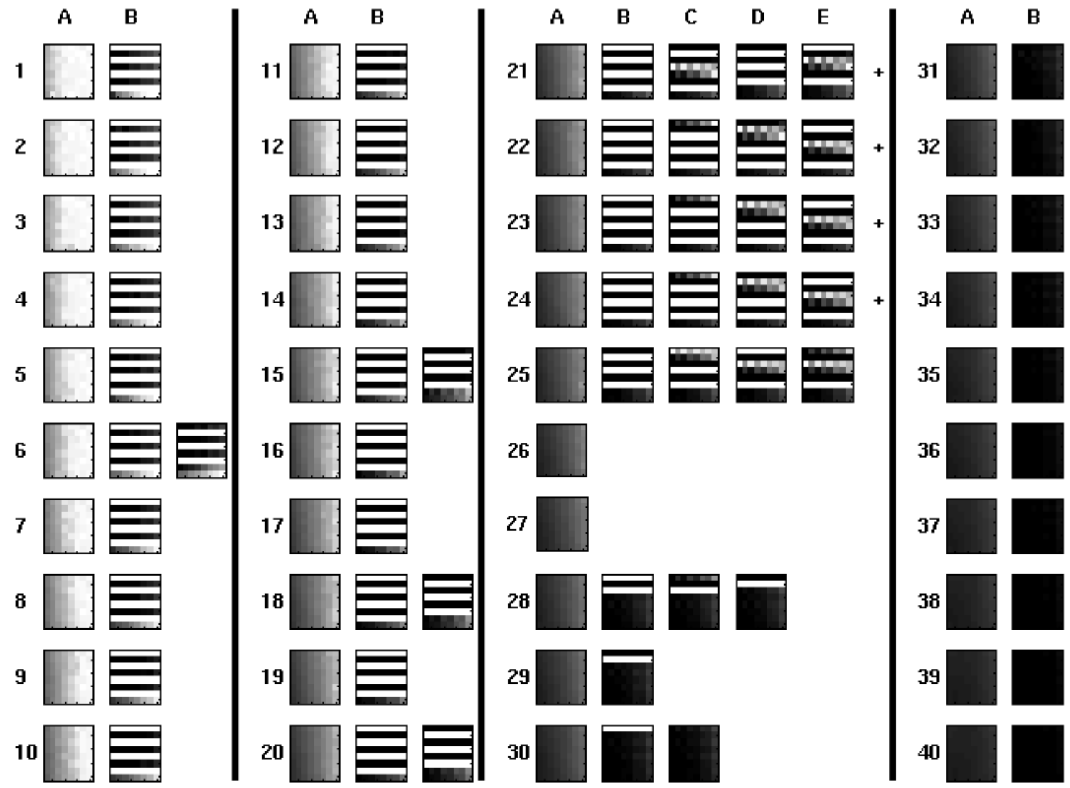

Figure 2.9: Stable output-patterns in the case of the first 40 input-frames. Column A contains the input pictures, Column B-E contain stable equilibrium points, which belong to the given input frame (if any). In accordance with Table 2.1, at the first 20 input frames only the 6th, 15th, 18th and 20th have two stable equilibria. In the cases of frames 26 and 27 there is no stable equilibrium point. The small "+" sign at the end of the 21st, 22nd, 23rd and 24th rows indicates that there are further stable equilibria, only the lack of space limits us here.

In Figure 2.9 we can clearly see that, during the first 20 input frames I have permanently one of the stable equilibrium points (of course they are 20 different EPs, but qualitatively, from an engineering point-of-view, they are really similar). In addition, at some input frames temporarily I have an other stable EP, which seems qualitatively the complement/opposite-pattern of the permanent one. What is more, on the basis of Table 2.1 I can observe that from frame-to-frame I have a really high fluctuating rate in the number of saddle-points. What can be the reason for this strange behavior? To uncover the details, first of all I would like to present the way I have computed the equilibrium points.

DOI:10.15774/PPKE.ITK.2014.007

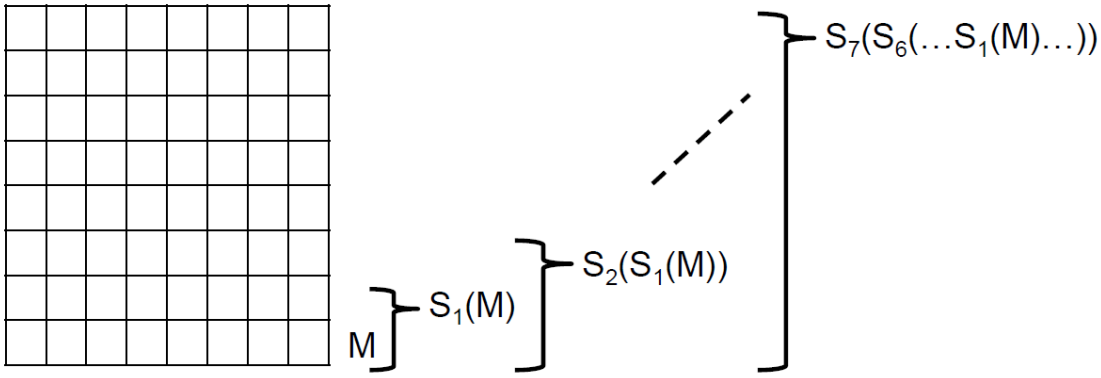

Figure 2.10: The schematic drawing of the 8×8 sized computing array can be seen on the left hand side. If we write down the equations of the underlying system of ordinary differential equations, we can group the equations in different dependence-ensembles. The bottom row works as an autonomous/independent dynamical system: the cells in it are connected only with each other. We can call it as the "master" of the system. The cells in the second row depend not only on each other, but also on the output of the appropriate cells below themselves. In this way we can say: the second row builds a "slave" system, depending on the bottom row.

Due to the special structure of the templates (there are only zeros in the top row of template A and template B), the cells are linked only with their left and right neighbors, plus with the bottom neighbor. In this way, I get a multiple-times repeating "master-slave" structure, as this dependence-relationship is depicted in Figure 2.10.

From this special "master-slave" structure it follows, that the waves ("effects") are propagating from bottom to top. The bottom row has only 1 stable equilibrium point (at least in the first 20 frames), the sometimes occurring duplication happens in the second row of the array. (In a lot of cases, it seems so that there is an other wave-direction on the computing array: from left to right.)

The basis of my equilibrium point computation is the division of the non-linear output function to its linear regions. In this way, every cell has three separate linear regions, resulting 3^N separate linear equation-systems in the case of N cells. In my case, every linear equation system has the form of Equation 2.12, where the coefficients of the saturated cells are built in c, the coefficients of the linear cells are in the matrix M. The solution (Equation 2.13) is based on the inversion of matrix M, where I have to check the newly computed x vector, whether the components of it get into the appropriate region (positively saturated, negatively saturated, linear), which was assumed during the set-up of Equation 2.12.

DOI:10.15774/PPKE.ITK.2014.007

$$Mx + c = 0 \tag{2.12}$$

$$x = M^{-1}(-c) \tag{2.13}$$

The stability of the solution can be defined with the eigenvalues of matrix M:

- if every eigenvalue has negative real part: the EP is stable (actually: attracting, a sink),
- if some of the eigenvalues have negative real part, some of them have positive: the EP is a saddle,
- if every eigenvalue has positive real part: the EP is unstable (actually: repelling, a source).

To make the structure clearer, now I will analyze only the first two rows on the bottom deeper. In this way I have a 16-dimensional system, but as I saw earlier: the second row produces the two different stable equilibrium points. The detailed description of my algorithm can be found in Appendix B.

2.5.2 Equilibrium points in the 16-dimensional system

In this subsection I will examine in details two input frames, the 7th and the 6th ones. I would like to kindly remind the Reader that, in the former case I have one stable EP, while in the latter case I have two stable EPs. (For the sake of simplicity, only the upper row will be presented in numerical form, because it represents the diversity on the top of the single stable equilibrium point of the bottom row.)

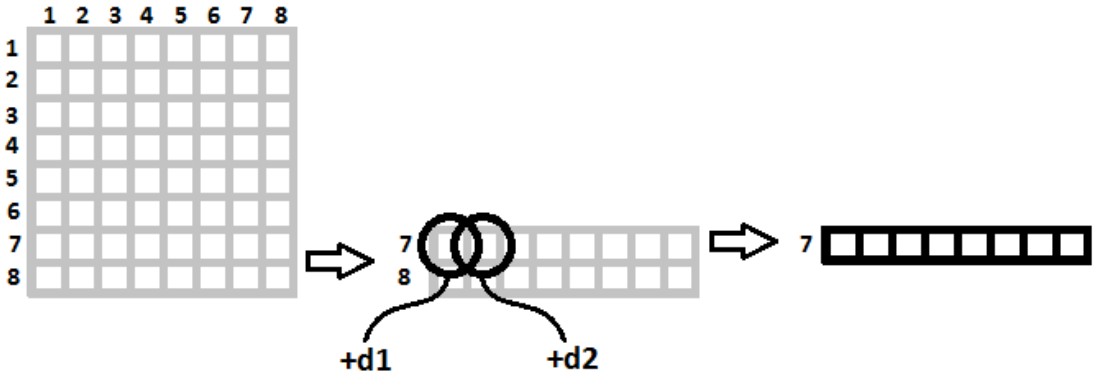

Figure 2.11: $d1$ and $d2$ as sweeping parameters throughout the detailed bifurcation-search experiments. $d1$ and $d2$ were used as additive terms in the cells' state (Equation 2.1). Only the state values of the 7th row are printed in Table 2.2 and Table 2.3, because the 8th row can express 1 stable point in its 8-dimensional subsystem.

DOI:10.15774/PPKE.ITK.2014.007

To clarify the strange fluctuation in Table 2.1 (the sudden/surprising change in the number of EPs), I have done some kind of smooth parameter sweeping to find regular bifurcation-curves. As Figure 2.11 shows, we have two additions on the positions $(7;1)$ and $(7;2)$ of the original array (in the following these additive parameters will be called $d1$ and $d2$), which can be identified as the first and second cells on the left hand side in the upper row of the 16-dimensional system. These two additive terms essentially modify my system (see Equation 2.1, the second double sum regarding Bu). The reason for choosing these two cells to modify is the directions of wave propagation on the array. At these two points I can do much smoother change in the values, than the difference between two captured images.

In the first case I would like to analyze the dynamics originating from the 7th input frame. According to Figure 2.9 and Table 2.1 I know that, here I have only one stable equilibrium point in the 64-dimensional system. Because the master-slave structure of the system (Figure 2.10), now we will see the 7th row of the 64-dimensional system. The 8th row will be fixed in its unique stable EP, only the emerging values in the 7th row will be shown. Table 2.2 and Table 2.3 summarize the possible EPs in the 7th row if the parameters are inside the ellipse shown in Figure 2.12, regarding the $d1d2$-plane. What we can see on the basis of Table 2.2 and 2.3 is the birth of a new stable equilibrium point. Figure 2.12 shows a clear structure with easily separable regions, where inside one region I can observe the same number of equilibria. This can be considered as a weak indication that the system is structurally stable, and the reason for the surprising changes in Table 2.1 is the unsuspected significant / qualitative difference between the visually quite similar input pictures. Furthermore, I can check the Euler-Poincare summation with the even versus odd parity convention in counting (Equation 2.14; equilibria-summation for differential equations, analogous to the Euler-formula for convex polyhedra), provided that the point infinitely far away is repulsive. In this case (eight-dimensional system) equilibria are classified according to the number of eigenvalues with positive real parts: if it is even, then it counts as $+1$, but if it is odd, then it counts as -1. I have checked Equation 2.14 along the whole $d1d2$-plane, and I have not observed any contradiction with it. In addition to this, I have checked the trajectories from the saddles in Table 2.3 with numerical simulations: they are not connected with each other (the first 6 saddles converge to the first stable equilibrium point, while the 7th saddle converges to the second stable EP).

DOI:10.15774/PPKE.ITK.2014.007

$$\#\{sources\} + \#\{sinks\} + \#\{saddles\} = 1 \tag{2.14}$$

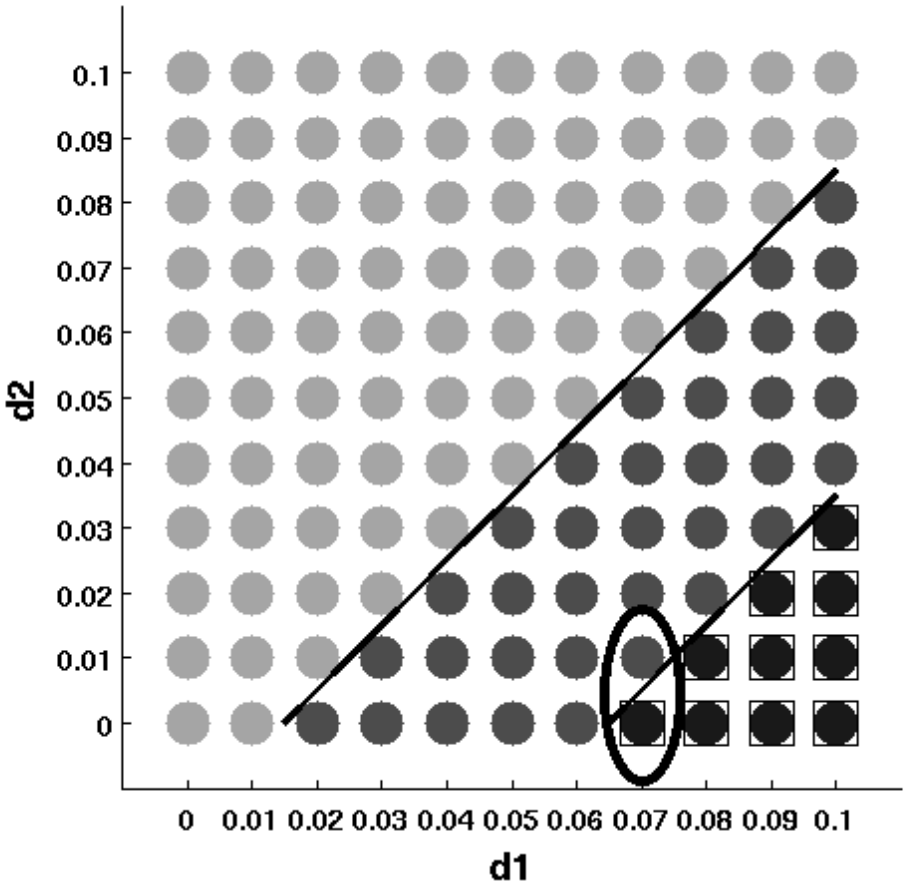

Figure 2.12: Bifurcation map regarding the 7th input frame. $d1$ and $d2$ were swept on the range $[0, 0.1]$. Light gray means 5 EPs, middle gray means 7 EPs, and dark means 9 EPs. There is only 1 stable equilibrium point on the plane, except those places which are framed with a small square (which actually fully coincides with the region where 9 EPs exist). The ellipse marks the place on the $d1d2$-plane, from where I present the numerical data in Table 2.2 and Table 2.3 regarding the EPs.

DOI:10.15774/PPKE.ITK.2014.007

Table 2.2: Equilibrium points, $d1 = 0.07$, $d2 = 0.01$ (see the ellipse in Figure 2.12). The rows represent different equilibrium points, which belong to the upper row of the 16-dimensional system (which is exactly the same as the 7th row of the 64-dimensional array, see Figure 2.10). The last column indicates the stability of the appropriate EP. The saddles have two different categories, namely "odd" and "even", where these words refer to the parity of the number of eigenvalues with positive real part. The boxes with light gray background mark the "together born" pairs: new pairs are born on the two sides of separating planes (conjecture: via saddle-node bifurcation).

-1.35272	-1.39692	-1.46778	-1.61709	-1.61327	-1.58997	-1.4613	-1.28179	stable
-0.679342	-1.04419	-1.46778	-1.61709	-1.61327	-1.58997	-1.4613	-1.28179	saddle, "odd"
-0.639166	-0.959824	-1.42358	-1.61709	-1.61327	-1.58997	-1.4613	-1.28179	saddle, "even"
-0.254091	-0.574749	-0.614925	-1.19351	-1.61327	-1.58997	-1.4613	-1.28179	saddle, "odd"
-0.078177	-0.398835	-0.439011	-0.824086	-1.41976	-1.58997	-1.4613	-1.28179	saddle, "even"
0.303424	-0.0172343	-0.0574105	-0.442485	-0.618399	-1.17021	-1.4613	-1.28179	saddle, "odd"
0.458162	0.137504	0.0973274	-0.287747	-0.463661	-0.845262	-1.29109	-1.28179	saddle, "even"

Table 2.3: Equilibrium points, $d1 = 0.07$, $d2 = 0.0$ (see the ellipse in Figure 2.12). The structure of this table is the same as of Table 2.2. In this table we are on the other side of a bifurcation curve, compared to Table 2.2. In the last two rows we have two new EPs: one saddle with an odd number of eigenvalues with positive real part; and one stable point.

-1.35272	-1.40692	-1.46778	-1.61709	-1.61327	-1.58997	-1.4613	-1.28179	stable
-0.679342	-1.05419	-1.46778	-1.61709	-1.61327	-1.58997	-1.4613	-1.28179	saddle, "odd"
-0.630075	-0.950733	-1.41358	-1.61709	-1.61327	-1.58997	-1.4613	-1.28179	saddle, "even"
-0.254091	-0.574749	-0.624016	-1.20351	-1.61327	-1.58997	-1.4613	-1.28179	saddle, "odd"
-0.069086	-0.389744	-0.439011	-0.814995	-1.40976	-1.58997	-1.4613	-1.28179	saddle, "even"
0.303424	-0.017234	-0.066501	-0.442485	-0.62749	-1.18021	-1.4613	-1.28179	saddle, "odd"
0.467253	0.146595	0.0973274	-0.278657	-0.463661	-0.836171	-1.28109	-1.28179	saddle, "even"
0.722789	0.402131	0.352864	-0.023120	-0.208125	-0.580635	-0.744464	-1.0007	saddle, "odd"
1.0007	0.678706	0.630075	0.253454	0.0690861	-0.30406	-0.467253	-0.723425	stable

In the second case, the dynamics originating from the 6th input frame will be considered. Figure 2.13 shows the bifurcation map on the $d1d2$ plane. Here both of $d1$ and $d2$ were in the range $[-0.1, 0.1]$. In this case I can observe also the well-defined regions for different numbers of equilibria. With the slight modification of $d1$ and $d2$ I can identify a region with only one stable equilibrium point. I have also checked Equation 2.14 on the whole $d1d2$-plane and I have not observed any contradiction with it. All of these arguments show in the same direction, namely that, for almost all parameters, our system is structurally stable, and the sudden/surprising changes in Table 2.1 are due to the differences between the (seemingly similar) input pictures.

DOI:10.15774/PPKE.ITK.2014.007

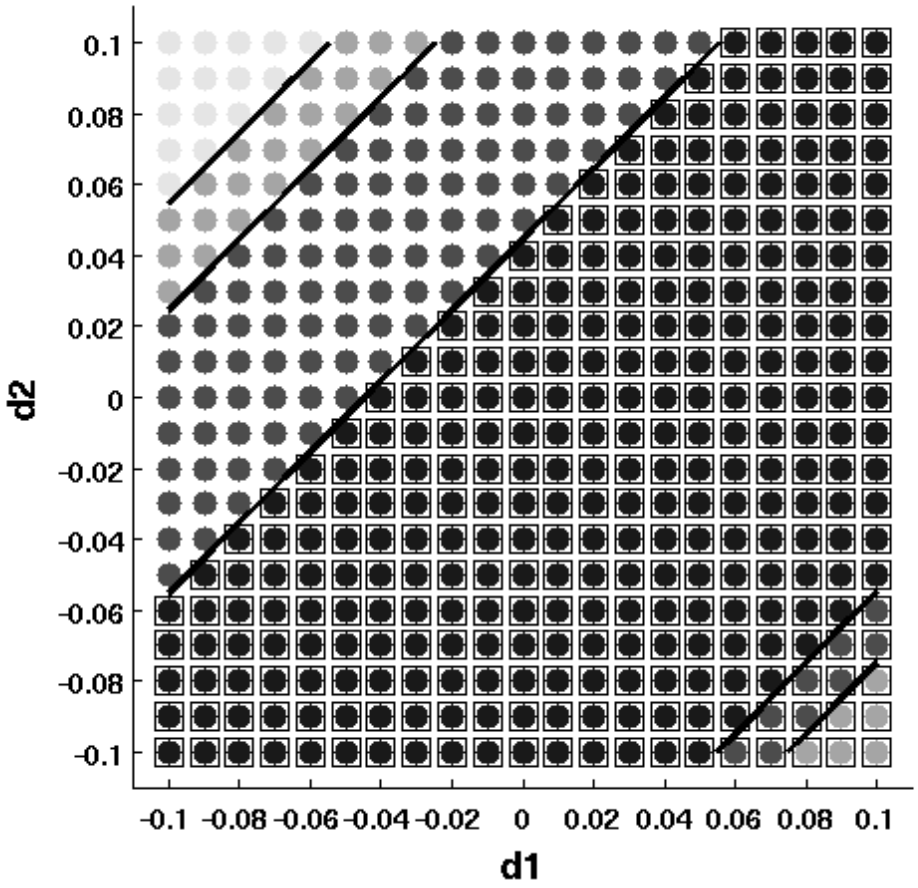

Figure 2.13: Bifurcation map regarding the 6th input frame. $d1$ and $d2$ were swept on the range $[-0.1, 0.1]$. Light gray means 3 EPs, the lighter middle gray means 5 EPs, the darker middle gray means 7 EPs, and dark means 9 EPs. There is only 1 stable equilibrium point on the plane, where the dots are not closed in a square, and there are 2 stable EPs where the dots are closed in a square.

2.5.3 Some numerical results on the basin of attraction

In the followings I will show some numerical results, which (according to my hope) can indirectly indicate the size or extent of attractive regions around equilibrium points. According to my opinion, the stable equilibrium point, which can appear as a "second" stable EP, has smaller basin of attraction, than the "first" one (which is present throughout the whole sequence). However, in the frame–by–frame case the special place / localization of the initial states can be inside these smaller regions resulting the convergence to these "second" stable EPs. This is really important, if I want to make somehow clear the original, strange behavior that I can observe in Figure 2.6.

DOI:10.15774/PPKE.ITK.2014.007

Two different cases will be investigated in the followings. In the first case, I use the 6th input frame as a basis, without applying any additive terms in $d1$ or in $d2$. In the second case, I will use the 7th input frame as a basis, with additive terms $d1 = 0.07$, $d2 = 0.0$ (this is the case inside the ellipse, in Figure 2.12, which I have carefully analyzed in the previous subsection from the viewpoint of EPs). In this way, both cases will have two stable equilibria.

Going into the details, I have to clarify the parameters of these simulations. Every time (in the simulations which will be referred in the rest of this subsection) the length of a single simulation is 400 time steps (assuming the cell time constant $\tau = 1$, this means 400 seconds). In every case, the sample size is 1000. In both of the main cases I will analyze the close (or not-so-close) vicinities of the input frame, of the "first" stable EP and of the "second" stable EP. 8-dimensional balls will be defined with different radii around these important points, and the initial states will be chosen randomly on the surfaces of these balls (more specifically: 1000 sample points from every different ball, to every different important point).

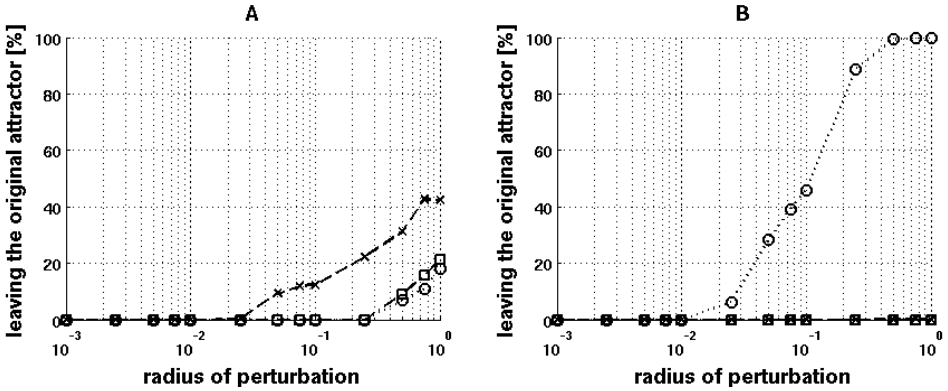

Figure 2.14: In this picture I illustrated the ratio of leaving the original attractor in the function of the increasing perturbation. Part A depicts the first main case (6th input frame with $d1 = d2 = 0.0$), while part B illustrates the second main case (7th input frame with $d1 = 0.07, d2 = 0.0$). The logarithmically scaled axes X indicates the increasing perturbation, while axes Y indicates the percentage of the leaving trajectories from their original attractor. Both in subfigure A and B, symbol 'x' marks the case when the system starts from the close vicinity of the appropriate input frame; symbol square marks the case when the system starts from the close vicinity of the "first" stable EP; while symbol circle refers to the initial point near to the "second" stable EP.

Figure 2.14 shows the leaving ratio of the original attractor in the function of the increasing perturbation. The figure on the left (A) involves the cases belonging to the 6th input frame, while the figure on the right (B) depicts the cases of the

DOI:10.15774/PPKE.ITK.2014.007

modified 7th input frame. Both of axes X show the increasing radii of the 8-dimensional perturbation-ball, while the different symbols refer to the different initial states. As we can see in the case of the unchanged 6th input frame (A), both of the stable equilibrium points have similar sized basins of attraction. In contrast to this, the modified 7th input frame (B) exhibits more asymmetrical basins: it is quite easy to go beyond the newly born second stable equilibrium point's basin of attraction.

The details as separate tables, and further comments can be found in Appendix C.

2.6 Other scenes and parameter regions

In this section I would like to describe a few scenes and parameter sets, which led me to the "final" template and the key example; however, with the vast majority of them the frameless computing mode was unable to outperform the frame–by–frame processing mode. This section has demonstrative purposes only, it is not a rigorous overview. I found a specific region in the parameter space, where the frameless processing provides the desired benefits for us, but unfortunately an arbitrary (randomly selected) scene not necessarily can be identified by this technique. Firstly I would like to show my initial template–values and describe the different scenes I have tested, then (omitting some not so expressive data–series) I would like to present the detailed data of a parameter-tuning experiment (which led me to the desired parameter values). All the details of this part of my work were published in [1].

$$\mathbf{T1}\text{: } s=1.1,\quad p=0.9,\quad q=-1.1,\quad r=-0.6,\quad b=1.0,\quad z=0 \tag{2.15}$$

$$\mathbf{T2}\text{: } s=1.1,\quad p=1.2,\quad q=-1.1,\quad r=-0.6,\quad b=1.2,\quad z=0 \tag{2.16}$$

$$\mathbf{T3}\text{: } s=1.1,\quad p=1.0,\quad q=-1.1,\quad r=0.3,\quad b=1.2,\quad z=0 \tag{2.17}$$

$$\mathbf{T4}\text{: } s=1.1,\quad p=1.4,\quad q=-1.1,\quad r=-0.35,\quad b=1.6,\quad z=-0.2 \tag{2.18}$$

On the basis of the presented parameter–regions in [17], I chose four parameter sets, presented here as templates **T1** – **T4** (see 2.15 – 2.18).

The investigated scenes were the followings:

– **scene 1**: a plain surface (parallel with the sensor array) moving inward to realize increasing covering rate with the panel. Other parts of the sensor array see "infinite" distance. (The object is moved by hand - rough steps.)

DOI:10.15774/PPKE.ITK.2014.007

- **scene 2**: like the previous input, but there is a counterpart plain surface, coming from the opposite side. Both of them move more and more underneath the sensor array. The area between them is seen as infinity by the sensors. (The object is moved by hand - rough steps.)
- **scene 3**: there are two plain surfaces on two opposite sides as well: one of them is still, the other one is moving more and more underneath the panel, toward the still one. (The object is moved by hand - rough steps.)
- **scene 4**: on the left side, there is a still, plain surface, along the whole width of the array, but on the opposite side, there is a narrow strip moving underneath the panel. (The object is moved by hand - rough steps.)
- **scene 5**: there is a still, plain surface, with approximately 40% masking/covering rate with the sensor array. The sensor array approaches from perpendicular direction to the scene. (The array is moved by machine - fine steps.)
- **scene 6**: there is a still, convex surface under the sensor array. The sensor array approaches from perpendicular direction to the scene. (The array is moved by machine - fine steps.)
- **scene 7**: there is a still, convex surface, and the sensor array moves from the margin to above it. (The array is moved by machine - fine steps.)
- **scene 8**: there is a still, concave surface, and the sensor array moves from the margin to above it. (The array is moved by machine - fine steps.)
- **scene 9**: there is a still, concave surface under the sensor array. The sensor array approaches from perpendicular direction to the scene. (The array is moved by machine - fine steps.)

These scenes were processed both with the frame–by–frame and with the frameless method. To evaluate the differences between the two output dynamics, I introduced four labels/measures and two derived ratio numbers. These were the followings:

- **[matching]**: the output dynamics of the two different systems are the same, when the frameless mode processes the next input frame of the frame–by–frame processing;

DOI:10.15774/PPKE.ITK.2014.007

– **[unsteady matching]**: a case of [matching], the two different systems have the same output dynamics; however, at the previous frame of the input flow they had different output dynamics;

– **[preservation]**: this attribute is defined in connection with the frameless method: the output dynamics does not change during the change of input frames;

– **[matching with preservation]**: in this case, [matching] and [preservation] occur at the same time: the output dynamics of the two different systems are the same, and there was no change in the output of the frameless mode during the update of the input frames;

– $\boldsymbol{\alpha}$: [unsteady matching] / [measurement case number], in my opinion this ratio expresses the equivalence level of the two systems: if this number is high, the frameless system is rather an input-dependent system;

– $\boldsymbol{\beta}$: ([preservation] - [matching with preservation]) / [measurement case number], according to my opinion this ratio expresses the level of dependence of the frameless system on the input: if this number is high, the frameless system is rather an autonomous system.

Unfortunately the derived ratio numbers did not show any global, significant pattern regarding specific template or specific scene. In the next step, I wanted to find an equilibrium between the "input–dependent" and the "autonomous" behavior. For this reason, I chose template **T1**, and swept the central element p of the feedback matrix A in the range [0.8, 1.4] (the other parameter values in **T1** remain unchanged). As we can see on the results in Table 2.4, the lower p values resulted in rather input–dependency, while higher p values showed rather autonomous behavior. The $p = 1.0$ value seems the most appropriate one, since in this case the α and β ratios seem equalized.

DOI:10.15774/PPKE.ITK.2014.007

Table 2.4: Sweeping the central-element (p) of the feedback matrix (A) in the case of template **T1**. The rows of the table denote the different measurement environments, the columns denote the used p value during the measurement, each of them is divided into two subsections to the index-numbers (α, β).

	$p=0.8$		$p=1.0$		$p=1.1$		$p=1.2$		$p=1.3$		$p=1.4$	
	α	β	α	β	α	β	α	β	α	β	α	β
scene 1	42%	16%	26%	37%	21%	42%	10%	74%	5%	89%	5%	89%
scene 2	43%	7%	36%	21%	29%	50%	14%	57%	7%	79%	7%	93%
scene 3	55%	10%	10%	55%	10%	45%	5%	5%	5%	0%	5%	0%
scene 4	63%	0%	6%	13%	6%	0%	6%	0%	6%	0%	6%	0%
scene 5	43%	11%	27%	24%	27%	27%	14%	30%	16%	43%	22%	54%
scene 6	33%	28%	18%	41%	21%	33%	15%	33%	15%	46%	10%	51%
scene 7	39%	11%	21%	34%	16%	37%	18%	39%	13%	45%	5%	68%
scene 8	68%	12%	32%	32%	24%	24%	20%	28%	16%	76%	12%	80%
scene 9	48%	15%	21%	27%	15%	30%	18%	48%	9%	45%	15%	55%

Throughout this section I presented different scenes and template parameter sets, with which I tried to achieve qualitative benefits of the frameless processing method. Later on, in the convex–bump identifying experiment I applied the $p=1.0$ value in **T1**, and I also changed the value of parameter r to -0.7, in which case I was able to unambiguously solve that identification problem.

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

Chapter 3

Measurement range tuning and complex movement detection with the depth measuring sensor array

In this chapter I present firstly a locally adaptive algorithm for the tuning of the measurement range of a depth measuring sensor array. Then, I would like to uncover the details of two CNN algorithms developed for depth measuring sensor arrays: one of them detects moving objects at constant speed but varying direction in space, while the other detects objects performing tilting movement. All of these algorithms were done only in simulation, in an environment differing (both in size and in operation) from my real infrared sensor array.

3.1 Measurement range tuning

I would like to kindly remind the reader that, my measurement-composition is interpreted in the depth-extension of the two-dimensional plane, assuming that the quality of the surfaces and the reflection coefficients are the same everywhere. The main goal is to adapt the different parts of the sensor array differently to make measurable the too wide depth dynamic-range.

In the literature of image processing, mapping of high dynamic range scenes is a well known problem. In the case of regular pictures, the different brightness-conditions on the different parts of the pictures make difficulties: one part of the pictures can be underexposed while the other part can be overexposed. There are more solutions for this problem in the literature [39], which problem is highly analogous to my depth-adaptation problem. Especially the doctoral work of Robert Wagner is relevant, who has solved this problem on a very similar computational architecture (CNN Wave Computer)[40]. In his work, the chip is a true vision chip,

DOI:10.15774/PPKE.ITK.2014.007

where the local adaptation is done by the fine tuning of the integration time of every sensing pixel. In our case, the increase/decrease of the light power works analogous to this.

The description of our algorithm:

- the increasing of the light power (I) can reduce the measured distance (d_m) of the farther objects
- the decreasing of the light power (I) can raise the measured distance (d_m) of the closer objects

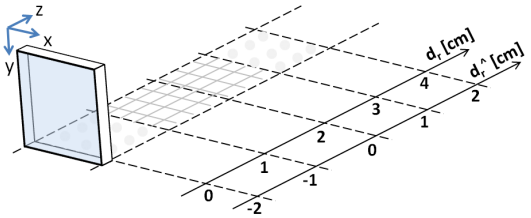

Figure 3.1: Schematic drawing of measurement ranges. The diagonal-crossing pattern denotes the measurement range with medium level light power, while the dotted areas indicate the extended measurement range. The scales of d_r and $\hat{d}_r$ stand for the real distance and the shifted real distance of the measured objects, respectively.

In Figure 3.1 we can see the default measurement range, and the two new regions obtained with light power increasing and decreasing. This measurement range extension makes measurable the double size of the original measurement range.

Notations:

- real distance: d_r; $d_r \in [0, 4]$cm (*interpretation: 0-closest, 4-farthest*).
- shifted real distance: $\hat{d}_r$; $\hat{d}_r = d_r - 2$cm; $\hat{d}_r \in [-2, 2]$cm
- light-power: I; $I \in [-1, 1]$ (*interpretation: -1-minimum light, 1-maximum light*). To set up appropriately the light-power on the hardware, we will use the following procedure:
  SetLightPower(I) – this procedure has not got any special role/function in the simulation, it indicates only the point in the algorithm, where the light power of the assumed sensor array should be modified.

DOI:10.15774/PPKE.ITK.2014.007

– measured distance: d_m; $d_m \in [-1, 1]$ (*interpretation: -1-closest, still measurable, 1-farthest, still measurable*). The measured distance depends on the real distance and on the applied light-power ($d_m = d_m(d_r, I)$). In this characteristic-exploration simulation, I will use the following procedure: $d_m = \text{ReadSensorValues}() = f_{pwl}(\hat{d}_r - I)$ – this procedure assumes that the applied light power and the measured distance have linear dependence with each other (between the two saturation endpoints). Of course this is not the real situation, but in a real measurement the nonlinear to linear correction can be wrapped in the *voltage – light power* characteristics of the light-sources (not analyzed deeper here).

Algorithm 1 Measurement range alignment

procedure AdaptLocally(d_m)
  $I(0) \leftarrow I_{max}/2$ ▷ medium level light power, equals zero
  SetLightPower($I(0)$)
  $d_m(0) \leftarrow$ReadSensorValues(0)
  $m \leftarrow 0.5$ ▷ slope multiplier in the characteristic; constant
  $th(0) \leftarrow th(1) \leftarrow 0.7$ ▷ threshold value in the characteristic; it will be increased
  $iternum \leftarrow 15$
  for $i \leftarrow 1, iternum$ **do**
    if $(d_m(i-1) < -th(i))$ **then** ▷ too close
      $I(i)' \leftarrow I(i-1) - m|d_m(i-1) + th(i)|$
    end if
    if $(-th(i) \leq d_m(i-1) \leq th(i))$ **then**
      $I(i)' \leftarrow I(i-1)$
    end if
    if $(d_m(i-1) > th(i))$ **then** ▷ too far
      $I(i)' \leftarrow I(i-1) + m|d_m(i-1) - th(i)|$
    end if
    $I(i) \leftarrow f_{pwl}(I(i)')$
    SetLightPower($I(i)$)
    $d_m(i) \leftarrow$ ReadSensorValues(i)
    $th(i+1) \leftarrow th(1) + (1 - th(1))(i/iternum)$
    if $(|d_m(i) - d_m(i-1)| \leq 0.02$ **OR** $|I(i) - I(i-1)| \leq 0.02)$ **then**
      break
    end if
  end for
end procedure

DOI:10.15774/PPKE.ITK.2014.007

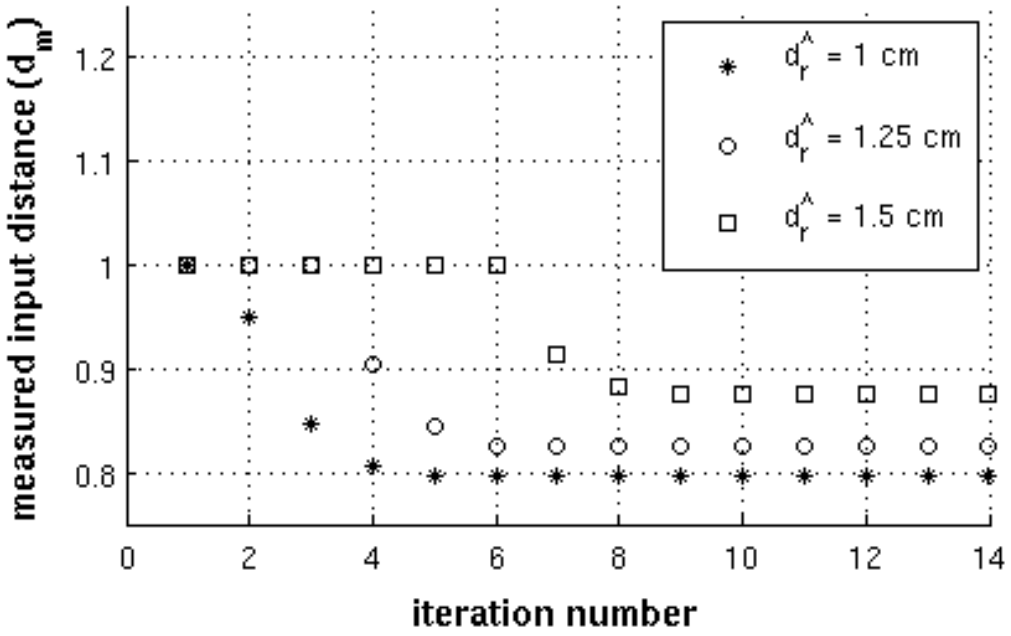

Figure 3.2: The successive change of the measured distance values as the iterative algorithm goes on. The measured distance values (d_m) belong to three different, shifted, real distance values ($\hat{d}_r = \{1\text{cm}, 1.25\text{cm}, 1.5\text{cm}\}$). The measured distance values (d_m) are along axis y, while the iteration number along axis x. (The real distance is measured perpendicularly from the sensor array, the zero point of the linearly shifted scale ($\hat{d}_r$) is at the center point of the dynamics range measured with medium light strength.)

As an illustrative picture of the algorithm see Figure 3.2. Here the measured distance (d_m) is plotted in the function of the iterations, in the case of three different real distances ($\hat{d}_r = 1\text{cm}$, $\hat{d}_r = 1.25\text{cm}$, $\hat{d}_r = 1.5\text{cm}$). Starting from medium-level light-strength, the points, which are closer to the maximal level of the measurable-distance, can be reached with fewer iteration steps than the farther ones. The purpose of this picture is to demonstrate the iterative approximation of the initially unmeasurable distances.

DOI:10.15774/PPKE.ITK.2014.007

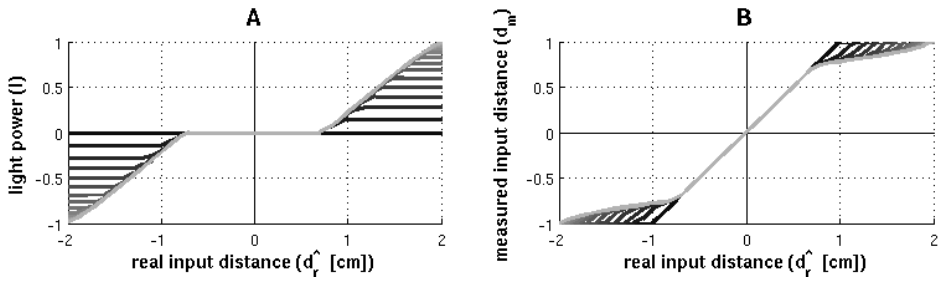

Figure 3.3: A) Light power characteristics during the progress of iterations; B) Measured input distance characteristics during the progress of iterations.

Figure 3.3 depicts the evolutions of the characteristics. The whole real distance ($\hat{d}_r$) is on axis x on both subfigures. The same colors on both subfigures belong together, representing the forthcoming iterations during the refinement algorithm (as the number of the iterations gets higher, the color of the line gets lighter). On Subfigure A) we can see the iterative change of the light-power (I) while on Subfigure B) we can observe the evolving measured distance (d_m) belonging to the updated light-strength. It is important to note that, I consider the measured input distance (d_m) as a dimensionless quantity. Although I can define an unambiguous correspondence with the nonlinear characteristics of *measured input distance(shifted real input distance)* (more precisely: $d_m(\hat{d}_r)$, as depicted on Figure 3.3/B); the aim of this algorithm is at the extension of the sensing range, and not at the accurate measurement on the extended sensing range. The algorithm stops the execution either if we reached the maximal number of allowed iterations (in the pesudo code above this means 15 iterations), or the last change of the measured distance or the last change of the light power is less than 0.02. In this sense, the algorithm every time converges to a specific value, which is either the nonlinearly mapped value of the real input distance inside the measurement range, or the saturation threshold itself. (The size of the measured distance refinement steps are higher rather than the maximal number of steps would not be enough.)

3.2 Detection algorithms with delayed CNN

3.2.1 Computational environment

The algorithms presented in this Section are based on standard delay-type templates [32], [62]. Here, the detections are realized without the frameless processing mode, just the appropriately chosen difference between the input frames which

DOI:10.15774/PPKE.ITK.2014.007

defines the temporal behavior. According to this, the delayed state equation (from the CNN literature) is described by Equation 3.1.

$$\begin{aligned}\dot{x}_{ij} = -x_{ij}(t) &+ \sum_{|k-i|\leq rd}\sum_{|l-j|\leq rd} A(i-k,j-l)y_{kl}(t) \\ &+ \sum_{|k-i|\leq rd}\sum_{|l-j|\leq rd} A^{\tau}(i-k,j-l)y_{kl}(t-\tau) \\ &+ \sum_{|k-i|\leq rd}\sum_{|l-j|\leq rd} B(i-k,j-l)u_{kl}(t) \\ &+ \sum_{|k-i|\leq rd}\sum_{|l-j|\leq rd} B^{\tau}(i-k,j-l)u_{kl}(t-\tau) + z\end{aligned} \tag{3.1}$$

As extending the standard CNN state equation (Equation 2.1), we have here additionally A^{τ} and B^{τ} as delayed feedback and delayed input synaptic operators, respectively. In the following algorithms I highly relied on the Template Library [34], only small modifications were necessary in some cases to achieve the results.

3.2.2 Simulation scene

The two algorithms will be presented in a common simulation scene. In this way I can see the specificity of them: none of them will detect other forms of movement except the desired one. The scene is built up of five objects, as depicted in Figure 3.4. The whole scene stands in front of the sensor array, which contains (in these simulations) 55×55 cells. The sensor array does not move, only the objects carry out different kinds of movements. The upper left rectangle presents the tilting movement. Under it, on the left center there is a still object. On the right center there are two rectangles: one of them is approaching the sensor array (from a perpendicular direction), the other one is going away from the sensor array (in a perpendicular direction). The object on the bottom moves at constant speed in space, from the left to the right. If we see it from the viewpoint of the sensor array, after a short parallel movement, first it increases its distance from the sensor array, than preserves its higher distance, then decreases its distance back to the original level. Finishing the excursion in the depth direction, the object turns back and finally gets closer to the sensor array.

DOI:10.15774/PPKE.ITK.2014.007

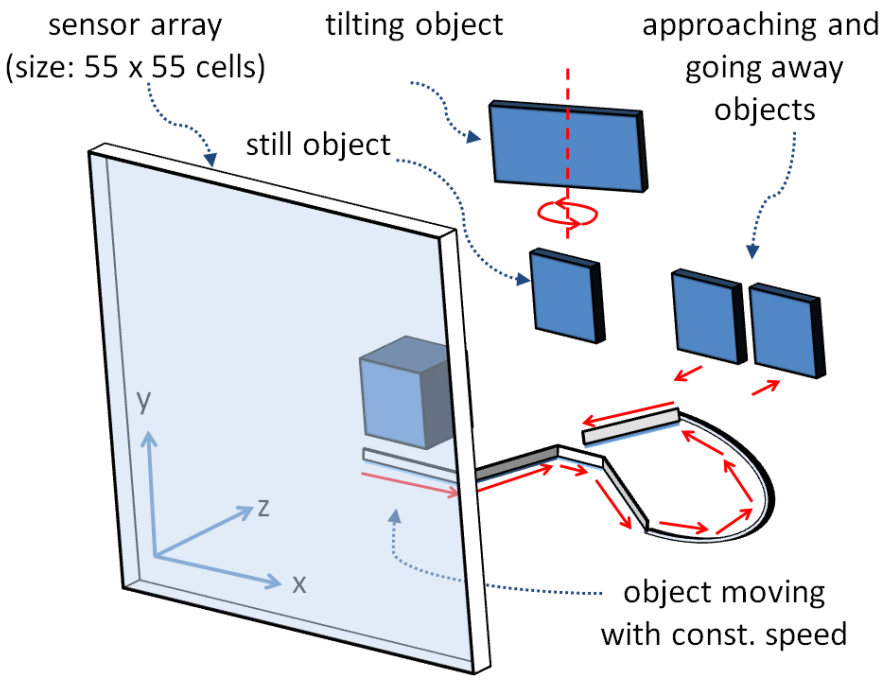

Figure 3.4: Simulation scene to test the detection algorithms. In the scene we can see a standing object, a tilting object, a perpendicularly moving pair of objects (getting closer and going farther), and an object moving on a spatial trajectory at constant speed.

In Figure 3.5 we can see the X, Y and Z position changes of the different objects as functions of time.

DOI:10.15774/PPKE.ITK.2014.007

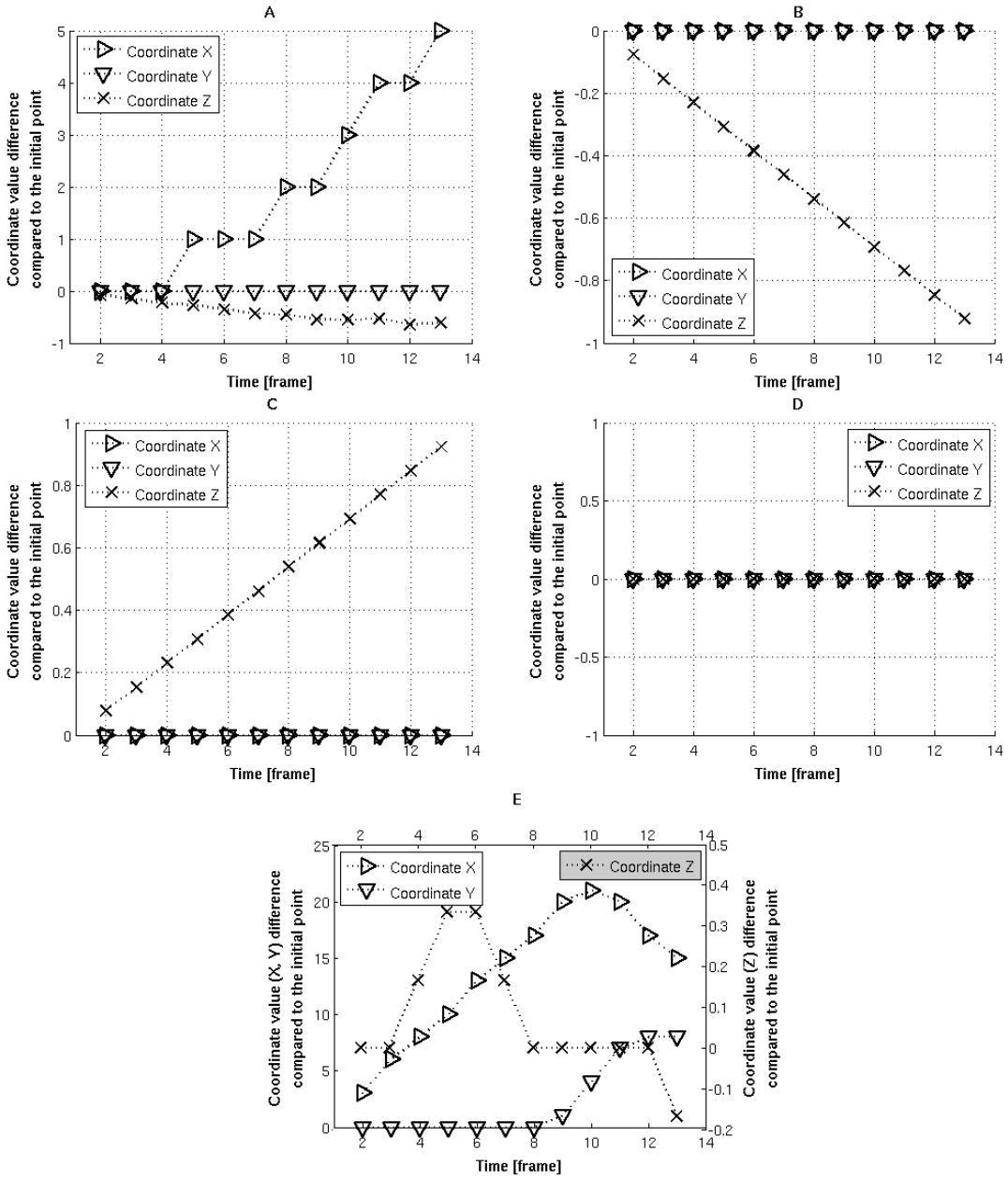

Figure 3.5: Different objects' coordinates as functions of time. Figure A belongs to the top left corner of the tilting object; Figure B and C belong to the approaching - going away objects pair, respectively; Figure D belongs to the still object; Figure E belongs to the object moving with constant speed. In every small figure, the right directed triangle means coordinate X, the bottom directed triangle means coordinate Y, and the symbol 'x' means coordinate Z (depth).

DOI:10.15774/PPKE.ITK.2014.007

3.2.3 Detection of objects moving at constant speed but in varying direction

In the followings my detection algorithm will be uncovered, which can recognize objects moving at constant speed in almost arbitrary direction. The scene, on which the measurement is simulated, can be seen in Figure 3.4, the precise coordinate functions in Figure 3.5.

The first step is the identification of the different base-components of the resultant velocity-vector. In this way, taking the summed squares of them, I can check whether the magnitude of the computed resultant is equal with the predefined constant value or not. This is described formally by Equation 3.2.

$$v_{res} = \sqrt{v_x^2 + v_y^2 + v_z^2} = \text{const.} \tag{3.2}$$

Because our sensors measure in the depth-direction, I can measure the velocity component in the "z" direction as easy as in the lateral directions ("x" and "y").

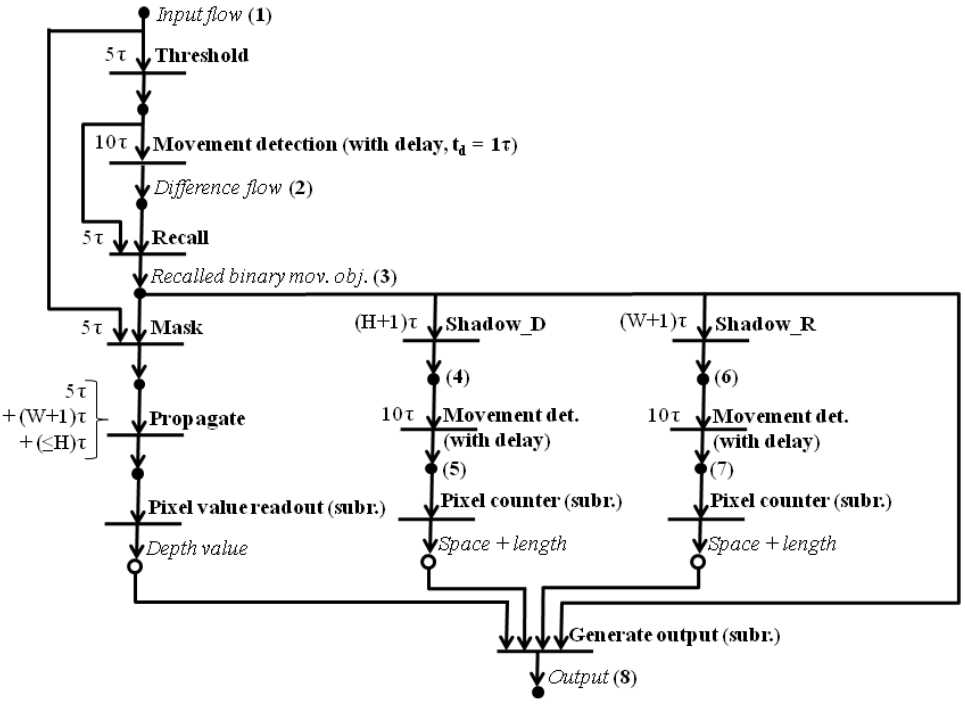

Figure 3.6: UMF diagram of the constant speed detection algorithm. Horizontal lines represent the different template-executions (subroutines) with the necessary execution time (expressed in the unit of the cells' time constant τ; W and H means the width and height of the processed image), while dots represent images/flows. The numbers next to some flow-point are links to Figure 3.7, where the result pictures of these stages are presented.

DOI:10.15774/PPKE.ITK.2014.007

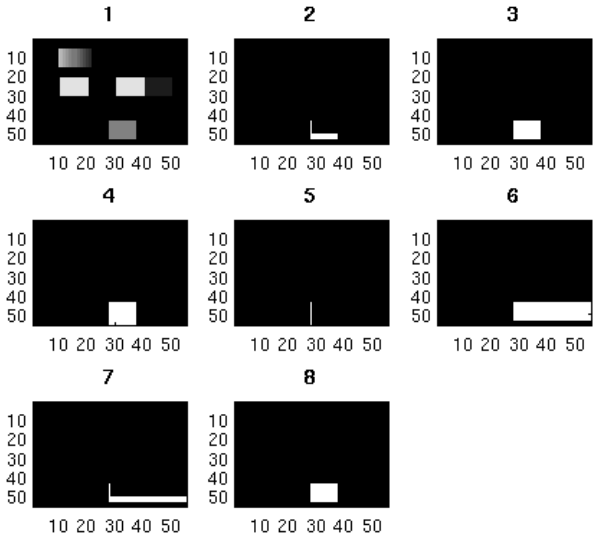

Figure 3.7: Different stages of the constant speed detection algorithm. The stages can be identified with the numbers in Figure 3.6, where the detailed processing-algorithm is presented. The time resolution of this image (here I mean time differences between the appropriate stages) can be expressed in the unit of the cells' time constant τ, the different stages of Figure 3.6 call for different processing time.

The UMF diagram (Universal Machine on Flows diagram – a representation of Virtual Cellular Machines) of the whole algorithm can be seen in Figure 3.6, while some of the inner stages/output-flows in Figure 3.7. The three main branches of the processing (measuring the depth-, the horizontal- and the vertical components of the movement) can be separately seen in Figure 3.6. The individual steps of this UMF diagram are coming either from the CNN Template Library [34] as a basic operation, or are created by me denoting small subroutines to complete the whole process. At the end of the three main branches I get two scalar values, which express the magnitude of the appropriate velocity-components. Analyzing the summed squares of them, I can state whether the object has the desired predefined constant speed or not.

At this point, I have to mention two main drawbacks of this algorithm. The first weakness is that the movement has to contain every time at least a minimal planar component, the solely depth movements are lost in the first difference flow computation. The other weakness is in connection with the propagation-type operations (**Propagate** and **Shadow** on the UMF): this algorithm can handle only one target in a given flow at the same time. Otherwise, during the propagation-type operations the different waves are merging, leading the computing array to confused projections. But, if the input flow contains only one moving object, the

DOI:10.15774/PPKE.ITK.2014.007

algorithm can detect it.

The speed-range of the detected moving object depends on different factors. The **Generate output** routine does the fine-tuning, namely, it defines the acceptance level and the accepted variance of the magnitude of the computed resultant. In this simulation, I leaned on the physical size and parameters of the existing sensor array. This means $1cm$ planar resolution in front of the panel, while the depth range is extended roughly between $1cm$ and $40cm$ (the existing sensor array can measure with confidence between $1-3cm$). Currently this algorithm detects the objects moving with the speed of $3cm/frame$, but modifying only the **Generate output** routine this can vary in the same order (approximately $1-10cm/frame$, assuming that the object displacement between two consecutive frames is not larger, than the size of the object). Considering the small value of the cells' time constant, this can be really fast. This value can be decreased, if the system omits regularly some of the input frames. Raising the delay (t_d) in the case of the **Movement detection** templates can decrease the detected speed of the moving object within limits.

The execution time of this algorithm can vary, depending on the underlying real architecture. If the executing architecture can parallel compute the branches, the whole algorithm takes 142τ (knowing that the size of the array is 55×55 cells). Of course this value does not contain the memory access time and the boundary row/column read-out time, and assumes some inner image memory, to store the necessary intermediate steps. If the execution parallellism is not present (as this is the case with the most sensor-processor arrays), we need more local image memories in order to store the partial results of the different branches. In this latter case, the algorithm takes 276τ (again, the size is *width*=*height*=55) for the execution. It is hard to estimate the *real* running time of this algorithm: unfortunately no such device exists, where a depth measuring sensor array is combined with a cellular processor array; I can lean on the performance data of similar cellular sensor-processor arrays only. For example, in Chapter 3 of [34] different types of architectures (Texas DaVinci DSP, Spartan 3 FPGA, Q-Eye and Xenon chips, IBM Cell, NVidia GTX280 GPU) are compared. On the basis of this comparison, I estimate the cells' time constant τ of a theoretical depth measuring sensor-processor array between $1-20\mu s$, resulting in (with an estimated array size of 128×128) $(5+11+5+5+5+129+128+129+11+129+11)*20\mu s = 568*20\mu s = 11.36msec$ execution time (approximately 88 fps). Of course these numbers are rough estimations, without knowing any bottleneck or constraint of the real implementations

DOI:10.15774/PPKE.ITK.2014.007

of a depth measuring sensor-processor array.

3.2.4 Detection of tilting-movement

During an approaching / going-away movement, the gradient on the difference-flow at the location of the moving object is uniform, only the edges produce salient value. Unlike approaching / going-away movements, the tilting movement has non-uniform surface on the gradient map of the difference flow (for the details: see the first sub-figure in Figure 3.9). I can detect it easily (there are more options for it), and I can use this attribute to identify tilting objects on the dynamically changing scene.

The UMF diagram of the complete algorithm can be seen in Figure 3.8, and snapshots from a running algorithm in Figure 3.9. The outputs of some processing-stages can be identified with the numbers after them in Figure 3.8.

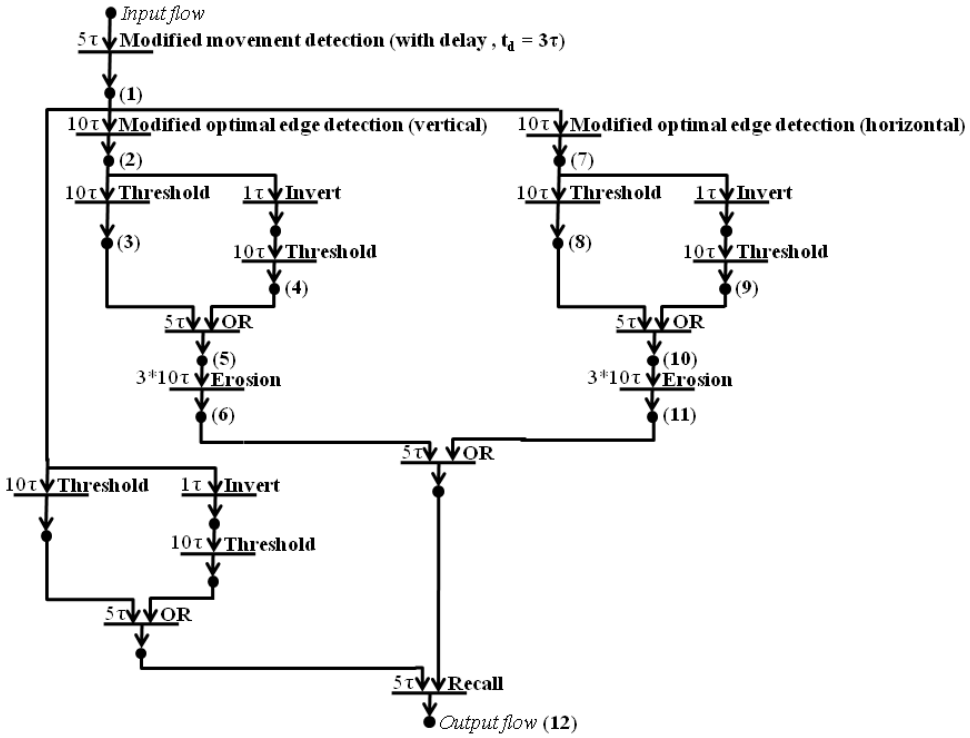

Figure 3.8: UMF diagram of the tilt-detection algorithm. Horizontal lines represent the different template-executions (subroutines) with the necessary execution time (expressed in the unit of the cells' time constant τ), while dots represent images/flows. The numbers next to some flow-point are links to Figure 3.9, where the result pictures of these stages are presented.

DOI:10.15774/PPKE.ITK.2014.007

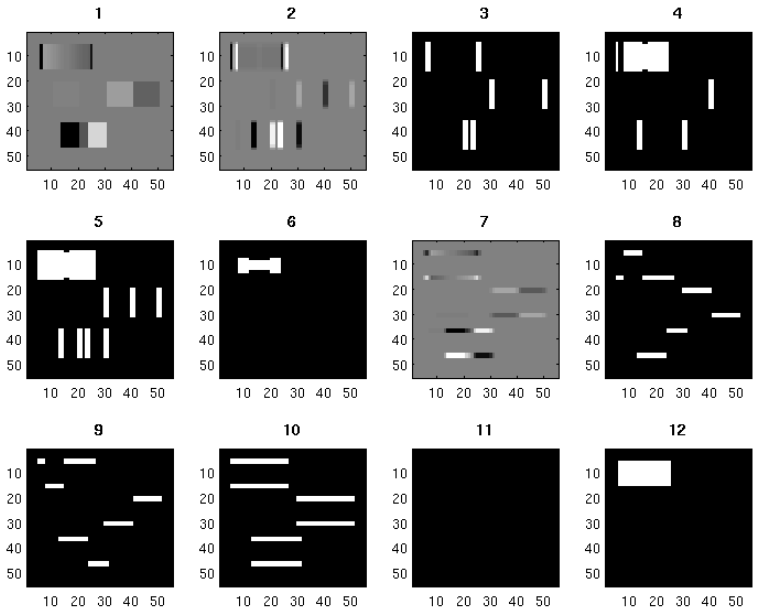

Figure 3.9: Different stages of the tilt-movement detection algorithm. The stages can be identified with the numbers in Figure 3.8, where the detailed processing-algorithm is presented. The time resolution of this image (here I mean time differences between the appropriate stages) can be expressed in the unit of the cells' time constant τ, the different stages of Figure 3.8 call for different processing time.

The main advantage of this algorithm is that, it can detect multiple tilting objects at the same time on the same flow. In addition to this, with the variation of the delay-value in the case of the first template-execution, I can detect a wide-range of rotating speed. Currently this algorithm detects approximately the rotations around 5°/frame, but considering the small value of the cells' time constant this can be really fast. We can either choose smaller delay in the first step (eg. $t_d = 1\tau$), or we can left out regularly some of the input pictures to achieve slower rotation speed detection.

The defined templates can be found in the Template Library [34], the three exceptions are as follows:

– **Modified movement detection (with delay):**

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ A^{\tau} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ B^{\tau} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ z = 0$$

The boundary condition is fixed-zero.

DOI:10.15774/PPKE.ITK.2014.007

– **Modified optimal edge detection (vertical):**

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ B = \begin{bmatrix} -0.3 & 0 & 0.3 \\ -0.75 & 0 & 0.75 \\ -0.3 & 0 & 0.3 \end{bmatrix};\ z = 0$$

The boundary condition is zero-flux.

– **Modified optimal edge detection (horizontal):**
It has the same values as the *vertical* detection, except that the values of matrix B are rotated around the central element with 90 degrees.

(Please note that none of these templates are delayed except the first one. This template extracts the temporal characteristic of the input flow, all of the other templates work either on simple images, or on the difference picture produced by this delayed template (extracting spatial features in both of the latter cases). This means, that Equation 3.1 is the general formula of the dynamics executed by my template-sequences, in most cases the delayed feedback (A^τ) and the delayed input synaptic (B^τ) operators are zero matrices.)

Execution time estimations of this algorithm can be also given. Fortunatelly this algorithm does not contain propagation-type operations, meaning that the execution time is independent from the size of the array. Assuming parallel execution of the separate branches, the whole algorithm takes 84τ, but I have to emphesize, this is not the realistic case by sensor-processor arrays. The serialized length of the algorithm is 176τ, which corresponds to $176*20\mu s = 3.52msec$ theoretical execution time (approximately 284 fps). At this point I would like to emphasize again, this is just an estimation, on the basis of the performance of other sensor-processor arrays.

DOI:10.15774/PPKE.ITK.2014.007

Chapter 4

Long transient metastable oscillations

In this chapter I would like to present my simulation results and numerical computations regarding the long transient metastable oscillations. As a first step after the review of the literature, I will describe this dynamical system as well as the characteristic oscillations arising in simulations. Then, I will show the details of my numerical eigenvalue computations in the context of a robustness analysis.

4.1 Review of the related literature

According to my best knowledge, the observation and analysis of long transient oscillations are relatively novel topics in the field of nonlinear dynamics and circuit theory. As (probably) the most similar research, I would like to mention the work of Yo Horikawa and Hiroyuki Kitajima (for example: [18],[19]), their study is on long transient metastable oscillations in an experimental electrical ring of oscillators. The authors have carefully analyzed the observed phenomenon in their system, figured out the kinematical model of the arising oscillations, and described heuristical estimations on the convergence of the waveforms and on the lengths of the transients. They built several different laboratory circuit prototype (with different cell activaton functions and interconnecting lines). The most significant difference between their works and ours is the different kind of output function at the elementary cells/neurons. More precisely, they apply (different kinds of) smooth output functions, while in our system the cells have the piecewise linear activation function.

The extensive studies (about the convergence properties of this system) of the Circuit Theory Research Group in Siena are probably the most strongly connected

DOI:10.15774/PPKE.ITK.2014.007

publications with this part of my dissertation, see [21], [22], [23], [24] and [25].

As a very comprehensive study on some dinamical behaviors of one- and two-dimensional CNN arrays, I would like to refer to Patrick Thiran's doctoral work [64]. The author carefully analyzes the arising, long lasting phenomena (for example, the long transient formation of the shapes between different regions), and analyzes the underlying dynamical system with current techniques (for example, Harmonic Balance technique). There is no direct connection between the long transient metastable oscillation and the work of Patrick Thiran, but some aspects of the different problems are closely analogous to each other.

In the autumn of 2011, I have heard a fascinating lecture of Zachary Peter Kilpatrick about dynamics in neural fields, where both the elementary mathematical model of the structures, and the interaction of dynamics were presented (for more details, please see [65]). In a rigorous sense, this work is really loosely coupled to the long transient metastable oscillations, but the accurate and expressive description of some low-level neural structure and mechanism inspired my thoughts into more biological (more bio-inspired) direction.

As an other long transient phenomenon, I would like to mention the work of Mária Ercsey-Ravasz and her colleagues in [16]: they solve constraint satisfaction problems with asymmetric continuous-time neural networks, where the inner state of the system can stay quite a long time in a transient chaos, before converging to an asymptotically stable equilibrium.

4.2 System description

The phenomenon is observed in a one-dimensinal CNN array with periodic boundary condition (a ring). The cells have first order dynamics, as described earlier by Equation 2.1, with the well known output function in Equation 2.2. In this case, only the feedback template A contains elements different from zero, as described by Equation 4.1.

$$A = [\alpha\ 0\ \beta], \alpha > 0, \beta > 0, \quad B = [0\ 0\ 0], \quad z = 0 \tag{4.1}$$

In the earliest experiments by *Luca Pancioni* and *Tamás Roska*, $\alpha = 3.5$ and $\beta = 2.5$ was chosen for the simulations, also with the initial condition $N/2$-times $\{+1\}$ and $N/2$-times $\{-1\}$ (N is even). With this setup, in an array with the size of 16 cells, the arising oscillation lasts hundreds of cycles, showing the waveform depicted in Figure 4.1.

DOI:10.15774/PPKE.ITK.2014.007

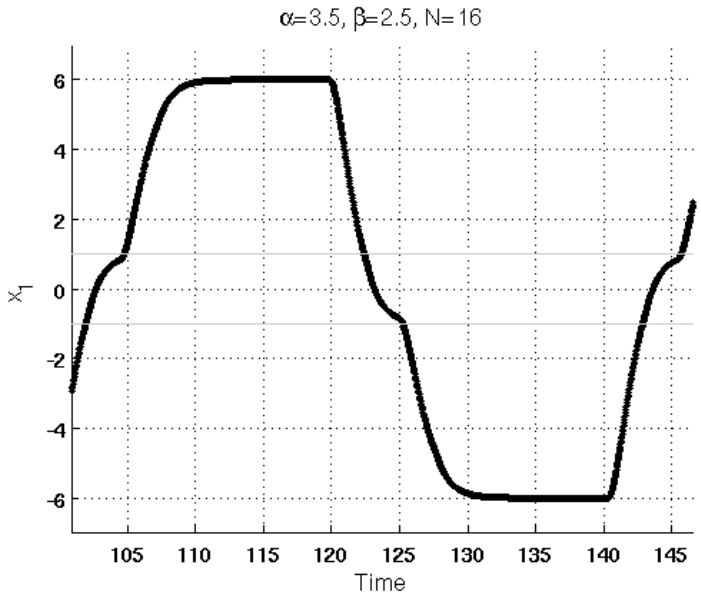

Figure 4.1: A typical waveform of the first cell's oscillation. The size of the array is 16, the coupling parameters are $\alpha = 3.5$, $\beta = 2.5$, the initial state is $N/2$-times $\{+1\}$ and $N/2$-times $\{-1\}$. Please note that the height of the plateaus is approximately $\pm 6 = \pm(\alpha + \beta)$.

It turned out that, with a wide range of the coupling parameters and for a wide set of initial conditions metastable periodic oscillations exist. This can be counted as an interesting experimental finding from the viewpoint of monotone dynamical systems. According to the Hirsch theorem (for the details the Reader is kindly referred to [20]), an eventually strongly monotone semiflow (ESM) converges to an equilibrium point, apart from a set of initial conditions with zero measure. Although our system is only a monotone semiflow (not an ESM) due to the squashing effect of the output function ([21]), the limit set dichotomy and most of the convergence properties of the ESM semiflows are still valid ([22][23][24][25]), in this way the "good" properties of the ESM semiflows are present here as well. This means that, the system must converge to an asymptotically stable equilibrium point in the long run, which can be actually observed in the experiments, too.

To remain at this example, the system has two asymptotically stable (AS) EPs ($+\alpha + \beta$ in all of the coordinates and $-\alpha - \beta$ in all of the coordinates) and a saddle (0 in all of the coordinates). For the sake of simplicity, let me introduce the notation $\{+1\}^p\{-1\}^q$, which means consecutively p-times $+1$, then q-times -1 ($p+q = N$, where N is the size of the array). With this notation, the AS EPs can be written as $\{+\alpha+\beta\}^N$ and $\{-\alpha-\beta\}^N$.

DOI:10.15774/PPKE.ITK.2014.007

Varying the size of the continuous positive and negative regions in the initial condition (at fixed ring-size), the arising waveform will reflect the structure of the initial condition. For two examples see Figure 4.2.

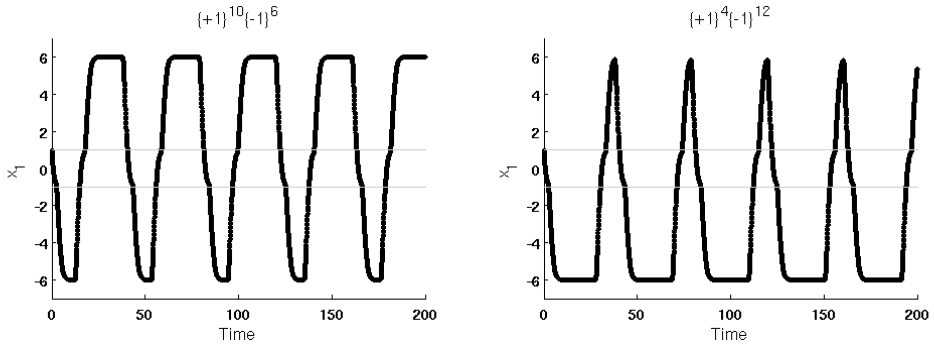

Figure 4.2: Different waveforms in the case of different initial conditions. In most cases, the arising oscillation's waveform follows the structure / combinatorics of the initial state.

Until I have only 1 positive and 1 negative plateau, the oscillation will converge to one of the AS EPs, on the way of the shorter plateau's decrease. However, if I have more positive and negative plateaus (meaning higher frequency in the intuitive sense), I can observe metamorphoses, as the frequency of the oscillation gradually decreases, up until I have 1 positive and 1 negative plateau, just before one of the AS EPs (latter ones can be considered as final stations). For an example, see Figure 4.3.

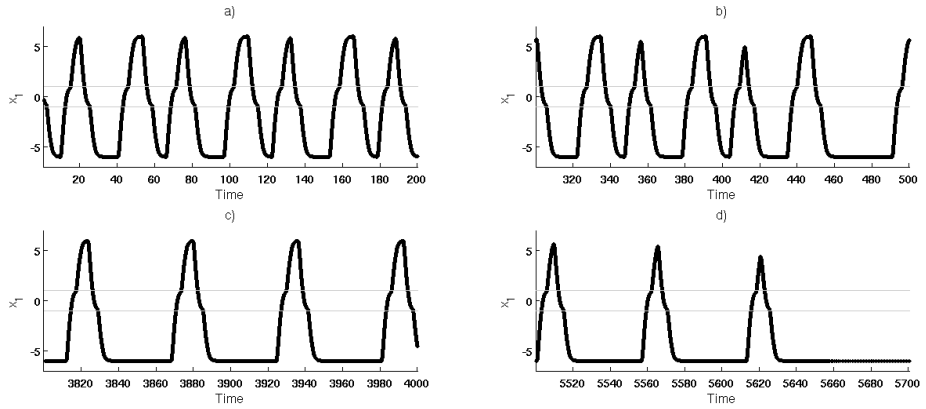

Figure 4.3: Metamorphoses of a waveform: different stages of an evolving oscillation. The coupling parameters are $\alpha = 3.5$, $\beta = 2.5$, the size of the array is 22, the initial state was $\{+1\}^5\{-1\}^8\{+1\}^4\{-1\}^5$.

DOI:10.15774/PPKE.ITK.2014.007

At this point I would like to mention a curious observation in the context of the different solvers and platforms. By the MATLAB environment, if we start the simulation from the symmetrical initial condition (eg. $\{+1\}^8\{-1\}^8$), the *ode45* solver of the system does not leave the periodic orbit, the oscillation tends to last forever. Implementing in MATLAB the Explicit Euler (EE) method, or in C++ language the EE or RK45 methods, all of them show the sharp attenuation and the convergence finally. The reason of this "forever lasting" oscillation by *ode45* should be some stabilizing mechanics/heuristics behind.

4.3 Robustness analysis with numerical eigenvalue computation

To numerically prove the metastability of the sometimes seemingly never-ending oscillations, I determined the Floquet eigenvalues and eigenvectors of the periodic orbits. The algorithm is based on the Poincaré return map, where practically we apply small perturbations on the trajectory, and observe the appearing deviation after one period from the original oscillation's trajectory. From these deviations one can construct the geometric Jacobian matrix, whose eigenvalues express the stability of the periodic orbit.

More precisely, I define a reference point and a reference section (called Poincaré section) in the N-dimensional state space. At this reference point, I add little perturbations to each coordinate separately, and measure the difference between the reference point and the first recurrences on the Poincaré section. From these difference vectors I can build the geometric Jacobian matrix, whose eigenvalues and eigenvectors are the Floquet eigenvalues and eigenvectors of the periodic orbit. In this case, the stability level is 1 (and not 0, as in the case of the stability of equilibra of linear differential equations). If the moduli of all Floquet eigenvalues — not counting the trivial Floquet eigenvalue which corresponds to perturbations perpendicular to the Poincaré section (and is always one) — are less than 1, the periodic orbit is stable. However, if any of them is slightly greater than 1 and none of them is significantly greater than 1, then the periodic orbit and all the neighboring oscillations are metastable.

Different kinds of tendencies and convergencies can be observed in connection with the strength of metastability. The first (and probably the most important) one comes from the size of the ring: fixing the parameter values, the metastability of the periodic orbit increases in exponential order with the size of the array, as

DOI:10.15774/PPKE.ITK.2014.007

we can see this in Table 4.1. *Professor Barnabás Garay* has analytically proved the existence of the periodic orbit and exponential asymptotics for the dominant Floquet eigenvalue as a function of the number of cells.

Table 4.1: The exponential convergence of the dominant Floquet eigenvalue in the function of the size of the array. The coupling parameters are $\alpha = 3.5$, $\beta = 2.5$, the initial condition is consecutively $N/2$–times $\{+1\}$ and $N/2$–times $\{-1\}$ (where N is even). The main message of this Table is that the dominant Floquet eigenvalue $\lambda_1 > 1$ decreases sharply with N, resulting in seemingly stable, long transient oscillations that change imperceptively on time intervals of order $1/(\lambda_1 - 1)$.

N	λ_1
6	2.5883
8	1.0985
10	1.00917
12	1.00089
14	1.00014
16	1.000097
18	1.000044

An other interesting tendency in the metastability's change can be observed, if we vary the coupling parameters. But before this, let me introduce the α, β parameter plane, as depicted in Figure 4.4.

DOI:10.15774/PPKE.ITK.2014.007

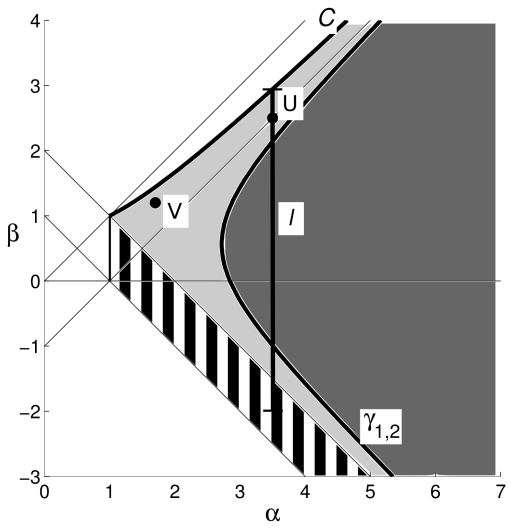

Figure 4.4: The existence of metastable periodic rotating waves on the parameter plane α, β of the feedback template A (in the region $\alpha \geq 1$, $-\alpha+1 \leq \beta \leq \alpha$); where the different shades of gray and the striped region indicate different types of metastable rotating waves. There is no metastable periodic oscillation in the white region.

The Figure is sketched on the coordinate system of the feedback template (A) elements, where certain negative values of β ($|\beta| \leq \alpha$) are defined as well, in this way – according to the author's best knowledge – extending the cooperative region analyzed in the literature so far. The different shades of gray indicate different types of connections between the states of the neighbouring cells; darker gray means "Type One", lighter gray means "Type Two" oscillations. "Type One" and "Type Two" oscillations on the parameter plane are separated by curve $\gamma_{1,2}$ (whose exact location actually depends on the number of cells (N) but, with $N \to \infty$, converges exponentially to the curve $\gamma_{1,2}$ depicted in Figure 4.4). Schematic drawings of "Type One" and "Type Two" oscillations can be seen in Figure 4.5. The names refer to the maximum number of cells simultaneously staying in the linear region ($x_i \in [-1, 1]$). The circuit measurements of the phenomenon are done in points $U = (3.5; 2.5)$ and $V = (1.7; 1.2)$ of the parameter plane; while the vertical line l marks the set of parameters, for which the dominant Floquet eigenvalues will be presented in details (Figure 4.6).

The relationship between the neighboring cells' states has further variations in the striped region, approaching the line $\beta = -\alpha + 1$ for N large. In the close vicinity of curve $\mathcal{C}$ (of equation $\beta = \frac{1+\alpha^2}{1+\alpha}$, $\alpha > 1$, *independently* of N) the metastable

DOI:10.15774/PPKE.ITK.2014.007

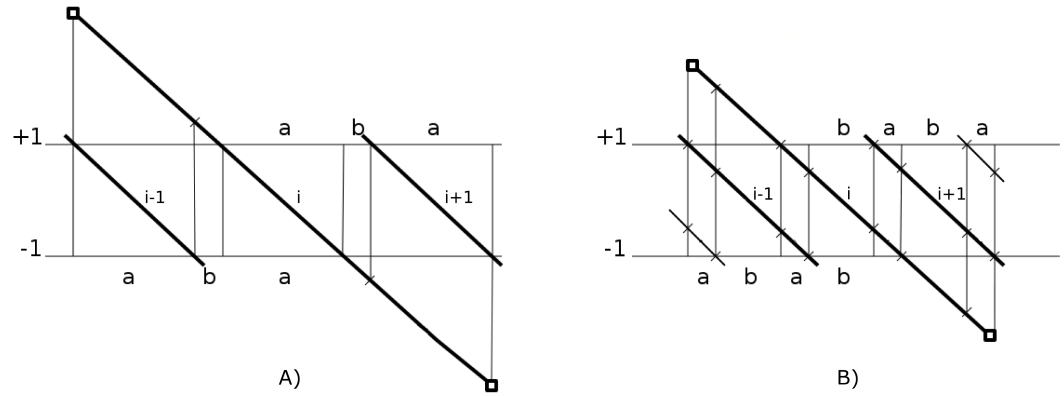

Figure 4.5: A) Type One oscillation; B) Type Two oscillation. The maximum number of cells simultaneously staying in the linear region ($x_i \in [-1, 1]$) is 1 and 2, respectively. The line segments a and b denote the regular repetition in the connection-structure of the neigboring cells' states.

oscillations get slower and slower, and finally, on the curve $\mathcal{C}$ itself, they disappear in a heteroclinic bifurcation.

Now consider the vertical line α=3.5 on the parameter plane for $-2<\beta<\frac{1+3.5^2}{1+3.5}\approx 2.9\ldots$. The nearness of the dominant Floquet eigenvalue to 1 for $2.5<\beta<2.9$ makes possible the circuit measurement of the phenomenon of metastable oscillations, even if I have a circuit built up of components with higher tolerances, as I will describe it in the next chapter.

DOI:10.15774/PPKE.ITK.2014.007

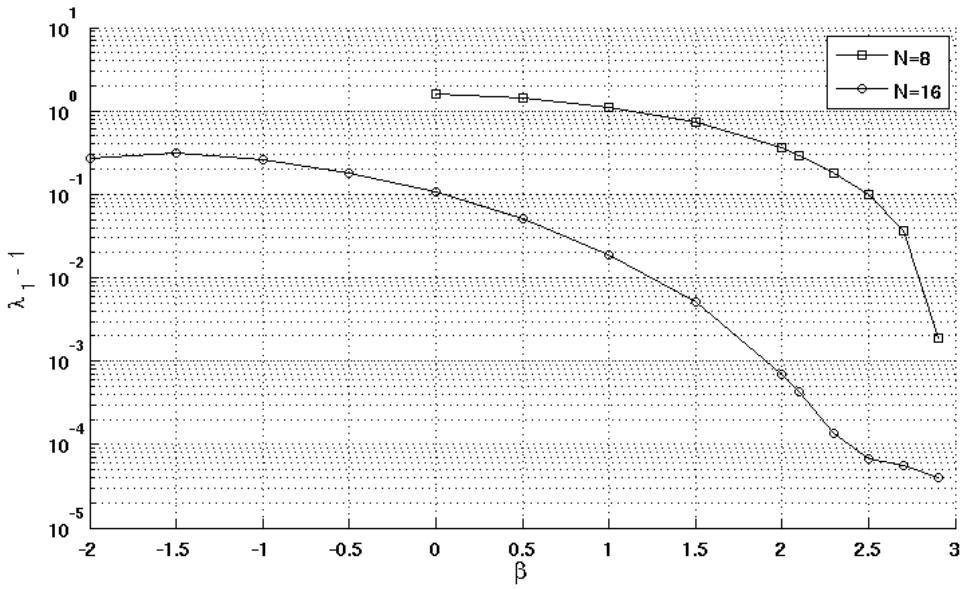

Figure 4.6: The dominant Floquet eigenvalue (λ_1) for N=8 and N=16 as a function of *beta* ($\alpha = 3.5$ is fixed). On the vertical axis I presented the value of $\lambda_1 - 1$, in order to appropriately separate the ending regions of the curves near β=2.9 on the logarithmic scale. These data were recorded from numerical computations, where the initial states were maximally symmetrical ($\{+1\}^4\{-1\}^4$ in the cases of $N = 8$ and $\{+1\}^8\{-1\}^8$ in the cases of $N = 16$). If $\beta < 0$, the $N = 8$ case becomes too unstable, resulting in the fail of the dynamics simulation.

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

Chapter 5

Metastable oscillations in circuit experiments

In this chapter I will present the paradigmatic experimental circuit which is applicable to produce long transient metastable oscillations. I use the coupling parameters that turned to be the most appropriate on the basis of the numerical exploration of the α, β parameter plane (Figure 4.4 and 4.6).

5.1 The circuit architecture

The circuit is built up by means of resistors, operational amplifiers and capacitances, the original version of the schematic was published in the Appendix of [26]. Figure 5.1 depicts the structure of a cell. Altogether 16 cells were built on a prototyping board, I could measure the phenomenon at different sizes of the array via reconfiguring the inter–cell lines.

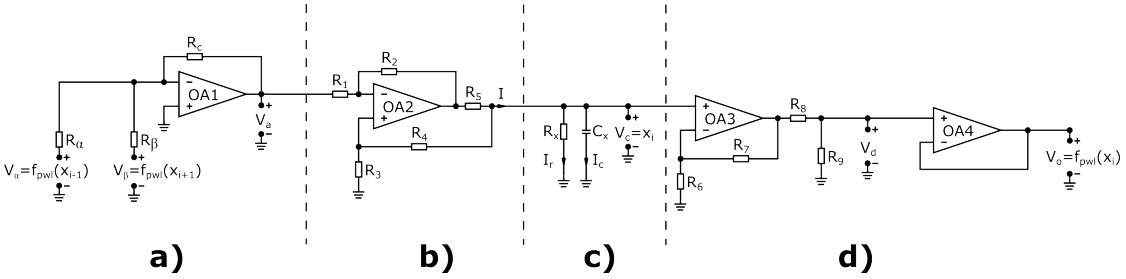

Figure 5.1: The schematic of a CNN cell built with the use of discrete components. The cell's functionality can be divided into four stages: summing amplifier (a), voltage controlled current source (b), inner state (c), and the realization of the piecewise linear output function (d) with a unity follower.

The architecture of a cell can be divided into four different stages, which will

DOI:10.15774/PPKE.ITK.2014.007

be described in the followings.

5.1.1 Summing amplifier

In the first step (part a) of Figure 5.1), I have an inverting adder implementing the weighted sum of the inputs to the i-th neuron. These inputs are the output voltages of the neighboring cells, $V_\alpha = f_{pwl}(x_{i-1})$ and $V_\beta = f_{pwl}(x_{i+1})$. Describing the output of this stage,

$$V_a = -\frac{R_c}{R_\alpha}V_\alpha - \frac{R_c}{R_\beta}V_\beta = -\alpha f_{pwl}(x_{i-1}) - \beta f_{pwl}(x_{i+1}) \tag{5.1}$$

i.e., the dimensionless positive interaction parameters α, β are obtained as

$$\alpha = \frac{R_c}{R_\alpha},\ \beta = \frac{R_c}{R_\beta}. \tag{5.2}$$

In the actual circuit I have chosen $R_c = 560\ \Omega$, so that the design equations for R_α and R_β are given as

$$R_\alpha = \frac{R_c}{\alpha} = \frac{560}{\alpha}\ \Omega,\ R_\beta = \frac{R_c}{\beta} = \frac{560}{\beta}\ \Omega. \tag{5.3}$$

5.1.2 Voltage controlled current source

The second stage (part b) of Figure 5.1) implements a voltage controlled current source (see Appendix of [26]), under the constraint

$$\frac{R2}{R1} = \frac{R4+R5}{R3} \tag{5.4}$$

that results in

$$I = -\frac{R2}{R1R5}V_a. \tag{5.5}$$

In the actual circuit I have chosen, in accordance with (5.4), $R1 = 1.8$ kΩ, $R2 = 2.7$ kΩ, $R3 = 1.8$ kΩ, $R4 = 1.2$ kΩ and $R5$=1.5 kΩ, leading to

$$I = -\frac{1}{1000}V_a = \frac{\alpha}{1000}f_{pwl}(x_{i-1}) + \frac{\beta}{1000}f_{pwl}(x_{i+1}). \tag{5.6}$$

5.1.3 The inner state of the cell

The third part (part c) of Figure 5.1) realizes the state x_i of the i-th neuron, which can be measured at the capacitor C_x. The time constant of the neuron is $\tau = R_xC_x$. By the Kirchhoff current law I obtain

$$I = I_c + I_r = C_x\dot{V}_c(t) + \frac{V_c(t)}{R_x} = C_x\dot{x}_i + \frac{x_i}{R_x}. \tag{5.7}$$

DOI:10.15774/PPKE.ITK.2014.007

I have chosen $R_x = 1$ kΩ, $C_x = 680$ nF, hence I have

$$680 \cdot 10^{-9}\, \dot{x}_i = -\frac{x_i}{1000} + \frac{\alpha}{1000} f_{pwl}(x_{i-1}) + \frac{\beta}{1000} f_{pwl}(x_{i+1}) \tag{5.8}$$

which coincides with Equation 2.1 describing the dynamics of the i-th neuron with a time constant $\tau = 6.8 \cdot 10^{-4}$ sec.

5.1.4 PWL-output with unity follower

The fourth step (part d) of Figure 5.1) realizes the PWL output-function $f_{pwl}(x_i)$ of the i-th neuron. This stage contains an amplification-division pair. Due to this overriding of the operational amplifier I am able to get the desired saturation-regions of the PWL output-function. Property $V_d = f_{pwl}(x_i)$ is ensured by constraint

$$\frac{R6 + R7}{R6} = \frac{1}{\frac{R9}{R8+R9}}\,. \tag{5.9}$$

In the circuit, the parameter values are chosen as $R6 = 1$ kΩ, $R7 = 18$ kΩ, $R8 = 18$ kΩ and $R9 = 1$ kΩ. Finally, the fourth operational amplifier implementing a voltage-follower with $V_o = V_d = f_{pwl}(x_i)$ is used for decoupling the voltage divider from the input stages of each connected neuron.

The initial condition is set by a switch at every cell, which is connected to the third stage's V_c point. When the switch is set in "load" position, it disconnects the third stage from the second one, moreover, it connects the third stage to the voltage generator providing initial condition. In "run" position it sets the normal setup, which is depicted on part c) of Figure 5.1. The evolving oscillations are recorded at node V_c (inner state) and V_o (output) of the cell numbered one.

DOI:10.15774/PPKE.ITK.2014.007

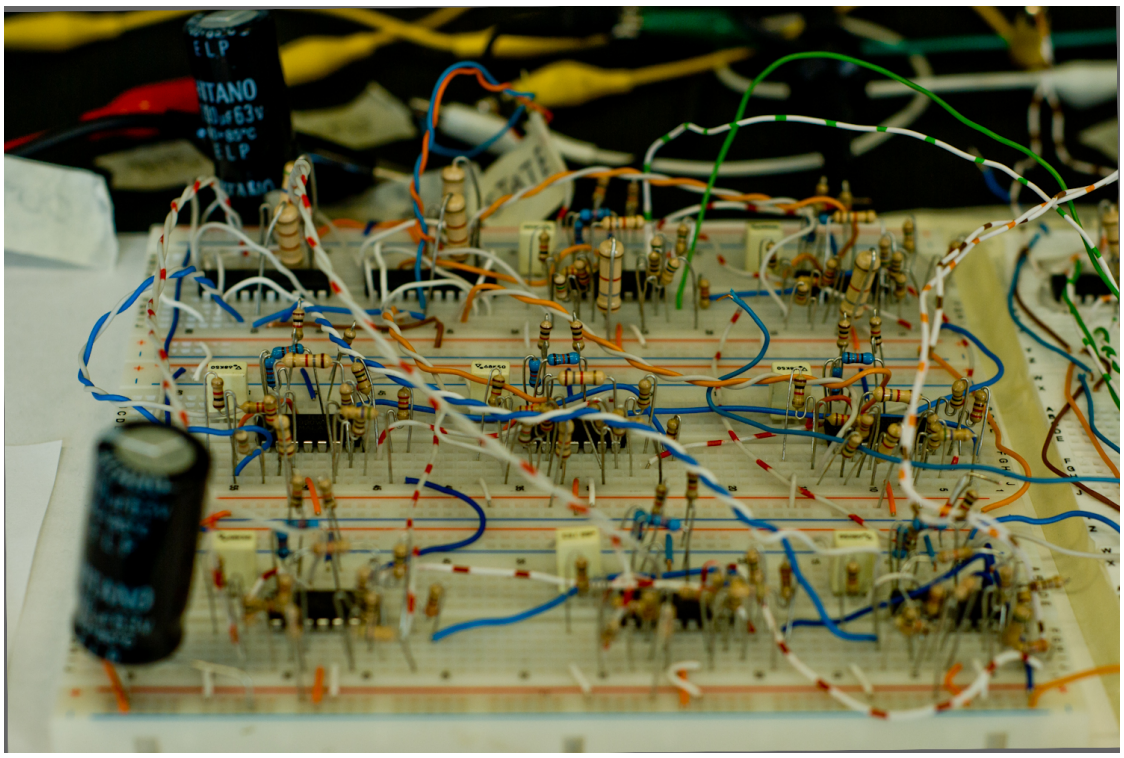

Figure 5.2: Half of the built circuit's 16 cells.

The implemented circuit took place on a breadboard, the half of it can be seen in Figure 5.2. The used resistors have 5% tolerances, the capacitors have 10% tolerances; the type of the operational amplifiers is TL084. The supply voltage for the operational amplifier was set to ±20 V. The MAX333 IC was used as switch, which has 130Ω series internal resistance.

5.2 Measurement results

In the circuit measurements, two different α, β pairs were tested, namely $\alpha = 3.5$, $\beta = 2.5$ and $\alpha = 1.7$, $\beta = 1.2$ (see also points U and V points in Figure 4.4) . By every example, the size of the array is 16. The results will be demonstrated together with the simulation waveforms, emphasizing the similar behaviors. More precisely:

- (a) waveform from MATLAB simulation of the theoretical model,
- (b) PSpice simulation of the circuit design,
- (c) waveform measured with oscilloscope on the built circuit.

DOI:10.15774/PPKE.ITK.2014.007

In general, as we will see in the figures, the transient length of the theoretical model (a) is in good agreement with the predicted behavior on the basis of the dominant Floquet eigenvalue in Table 4.1. In contrast to this, the circuit simulator (b) takes into consideration the non–idealities of the different types of components; while the real measurement contains all of the inaccuracies of the actual measurement, especially the imprecise and asyncronous setup of the initial state.

Differing from the earlier presented simulation waveforms, now I changed the time constant (τ) of the MATLAB simulator to $6.8 \cdot 10^{-4}$ sec., in order to have the same time-scale (*time-scale resolution*) on axis x.

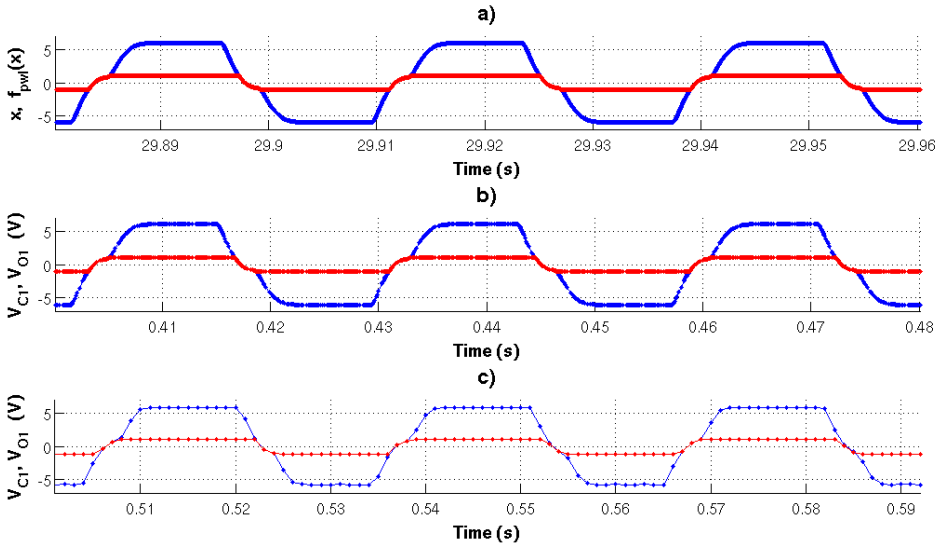

Figure 5.3: The coupling parameters are $\alpha = 3.5$, $\beta = 2.5$, the initial condition is $x(0) = \{+1\}^8\{-1\}^8$. The MATLAB simulation (a) was stopped after 1000 periods approximately; the PSpice simulation (b) was stopped after 74 cycles; the oscillation of the circuit (c) disappeared after 74 periods, and the dynamics converged quickly to one of the asymptotically stable equilibria thereafter. The blue and the red curves portray the inner state and the output of the first cell, respectively.

Figure 5.3 corresponds to point U in Figure 4.4. With this symmetrical initial condition and with this large size ($N = 16$) of the simulated ring, even the PSpice simulation (knowing the non-idealities) can result more than hundred cycles. However, the length of a PSpice oscillation will be every time shorter than the length of the simulation of the ideal model in MATLAB, due to the inclusion of non-ideal characteristics of the operational amplifiers and other "real" effects in the circuit simulator. Actually, at this point the Reader is kindly reminded to the aforementioned observation that the *ode45* solver of MATLAB does not show any

DOI:10.15774/PPKE.ITK.2014.007

convergence in the cases of symmetrical initial conditions. Apart from this exception, all of the oscillations on every platform have to converge to one of the stable equilibria, but they attenuate sharply, in the last few cycles only. Figure 5.3 emphasizes the similarity of the waveforms, the convergence of the PSpice oscillation was not accurately measured.

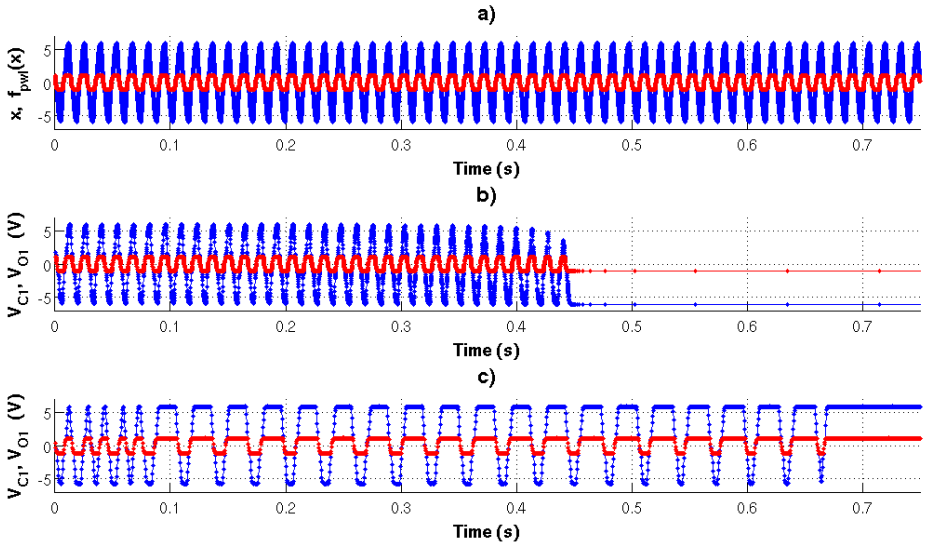

Figure 5.4: The coupling parameters are $\alpha = 3.5$, $\beta = 2.5$, the initial condition is $x(0) = \{+1\}^4\{-1\}^4\{+1\}^4\{-1\}^4$. The waveform follows the combinatorical structure of the initial condition. The MATLAB simulation (a) was stopped at 1 sec.; the PSpice simulation (b) converged to the negatively saturated AS EP after approximately 16 periods; the oscillation of the circuit (c) after 2.5 periods had a metamorphosis to an other oscillation (probably the waveform belonging to the initial condition $\{+1\}^{10}\{-1\}^6$), then had further 19 periods before converging to the positively saturated AS EP. The difference between the behavior of (b) and (c) is analyzed in the main text.

Figure 5.4 shows the second example corresponding also to point U in Figure 4.4 this time with initial condition $x(0) = \{+1\}^4\{-1\}^4\{+1\}^4\{-1\}^4$. The MATLAB simulation of the theoretical model seems to oscillate endlessly; while the other two oscillations show qualitatively different waveforms. It seems to me that this is due to the different directions of escapes: the PSpice simulator can be handled more reliably about the exact waveform than the real oscillation of the experimental circuit with components of higher tolerances. For a more exact answer, I made small perturbation on the original waveform in MATLAB. The results are in Figure 5.5.

DOI:10.15774/PPKE.ITK.2014.007

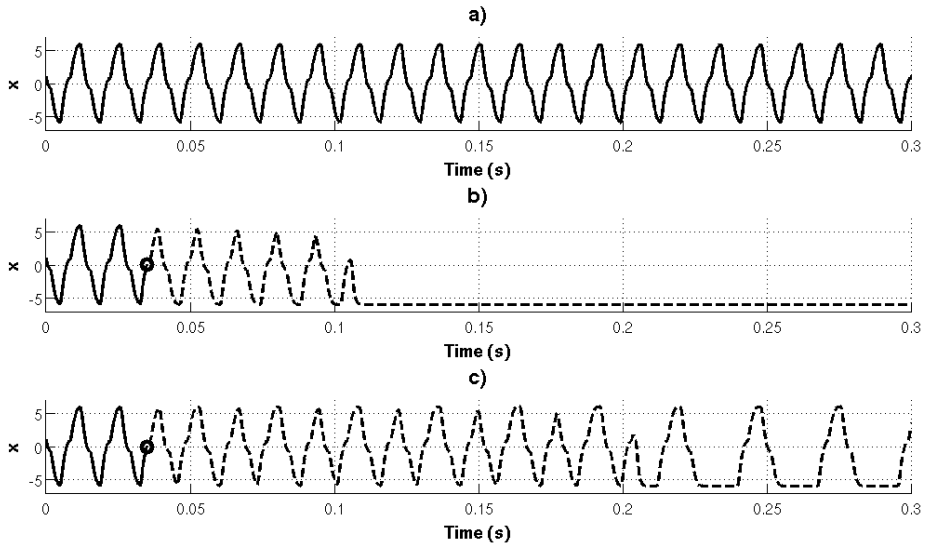

Figure 5.5: Perturbation analysis of the oscillation arising from $x(0)=\{+1\}^4\{-1\}^4\{+1\}^4\{-1\}^4$, $\alpha=3.5$, $\beta=2.5$. (a) is the original oscillation; (b) shows the waveform after perturbation into the direction of the first Floquet eigenvector; (c) shows the waveform after perturbation into the direction of an element of the real eigenplane corresponding to the complex conjugate pair of Floquet eigenvalues $\lambda_{2,3}$. The small circles mark the place of perturbation on the first coordinate-function.

I have computed the Floquet eigenvalues and eigenvectors of this oscillation, when the first coordinate-function crosses the zero-level (at $t = 0.03495$ sec.): I have three metastable eigenvalues, $\lambda_1 = 1.20665$ and $\lambda_{2,3} = 1.09632 \pm 0.10317i$. The related eigenspaces can be handled as "escape" directions. I have added a strong perturbation to the original oscillation at $t = 0.03495$ sec.: on subfigure (b) – the fivefold of the first unit eigenvector, while on subfigure (c) – the fivefold of a unit vector belonging to the real eigenplane of $\lambda_{2,3}$ (the small black circles show the point of perturbation in the figures). As I can see in the Figure, the different directions lead to different forms of metamorphoses: the first one results in equalized attenuations on both of the upper plateaus within one period, while the second one suppresses only one of the plateaus within a period, resulting in an asymmetric waveform. According to my opinion, the dynamics on the (c) part of Figure 5.4 shows the "random way" during the convergence, due to the high tolerances of the components and the inaccurate setup of the initial condition.

DOI:10.15774/PPKE.ITK.2014.007

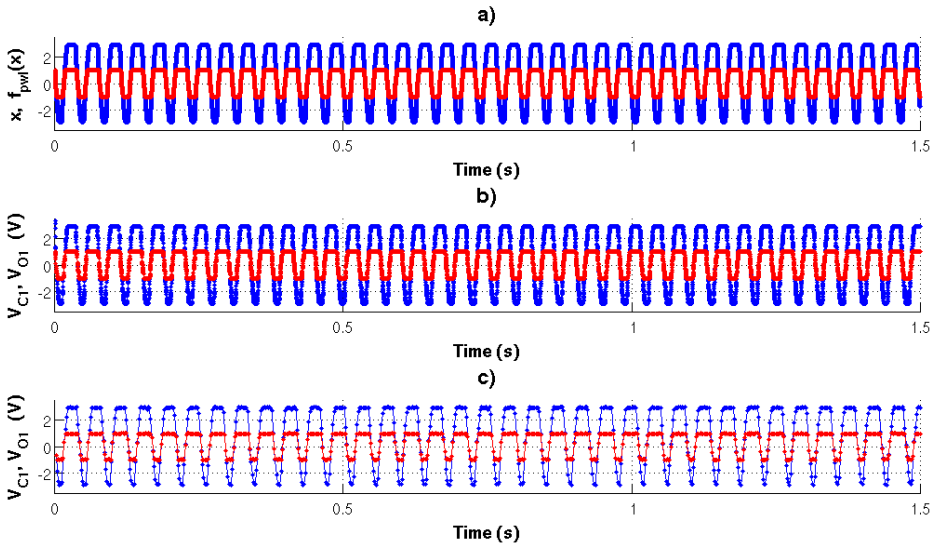

Figure 5.6: Oscillations for coupling parameters $\alpha = 1.7$, $\beta = 1.2$ and initial condition $x(0) = \{+1\}^{10}\{-1\}^{6}$. The waveform follows the combinatorical structure of the initial condition. The MATLAB simulation (a) was stopped at 1.5 sec.; the PSpice simulation (b) was stopped after 40 cycles; the oscillation of the circuit (c) converged to the positively saturated AS EP after 66 cycles approximately. Please note in these figures the changed height of the plateaus: it is approximately ± 2.9 which is equal to $\pm(\alpha+\beta)$.

The third example is depicted in Figure 5.6. In this case, the initial condition was $x(0) = \{+1\}^{10}\{-1\}^{6}$, the coupling parameters $\alpha = 1.7$, $\beta = 1.2$ (point V of the parameter plane in Figure 4.4).

DOI:10.15774/PPKE.ITK.2014.007

Chapter 6

Conclusions

The results of the research on detection with frameless processing are presented in Chapter 2. It is shown by the deeper analysis of the phenomenon beyond that the structure of equilibria and the changes of the basins of attractions are the real causes of the first time seemingly strange behavior. Furthermore, it is also demonstrated by the presented example how frameless processing can be efficiently used as the detector of spatial-temporal events.

In Chapter 3, the details of a measurement range-tuning algorithm are uncovered for the model of an infrared sensor array with activation light sources, which can be used for the observation of scenes with higher depth-dynamic range on the way of locally adaptive measurements. Also in this chapter, two standard CNN-algorithms are presented for the detection of specific spatial movements.

In Chapter 4 and 5 long transient metastable oscillations are examined from different points of view. In the former one, the existence and the different combinatorics of the oscillations' waveforms are showed as well as the robustness of them by numerical computations throughout the parameter-space of the coupling parameters. In the latter one, the robustness of these oscillations is demonstrated with a paradigmatic laboratory circuit, where the high tolerance of the components and the inaccurate setup of the initial conditions meant in themselves the noisy circumstances, which allow the existence of the more robust phenomena only.

DOI:10.15774/PPKE.ITK.2014.007

Summary

6.1 New scientific results

1. Thesis: *Recognition of a feature spanning frames (in space and/or in time) with a CNN Wave Computer interfaced with a two–dimensional, depth measuring sensor array equipped with own lightsources.*

Our measurement and processing differs from the input recording and processing method of the original CNN Wave Computer in two major points. In contrast to the ordinary, *passive* input recording, the picture of the measured object is produced *actively* by us: there is an LED next to every phototransistor, which measures the reflected light of the LED from the environment. This measurement method is novel in the sense that, the global solution evolving on the computing array can be influenced during the transient of the computation (an earlier, inspiring paper [14]). In the other hand, the process of the computation is continuous, not discrete from frame to frame (frame–by–frame). The individual pictures are not separately processed and evaluated, but a single, continuous processing flow exists, which gets from time to time the appropriate input picture, and produces a static or oscillating pattern at the output as the result of the computation. Another, frameless processing example is [15], and an analogous example is [16].

We have used both of the above mentioned measurement and processing specialities for solution of detection problems, but in this dissertation I will present only the results connected to the continuous processing method. We believe that these computation methods have a key role in time–critical object and event recognition tasks.

Published in: [3], [9], [10], [13].

1.1. Applying a specific coupled template class with few non–zero elements on a CNN Wave Computer interfaced with a two–dimensional, depth measuring sensor array equipped with own lightsources, I showed that, in contrast to the frame–by–frame input processing, the continuous input flow makes possible to the evolving dynamics

DOI:10.15774/PPKE.ITK.2014.007

of the computing array the unambiguous identification of object– or scene–properties spanning frames (in space and/or in time). I provided a solution for detecting a given, oversized terrain feature; I have verified the applicability of my method with measurements.

I was able to detect a property with frameless processing mode, which is far more difficult when we process the input frame–by–frame. The sensor array was passed over a terrain bump with size three and a half times larger, than the size of the array itself (for the measurement setup the Reader is kindly referred to Figure 2.5); the activation sources were constantly lighting. In the three main regions of the bump (uphill, plateau, downhill) three different output patterns became solely dominant, which were preserved till the end of the specific regions. By the frame–by–frame processing of the input flow (instead of a long–continuous computation, separate short ones), more different patterns (even oscillations) were able to emerge in the different regions. This means that, from one / a few separate input pictures we are unable to unambiguously decide over which region stands the array. Although with a statistics of more still input pictures we could overcome this problem, this needs additional memory and post processing routines. On the other hand, in the case of frameless processing, the evolution and inner state of the dynamics itself stands for the solution, it needs comparably less processing, it is easier. The executing architecture "involves" this dynamics. If the inner state of the computing is easily readable from outside, then the result of the computation can be known without any special device during the computation itself.

On the computing array we applied the template–family according to Equation (2.3), with the parameter values in Equation (2.4). In the literature, the templates contain generally more values which are different from zero, the filling rate of this template is sparse. The computing array had zero-flux boundary condition, the initial state was the first input picture. This template appeared in István Petrás's PhD dissertation [17], where the Author processed only separate, still input pictures. He discovered the different template–value regions according to different output dynamics (stable equilibrium point, stable oscillation, chaotic behavior), which results I used as initial values during parameter–tuning.

Measuring the performance of detection with this phenomenon is difficult, there is no ready specific architecture which could run the computation in real time. But if we consider that no read-out and memory moduls are necessary, we can assume that this solution would outperform other, general systems in the context of time- and energy-management.

DOI:10.15774/PPKE.ITK.2014.007

2. Thesis: *Measurement range tuning and complex movement detection with a CNN Wave Computer interfaced with a two–dimensional, depth measuring sensor array equipped with own lightsources.*

Published in: [1], [4], [5], [7], [11].

2.1. I presented a locally adaptive algorithm for the tuning of the range of depth dynamics, in the proposed architectural setup where a two–dimensional, depth measuring sensor array equipped with own lightsources is interfaced to a CNN Wave Computer.

Current imaging sensors are unable to take a picture of scenes with high light dynamic range. This is due to the linear relationship between the amount of the incoming light and the amount of the recorded light. In this way only two or three orders of magnitude can be covered. There are specific solutions, where either the sensitivity of the sensor is logarithmic, or the light capture process happens on an adaptive (or rather on a locally adaptive) way. These solutions can result in that, there will not be any part of the picture in saturation. In the case of CNN Wave Computer architecture, there exists already a locally adaptive solution in the doctoral dissertation of Róbert Wagner, which solution is based on the locally adaptive settings of the cells' integrating time on the whole array.

The architectural difference and the depth range are the novelties in my solution. I work with the cells' light activation strength of the depth measuring sensor array in a frame–by–frame iterative algorithm. The integrating time of the sensing cells is not varied. As an advantage, my method needs only the logical unit of the cell (thinking in the architecture of the CNN-UM) to the adaptive calibration of the light-conditions meaning that the analog unit has to take care of the "main" objective (CNN-algorithm) only. Because our sensor array supports binary activation patterns (On / Off), this algorithm was implemented only in simulation.

2.2. I presented CNN–algorithms for the detection of moving objects with constant speed but varying direction, and for the detection of tilting movement, in the proposed architectural setup where a two–dimensional, depth measuring sensor array equipped with own lightsources is interfaced to a CNN Wave Computer.

There are already known algorithms in the literature to detect movement, or to detect moving objects at a given speed. The basis of them essentially is the difference computation of two consecutive frames with a specific delay–distance in the input flow.

DOI:10.15774/PPKE.ITK.2014.007

Due to the speciality of our system (depth measurement) we are able to observe the depth–displacement (getting closer or farther) of the moving object. In the case of the first algorithm (*detection of moving object at constant speed but varying direction*) we compute first the planar (in the plane of axes x–y) and the depth (axis z) components, then we take the resultant of them.

The second algorithm detects the objects with tilting movement, when a surface tilts around an axis. The plane of the axis and the surface of the sensor array must be parallel. The algorithm begins with a difference computation on two images of the input flow with specific delay between them. On the resulting image, a modified edge–detecting method is run, which results in bigger patches at those places, where the neighboring rows/columns are detected with continuously changing intensity (due to the tilting movement).

Figure 3.4 shows a complex simulation scene of objects with different locomotions. I tested the complex movement detection algorithms on this scene, analyzing the specificity with counterexamples, too.

DOI:10.15774/PPKE.ITK.2014.007

3. Thesis: *Long transient, metastable periodic oscillations in simulations and in circuit experiments.*

In the literature we can see qualitatively similar phenomena with other systems, like those investigated in [18] and in [19].

In our case, the phenomenon is observed in a one–dimensional, autonomous CNN array with periodic boundary condition. The cells are regular, first order ones. The neighborhood connection is realized by the template according to Equation 4.1. The output function of the cells is the well known piecewise linear activation function, described by Equation (2.2).

According to [20] an eventually strongly monotone semiflow (ESM) converges to an equilibrium point, apart from a set of initial conditions with zero measure. Although our system is only a monotone semiflow (not an ESM) due to the squashing effect of the output function ([21]), the limit set dichotomy and most of the convergence properties of the ESM semiflows are still valid ([22][23][24][25]), in this way the "good" properties of the ESM semiflows are present here as well. This means that the system must converge to an asymptotically stable equilibrium point.

In contrast to this, both in simulations and in circuit measurements we were able to observe unexpectedly long oscillations (lasting even hundreds of cycles) on a wide set of the coupling parameters (α, β). But these periodic oscillations are not stable, I prove their metastability with the computation of their Floquet eigenvalues.

Published in: [2], [3], [6], [8], [12].

3.1. I have numerically proven the existence of long transient metastable periodic oscillations in the dynamics of a one–dimensional, coupled, autonomous CNN Wave Computer with periodic boundary condition (ring). With the numerical eigenvalue computations I have defined the region of the strongest metastability on the parameter plane, where the phenomenon is certainly reproducible even with an electrical circuit by means of higher tolerance components.

In the case of the periodic oscillations with stronger metastability, I determined the Floquet eigenvalues and eigenvectors on the basis of the Poincaré return map.

Fixing the parameter values, the metastability (in Floquet–sense) of the periodic orbit increases in exponential order with the size of the array, as we can see this in Table 4.1. *Professor Barnabás Garay* has analytically proved both the real

DOI:10.15774/PPKE.ITK.2014.007

metastability and the asymptotical behavior of the metastability in the function of the number of the cells.

Figure 4.4 depicts the parameter space of α and β from the feedback template A, where certain negative values of β ($|\beta| \leq \alpha$) are defined as well, in this way – according to the author's best knowledge – extending the cooperative region analysed in the literature so far. The waves evolving on the cells can differently connect to each other. These different forms are indicated on the figure with different shades of gray, $\gamma_{1,2}$ is the borderline between them (the exact location of it depends on the number of the cells (N), in the function of N it exponentially converges to the curve indicated on the figure). The relationship between the neighboring cells' waves has further transformations in the striped region, approaching the line $\beta = -\alpha + 1$ (the exact borderlines are known, this figure has only illustrative purposes). In the close vicinity of curve $\mathcal{C}$ the oscillation gets slower and slower, on the curve it dies with a heteroclinic bifurcation. The circuit measurements of the phenomenon are done in points $U = (3.5; 2.5)$ and $V = (1.7; 1.2)$ of the parameter plane (see subthesis 3.2), in whose vicinity the metastability can be observed by means of an electrical circuit.

3.2. I built a paradigmatic experimental electrical circuit with the use of discrete components, which can realize the functionality of a one–dimensional, autonomous CNN Wave Computer. Appropriately connecting the cells I realized an electrical implementation, which is applicable to reproduce the phenomenon of long transient metastable oscillations. In this way I measured the phenomenon in the electrical circuit as well, experimentally demonstrating its robustness.

The circuit is built up by means of resistors, operational amplifiers and capacitances, the original version of the schematic was published in the Appendix of [26]. Figure 5.1 depicts the structure of a cell. Altogether 16 cells were built on a prototyping board, in this way we measured the phenomenon at different sizes of the ring with the reconfiguration of the inter–cell lines. The initial states were set with a switching circuit at every cell's V_c node; the switching circuits were uniformly controlled by a function generator. The resulting oscillation was recorded at node V_c (inner state) and V_o (output) of the cell numbered one.

DOI:10.15774/PPKE.ITK.2014.007

6.2 Application of the results

The scope of the first and the second thesis group (object and event recognition, detection of complex movement) covers a wide range of possible applications. Inside mobile robotics, numerous unsolved (or just partially solved) problems exist, where a more efficient solution (compared to classical image processing) could be advantageous. Good examples are the tasks of the recent DARPA Challenge. In the case of walking on a rough surface, if we could tell some critical features of the ground just before putting the foot down, then we can stabilize the balance of the robot itself. In the case of climbing up a ladder in an industrial environment, a sensor array mounted on the foot can measure the outlines and characteristics of the rungs (surface, slope, accurate position) just before the step, resulting the reliable realization of the planned movement.

I have analyzed different situations in simulation as well as in real hardware measurement, when a sensor array supposedly mounted on the foot of a robot detects an unexpected salient object, or an unexpectedly steep slope, or an other moving object. In these cases the system has notified the central movement controlling unit with a "STOP" command. I have created a simulation environment, in which an autonomous mobile robot tries to get a predefined point in the changing environment with other moving objects. On the basis of the measurements of the distance measuring infrared sensor array mounted on the front, the agent tries to avoid adaptively the other objects in order to get the end point without collision.

In the case of the third thesis group (metastable periodic oscillation), beyond the detailed analysis of the oscillation we can expect information representation and processing (due to the biological motivation, for example [27], [28], [29]). Series of different waveforms and/or frequencies could identify a (quasi-)periodic series of phenomena/events; or a metastable oscillation could encode an engram in the memory with appropriate architecture. These examples try to mimic the information processing and representation observed or supposed in biology.

I have done other simulations with a slight modification of the computing architecture presented in Thesis 3. The objective was the detection of periodic oscillations with different patterns and speed (frequency): if the system got the desired signal-sequence on its input terminal, then the metastable oscillation became stabilized; however in the case of other sequences the waveform went wrong or the inner state of the system converged to a stable equilibrium point.

DOI:10.15774/PPKE.ITK.2014.007

Appendix A

Terms, definitions, theorems connected to the dissertation

For a more comprehensive and detailed overview, the Reader is kindly referred to [66] and [20].

– $\Phi : \mathbb{T} \times \mathbb{R}^n \to \mathbb{R}^n$ **dynamical system**

I. Φ is continuous, jointly in both variables

II. $\Phi(0, x) = x \quad \forall x \in \mathbb{R}^n$

III. $\Phi(t, \Phi(s, x)) = \Phi(t+s, x) \quad \forall t \forall s \in \mathbb{T}, \quad \forall x \in \mathbb{R}^n$

Here time $\mathbb{T} = \mathbb{R}$ (continuous time) or $\mathbb{T} = \mathbb{Z}$ (discrete time).

The basic example for a dynamical system is the solution operator $\Phi{:}\mathbb{R}\times\mathbb{R}^n\to \mathbb{R}^n$ of an autonomous differential equation $\dot{x} = f(x)$ (f sufficiently smooth, e.g., $f \in C^1$).

– **Eigenvalues and stability of an equilibrium point**

$\dot{x} = f(x)$ autonomous differential equation,
$f(x_0) = 0 \Leftrightarrow x_0$ is an equilibrium point.
The eigenvalues of the equilibrium point are the eigenvalues of the Jacobian matrix $J = f'(x_0)$ evaluated at x_0 .
Notation: $\lambda_1, ..., \lambda_n$ for the eigenvalues, and v_i for an eigenvector associated to λ_i.
Stability criteria for equilibrium point x_0:
$\forall i \quad Re(\lambda_i) < 0 \Rightarrow \quad x_0$ is exponentially stable;

DOI:10.15774/PPKE.ITK.2014.007

$\exists i \quad Re(\lambda_i) > 0 \Rightarrow$ x_0 is unstable (and then the direction of v_i is repulsive);
$maxRe(\lambda_i) = 0 \Rightarrow$ further investigations are necessary.

– **Eigenvalues and stability of a periodic orbit**

As before $\dot{x} = f(x)$ denotes an autonomous differential equation,
and $\Phi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ stands for the solution operator.
$p_0 \in \mathbb{R}^n$ point is a periodic point of Φ and $\tau_0 \in \mathbb{T}$, $\tau_0 > 0$ is the minimal period, if $\Phi(\tau_0, p_0) = p_0$ and $\forall \tau \in \mathbb{T}$, $0 < \tau < \tau_0$: $\Phi(\tau, p_0) \neq p_0$.
The periodic orbit is $\Gamma = \{\Phi(t, p_0) \in \mathbb{R}^n | t \in \mathbb{T}\} = \{\Phi(t, p_0) \in \mathbb{R}^n | t \in \mathbb{T}, 0 \leq t \leq \tau_0\}$.
The periodic solution is $p : \mathbb{R} \to \mathbb{R}^n$, $p(t) = \Phi(t, p_0)$.
The Poincare section to Γ at $p_0 \in \Gamma$ is the codimension one hyperplane $\Sigma = \{x \in \mathbb{R}^n | \langle x - p_0, f(p_0) \rangle = 0\}$.
The first return time function $\tau : \mathcal{N}_{p_0} \to \mathbb{R}$ is the local solution $\tau = \tau(x)$ of equation $H(\tau, x) = 0$ satisfying $\tau(p_0) = \tau_0$. Here $\mathcal{N}_{p_0}$ is a neighborhood of p_0 in Σ and $H(\tau, x) = \langle \Phi(\tau, x) - p_0, f(p_0) \rangle$, $(\tau, x) \in \mathbb{R} \times \Sigma$. (Conditions $H(\tau_0, p_0) = 0$ and $H'_\tau(\tau_0, p_0) \neq 0$ of the implicit function theorem hold true.)
The Poincare return map is the operator $\pi : \mathcal{N}_{p_0} \to \Sigma$, $x \to \pi(x) = \Phi(\tau(x), x)$, which can be considered (via identifying the Poincare section Σ with the subspace $\mathbb{R}^{n-1}$) as an $\mathbb{R}^{n-1} \to \mathbb{R}^{n-1}$ mapping.
Eigenvalues $\kappa_1, \kappa_2, ..., \kappa_{n-1}$ of matrix $\pi'(p_0)$ do not depend on the choice of $p_0 \in \Gamma$ and are called the Floquet eigenvalues (sometimes also termed as Floquet multipliers) of the periodic orbit. (The trivial Floquet eigenvalue is $\kappa_0 = 1$.) (In the main text, I termed $\pi'(p_0)$ as the *geometric Jacobian*.)
Stability criteria for periodic orbit Γ:
$\forall i, i > 0 \quad |\kappa_i| < 1 \Rightarrow$ Γ is orbitally asymptotically stable;
$\exists i, i > 0 \quad |\kappa_i| > 1 \Rightarrow$ Γ is orbitally unstable;
$max_i |\kappa_i| = 1 \Rightarrow$ further investigations are necessary.
Roughly speaking, the periodic orbit Γ is metastable, if all Floquet eigenvalues satisfy $|\kappa_i| < 1 + \varepsilon$ and most Floquet eigenvalues satisfy $|\kappa_i| < \varepsilon$ for an $\varepsilon > 0$ very small. The precise definition we use is given for a family of periodic orbits $\Gamma_n \in \mathbb{R}^n$, $n = 2,3, \ldots$. It concerns a sequence of differential equations $\dot{x} = f_n(x)$ in $\mathbb{R}^n$ and requires that ε as a function of n is exponentially small in n. It is usually also required that the number of the unstable Floquet eigenvalues is one.

– **The ω-limit set**

DOI:10.15774/PPKE.ITK.2014.007

As before $\dot{x} = f(x)$ denotes an autonomous differential equation,
and $\Phi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ stands for the solution operator
From now on, we assume that $\dot{x} = f(x)$ is dissipative in the sense that the point at infinity is repulsive or, equivalently, we assume that all trajectories enter a ball B and remain there for all time large enough.
$x \in \mathbb{R}^n$
$\omega(x) = \{y \in \mathbb{R}^n \mid \exists\, t_n \to \infty$ sequence, if $n \to \infty$ then $\Phi(t_n, x) \to y\}$
Examples are: equilibrium point, limit cycle, heteroclinic cycle (these are the only examples in $\mathbb{R}^2$), invariant tori, chaotic attractors (only in $\mathbb{R}^n$, $n > 2$).
ω-limit sets are nonempty, closed, bounded, connected and invariant (i.e., the union of entire trajectories)

For different types of monotone dynamical systems, ω-limit sets have a particularly simple structure and, forgetting about a nowhere dense set of measure zero, all the complicated possibilities are ruled out.

– $\leq$ **closed partial order over** $\mathbb{R}^n$

I. $x \leq x$ (reflexivity)

II. $x \leq y$ and $y \leq z$ then $x \leq z$ (transitivity)

III. $x \leq y$ and $y \leq x$ then $x = y$ (antisymmetry)

IV. ($x \leq y$ and $z \leq u$ then $x + z \leq y + u$)

V. ($x \leq y$ and $\lambda \geq 0$ then $\lambda x \leq \lambda y$)

VI. $\leq$ (closedness) if $x_k \leq y_k$, $x_k \to x$ and $y_k \to y$ then $x \leq y$

The basic example is the standard closed partial order defined by letting $x \leq y$ if and only if $x_i \leq y_i$ for each coordinate $i = 1,2,\ldots,n$. (Here the coordinate notation $x = (x_1, x_2, \ldots, x_n)$, $y = (y_1, y_2, \ldots, y_n)$ is used.)

– **The** $\Phi : \mathbb{T} \times \mathbb{R}^n \to \mathbb{R}^n$ **dynamical system is**

- monotone
  if $x \leq y$ then $\Phi(t, x) \leq \Phi(t, y) \quad \forall t \geq 0$
- strongly monotone
  if $x \neq y$ and $x \leq y$ then $\Phi(t, x) \ll \Phi(t, y) \quad \forall t > 0$
  where $x \ll y$ means $x_i < y_i \quad i = 1,2,\ldots,n$

DOI:10.15774/PPKE.ITK.2014.007

- eventually strongly monotone
  if $x \neq y$ and $x \leq y$ then $\exists t_0 > 0$ such that $\Phi(t_0, x) \ll \Phi(t_0, y)$

– **Hirsch theorem**

Given $\dot{x} = f(x)$ and the solution operator $\Phi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$,
assume that

(i) Φ is eventually strongly monotone,

(ii) $\Phi \in C^1$ (this is implied by the property $f \in C^1$),

(iii) Φ is dissipative in the sense that the point at infinity is repulsive.

Then

(1) for almost all $x \in \mathbb{R}^n$, $\omega(x)$ is an equilibrium point of $\dot{x} = f(x)$ (the exceptional set is of measure zero and is nowhere dense),

(2) (1) is valid for all non-periodic points x with the property that $x \leq \Phi(T, x)$ for some $T > 0$,

(3) periodic orbits cannot be asymptotically stable,

(4) if $x \leq y$ then
either $\omega(x) = \omega(y) = \{e\}$ (the same equilibrium point for x and y),
or $z \leq w$ for each $z \in \omega(x)$ and $w \in \omega(y)$ (and $z \neq w$ for $x \neq y$),

(5) if $y, z \in \omega(x)$ and $y \neq z$ then
$y \leq z$ fails.

– **Application of Hirsch theorem**

For parameters $\alpha, \beta > 0$, consider a bidirectionally coupled CNN ring of equations
$\dot{x_i} = -x_i + \alpha\sigma_{arctan}(x_{i-1}) + \beta\sigma_{arctan}(x_{i+1})$, $i = 1,2,\ldots,n$.
The the conditions of Hirsch theorem are all satisfied.

Moreover, as proven by the Siena group (see [21], [22], [23], [24], [25]), all conclusions of Hirsch Theorem remain valid if the smooth sigmoid function σ_{arctan} is replaced by the piecewise linear and saturated sigmoid function σ_{abs}. Note that this weakening of the assumptions on the sigmoid output function implies that conditions (i)-(ii) of Hirsch Theorem are violated.

DOI:10.15774/PPKE.ITK.2014.007

– **Perron-Frobenius theorem, a result related to Hirsch theorem**

Let $M = \{m_{ij}\}_{i,j}$ be an $n \times n$ non-negative matrix: $m_{ij} \geq 0 \quad i,j = 1,2,\ldots,n$. If M is irreducible, there exists a dominant eigenvalue $\lambda = \lambda_{dom} > 0$ and a dominant eigenvector $s = s_{dom} = (s_1, s_2, \ldots, s_n)$ with the properties that $\lambda \geq |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_n|$ and $s_i > 0$ for $i = 1,2,\ldots,n$. The dominant eigenspace is one-dimensional.
If M is primitive,then the dominant eigenvalue satisfies $\lambda > |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_n|$.

A positive matrix is primitive and a primitive matrix is irreducible. Irreducibility means that matrix M is not similar via a permutation to a block upper triangular matrix (that has more than one block of positive size). Irreducibility makes sense in graph theory, too: a directed graph is strongly connected if and only if its adjacency matrix is irreducible. The non-negative matrix M is primitive if its k-th power M^k is positive for some fixed k.

Recall the last sentences on "Eigenvalues and stability of a periodic orbit".
In our application to the long-transient periodic orbit of CNN rings, the dominant Floquet eigenvalue is metastable.

(Applications of Perron-Frobenius theorem to stochastic matrices (finite Markov chains) are irrelevant.)

DOI:10.15774/PPKE.ITK.2014.007

DOI:10.15774/PPKE.ITK.2014.007

Appendix B

Computation of equilibrium points

This appendix reviews in details the computations of equilibrium points, which method was referred throughout Section 2.5.

The dynamics of every cell is according to the following equations (as earlier in Equation 2.1 and in Equation 2.2):

$$\dot{x}_{ij} = -x_{ij}(t) + \sum_{|k-i|\leq rd} \sum_{|l-j|\leq rd} A(i-k, j-l) y_{kl}(t) + \sum_{|k-i|\leq rd} \sum_{|l-j|\leq rd} B(i-k, j-l) u_{kl}(t) + z \quad \text{(B.1)}$$

$$y = f_{pwl}(x) = \frac{1}{2}(|x+1| - |x-1|) \quad \text{(B.2)}$$

where (as in Equation 2.3 and in Equation 2.4):

$$A = \begin{bmatrix} 0 & 0 & 0 \\ s & p & q \\ 0 & r & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; z = z \quad \text{(B.3)}$$

with

$$q = -s. \quad \text{(B.4)}$$

Let us introduce the following notations

$$\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_{64}), \quad \text{(B.5)}$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_{64}), \quad \text{(B.6)}$$

$$\mathbf{c} = (c_1, c_2, \ldots, c_{64}), \quad \text{(B.7)}$$

with the relation

$$M(\boldsymbol{\xi})\mathbf{x} + \mathbf{c} = \mathbf{0}. \quad \text{(B.8)}$$

DOI:10.15774/PPKE.ITK.2014.007

Matrix $M(\boldsymbol{\xi})$ contains the logical case-separation (later in details) and the boundary conditions, while column vector $\mathbf{c}$ involves the measurement data. The coordinates of the index vector $\boldsymbol{\xi}$ are

$$\xi_i = \begin{cases} -1 & \text{whenever} \quad x_i \leq -1 \\ 0 & \text{whenever} \quad -1 < x_i < 1 \\ 1 & \text{whenever} \quad 1 \leq x_i \end{cases}. \tag{B.9}$$

Since x_i is obtained as a coordinate of the solution of a linear algebraic equation with real constant coefficients (vector $\mathbf{c} \in \mathbb{R}^{64}$ in B.8 is given by the input), the possibility of $x_i = \pm 1$ can be neglected. For the same reason, non-differentiability of the piecewise-linear function (Equation B.2) does not make any complications.

Since our piecewise-linear function has three linear regions, this means 3^{64} combinatorial cases to consider. However, the special structure of templates A and B allows us to divide the system into smaller parts, according to the followings:

$$\boldsymbol{\xi} = (\boldsymbol{\xi}^1, \boldsymbol{\xi}^2, \ldots, \boldsymbol{\xi}^8), \tag{B.10}$$

$$\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^8), \tag{B.11}$$

$$\mathbf{c} = (\mathbf{c}^1, \mathbf{c}^2, \ldots, \mathbf{c}^8), \tag{B.12}$$

where

$$\begin{aligned} \boldsymbol{\xi}^1 &= (\xi_1, \xi_2, \ldots, \xi_8), \\ \boldsymbol{\xi}^2 &= (\xi_9, \xi_{10}, \ldots, \xi_{16}), \\ &\vdots \\ \boldsymbol{\xi}^8 &= (\xi_{57}, \xi_{58}, \ldots, \xi_{64}), \end{aligned}$$

and the coordinates of $\mathbf{x}$ and $\mathbf{c}$ are defined on the same way. With this notation, the linear algebraic system of equations can be rewritten in a lower triangular block matrix form

$$M(\boldsymbol{\xi})\mathbf{x}+\mathbf{c}=\mathbf{0} \Leftrightarrow \begin{bmatrix} M_1(\boldsymbol{\xi}^1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ rI & M_2(\boldsymbol{\xi}^2) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & rI & M_3(\boldsymbol{\xi}^3) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & rI & M_4(\boldsymbol{\xi}^4) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & rI & M_5(\boldsymbol{\xi}^5) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & rI & M_6(\boldsymbol{\xi}^6) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & rI & M_7(\boldsymbol{\xi}^7) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & rI & M_8(\boldsymbol{\xi}^8) \end{bmatrix} \begin{bmatrix} \mathbf{x}^1 \\ \mathbf{x}^2 \\ \mathbf{x}^3 \\ \mathbf{x}^4 \\ \mathbf{x}^5 \\ \mathbf{x}^6 \\ \mathbf{x}^7 \\ \mathbf{x}^8 \end{bmatrix} + \begin{bmatrix} \mathbf{c}^1 \\ \mathbf{c}^2 \\ \mathbf{c}^3 \\ \mathbf{c}^4 \\ \mathbf{c}^5 \\ \mathbf{c}^6 \\ \mathbf{c}^7 \\ \mathbf{c}^8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{B.13}$$

DOI:10.15774/PPKE.ITK.2014.007

The solution coordinate octets can be computed as

$$\mathbf{x}^1 = -M_1^{-1}(\boldsymbol{\xi}^1)\mathbf{c}^1 \tag{B.14}$$
$$\mathbf{x}^2 = -M_2^{-1}(\boldsymbol{\xi}^2)[\mathbf{c}^2 + r\mathbf{x}^1] \tag{B.15}$$
$$\mathbf{x}^3 = -M_3^{-1}(\boldsymbol{\xi}^3)[\mathbf{c}^3 + r\mathbf{x}^2] \tag{B.16}$$
$$\mathbf{x}^4 = -M_4^{-1}(\boldsymbol{\xi}^4)[\mathbf{c}^4 + r\mathbf{x}^3] \tag{B.17}$$
$$\mathbf{x}^5 = -M_5^{-1}(\boldsymbol{\xi}^5)[\mathbf{c}^5 + r\mathbf{x}^4] \tag{B.18}$$
$$\mathbf{x}^6 = -M_6^{-1}(\boldsymbol{\xi}^6)[\mathbf{c}^6 + r\mathbf{x}^5] \tag{B.19}$$
$$\mathbf{x}^7 = -M_7^{-1}(\boldsymbol{\xi}^7)[\mathbf{c}^7 + r\mathbf{x}^6] \tag{B.20}$$
$$\mathbf{x}^8 = -M_8^{-1}(\boldsymbol{\xi}^8)[\mathbf{c}^8 + r\mathbf{x}^7]. \tag{B.21}$$

This structure is the reason why we can compute the solution in eight successive cycles. It is even more important that the overwhelming majority of the 3^{64} theoretical cases is automatically excluded, partly because $M_i(\boldsymbol{\xi}^i)$'s are of rank < 8 and thus making the linear algebraic equation $M_i(\boldsymbol{\xi}^i)\mathbf{x}^i + \mathbf{c}^i = \mathbf{0}$ unsolvable (vector $\mathbf{c}^i \in \mathbb{R}^8$ is given by the input), partly because the x_i's computed (if computable) conflict with preassumption B.9. Please see the actual numbers of such cases in the numerical example given at the end of this Appendix. All in all, the probability when the coordinate octets $\mathbf{x}_1, \ldots, \mathbf{x}_8$ of the state variable are actually computable (i.e., if the $M_i(\boldsymbol{\xi}^i)$'s are invertible, and preassumption B.9 is satisfied) is almost zero.

Establishing the connection between the cell indexes and the position of the pixels, the Reader is kindly reminded of Figure 2.10 and 2.11. The bottom row of the sensor array (8th row in Figure 2.11) is indexed in this mathematical description as $\mathbf{x}^1 = (x_1, \ldots, x_8)$; the positions of coordinates of $\boldsymbol{\xi}$ and $\mathbf{c}$ are similarly.

The four main steps of the equilibrium point (EP) computing algorithm are as follows:

I. The initial step: generating all of the possible index vectors to the bottom row of the sensor array ($N = 8$ cells in the bottom row, 3^N index vectors):

$$\xi_1, \xi_2, \xi_3, ..., \xi_8 : \quad \xi_i \in \{-1; 0; 1\},\ \xi_i = \begin{cases} -1 & \text{whenever} \quad x_i \leq -1 \\ 0 & \text{whenever} \quad -1 < x_i < 1 \\ 1 & \text{whenever} \quad 1 \leq x_i \end{cases}$$

II. Computations for every index vector (regarding the bottom row of the sensor array) is as follows:

DOI:10.15774/PPKE.ITK.2014.007


```
1: M ← zeros(8, 8)
2: c ← zeros(8, 1)
3: for i ← 1, 8 do
4:     M_{i,i} ← −1
5: end for
6: for i ← 1, 8 do
7:     //                                   ▷ intentionally left blank
8:     //                                   ▷ intentionally left blank
9:     //                                   ▷ intentionally left blank
10:    if ξ_i == 0 then
11:        M_{i,i} ← M_{i,i} + (p + r)
12:        if i == 1 then
13:            M_{i,i} ← M_{i,i} + s
14:            M_{i+1,i} ← M_{i+1,i} + s
15:        end if
16:        if 1 < i < 8 then
17:            M_{i−1,i} ← M_{i−1,i} − s
18:            M_{i+1,i} ← M_{i+1,i} + s
19:        end if
20:        if i == 8 then
21:            M_{i,i} ← M_{i,i} − s
22:            M_{i−1,i} ← M_{i−1,i} − s
23:        end if
24:    else
25:        c_i ← c_i + (p + r)ξ_i
26:        if i == 1 then
27:            c_i ← c_i + sξ_i
28:            c_{i+1} ← c_{i+1} + sξ_i
29:        end if
30:        if 1 < i < 8 then
31:            c_{i−1} ← c_{i−1} − sξ_i
32:            c_{i+1} ← c_{i+1} + sξ_i
33:        end if
34:        if i == 8 then
35:            c_i ← c_i − sξ_i
36:            c_{i−1} ← c_{i−1} − sξ_i
37:        end if
38:    end if
39: end for
40: for i ← 1, 8 do
41:    c_i ← c_i + bu_i + z
42: end for
43:        ▷ at this point, matrix M and vector c are filled with the normal setup
    Mx + c = 0
```

DOI:10.15774/PPKE.ITK.2014.007

44: $M^{-1} \leftarrow$ invertmatrix(M) ▷ 'rmatrixinverse' routine in the ALGLIB package
45: **for** $i \leftarrow 1, 8$ **do**
46: $c_i \leftarrow -c_i$
47: **end for**
48: $x' \leftarrow M^{-1}c$ ▷ matrix-vector product
49: valid $\leftarrow TRUE$
50: **for** $i \leftarrow 1, 8$ **do**
51: **if** $(\xi_i == 1$ **AND** $x'_i < 1)$ **OR**
$(\xi_i == -1$ **AND** $x'_i > -1)$ **OR**
$(\xi_i == 0$ **AND** abs$(x'_i) > 1)$ **then**
52: valid $\leftarrow FALSE$
53: **break**
54: **end if**
55: **end for**
56: **if** NOT valid **then**
57: **break** ▷ return to the beginning, to the next index vector
58: **else**
59: $\lambda \leftarrow$ eigenvaluecomp(M) ▷ 'rmatrixevd' routine in the ALGLIB package
60: **save**(λ) ▷ permanently store these newly computed eigenvalues
61: $x \leftarrow x'$
62: **save**(x) ▷ permanently store this newly computed solution vector
63: **end if**

III. Similarly to Step I: generating all of the possible index vectors to the second bottom row of the sensor array (in Figure 2.11 the 7th row). The indexes will be reused $(i : 1, 2, ..., 8)$ for x, ξ and u (as an example: the first pixel of the 7th row in the input frame is u_1). Until line 48, x' will preserve the coordinate octet of the EPs, previously computed to the underlying row.

IV. Considering every EP of the bottom row of the sensor array (8th row in Figure 2.11) as a boundary value from the viewpoint of the 7th row, we analyze the newly generated index vectors. The computations are really similar to the aforementioned procedure (in Step II), we have to modify only three different parts of the procedure:

1. in the three blank line beginning with line 7, we have to insert the following snippet:

DOI:10.15774/PPKE.ITK.2014.007

if $x_i' \leq -1$ **then** $c_i \leftarrow c_i - r$ **end if**

if $x_i' \geq 1$ **then** $c_i \leftarrow c_i + r$ **end if**

if $-1 < x_i' < 1$ **then** $c_i \leftarrow c_i + r x_i'$ **end if**

2. at line 11 we have to exclude parameter r from the summation, because we inserted the bottom neighbor's state earlier, as a constant. Namely, we have

$M_{i,i} \leftarrow M_{i,i} + p$

3. at line 25 we have to exclude parameter r, again. Namely, we have

$c_i \leftarrow c_i + p\xi_i$

The algorithm computes first the first octets of the EP vectors and eigenvalues. These coordinates belong to the bottom row of the sensor array. Then, for every octet (as boundary condition) the algorithm determines the possible next octets (in the second bottom row of the sensor array). Repeating Step III and Step IV together, we can compute the whole array (solution octet combinations from the different rows, and their stability), from bottom to top.

The ALGLIB numerical library is available[1] for C++ (among other platforms).

The mentioned 'rmatrixinverse' routine indicates the singular cases of the original matrix, in those situations I omitted the underlying $\boldsymbol{\xi}$ combination-vector. Once again, I would like to emphasize that, matrix M does not depend on the measurement data, only on the index vector $\boldsymbol{\xi}$.

The 'rmatrixevd' routine indicates, if the eigenvalue computation did not converged, but during my simulations no such event occured. This routine is based on the QR algorithm with multiple shifts; the implementation (in ALGLIB) is based on the LAPACK 3.0 library.

As noted in Appendix A, the stability of an EP connected to an autonomous differential equation is determined through the eigenvalues of its Jacobian matrix $J = f'(x_0)$ evaluated at x_0. As mentioned above, in our case points of non-differentiability can be neglected. In the followings I describe the Jacobian matrix

[1] http://www.alglib.net/

DOI:10.15774/PPKE.ITK.2014.007

belonging to the index vector $\xi_i = 0; i = 1, ..., 8$ of the bottom row, when every cells are in the interior of the linear region.

$$J(x_0) = \begin{bmatrix} -1+p+r+s & -s & 0 & 0 & 0 & 0 & 0 & 0 \\ s & -1+p+r & -s & 0 & 0 & 0 & 0 & 0 \\ 0 & s & -1+p+r & -s & 0 & 0 & 0 & 0 \\ 0 & 0 & s & -1+p+r & -s & 0 & 0 & 0 \\ 0 & 0 & 0 & s & -1+p+r & -s & 0 & 0 \\ 0 & 0 & 0 & 0 & s & -1+p+r & -s & 0 \\ 0 & 0 & 0 & 0 & 0 & s & -1+p+r & -s \\ 0 & 0 & 0 & 0 & 0 & 0 & s & -1+p+r-s \end{bmatrix} \tag{B.22}$$

In the main diagonal, the r coefficients (regarding the bottom neighbors) are represented due to the zero-flux boundary condition. Similarly, in the top-left and bottom-right corners the virtual cells' coefficients are also presented. If some of the cells are assumed to be in the interior of the saturated regions of the output function ($abs(x_i) > 1$), the related coefficients must be replaced by zeros in the matrix. Please note that matrix $J(x_0)$ can occure in any position of the diagonal of the block matrix in B.13. In addition observe that the 8×8 Jacobian matrix J is the same as matrix $M_i(\boldsymbol{\xi}^i)$, and also coincides with matrix M (without any indices) in the pseudo-code above.

In the context of the bottom row, it is impossible to generate a singular matrix M (thanks to the presence of parameter r and the given template values). However, in the context of the remaining rows, 4593 out of 6561 (3^8) combinations resulted in singular matrices at every row. As two other examples, I would like to show the two main types of these matrices. In the first case, the index vector is $\boldsymbol{\xi}^2 = \{-1; -1; -1; -1; -1; -1; 0; -1\}$, where the zero element between the two non-zero elements generates the problematic part of the matrix. This index vector enforces the following matrix:

$$M(\boldsymbol{\xi}^2) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.1 & -1 \end{bmatrix}. \tag{B.23}$$

As we can see, the main problem here is the zero row, which leads to the zero value of the determinant of matrix $M(\boldsymbol{\xi}^2)$ automatically. At this point, I have to mention that column vector $\mathbf{c}^2$ contains a nonzero element in its appropriate coordinate (not counting the case with almost zero probability, when the appropriate input pixel (belonging to this cell) is represented by 0 on the real-valued scale of interval $[-1; 1]$).

DOI:10.15774/PPKE.ITK.2014.007

The other main type of the singular matrices can be enforced with the following type of index vector pattern: $\boldsymbol{\xi}^3=\{-1;-1;-1;-1;0;0;0;-1\}$, where at least three zero values follow each other. In this case the generated matrix is

$$M(\boldsymbol{\xi}^3)=\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 & -1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.1 & -1 \end{bmatrix}. \tag{B.24}$$

In this case, rows 5 and 7 of matrix $M(\boldsymbol{\xi}^3)$ are linearly dependent, which leads to the zero value of the determinant of matrix $M(\boldsymbol{\xi}^3)$, again.

Closing this appendix I would like to review the numerical details of a specific example. Throughout Section 2.4 and Section 2.5 I analyzed a specific measurement in details. In the case of that measurement, the EP-computation at the 6th input frame resulted the following numbers of combinatorial cases:

sensor array		
Row 1	3^8 / 4593 / 1963	3^8 / 4593 / 1967
Row 2	3^8 / 4593 / 1967	3^8 / 4593 / 1963
Row 3	3^8 / 4593 / 1963	3^8 / 4593 / 1967
Row 4	3^8 / 4593 / 1967	3^8 / 4593 / 1963
Row 5	3^8 / 4593 / 1963	3^8 / 4593 / 1967
Row 6	3^8 / 4593 / 1967	3^8 / 4593 / 1963
Row 7	3^8 / 4593 / 1959	
Row 8	3^8 / 0 / 6560	

The left column symbolizes the rows of the sensor array. The triplets in each row mean the following case-numbers: *total number of index vectors / number of singular matrices from* $M(\boldsymbol{\xi}^i)$ */ number of contradictions with preassumption B.9.* The columns should be read from bottom to top, from one row to the next level only the stable EP coodinate octets are continued. In the 7th row of the sensor array we get 2 stable octets and the right triplet column is also the continuation of the first two triplets in the left triplet column. Both triplet columns lead to one single equilibrium each. As it is seen, the maximal number of combinatorial branches left at each row is $9=3^8-4593-1959$.

DOI:10.15774/PPKE.ITK.2014.007

Appendix C

Detailed results on the basin of attraction

This appendix overviews the numerical results of the basin-of-attraction exploration, which results were briefly presented in Subsection 2.5.3.

Table C.1: Convergence results in the first main case (6th frame; $d1 = 0$, $d2 = 0$), when the initial states are around the input frame.

radius	**0.001**	**0.0025**	**0.005**	**0.0075**	**0.01**	**0.025**	**0.05**	**0.075**	**0.1**	**0.25**	**0.5**	**0.75**	**1**
went away from the "second" stable EP	0%	0%	0%	0%	0%	0.3%	9.3%	12.0%	12.4%	22.3%	31.1%	42.6%	42.4%
arrived to the "first" stable EP	0%	0%	0%	0%	0%	0.2%	8.0%	9.5%	9.1%	20%	29%	39.7%	40.4%

Table C.2: Convergence results in the first main case (6th frame; $d1 = 0$, $d2 = 0$), when the initial states are around the "first" stable EP.

radius	**0.001**	**0.0025**	**0.005**	**0.0075**	**0.01**	**0.025**	**0.05**	**0.075**	**0.1**	**0.25**	**0.5**	**0.75**	**1**
went away from the "first" stable EP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	9.1%	15.6%	21.4%
arrived to the "second" stable EP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	8.5%	14.0%	20.2%

DOI:10.15774/PPKE.ITK.2014.007

Table C.3: Convergence results in the first main case (6th frame; $d1 = 0$, $d2 = 0$), when the initial states are around the "second" stable EP.

radius	0.001	0.0025	0.005	0.0075	0.01	0.025	0.05	0.075	0.1	0.25	0.5	0.75	1
went away from the "second" stable EP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	6.9%	11.0%	17.9%
arrived to the "first" stable EP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	6.0%	9.9%	16.4%

The results regarding the first main case — when the system is defined with the 6th input frame, without any additive term ($d1 = 0$, $d2 = 0$) — are as follows: Table C.1 summarizes the case, when the initial state is around the input frame. In Table C.2 we can see the case, when the initial state is the "first" stable EP. The results regarding the "second" stable EP are in Table C.3. Please remember (on the basis of Figure 2.6), that normally (without any random noise) the system, started from the input frame, converges to the "second" stable EP.

What I can see in Table C.2 and Table C.3: the attractive basins' sizes of the two different stable EPs are more or less similar. Maybe this ball-model around the equilibrium points is rather unrealistic; however, due to the high number of sample points on the surface I think this can be a measure of the basin of attraction.

From Table C.1 I know that the convergence from the input frame to the "second" stable EP is not accidental, the size of the convergence-region around it is acceptable.

The results regarding the second main case — when the system is defined on the 7th input frame with additive terms $d1 = 0.07$, $d2 = 0$ — can be seen as follows: Table C.4 summarizes the results, when the simulations are started from the modified input frame. In Table C.5 we can see the convergence results, when the initial states are around the "first" stable EP. Table C.6 shows those results, when the initial states are around the "second" stable equilibrium point. Here I have to mention a very important difference compared to the first main case (Table C.1, C.2 and C.3): in this main case, the dynamical system converges to the "first" stable EP, if I start it from the modified input frame. Although I have two stable equilibria, this system does not follow the earlier observed behavior, which can be read out from the combined (paired) rows of Figure 2.6, Figure 2.9 and Table 2.1.

DOI:10.15774/PPKE.ITK.2014.007

Table C.4: Convergence results in the second main case (7th frame; $d1 = 0.07$, $d2 = 0$), when the initial states are around the modified input frame.

radius	**0.001**	**0.0025**	**0.005**	**0.0075**	**0.01**	**0.025**	**0.05**	**0.075**	**0.1**	**0.25**	**0.5**	**0.75**	**1**
went away from the "first" stable EP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0.1%	0.3%	0%
arrived to the "second" stable EP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0.1%	0%

Table C.5: Convergence results in the second main case (7th frame; $d1 = 0.07$, $d2 = 0$), when the initial states are around the "first" stable EP.

radius	**0.001**	**0.0025**	**0.005**	**0.0075**	**0.01**	**0.025**	**0.05**	**0.075**	**0.1**	**0.25**	**0.5**	**0.75**	**1**
went away from the "first" stable EP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
arrived to the "second" stable EP	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table C.6: Convergence results in the second main case (7th frame; $d1 = 0.07$, $d2 = 0$), when the initial states are around the "second" stable EP.

radius	**0.001**	**0.0025**	**0.005**	**0.0075**	**0.01**	**0.025**	**0.05**	**0.075**	**0.1**	**0.25**	**0.5**	**0.75**	**1**
went away from the "second" stable EP	0%	0%	0%	0%	0%	6.2%	28.2%	39.1%	45.9%	88.8%	99.3%	99.9%	100%
arrived to the "first" stable EP	0%	0%	0%	0%	0%	6.2%	28.2%	39.1%	45.5%	68.6%	92.5%	98.6%	99.9%

What I can see on the basis of Table C.5 and Table C.6: the "second" stable EP has a much smaller basin of attraction than the "first" stable EP. This is not

DOI:10.15774/PPKE.ITK.2014.007

so surprising, if I remember that the bifurcation curve is really near, where I loose my "second" stable EP (see Figure 2.12). Using the input frame as initial state, I was unable to reach the "second" stable EP (Table C.4), meaning that this input frame is deeply inside the convergence region of the "first" stable EP.

I have to note one more indirect observation, regarding the combined (paired) rows of Figure 2.6, Figure 2.9 and Table 2.1. The emerging "second" stable equilibrium points can have enough large basin of attraction, in this way not just "accidentally" influencing the outcome of the frame-by-frame simulation.

DOI:10.15774/PPKE.ITK.2014.007

References

The author's journal publications

[1] **M. Koller** and Gy. Cserey, "Spatial-temporal active wave computing using infrared proximity array," *International Journal of Circuit Theory and Applications*, vol. 40, no. 12, pp. 1209–1218, 2012.

[2] M. Forti, B. Garay, **M. Koller**, and L. Pancioni, "Long transient oscillations in a class of cooperative cellular neural networks," *International Journal of Circuit Theory and Applications*, 2014.

[3] A. Horváth, **M. Koller**, A. Stubendek, and T. Roska, "Spatial-temporal event detection via frameless cellular wave computing – a review," *Nonlinear Theory and Its Applications*, 2014. accepted.

The author's international conference publications

[4] A. Tar, **M. Koller**, and Gy. Cserey, "3D geometry reconstruction using large infrared proximity array for robotic application," in *Proceedings of IEEE International Conference on Mechatronics, ICM 2009*, (Malaga, Spain), 2009.

[5] **M. Koller** and Gy. Cserey, "CNN computational abilities of large infrared proximity arrays," in *Proceedings of the 12th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2010*, (Berkeley, CA), 2010.

[6] M. Forti, B. Garay, **M. Koller**, and L. Pancioni, "An experimental study on long transient oscillations in cooperative CNN rings," in *Proceedings of the 13th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2012*, (Torino, Italy), 2012.

DOI:10.15774/PPKE.ITK.2014.007

[7] **M. Koller**, "3D spatio-temporal movement detection with adaptively tuned 2D active sensorarray," in *Proceedings of the IRUN Winter School on Nonlinear Dynamics in Cellular Wave Computing*, (Budapest, Hungary), 2013.

[8] M. D. Marco, M. Forti, B. Garay, **M. Koller**, and L. Pancioni, "Multiple metastable rotating waves and long transients in cooperative CNN rings," in *Proceedings of the European Conference on Circuit Theory and Design, ECCTD 2013*, (Dresden, Germany), 2013.

[9] **M. Koller**, A. Horváth, and T. Roska, "Frameless computing for spatial-temporal events," in *Proceedings of the European Conference on Circuit Theory and Design, ECCTD 2013*, (Dresden, Germany), 2013.

[10] **M. Koller** and T. Roska, "Activation light pattern helps detection," in *Proceedings of the 14th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2014*, (Notre Dame, IN), 2014.

[11] **M. Koller** and Gy. Cserey, "Uncertain ground detection by CNN based infrared proximity arrays," in *Proceedings of the 14th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2014*, (Notre Dame, IN), 2014.

The author's other publications

[12] M. Forti, B. Garay, **M. Koller**, and L. Pancioni, "Floquet multipliers of a metastable rotating wave," Tech. Rep. 2013-2, Department of Information Engineering and Mathematical Sciences, University of Siena, 2013.

[13] **M. Koller** and T. Roska, "Frameless spatial–temporal event detection via lighting activation," Tech. Rep. JLR – 4 / 2013, The Jedlik Laboratories, Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, 2013.

Publications connected to the dissertation

[14] T. Roska and Á. Zarándy, "Proactive, adaptive, cellular sensory-computer architecture via extending the CNN univesal machine," in *Proceedings of the European Conference on Circuit Theory and Design, ECCTD 2003*, (Krakow, Poland), 2003.

DOI:10.15774/PPKE.ITK.2014.007

[15] A. Horváth and T. Roska, "Frameless spatial-temporal event detection via delay-templates," Tech. Rep. JLR – 2 / 2013, The Jedlik Laboratories, Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, 2013.

[16] B. Molnár and M. Ercsey-Ravasz, "Asymmetric continuous-time neural networks without local traps for solving constraint satisfaction problems," *PLoS ONE*, vol. 8, 2013.

[17] I. Petrás, *Spatio-temporal patterns and active wave computing.* PhD thesis, Interdisciplinary Doctoral School, FITB, Pázmány Péter Catholic University, Budapest, Hungary, 2005.

[18] Y. Horikawa and H. Kitajima, "Duration of transient oscillations in ring networks of unidirectionally coupled neurons," *Physica D: Nonlinear Phenomena*, vol. 238, no. 2, pp. 216 – 225, 2009.

[19] Y. Horikawa, "Metastable dynamical patterns and their stabilization in arrays of bidirectionally coupled sigmoidal neurons," *Physical Review E*, vol. 88, no. 6, p. 062902, 2013.

[20] M. Hirsch and H. Smith, "Monotone dynamical systems," in *Handbook of Differential Equations, Ordinary Differential Equations (second volume)* (A. Canada, P. Drabek, and A. Fonda, eds.), Elsevier, 2005. available at: `http://math.la.asu.edu/halsmith/book.html`.

[21] M. D. Marco, M. Forti, M. Grazzini, and L. Pancioni, "The dichotomy of omega-limit sets fails for cooperative standard CNNs," in *Proceedings of the 12th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2010*, (Berkeley, CA), 2010.

[22] M. D. Marco, M. Forti, M. Grazzini, and L. Pancioni, "Limit set dichotomy and convergence of semiflows defined by cooperative standard CNNs," *International Journal of Bifurcation and Chaos*, vol. 20, no. 11, pp. 3549–3563, 2010.

[23] M. D. Marco, M. Forti, M. Grazzini, and L. Pancioni, "Limit set dichotomy and convergence of cooperative piecewise linear neural networks," *IEEE Transaction on Circuits and Systems I.*, vol. 58, no. 5, pp. 1052–1062, 2011.

DOI:10.15774/PPKE.ITK.2014.007

[24] M. D. Marco, M. Forti, M. Grazzini, and L. Pancioni, "Convergence of a class of cooperative standard Cellular Neural Network arrays," *IEEE Transaction on Circuits and Systems I.*, vol. 59, no. 4, pp. 772–783, 2012.

[25] M. D. Marco, M. Forti, M. Grazzini, and L. Pancioni, "Further results on convergence of cooperative standard Cellular Neural Networks," in *IEEE International Symposium on Circuits and Systems, ISCAS 2011*, (Rio de Janeiro), 2011.

[26] L. O. Chua and L. Yang, "Cellular Neural Networks: Theory and applications," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1257–1290, 1988.

[27] J. S. Kelso, "Multistability and metastability: understanding dynamic coordination in the brain," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 367, no. 1591, pp. 906–918, 2012.

[28] G. Werner, "Metastability, criticality and phase transitions in brain and its models," *Biosystems*, vol. 90, no. 2, pp. 496–508, 2007.

[29] B. Deng, "Metastability and plasticity in a conceptual model of neurons," *Journal of Integrative Neuroscience*, vol. 9, no. 1, pp. 31–47, 2010.

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

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

[32] T. Roska and L. O. Chua, "Cellular Neural Networks with nonlinear and delay-type template elements," in *Proceedings of IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 1990*, (Budapest, Hungary), pp. 12–25, IEEE, 1990.

[33] L. O. Chua and T. Roska, *Cellular Neural Networks and visual computing, Foundations and applications.* Cambridge University Press, 2002.

[34] K. Karacs, Gy. Cserey, Á. Zarándy, P. Szolgay, Cs. Rekeczky, L. Kék, V. Szabó, G. Pazienza, and T. Roska, *Software library for cellular wave computing engines.* Cellular Sensory Wave Computers Laboratory, HAS and the Jedlik Laboratories, PPCU, Budapest, Hungary, 2010.

DOI:10.15774/PPKE.ITK.2014.007

[35] Á. Rodríguez-Vázquez, R. Domínguez-Castro, F. Jiménez-Garrido, S. Morillas, J. Listán, L. Alba, C. Utrera, S. Espejo, and R. Romay, "The Eye-RIS CMOS vision system," in *Analog circuit design*, pp. 15–32, Springer, 2008.

[36] Á. Zarándy and Cs. Rekeczky, "Bi-i: A standalone cellular vision system, Part I. Architecture and ultra high frame rate processing examples," in *Proceedings of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2004*, (Budapest, Hungary), 2004.

[37] Á. Zarándy and Cs. Rekeczky, "Bi-i: a standalone cellular vision system, Part II. Topographic and non-topographic algorithms and related applications," in *Proceedings of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2004*, (Budapest, Hungary), 2004.

[38] A. Rodriguez-Vázquez, S. Espejo, R. Dominguez-Castro, J. L. Huertas, and E. Sánchez-Sinencio, "Current-mode techniques for the implementation of continuous- and discrete-time cellular neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 3, pp. 132–146, 1993.

[39] S. Kavusi and A. E. Gamal, "A quantitative study of high dynamic range image sensor architectures," *IS&T/SPIE Electronic Imaging*, vol. 5301, pp. 264–275, 2004.

[40] R. Wagner, *Mammalian Retina and the CNN universal machine: Locally adaptive algorithms and examining ON-OFF interactions.* PhD thesis, Interdisciplinary Doctoral School, FITB, Pázmány Péter Catholic University, Budapest, Hungary, 2007.

[41] A. N. Zaikin and A. M. Zhabotinsky, "Concentration wave propagation in two-dimensional liquid-phase self-oscillating system," *Nature*, vol. 225, pp. 535–537, 1970.

[42] F. Corinto, V. Lanza, and M. Gilli, "Spiral waves in bio-inspired oscillatory dissipative media," *International Journal of Circuit Theory and Applications*, vol. 36, no. 5-6, pp. 555–571, 2008.

[43] A. C. Scott, "The electrophysics of a nerve fiber," *Rev. Mod. Phys.*, vol. 47, pp. 487–533, Apr 1975.

DOI:10.15774/PPKE.ITK.2014.007

[44] P. Newell and J. Reissig, "Microbial Interactions (Receptors and Recognition, Series B)," 1977.

[45] B. Goodwin, *How the leopard changed its spots: The evolution of complexity*. Princeton Univ Pr, 2001.

[46] L. Chua, *CNN: A paradigm for complexity*. World Scientific Pub Co Inc, 1998.

[47] L. Chua, "CNN: A vision of complexity," *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, vol. 7, no. 10, p. 2219, 1997.

[48] R. Dogaru and L. O. Chua, "Edge of chaos and local activity domain of the gierer-meinhardt CNN," *Int. J. of Bifurcation and Chaos*, vol. 8, no. 12, pp. 2321–2340, 1998.

[49] R. Dogaru and L. O. Chua, "Edge fo chaos and local activity domain of fitzhugh-nagumo equation," *Int. J. of Bifurcation and Chaos*, vol. 8, no. 2, pp. 211–257, 1998.

[50] R. Dogaru and L. O. Chua, "Edge of chaos and local activity domain of the brusselator CCN," *Int. J. of Bifurcation and Chaos*, vol. 8, no. 6, pp. 1107–1130, 1998.

[51] L. O. Chua, "Passivity and complexity," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 46, pp. 71–82, Jan. 1999.

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

[53] "Leonar3do system software 2.3 and leonar3do hardware kit – user guide." `http://downloads.leonar3do.com/LeoStore/LeoSystemManual.pdf`, cited Jun 2014.

[54] "Leap motion controller." `https://www.leapmotion.com/`, cited Jun 2014.

[55] M. Andersen, T. Jensen, P. Lisouski, A. Mortensen, M. Hansen, T. Gregersen, and P. Ahrendt, "Kinect depth sensor evaluation for computer vision applications," Tech. Rep. ECE-TR-6, Department of Engineering – Electrical and Computer Engineering, Aarhus University, 2012.

DOI:10.15774/PPKE.ITK.2014.007

[56] "Company canesta article on wikipedia." `http://en.wikipedia.org/wiki/Canesta`, cited Jun 2014.

[57] P. M. Novotny and N. J. Ferrier, "Using infrared sensors and the phong illumination model to measure distances," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1644–1649, IEEE, 1999.

[58] G. Benet, F. Blanes, J. E. Simó, and P. Pérez, "Using infrared sensors for distance measurement in mobile robots," *Robotics and autonomous systems*, vol. 40, no. 4, pp. 255–266, 2002.

[59] V. Pavlov, H. Ruser, and M. Horn, "Feature extraction from an infrared sensor array for localization and surface recognition of moving cylindrical objects," in *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*, pp. 1–6, IEEE, 2007.

[60] Á. Tar, *Low resolution infrared proximity array based 3D object and force reconstruction, and modular oscillatory arrays.* PhD thesis, Interdisciplinary Doctoral School, FITB, Pázmány Péter Catholic University, Budapest, Hungary, 2012.

[61] Y. Do and J. Kim, "Infrared range sensor array for 3d sensing in robotic applications.," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[62] T. Roska, T. Boros, A. Radványi, P. Thiran, and L. O. Chua, "Detecting moving and standing objects using Cellular Neural Networks," *International Journal of Circuit Theory and Applications*, vol. 20, no. 5, pp. 613–628, 1992.

[63] "DARPA Robotics Challenge website." `http://www.theroboticschallenge.org/`, cited May 2014.

[64] P. Thiran, *Dynamics and self–organization of locally coupled neural networks.* PhD thesis, École Polytechnique Fédérale de Lausanne, Département D'Électricité, Lausanne, Switzerland, 1996.

[65] Z. P. Kilpatrick, *Spatially structured waves and oscillations in neuronal networks with synaptic depression and adaptation.* PhD thesis, Department of Mathematics, The University of Utah, Salt Lake City, UT, 2010.

DOI:10.15774/PPKE.ITK.2014.007

[66] B. Garay, "Nonlinear dynamical systems," 2013. available at: http://digitus.itk.ppke.hu/~garay/NDS_jegyzet/NDS_jegyzet_Garay_final_v4.pdf.