KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, — B-3001 Leuven

Optimisation and Robustness of Cellular Neural Networks

Promotoren :
Prof. Dr. ir. J. SUYKENS
Prof. Dr. ir. J. VANDEWALLE

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Samuel XAVIER DE SOUZA

June 2007

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, — B-3001 Leuven

Optimisation and Robustness of Cellular Neural Networks

Jury :
Prof. H. Van Brussel, voorzitter
Prof. J. A. K. Suykens, promotor
Prof. J. Vandewalle, promotor
Prof. A. Barbé
Prof. J. Van Impe
Prof. M. Van Hulle
Prof. T. Roska (Hungarian Academy of Sciences)
Prof. G. Gielen

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Samuel XAVIER DE SOUZA

U.D.C. 681.3∗I4

June 2007

D/2007/7515/58
ISBN 978–90–5682–824–0

Foreword

Scientists study the world as it is;
engineers create the world that has never been.
Theodore von Kármán

I would like to thank many people. Not only those who helped me directly and indirectly during the course of my Ph.D. but also those who have supported me in my life and made possible for me to have reached this point.

Eu agradeço à minha mãe por todo o seu amor e esforço indispensáveis à minha formação, e ao meu pai por ter me ensinado o que é ser bom de coração. Também agradeço às minhas tias Maria e Graça por todo o apoio. Serei eternamente grato.

I thank my two promoters, Prof. Johan A. K. Suykens and Prof. Joos Vandewalle, for the opportunity to do this Ph.D. and for their help and patience while advising me with my work and correcting my manuscripts. I also thank my co-authors in our scientific publications: Prof. Desiré Bollé, Dániel Hillier, Prof. Müştak E. Yalçın, and my master thesis student Michiel Van Dyck. I have enjoyed my time immensely working with them. A special thanks to Dr. João Ramos for the motivation given on many aspects of this thesis.

Special thanks to my advisers and member of the reading committee, Prof. André Barbé, Prof. Marc Van Hulle, and Prof. Jan Van Impe. Their comments have been always interesting, valuable and constructive.

Thank you to the other members of the jury, Prof. Georges Gielen, who I am looking forward to collaborating with in the future, Prof. Herman Van Brussel, for kindly accepting to be the jury chairman, and Prof. Támas Roska of the Hungarian Academy of Sciences, for accepting the invitation to come to Belgium and be part of the jury. Our motivational chats have

been invaluable. His thoughts and advice have inspired me throughout my Ph.D.

For my appetite for scientific research, I acknowledge my friend and former graduation adviser Prof. Adrião D. Dória Neto and Prof. José Alfredo F. Costa.

For giving me the opportunity to come to Belgium and start my internship at IMEC, thanks to my friend Prof. Carlos Valderrama. Without his help, this doctorate would not have been possible.

I also enjoyed working with my colleagues at SISTA, specially those regulars at the Alma lunch break for keeping the healthy balance between research and non-sense. The friendly support of Péla Noé, Ilse Pardon and Ida Tassen, from SISTA, and Evelyn Dehertoghe, Eliane Kempenaars, and Lut Vanderbracht, from the department, is greatly appreciated.

I am grateful for the help and support that I have received from my beloved parents-in-law, Joris and Mieke, and my little sister for the years that she lived with us and supported us in Belgium.

I thank Davis Goodman, the crazy Canadian, for the countless hours spent correcting my English in this thesis.

Last and most importantly, I thank my family: my wife Sofie, our daughter Lena, and our son Enio. I thank them for being everything to me and for all the love and support that I received during this doctorate.

Abstract

In this thesis we present new methodologies for improving the robustness of analog VLSI visual processors which are based on Cellular Neural Networks (CNN). Such a system can process information at very high speeds, only comparable to today's supercomputers. The regular lattice architecture of CNNs allows massive parallelism, which makes it very suitable for performance-demanding applications in image processing. Its reduced size and power consumption make it easy to embed in portable appliances. The only disadvantage of today's programmable CNNs relates to the analog VLSI technology, which despite remarkable recent advances, cannot guarantee a sufficiently high accuracy and reliability that is needed in many applications. In this thesis we describe methodologies for customised tuning of CNN chips and learning of new complex operations. Based on well established methods such as design centring and trajectory learning, the techniques described here prove to be very useful in reducing the effects of parameter deviation and post-manufacturing interference in the operation of CNN-based processors. We show that on-chip learning is not only viable but also better than simulation-based learning for being much faster. We also present a new global optimisation method that is suitable for CNN optimisation. The new method of Coupled Simulated Annealing makes use of coupling in order to allow multiple Simulated Annealing (SA) processes to cooperate toward finding the global optimum of multi-modal and multi-dimensional optimisation problems. A number of proof-of-concept applications is presented in order to show the effectiveness of our methodologies. These applications serve to demonstrate the potential for future VLSI CNN systems toward ultra-fast visual applications such as quality control in agricultural, semiconductors, textile and other industries, surveillance and traffic analysis, biochemical process inspection, intelligent systems in the automotive industry, visual computer/gaming interaction and others.

Notation

Mathematical notation

$a, b, c \in \mathbb{R}$	Scalar variables
$\mathbf{y}, \mathbf{x} \in \mathbb{R}^N$	Vector variables
x_i	The i^{th} element of $\mathbf{x} \in \mathbb{R}^N$
$A, B \in \mathbb{R}^{(2r+1)\times(2r+1)}$	Template matrices with $r \in \mathbb{N}^*$, typically $r = 1$
$a_{i,j}$	The element of the i^{th} row and j^{th} column of $A \in \mathbb{R}^{(2r+1)\times(2r+1)}$
$\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{M\times N}$	Matrix variables
$x_{i,j}$	Element of the i^{th} row and j^{th} column of a matrix $\mathbf{X} \in \mathbb{R}^{M\times N}$ or of a vectorised matrix $\mathbf{x} \in \mathbb{R}^{MN}$
$N(c)$	Set of indices of all cells connected to the cell c in a CNN
$\|\|\mathbf{y}\|\|_2$	Euclidean norm of $\mathbf{y} \in \mathbb{R}^N$
$\min_{A,B,z}$	Function minimisation over A, B, and z

Acronyms and abbreviations

AMC	Analogic Macro Code
ASA	Adaptive Simulated Annealing
AWC	Active Wave Computing
CLM	Coupled Local Minimisers
CNN	Cellular Neural Networks
CNN-UM	CNN Universal Machine
CSA	Coupled Simulated Annealing
CSA-BA	Blind Acceptance
CSA-M	Coupled Simulated Annealing-Modified
CSA-MuSA	Multi-state Simulated Annealing
CSA-MwVC	CSA-M with Variance Control
DSP	Digital Signal Processor
FSR	Full Signal-Range
GA	Genetic Algorithms
HPC	High Performance Computing
IPC	Inter-Process Communication
ISR	Improved Signal-Range
MPI	Message Parsing Interface
MSA	Multi-start SA
RNN	Recurrent Neural Networks
SA	Simulated Annealing
SDK	Software Development Kit
SIMD	Single-Instruction Multiple-Data
TS	Training Set
VLSI	Very Large-Scale Integration
VRML	Virtual Reality Modelling Language
VSoC	Vision System on a Chip

para Lena e Enio.

Contents

Chapter 1

Introduction

1.1 Cellular neural networks as a platform for computation

This thesis concerns Cellular Neural Networks (CNN) and optimisation. In other words, we use optimisation to improve existing CNN implementations. These implementations are integrated electronic circuits based on Very Large-Scale Integration (VLSI) technology. These circuits, commonly called CNN chips, can be used to perform highly efficient computations. As the name suggests, CNNs are composed of cells that are connected to their neighbours forming a network. Each of these cells is a dynamical system in its own, which means that its state evolves in time according to a specific rule. Up to a certain degree, with a programmable rule, a single cell can also be used to perform computation. In cooperation with the other cells in the network, the computation potential of such chips is only comparable to that of today's supercomputers.

There are other advantages of CNN-based computation in comparison with today's classical digital computers. While the latter is based on digital signals and binary logic, the former is based on analogue signals and connection or coupling rules, called templates in CNN terminology. CNN chips are inherently parallel processors while the majority of the computation realised today in the digital world is sequential. Additionally, although there exists a trend in better exploiting the benefits of parallelism in digital computing [71, 44], digital processors are also relatively larger and much more power hungry than CNN-based processors. The latter are many orders of

magnitude more efficient than the former while only using a fraction of the silicon area that is needed in comparison with the digital technology. The reasons why CNN-based computation is so much more efficient can roughly be summarised into two inherent principles of CNNs.

The first principle is the analogue processing nature of the cells. Being a continuous dynamical system, the time evolution of the state of a cell is defined in the continuous time domain[1]. This means that the state of the cells does not need an external *clock* stimulus to evolve as digital processors do. This fact in its own not only allows faster computation but also permits a lower power consumption. The main reason why digital processors consume increasingly more power is that the energy dissipated by such processors scales approximately quadratically with the clock frequency due to the quadratic relation between power and voltage. Doubling the processor frequency from 1 GHz to 2 GHz would roughly mean an increase in power consumption with a factor of 4. This is certainly one of the reasons for the existence of a new trend in the processor industry toward multi-core processors. The benefits of such trend is very encouraging. For instance, with 8 processing cores per chip, the new IBM Cell processor [15, 102] consumes 60 to 80 Watts at 4 GHz. To achieve the same performance in a single-core processor, it would be needed a factor 8 increase in the clock frequency resulting in hypothetical value of $8 \times 4 = 32$ GHz, which would increase the power consumption with a factor $8^2 = 64$ resulting in the impractical values of 3.84 to 5.12 kilowatts. Parallel or distributed processing is therefore a key feature in the future of digital computation. In CNN-based processors, this feature is present since the invention of CNNs. In fact, today's CNN-based processors features more than 25,000 processing cells per chip distributed in a 176×144 regular grid.

The second principle responsible for the better efficiency of CNN-based processors is the local nature of the couplings between the processing cells. Because these connections are by definition local and therefore restricted to the neighbouring cells, the physical circuit connections are simpler to build in VLSI technology and therefore permits the fabrication of large numbers of cells per chip. With more than 25,000 processing cells per chip, a CNN-based processor can achieve extraordinary performances in

[1]Discrete time cellular neural networks is a type of CNNs whose state is defined in discrete time instead of continuous time. Although the methods described in this thesis may also be used for this type of CNNs, we only approach the continuous case.

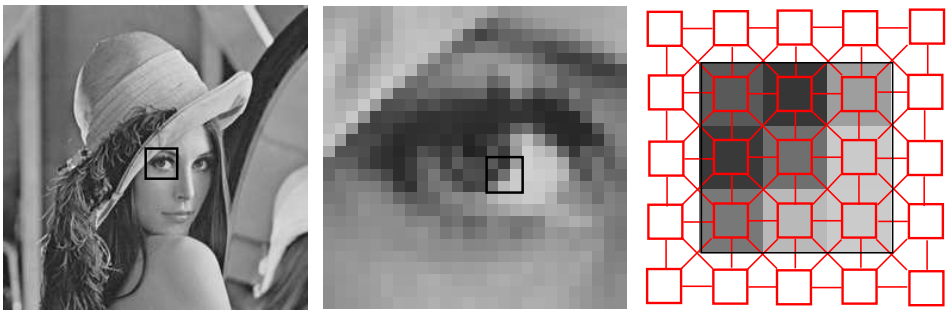


Figure 1.1: CNNs are very suitable for image processing thanks to the one-to-one relationship between pixels and cells.

the order of Tera operations per second, this with a power consumption in the order of hundreds of milliwatts. Moreover, the local nature of the couplings and the resulting network architecture make CNNs very suitable for image processing. One of the reasons is that each pixel in an image can be processed by each cell as depicted in Figure 1.1. Another reason is that today's CNN chips behave like Single-Instruction Multiple-Data (SIMD) parallel computers. That means that the chip executes a single instruction or template across the whole network. This feature is very desirable in image processing applications.

After much said about the advantages of CNNs over classical digital computation, it remains to be explained why we still need to use optimisation in order to improve the existing CNN VLSI implementations. Although CNN-based processors are much more efficient than their digital counterparts w.r.t. processing power and speed per power dissipation and silicon area, unfortunately, robustness is today still a weak point. This is the main point of focus in this thesis and the reason to use optimisation to make robustness a stronger aspect in these systems.

1.2 Toward robust CNN-based processors

With the actual analogue VLSI technology, considerably larger CNN-processors can be implemented in a single chip [85, 84]. Such a chip can perform image processing tasks with extremely high throughput data rates, which make it very suitable for a wide range of image processing tasks, especially

for real-time applications [30]. Nevertheless, each CNN chip has analogue parameters that are slightly different from those ideal ones used on simulators. The causes are mainly noise in electrical components of cells as well as imperfections in the fabrication process. This may often result in erroneous behaviour of some cells. These differences between ideal structures and real chips prevent the cells in the CNN chip to react in the same way as in the simulators and causes sometimes serious differences between simulator results and chip results. Tests of the early templates [115], which were developed for simulators, on VLSI chips proved that many templates worked incorrectly [139]. Consequently, new template design methods were developed [96, 90, 68, 162] with the purpose of generating templates that are more tolerant against inherent parameter deviations and noise without taking into account specific characteristics of an individual chip.

However, the degree of robustness for different operations are not the same [96]. While templates with a high degree of robustness allow a correct chip response for the given operation, other templates with lower robustness still cause erroneous operation in CNN chips. The robustness value, or degree of robustness, of a template gives a measure of how tolerant the template values are to parameter deviations. When the deviation is larger than the corresponding tolerance range of the given chip parameter, the template does not react properly and produces unexpected and undesirable results. Therefore, even the most chip-independent robust templates will not guarantee a fully correct behaviour for a given CNN chip unless its robustness is sufficiently large to overcome the parameter deviations of the chip. Nevertheless for some applications, one can manually and empirically attempt to tune the templates of a given chip and attempt to make it respond correctly for a given task. Yet, even if the goal is achieved, there is no guarantee that a final template will work for other similar chips. In addition, manually tuning each template used in an application might be a long and very tedious task.

In contrast with the other types of template generation methods, like *design* and *learning* that ignore specific chip characteristics, Földesy *et al.* [38] proposed a method for template *optimisation* and *decomposition* that uses measurements of a specific chip and therefore takes into account its inherent characteristics. Although it was very well defined, the work of Földesy *et al.* presented a few key limitations and therefore was used as the starting point for our research. In the next Section we describe the objectives that

we focused during the evolution of this thesis.

1.3 Objectives

As already stated in the Section 1.1, optimising the robustness of CNN implementations was the main objective pursued during the development of this thesis. We have defined this main goal as a composition of other subgoals. These subgoals were defined in the beginning of our research and sometimes redefined along the way in order to fit the needs that were encountered by our intermediate results. Our final subgoals are defined as follows:

- Establish the causes of erroneous behaviour in CNN chips. There can exist many causes for malfunctioning, such as parameter deviations during the manufacturing processes as well as environment disturbances like temperature variations and electrical noise. Our objective is to define which are the causes of errors and up to what extent they affect the robustness.

- Create a methodology to optimise CNN chips toward robustness. Our objective is to develop such a methodology based on existing optimisation techniques that can use measurements of CNN chips in order to tune their free parameters and in this way reduce or even eliminate the existing operational errors.

- Create a methodology for on-chip CNN learning of spatiotemporal behaviour. It is our objective to develop such a technique in order to give support to the new paradigm of active wave computing [126]. This new computation paradigm uses the spatiotemporal dynamics of CNN to perform computation.

- Create a general purpose global optimisation technique that can be used as the optimisation core of the CNN optimisation techniques developed along this thesis. This subgoal has grown from the need to suppress the weak points of the optimisation method that we had been using for learning and tuning of CNNs. We have noticed that in order to achieve good results it was necessary to involve a lot of human interaction during the optimisation process. This, mainly due to premature convergence to a non-optimal solution. Inspired by the

benefits of coupling in Coupled Local Minimisers (CLM) [131], we establish the objective of developing a global optimisation methodology that via coupling is able to perform cooperative behaviour and play the role of human interaction in the optimisation method that we used originally for CNN optimisation.

- Integrate all the methods developed so far into a single framework for learning, tuning, and robustness optimisation of CNNs using the new coupled global optimisation method.

- Develop proof-of-concept applications to demonstrate the working of new techniques and the potential that CNN-based processors can present when these new post-manufacturing techniques are used.

1.4 Chapters and main contributions

In this Section we provide an overview of the main contributions of the thesis and the organisation of the Chapters. Except for Chapter 2, which contains an introduction to cellular neural networks, all other Chapters are the results of the research done in this thesis. In Chapter 7 we present a collection of applications that were developed using the techniques developed during the course of our research. Although the development of these applications depends on the techniques described in other chapters, it can be read independently. In Figure 1.2 the contents and links between the Chapters are shown.

What follows is an overview of the Chapters and their association with the contributions in this thesis.

Chapter 2 This is a background Chapter where we present the working principles of CNNs and the different cell models and network architectures. We also present a brief history and the state-of-art in VLSI CNN implementations.

Chapter 3 In this Chapter we introduce *chip-specific robustness for CNN templates*, a methodology that does not rely on gradient optimisation methods to find chip-specific optimal propagating and non-propagating CNN templates. The templates are not only tuned for optimality but also for robustness. The resulting chip-specific robust

templates have their parameter values driven to the middle of a correct operating range. This not only suppresses manufacturing imperfections but also minimises the errors caused by post-manufacturing factors such as temperature variations and electrical noise, which may cause the parameter values to fall outside the correct working range. The template values are tuned using a global optimisation technique according to measurements obtained from a specific target chip. This contribution is also described in the references [156, 155].

Chapter 4 In this Chapter we introduce a methodology for *learning of CNN spatiotemporal dynamics*. This methodology is based on the methodology described in Chapter 3. CNN optimisation methods relying on chip measurements to compute cost functions are naturally faster than those relaying on simulations. For this reason, such methods also provide a very suitable platform for learning spatiotemporal behaviour, which demands a large amount of resources. With on-chip cost function calculation, learning of CNN spatiotemporal patterns takes a fraction of the time it would take in modern digital computers. We have developed a methodology for learning these patterns. Spatiotemporal CNN operations are especially important in the new computation paradigm of active wave computing. Besides on-chip learning of new active wave operations, the method we have developed can also be used to optimise the speed of existing CNN operations. This contribution is also described in the references [152, 151].

Chapter 5 In this Chapter we present *Coupled Simulated Annealing (CSA)*, a general purpose class of methods that, inspired by the working of coupling in CNNs, uses cooperative behaviour to guide simulated annealing processes toward the global optimum of a given cost function. The new class of algorithms is characterised by an ensemble of optimisation processes that are coupled to each other by a coupling term. The key difference between a CSA process and a classical Simulated Annealing (SA) process is in the acceptance probability functions. These new functions embed a coupling rule which guides optimisation processes. The CSA acceptance functions can be considered as a generalisation of classical SA acceptance probabilities. The coupling term, which is embedded in the acceptance functions, is defined by a function of the energies of all current solutions in the different

processes. We exemplify the class of algorithms described by CSA with three instance methods. Additionally, we demonstrate how coupling can be used to steer the overall performance of the optimisation by adaptively adjusting the acceptance temperature in a very simple way. Results lead to the conclusion that by choosing the right coupling, considerably better results can be achieved w.r.t. other CSA schemes and classical SA. This contribution is also described in the references [154, 153].

Chapter 6 In this Chapter we proposed a unification of our previous chip-specific CNN optimisation approaches for *systematic CNN learning and optimisation* of spatiotemporal dynamics. The proposed method extends the previous approaches in three main aspects. First, hardware parameters of CNN chips are included in the optimisation. This opens the way to run on actual CNN chips some templates so far believed to be very unstable. Second, we use our own CSA methodology as the optimisation core for learning and tuning, which improves learning speed significantly. Third, the resulting framework for systematically learning and optimisation is presented as a new Matlab toolbox so that the task of the CNN algorithm designer is reduce to defining the operation to be learned as a training set for the optimisation process. Training set design is the most crucial issue of this approach, thus some basic design rules are presented. The proposed framework may become a valuable tool to find new CNN templates and robustly implement them on chip. This contribution is also described in the references [57, 59, 58].

Chapter 7 A number of prototype *applications and demonstrators* have been developed in our group using our chip-specific optimisation techniques. These include: real-time object tracking [150], airborne fingertip mousing [43], and hands-free wheelchair driving [148, 149]. Echocardiogram contour extraction [56], and medical eye tracking and gaze estimation are other examples of prototype applications that are being built using templates tuned with our approach. In this Chapter we present a couple of these applications which used techniques described in Chapters 3, 4, and 6 for tuning and learning of templates.

Chapter 8 Finally, in this Chapter, we present the general conclusions of

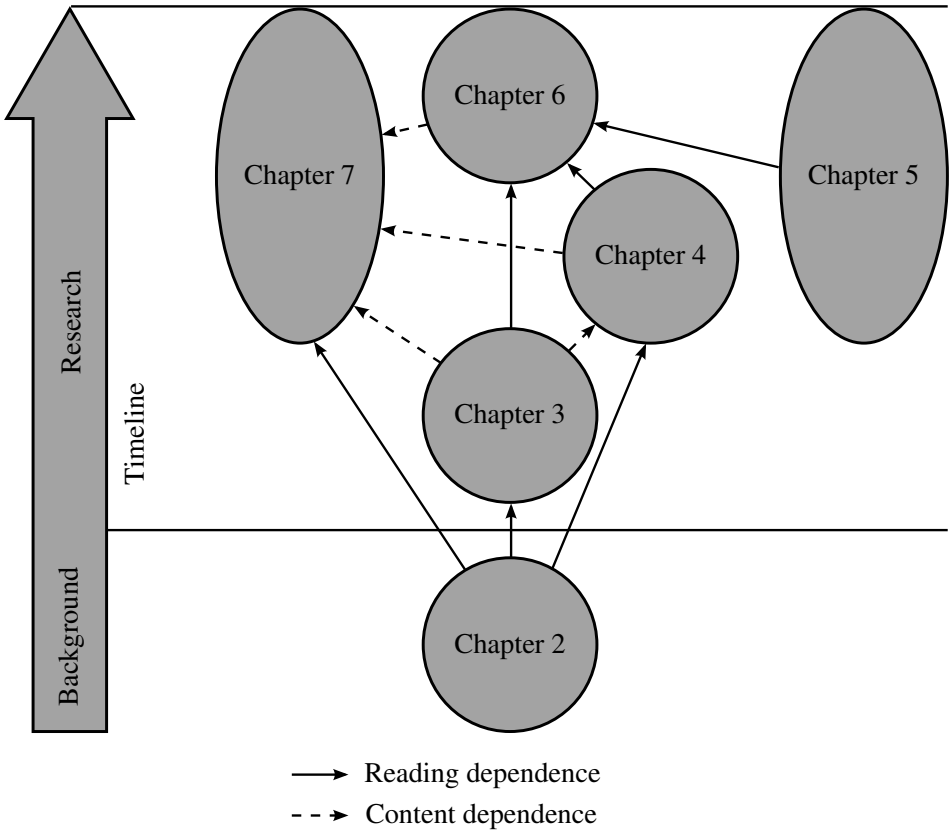


Figure 1.2: Reading and content dependences of the Chapters in this thesis. The research timeline is also depicted here.

this thesis and describe some challenges for future research.

Chapter 2

Coupling, Local Activity, and Cellular Neural Networks

The fundamental limits of science have been continuously expanded along the past centuries. From the discovery that the Earth is round to the finding, at the time, of the most elementary building block of the universe, the atom, these limits have been extended at an extraordinary rate. More and more is known about little things that compose giants and about gigantic things made by little ones [78]. What remains generic is that interactions within a specific level generate larger levels. There are two fundamental principles that form systems within a level: elementary units, here called *cells*; and interactions between cells, here called *coupling*. In fact, a system of coupled cells may represent a cell at larger levels. Coupling is therefore present in all universal levels.

In many systems in nature, elementary cells are modelled as a dynamical system governed by a series of differential equations. Depending on the model properties of these cells, interesting complex dynamics can be observed to exist. Many scientists have studied these properties in attempt to describe the foundations of complexity and its equivalent terminology, e.g. emergence, self-organisation, collective behaviour, etc. Recently, in contrast to these previous studies, Chua [23] has proposed a quantitative, rather than qualitative approach for complexity. In his approach, Chua defines the principle of *local activity*, to which he attributes the origin of

complexity.

Chua also presents *Cellular neural networks* (CNNs) as a paradigm for complexity. CNNs can be used to demonstrate the underlying principle of complexity and therefore it is also a suitable platform for studying various complex phenomena. Besides modelling and simulation of complex and natural dynamics, CNNs have been also used to perform computation. The local characteristic of its connections makes CNNs very appropriate for Very Large-Scale Integration (VLSI) implementations.

In this Chapter we explain how the principles of coupling and local activity are joined together into the versatile platform of cellular neural networks. Theory and implementation of CNNs are explained and discussed with relation to different aspect that are considered relevant to framework of this thesis.

2.1 Coupling and local activity

Universality and complexity are two fundamental principles of science which have been widely studied. *Coupling*, or interaction between individuals or systems, is universal. *Local activity* is the origin of complexity. Although many scientists consider coupling to be ubiquitous in many different fields, only part of these interactions involves complex individuals or systems, which can then generate larger complex systems. The origin of complexity has been recently attributed to the principle of local activity. Previous studies have described complexity, emergence, and its other equivalent terminologies in a *qualitative* way. In contrast, local activity has a *quantitative* mathematical description to characterise complexity. Moreover, in the context of this thesis, perhaps the most important finding w.r.t. complexity and universality is that locally active individuals which are not complex individually may present complexity when coupled together.

In order to establish a quantitative approach for complexity, in his book *CNN: A Paradigm for Complexity* [23], Chua described the principle of local activity. We quote the definition of CNN from his book:

> "CNN is an acronym for either **C**ellular **N**eural **N**etwork when used in context of brain science, or **C**ellular **N**onlinear **N**etwork when used in the context of *coupled dynamical systems*. A CNN is defined by two mathematical constructs:

1. A spatially discrete collection of continuous nonlinear dynamical systems called *cells*, where *information* can be encrypted into each cell via three independent variables called *input*, *threshold*, and *initial state*.
2. A *coupling law* relating one or more relevant variables of each cell $C_{i,j}$ to all neighbour cells $C_{k,l}$ located within a prescribed sphere of influence $S_{i,j}(r)$ of radius r, centred at $C_{i,j}$."

Although Chua focuses on reaction-diffusion CNN equations in the description of local activity, he advocated that the principle is universal and that it holds for any system exhibiting complexity which can be represented by a mathematical model composed of *cells* and *coupling laws*.

Prior to Chua's work, other scientists also have tried to describe complexity although using different terminologies. Chua cites [24] for example Schrödinger's necessary condition for the emergence of life as being the "exchange of energy" from *open* systems [123]; Prigogine's new principle of nature called "the instability of the homogeneous" [104]; Turing's "symmetry breaking" mechanism for morphogenesis [143]; and Smale's question about the axiomatic properties necessary to make the Turing interacting system instable [128]. According to Chua, jargons such as emergence, self-organisation, synergetics, collective behaviour, non-equilibrium phenomena, among others are equivalent terms to describe complexity. Although Chua's initial approach to describe local activity requires circuit theory as a framework, in [24], he provides a proof of the local activity theorem that is mathematically self-contained.

Uncoupled locally active cells can generate complex behaviour such as limit cycles and chaos. When such cells are connected to each other, it is not surprising that complex spatiotemporal dynamics may appear. However, the effect of coupling can go beyond simple spatial propagation of existing single-cell complexity. In fact, 30 year ago Smale posed a paradoxical example of two "mathematically dead" cells becoming "alive" when connected to each other by a diffusive coupling. Smale was confronted with two asymptotically stable cells which begin to oscillate when diffusively coupled. He posed the problem of finding the conditions under which this phenomenon would occur. Pogromsky *et al.* [103] have proposed a solution to Smale's paradox in the context of passivity and minimum phaseness and have shown that, in Smale's terminology, each cell by itself cannot be to-

tally dead in order to become alive after coupled to each other, in the sense that each system alone must have unstable dynamics consistent with some external state constraint. Chua comes to a similar conclusion when solving Smale's problem in the context of the local activity theory. Having formally divided the parameter space into locally passive and locally active disjoint subsets, Chua identifies a subset of the active parameter space called the "sharp edge of chaos" [24] which fulfils Smale's quest for the axiomatic properties necessary to effect oscillations via diffusive coupling. Smale's dead cells, in Chua's view, need to be locally active in the region of the sharp edge of chaos, or in the view or Pogromsky, need to have unstable dynamics consistent with some external state constraint in order to become alive when coupled together.

Coupling and local activity are therefore instruments for generation of spatiotemporal complex behaviours in CNN-like spatially distributed systems. In Chapter 4 we give an example of such behaviours and describe a methodology to learn them with CNNs.

Besides modelling complexity, CNNs can also be used to simulate such phenomena. In fact, in the past years, even locally passive CNNs have been used to address a number of problems. As it is shown in the following Section, CNNs are not only a paradigm for complexity but also a platform to solve and simulate a large variety of complex and non-complex challenges.

2.2 Cellular neural networks

Cellular neural networks were first described by Chua and Yang as a new circuit architecture in the framework of neural networks with important applications in image processing [30], pattern recognition and other areas [28, 27]. A *cell* is the elementary circuit unit of a CNN and it is composed of linear resistors and capacitors; and linear and nonlinear controlled and independent sources. The cells are connected structurally to form a network in such a way that only neighbour cells are connected to each other.

Many different network architectures [63, 159] and cell models [54, 36] have been devised as extensions of Chua and Yang's original proposal. Among all these architectures and cell models, the locality of the connections remains the most important aspect. Despite the fact that such networks are only locally connected, many different types of dynamics have

been reported to emerge [25]. While adjacent cells interact directly only with neighbouring cells, cells that are not immediately connected together can still be affected indirectly because of propagation properties of the dynamics of the network.

In the years following Chua and Yang's papers, a strong relation of CNNs with nonlinear system theory became increasingly more evident [119]. Researchers gradually also adopted the CNN acronym to represent not only Cellular Neural Network but also Cellular Nonlinear Network, especially in the context of coupled dynamical systems.

The most usual CNN architecture is a regular two-dimensional grid as originally proposed by Chua and Yang. In this architecture the cells are disposed regularly in a rectangular grid. Each cell is connected directly to each neighbouring cell located within a certain *neighbourhood* or *sphere of influence*. Starting from this architecture, other n-dimensional grids can easily be devised. An overview of the main types of CNN architectures can be seen in Figure 2.1.

Besides structure, CNNs can also vary according to its cell models. For instance, rather than continuous time dynamics, Harrer and Nossek have proposed a CNN with discrete time steps [54]. This model has been used in application in various areas [18, 95]. Arena *et al.* also proposed a modified CNN model that has an extra coupling law directly dependent on the state of the network [9]. This model was used to realise Chua's circuits [9] and n-double scroll attractors [10]. Some generalised cell models [49, 112, 48] have also been proposed, with applications on detection of moving objects [111, 119] and other image processing tasks [64].

In this Chapter we describe two of the many CNN models existing in the literature, namely, the original Chua and Yang's model and the full-range model, used frequently in CNN integrated circuit implementations. We consider these two models the most relevant ones w.r.t. to the methods that are described in this thesis.

2.2.1 Chua and Yang's CNN model

The original CNN model was derived from the circuit in Figure 2.2 representing an elementary cell. The instantaneous current through the capaci-

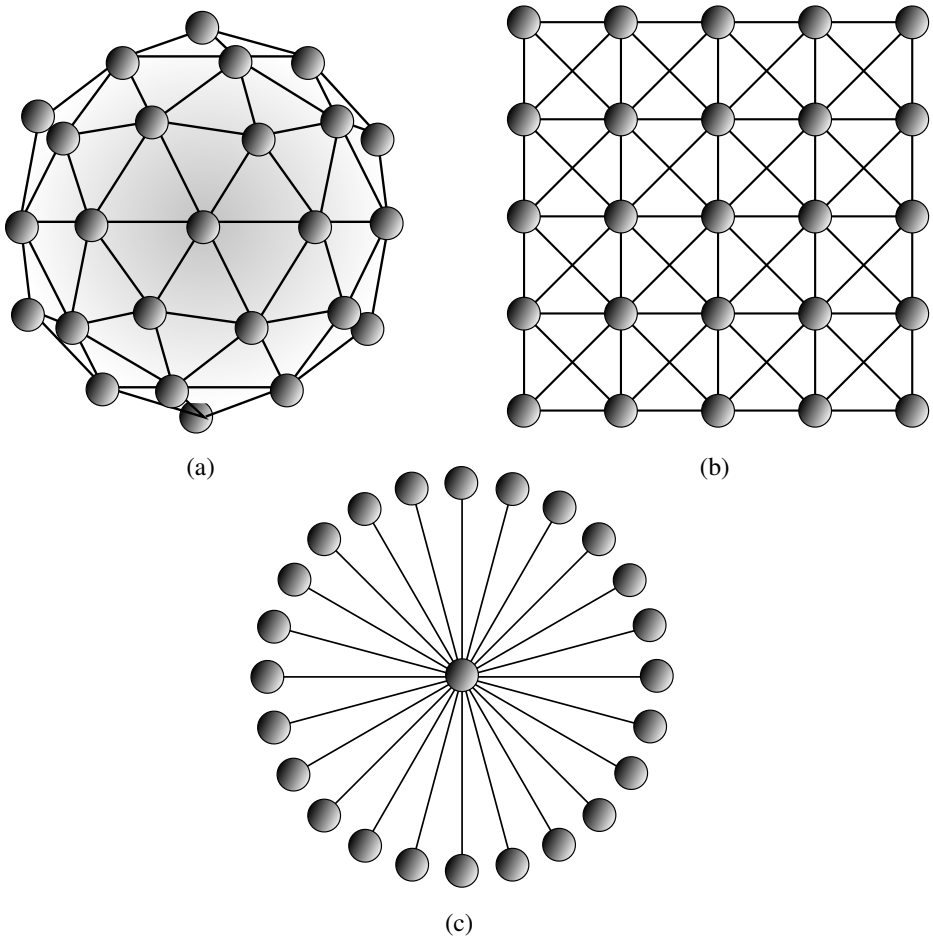


Figure 2.1: Main types of CNN architectures: in (a), the spherical CNN [159]; in (b), the rectangular 8-connected CNN that is very commonly used in simulations and VLSI implementations; and in (c) the star CNN [63].

tor in the circuit of Figure 2.2 is driven by the following equations:

$$\begin{aligned} C\frac{dv_{xij}(t)}{dt} &= -\frac{1}{R_x}v_{xij}(t) \\ &\quad + \sum_{k,l\in N(c)} I_{xy}(i,j;k,l;t) \\ &\quad + \sum_{k,l\in N(c)} I_{xu}(i,j;k,l) + I_z, \end{aligned} \tag{2.1}$$

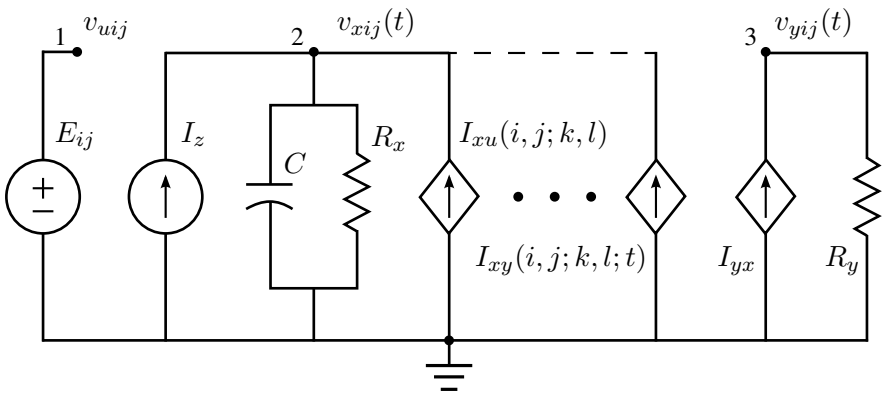


Figure 2.2: Original Chua and Yang's CNN model. The circuit describes an elementary cell of the network indexed by i, j. C is a linear capacitor; R_x and R_y are linear resistors; I_z and E_{ij} are independent sources of current and voltage, respectively; $I_{xu}(i,j;k,l)$ and $I_{xy}(i,j;k,l;t)$ are two linear voltage-controlled current sources with characteristics $B(i,j;k,l)v_{uij}$ and $A(i,j;k,l)v_{yij}(t)$, respectively; The only nonlinear part of the circuit is the piecewise-linear voltage-controlled current source I_{yx} with characteristic function $I_{yx} = \frac{1}{R_y} f(v_{xij})$, where $f(\cdot)$ holds the nonlinearity.

where the $N(c)$ is the set of indices of all cells connected to the cell c, indexed in (2.1) by the indices i, j representing the positions of a rectangular regular grid of $M \times N$ cells. The value $v_{xij}(t)$ is the voltage across the capacitor C and it is called the state of the cell. The current I_z is an independent current source and the currents $I_{xy}(i,j;k,l;t)$ and $I_{xu}(i,j;k,l)$ are linear voltage-controlled sources whose values obey the following characteristics:

$$I_{xy}(i,j;k,l;t) = A(i,j;k,l)v_{ykl}(t), \tag{2.2}$$

$$I_{xu}(i,j;k,l) = B(i,j;k,l)v_{ukl}, \tag{2.3}$$

where v_{ukl} and $v_{ykl}(t)$ represent the voltage values on the nodes 1 and 3 the cell indexed by k, l, and are called the input and the output of this cell, respectively. The terms $A(i,j;k,l)$ and $B(i,j;k,l)$ represent the conductance values or weights of the connections between the cell indexed by i, k and its neighbours, indexed by k, l. A and B refers to the two different types of connections. While A refers to connections with the *output*, B refers to connections with *input* of neighbours cells.

The only nonlinearity present in the circuit of Figure 2.2 is a piecewise-linear voltage-controlled current source I_{yx} with a characteristic function described by the following equation:

$$I_{yx} = \frac{1}{2R_y}\left(|v_{xij}(t)+1| - |v_{xij}(t)-1|\right). \tag{2.4}$$

If (2.2) and (2.3) are replaced in (2.1), we find the following expression, commonly used to describe the state of a Chua and Yang's CNN cell:

$$\begin{aligned} C\frac{dv_{xij}(t)}{dt} &= -\frac{1}{R_x}v_{xij}(t) \\ &\quad + \sum_{k,l\in N(c)} A(i,j;k,l)v_{ykl}(t) \\ &\quad + \sum_{k,l\in N(c)} B(i,j;k,l)v_{ukl} + I_z, \end{aligned} \tag{2.5}$$

where $v_{yij}(t)$ is obtained from (2.4):

$$v_{yij}(t) = \frac{1}{2}\left(|v_{xij}(t)+1| - |v_{xij}(t)-1|\right). \tag{2.6}$$

Chua and Yang stated the following assumptions w.r.t. the cell circuit of Figure 2.2:

1. Initial state voltages are bounded to one, $|v_{xij}(0)| \leqslant 1$,
2. Input voltages are bounded to one, $|v_{uij}| \leqslant 1$,
3. Feedback gain matrices must be symmetric[1], $A(i,j;k,l) = A(k,l;i,j)$,
4. $C > 0$, $R_x > 0$.

Following the common CNN methodologies, we avoid unnecessary clutter hereforth by making the following assumptions and changes in notation without compromising generality:

1. The values of the capacitor C and the resistor R_x of the cell are unitary: $C = 1$, $R = 1$;

[1]This assumption can be imposed in order to ensure stability of the CNN. Nowadays, unstable CNNs also play a significant role and therefore, in a broader context, this assumption can be neglected.

2. The voltages v_{xij}(t), v_{uij}, and $v_{yij}(t)$ are denoted by $x_{i,j}(t)$, $u_{i,j}$, and $y_{i,j}(t)$, respectively; and

3. The current I_{ij} is denoted by z_{ij}.

This way, our notation becomes independent from circuit theory notation. The resulting CNN state equation can then be written as follows

$$\begin{aligned} \frac{dx_{i,j}(t)}{dt} &= -x_{i,j}(t) \\ &\quad + \sum_{k,l \in N(c)} A(i,j;k,l) y_{k,l}(t) \\ &\quad + \sum_{k,l \in N(c)} B(i,j;k,l) u_{k,l} + z_{i,j}, \qquad (2.7) \\ y_{i,j}(t) &= \frac{1}{2}\left(|x_{i,j}(t)+1| - |x_{i,j}(t)-1|\right). \qquad (2.8) \end{aligned}$$

The equations above are considered to be the standard CNN model by many researchers in this field. Many modified versions of these exist. The modifications are mostly related to the nonlinearity in (2.8), and specific rules for the boundary conditions for the cells at the edge of the grid.

Examples of CNN nonlinearities

Although the output nonlinearity proposed originally by Chua and Yang is still widely used, some authors use approximations of the original piecewise-linear output function, *e.g.*. $\tanh(x_{i,j}(t))$. This is especially the case in learning methods which use gradient descent in the learning process because of the need of a differentiable error function. Other authors consider the use of a different type of piecewise-linear function in the state variable in CNN equation in order to eliminate the need for the piecewise-linear function at the output of the cell. This is important to achieve simpler cell circuitry which are more suitable for VLSI CNN implementations. This is detailed further in Section 2.2.2. Figure 2.3 present the characteristics of the main types of CNN nonlinearities.

Boundary conditions

The analysis of the dynamical behaviour of coupled CNN cells most of the time considers an infinite number of cells disposed in an infinite chain, in

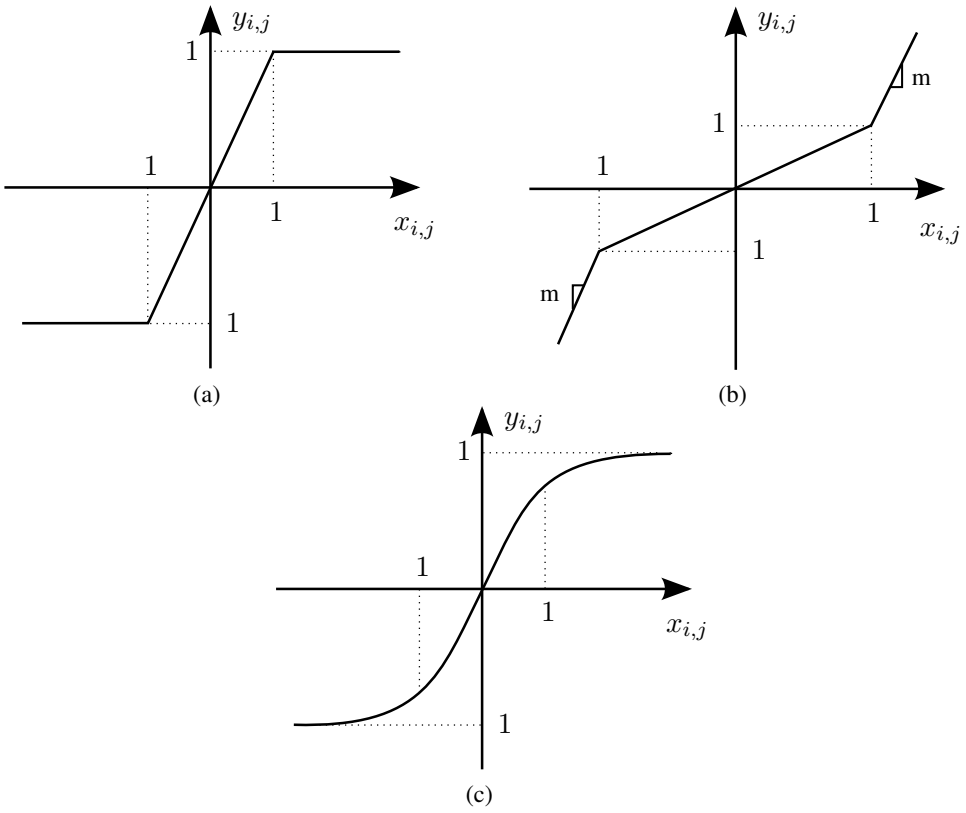


Figure 2.3: Main types of CNN nonlinearities: (a) original output function proposed by Chua and Yang [28], $y_{i,j}(t) = \frac{1}{2}\left(|x_{i,j}(t)+1| - |x_{i,j}(t)-1|\right)$; (b) nonlinearity used in VLSI suitable CNN models, see Section 2.2.2; and (c) tangent hyperbolic function used as approximation of (a) especially in CNN learning methodologies that use gradient descent techniques.

the case of one-dimensional CNNs, an infinite grid, for two-dimensional CNNs, etc. However, when considering a more specific analysis or a real implementation, it is necessary to define the values that the network need to assume around its boundaries. The behaviour of the network depends on the conditions on which these values are defined [140]. There exist three main types of boundary conditions that can be used in order to establish the interactions in at the boundary or the network. We list here these three types of conditions as mentioned elsewhere [23].

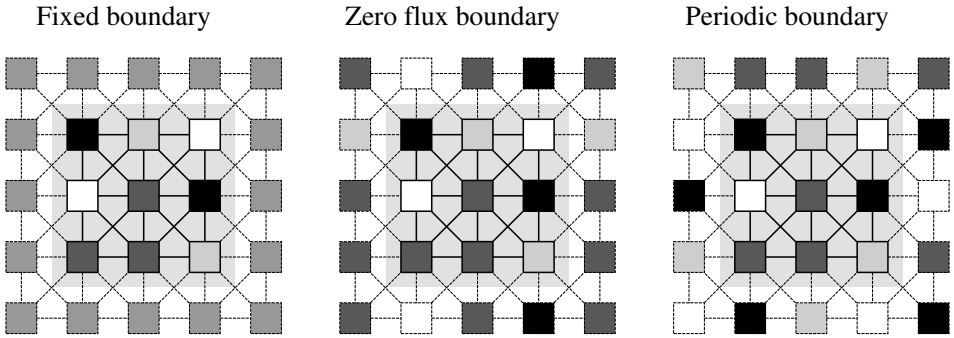


Figure 2.4: Illustration of three types of boundary conditions commonly used in CNNs. The output value of the outer *virtual* boundary cells are specified according to the values of the inner cells.

Fixed boundary condition, also called *Dirichlet boundary condition*, establishes that the state $x_{q,r}$ of outer cells at the boundaries of the network has a fixed value. *i.e.* $x_{q,r} = \text{constant}\ \forall q, r$ indices of outer neighbours of a cell in the boundaries of the network.

Zero flux boundary condition, also called *Neumann boundary condition*, establishes that the state of an outer cell at the boundaries of the network has its value mirrored to its inner neighbour cell that lies in the direction orthogonal to the boundary, *i.e.* $x_{q,r} = x_{i,j} \forall q, r$ indices of outer neighbour cells at the boundary of the network, and i, j indices of the inner neighbour cells in the direction orthogonal to the boundary.

Periodic boundary condition, also referred as *toroidal boundary condition*, establishes that the state of an outer cell at the boundaries of the network has its value mirrored to the inner cell at the opposite boundary of the network in the direction orthogonal to the boundary. That means for instance that grid networks are folded to become a toroid.

Figure 2.4 illustrate these three types of boundary conditions.

2.2.2 Full-signal range CNN model

The local properties of CNN weights allows for easier implementation in VLSI integrated circuits resulting in increased cell density per unity of silicon area. However, integrated circuit implementation of variants from the original CNN model presented a few drawbacks. Because the original model works on voltage-mode and due to the fact that photo-sensory devices work with current outputs, these implementations [31, 51] need conversion of these output to voltages in order to be applied to image processing designs. Additionally, differences in internal voltage and current ranges makes the design of the electrical cell nontrivial because of transistor nonlinearities. Finally, because of the combination between internal voltage and current signals, high-impedance nodes are necessary. Therefore, it results in slow circuits because of the necessarily large time constants.

Rodríguez-Vázquez *et al.* have proposed a CNN model that uses current-mode techniques to avoid the drawbacks of the original model for implementation of continuous- and discrete-time CNNs. At the same time, another current-mode approach was proposed by Varrientos *et al.* [144]. The key difference between these two current-mode CNN models is the addition of another nonlinearity to the state of the cell by Rodríguez-Vázquez *et al.* The result of such measure is a reduction in area and power consumption.

The cell model proposed by [108] is described by the following equations:

$$\begin{aligned}\frac{dx_{i,j}(t)}{dt} &= -g_m(x_{i,j}(t)) \\ &\quad + \sum_{k,l\in N(c)} A(i,j;k,l)y_{k,l}(t) \\ &\quad + \sum_{k,l\in N(c)} B(i,j;k,l)u_{k,l} + z_{i,j}, \end{aligned} \tag{2.9}$$

with

$$\begin{aligned} y_{i,j}(t) &= f(x_{ij}(t)) \\ &= \begin{cases} 1, & \forall\ x_{i,j}(t) \geqslant 1, \\ x_{i,j}(t), & \forall\ |x_{i,j}(t)| < 1, \\ -1, & \forall\ x_{i,j}(t) \leqslant -1. \end{cases} \end{aligned} \tag{2.10}$$

$$g_m(x_{i,j}(t)) = \begin{cases} m(x_{i,j}(t)-1)+1, & \forall\ x_{i,j}(t) > 1, \\ x_{i,j}(t), & \forall\ |x_{i,j}(t)| \leqslant 1, \\ m(x_{i,j}(t)+1)-1, & \forall\ x_{i,j}(t) < -1, \end{cases} \tag{2.11}$$

where $f(\cdot)$ is equivalent to (2.8) and is represented in differently for purpose of comparison with $g_m(\cdot)$ in (2.11). The variable m defines the slope of the linear segments in the extremes of $g_m(\cdot)$ characteristic function, which can be seen in Figure 2.3(b). The original model is called the *Full Signal-Range* (FSR) *cell* as originally proposed [108] and assumes $g_m(x_{i,j}(t))$ in the limit of $m \to \infty$. Later, a generalised model called *Improved Signal-Range* (ISR) *cell* was proposed as a natural transition model between Chua and Yang's cell and the FSR cell [36]. The ISR cell simply drops the mathematical limit $m \to \infty$ in $g_m(x_{i,j}(\cdot))$ and assumes the range $m \geqslant 1$. If $m = 1$, the ISR cell is reduced to the original CNN model in (2.7) and (2.8). The ISR cell holds stability properties that are very similar to the original model, and so does the FSR cell, consequently. Moreover, a specific property of the FSR cell made this cell a standard for following successful implementations of large VLSI CNN chips [85, 84, 109]. The property we refer to is due to the elimination of the output piecewise-linear function at the output of the cell. This is possible thanks to the following property, proved in [36]:

Given a regular array of $M \times N$ FSR cells, *i.e.* cells described by (2.9) and (2.10) with $g_m(x_{i,j})$ defined by

$$g_m(x_{i,j}(t)) = \lim_{m\to\infty} \begin{cases} m(x_{i,j}(t)-1)+1, & \forall\ x_{i,j}(t) > 1, \\ x_{i,j}(t), & \forall\ |x_{i,j}(t)| \leqslant 1, \\ m(x_{i,j}(t)+1)-1, & \forall\ x_{i,j}(t) < -1. \end{cases} \tag{2.12}$$

If the initial state vector lies within the hypercube $[-1,1]^{M\times N}$, *i.e.* $-1 \leqslant x_{i,j}(0) \leqslant 1,\ \forall i = 1,\cdots,M; j = 1,\cdots,N$, then the evolution of the state vector is also bounded into the same hypercube, *i.e.* $-1 \leqslant x_{i,j}(t) \leqslant 1,\ \forall t > 0; i = 1,\cdots,M; j = 1,\cdots,N$, independently of the particular CNN coefficients A, B, and z.

Due to this property, the state vector is equivalent to the output vector, $x \equiv y$, and therefore, (2.9) and (2.10) are reduced to

$$\begin{aligned} \frac{dx_{i,j}(t)}{dt} &= -g_m(x_{i,j}(t)) \\ &\quad + \sum_{k,l\in N(c)} A(i,j;k,l)x_{k,l}(t) \\ &\quad + \sum_{k,l\in N(c)} B(i,j;k,l)u_{k,l} + z_{i,j}, \end{aligned} \tag{2.13}$$

and so the nonlinearity is eliminated from the output which is now equivalent to the state. An additional advantage of such a model is that the output, input and state ranges are normalised to a single range, which reduces the complexity of circuit implementation.

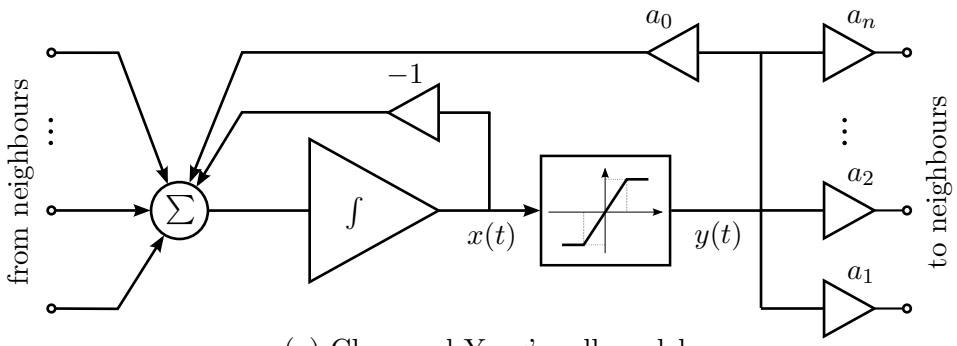
In practise, the block diagram of the original cell circuitry depicted in Figure 2.5(a) is reduced to the circuit illustrated in Figure 2.5(b).

2.2.3 Space-invariant CNNs

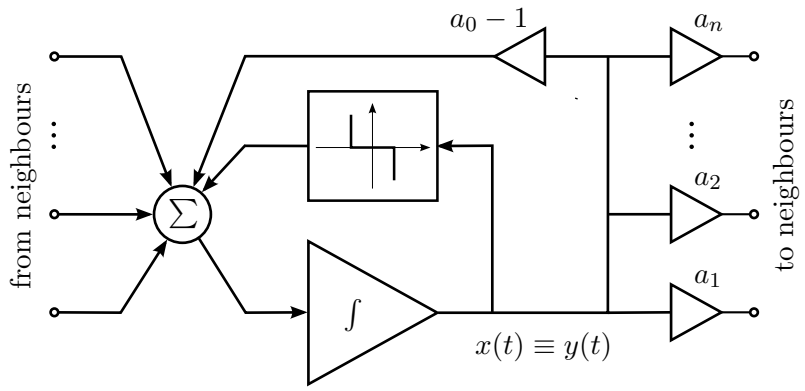
So far, we have only discussed advantages of CNNs for integrated circuit implementation that are related to the locality of the connections, which, indeed, makes wiring much simpler in VLSI design. Nevertheless, this is not the only major feature in favour of CNNs. While the interconnection matrices $A, B \in \mathbb{R}^{(MN)\times(MN)}$ in CNNs are always local, they also may be space-invariant, and so does the bias term $\mathbf{z} \in \mathbb{R}^{(M\times N)}$. This means that A and B can be represented by single local connectivity matrices, and $\mathbf{z}$ can be represented by a scalar, i.e. in this case $A, B \in \mathbb{R}^{n+1}$ and $\mathbf{z} \equiv z \in \mathbb{R}$, with n denoting the number of neighbours of a cell. This way, CNNs can be used to perform different types of Single-Instruction Multiple-Data (SIMD) parallel operations, which are coded into the values of the local versions A, B, and $\mathbf{z}$. In this form, the coefficients of a CNN are called CNN genes, or CNN cloning templates, or simply CNN templates. A template has a fixed structure with a fixed number of values. This small set of values defines the evolution of the dynamics of the whole network.

A space-invariant CNN implies a homogeneous or uniform CNN, which means that the physical connections must obey a regular pattern across the network. Each cell, with exception of the boundary ones, must have the same number of connections and these must hold the same topological structure. These assumptions define a large number of CNN topologies that qualifies to be space-invariant. For illustration, in Figure 2.6, we present three of these topologies. The number of neighbours of a uniform CNN depends on the size of the radius $r \in \mathbb{N}$ of the *sphere of influence* $S(r)$. In all cases of Figure 2.6, the value of the radius is $r = 1$, and therefore, the number of neighbours of a cell is equivalent to the number of actual adjacent neighbours. In the case of $r = 2$, for example, n would equate the sum of the number of adjacent neighbours and their neighbours, mutually-exclusive.

In the case of regular rectangular CNNs with $r = 1$ and $n = 8$, a



(a) Chua and Yang's cell model



(b) FSR cell model

Figure 2.5: Block diagrams from the original Chua and Yang's cell circuit (a), and from the full signal-range model (b). Input and bias connections are omitted to avoid clutter. The gains a_0 and $a_h, \forall h = 1, \cdots, n$ are associated with the corresponding elements of the connection matrix A representing self-feedback and feedback to neighbours, respectively, with n being the number of connected neighbours. The output nonlinearity disappears from the FSR cell. Instead, the state is used as output and is fed back to the incoming sum after passed through a saturation block. This block makes sure that the states of the network remains with in the unity hypercube and keeps the internal signals within the same range.

template is composed by the following pieces:

$$A = \begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix}, \quad B = \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix}, \quad z, \tag{2.14}$$

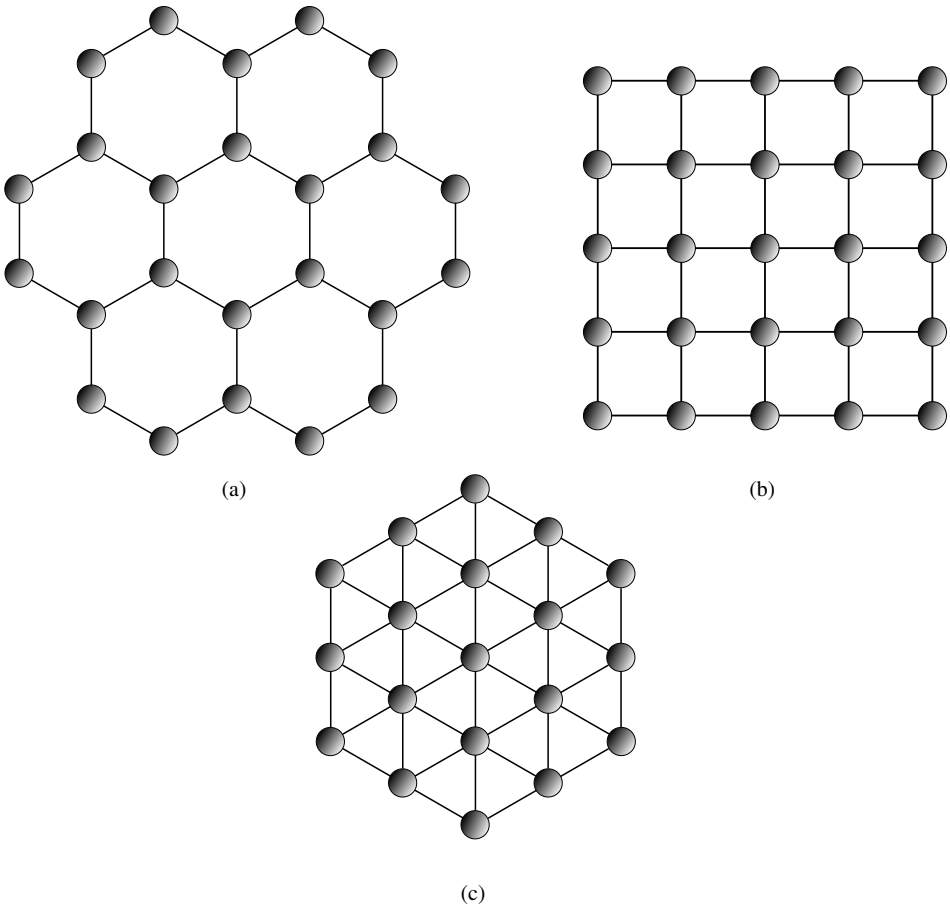


Figure 2.6: Three types of homogeneous CNNs. A triangular CNN where the number of neighbours of each cell $n = 3$ (a); A rectangular CNN, with $n = 4$ (b); and a hexagonal CNN [88], with $n = 6$ (c).

where $a_{k,l}, b_{k,l}, \forall k, l \in \mathbb{Z} \times \mathbb{Z}$ are weights of neighbour connections with $a_{0,0}$ and $b_{0,0}$ being auto-weights, w.r.t. to the *feedback* and *control* matrices, A and B, respectively. The condensed form $|k, l| \leqslant r$ refers to both $|k| \leqslant r$ and $|l| \leqslant r$ expressions, simultaneously. The topology equivalent to this template structure is illustrated in Figure 2.1(b).

Finally, the equation that defines space-invariant CNN with the type of

templates in (2.14) can be derived from (2.7) and written as follows

$$\begin{aligned}\frac{dx_{i,j}(t)}{dt} &= -x_{i,j}(t) \\ &\quad + \sum_{|k,l|\leqslant r} a_{k,l} y_{i-k,j-l}(t) \\ &\quad + \sum_{|k,l|\leqslant r} b_{k,l} u_{i-k,j-l} + z, \qquad (2.15)\\ y_{i,j}(t) &= \frac{1}{2}\left(|x_{i,j}(t)+1| - |x_{i,j}(t)-1|\right), \qquad (2.16)\end{aligned}$$

where (2.16) is repeated here for the reader's convenience.

Hereafter in this thesis, when referring to a cellular neural network, we always mean a space-invariant CNN, unless explicitly expressed otherwise.

2.2.4 Cellular neural network universal machine

The CNN paradigm with space-invariant templates defined an ideal framework for analog SIMD or array computing. Dozens of these templates where designed along the years following the CNN invention and compose today a large multipurpose CNN template library. Many dedicated analog chips were developed to perform many single template operations, mainly image processing tasks, in order to profit from the high parallelism of CNNs. As a natural conceptual continuation of these chips, the *CNN Universal Machine and supercomputer* (CNN-UM) was created with the objective to provide a programmable CNN environment to combine multiple template execution in a single chip instead of building different ones of each template.

The CNN universal machine and supercomputer was described as "the first algorithmically programmable analog computer having real-time and supercomputer power on a single chip" [113, 26, 117]. Its design incorporates global and distributed analog memory and logic. Simulation of partial differential equations were also included in its design with use of appropriate complex cells. With applications on many areas, which include neuromorphic computing and the so called *programmable physics, chemistry, and bionics*, the CNN-UM was presented as a unique choice for building a large-scale programmable analog array computer.

One of the key ideas behind the design of the CNN-UM was called the *dual* or *analogic computing*, which comes from a contraction between

the words *analog* and *logic*. This idea combines analog array processing with logic operations by incorporating distributed analog memory and programmability. Many exiting algorithms and physical phenomena could then be translated into CNN analogic algorithms and implemented or simulated using a programmable kernel function with a finite spatial window in appropriate analogic circuits.

The special purpose CNN analog and mixed-signal mode chips that were implemented before the CNN-UM was designed for use in well defined applications presented superior efficiency for those tasks when compared to their digital counterparts. However, digital systems were still more widely used. The advent of integrated circuit implementation of the CNN-UM has the advantage of operating with store-program capability as opposite to those special purpose CNN chips. CNN-UM chips were expected to offer a viable complement or even an alternative to conventional digital computing. Indeed, the digital computing paradigm is persistently used in areas where its mainly sequential nature is not adequate.

Encouraged in the design of the CNN-UM, many scientists have developed and implemented several analog VLSI CNN systems. From all these chips, perhaps the *ACE* family [86, 109, 19, 20, 83, 82, 81, 84, 85, 35] was the most universal. These chips allowed unprecedented analogic operations of large grids of cells, e.g. $4,000$ and $16,000$ cells, with control and feedback template programmability. Next Section is dedicated to describe the evolution of these chips and present the state-of-the-art of their successors.

2.3 Brief history and state-of-the-art of CNN-UM implementations

The first CNN-UM chips [66, 65, 69, 34, 37] ware implemented after encouraging implementations of various successful special purpose CNN systems with increasing complexity [31, 55, 107, 51]. Shortly after, following the same path, many researchers have developed other chips with stored-program capability [32, 67]. Additionally, many actions were taken to give support to the development of those chips, such as the creation of a suite of design tools which includes a high level language, called *Alpha*, to describe analogic algorithms, compiler, operating system, chip prototyping system, and designer's toolkit [118, 120, 135]. These tools, which later became commercial development kits [164, 7, 163], supported mainly the series of

chips called the *ACE family*, which got a boost development during the European project called DICTAM [2].

2.3.1 The ACE chip family

The first of the ACE chips [35], called ACE400, was composed of 20×22 CMOS cells, arranged in a rectangular regular grid. The external control interface of this chip was fully digital whereas the internal programming signals of these chips were analogue. It was manufactured in a $0.8\mu m$ single-poly double-metal technology with around 7 bits accuracy in analog processing operations. All 19 template values of a 3×3 neighbourhood were fully programmable. Although this chip was not remarkable by its cell/area ratio when compared with an existing chip at the time [67], it presented reasonably more functionalities.

The follower of the ACE400, the ACE4k, featured over $4,000$ cells disposed in a regular 64×64 grid with distributed optical acquisition, and distributed image memory cache on the same chip on the same silicon substrate [85]. Implemented in a $0.5\mu m$ standard CMOS technology, it contains approximately 1 million transistors operating in analog mode. This chip is reported to be capable to perform complex spatiotemporal image processing tasks with time execution as short as 300ns using as little as 0.3mW per cell, or 1.2W of power for the whole chip.

The third generation of ACE chips is the ACE16k [109, 84, 83, 82, 81], a vision chip that like his predecessors is flexible, ultra-fast, and power and area efficient. This chip was built in a standard $0.35\mu m$ CMOS technology and features a moderate accuracy of 8 bits and was particularly designed to overcome some limitations of its predecessors. The improvements includes, a better internal organisation of the processing cells, better management of the analog memories, a reconfigurable optical interface, incorporation of an address event detection scheme to simplify information of black and white images, and an improved power consumption management, which allows even more energy efficiency.

Belonging to the same ACE family but targeting a different gamut of applications, the CACE1k chip [20, 19] is a second-order bio-inspired processor for focal-plane dynamic image processing. Conceptually, it is composed of grid of cells organised in two layers of 32×32 cells, where each layer is very similar to the other single-layer ACE chips but between the layer there exists a coupling between immediately superposed cells. Built

in $0.5\mu m$ CMOS technology, the chip was designed inspired in studies of the working of the mammalian retina in order to mimic the way in which images are processed at the front-end of natural pathways. This is possible because the chip allows programming of complex spatiotemporal dynamics.

During the course of the work developed for this thesis, we have performed some experiments with the ACE4k and two versions of the ACE16k. The results of some of these experiments can be seen in the Chapters 3, 4, and 7.

2.3.2 The Q-Eye processor

Today, the state-of-the-art of CNN-based visual analog processors is the Q-Eye chip. Built on $0.18\mu m$ CMOS technology in a QCIF[2] format, the Q-Eye is composed of a 176×144 grid of cells and is considered by its manufacturer [1] as a "ACE-like" processor, who claims a frame rate above $10,000$ fps. So far there has been no existing academic publication on this chip although it is considered by the CNN academic community as the new generation of Vision System on a Chip (VSoC).

The application areas of the Q-Eye include: intelligent security cameras; consumer electronics and consumer robotics; in the automotive industry: smart airbag deployment, blind spot detection, navigation, collision warning, etc.; in machine vision: ultra-high speed monitoring in production lines, *e.g.* in textile, micro-electronic, and pharmaceutical industries; and other areas overlapping CNN application areas.

[2]QCIF stands for "Quarter CIF". CIF or *Common Intermediate Format* was designed to allow easy conversion between different image standards.

Chapter 3

Optimising CNNs

As seen in the previous Chapter, Cellular Neural Networks (CNN) are progressively becoming a more attractive alternative to conventional digital computation. The computational power available in a single modern CNN Universal Machine (CNN-UM) chip is comparable with that of supercomputer systems with many digital processors in parallel. CNN-UMs can only perform Single-Instruction Multiple-Data (SIMD) operations. However, this architecture is sufficiently useful to suit many applications. CNN image processing for instance is a research field studied worldwide. There are many factors that contribute to such extraordinary computing power. They include the local nature of the connections in the CNNs and, more practically, advances in analogue and mixed-signal circuit technology, which allow large arrays of CNN cells to be placed in a single Very Large-Scale Integration (VLSI) chip. In fact, it is also the analogue circuitry that boosts the speed and processing power of VLSI CNN systems. However, what makes these systems attractive is also what makes its weakest feature. Analogue VLSI technology today is still much susceptible to parameter deviations during the manufacturing process. The result of such imperfections in the CNN operation often cannot be neglected.

The objective of this Chapter is to explain the different causes of the errors observed in CNN-UM implementations as well as to describe the existing methods that deal with these problems either by design or by optimisation of CNN templates and other run-time parameters. Although these methods are still in the development phase, the early results that we present here are encouraging enough to motivate further development.

The maturity of these techniques and the existing rapid improvements in mixed-signal and analogue VLSI methodology are essential and symbiotic steps to give CNN technology a place in application areas today dominated by digital computers and even beyond.

3.1 Erroneous behaviour

Erroneous behaviour observed in VLSI implementations of CNN-UM may be caused by a combination of reasons.

3.1.1 Causes of errors

Manufacturing process variations and environmental effects such as temperature variation and electrical noise is perhaps the most important cause of erroneous behaviour in VLSI CNN-UM chips. Although adaptive techniques have been employed in CNN-UM chip implementations to ensure accurate external control and system robustness against parameter variations [37], analogue VLSI implementations can only guarantee a rough accuracy, about 5-10%, in relation to ideal parameter values. Moreover, template parameters have a discrete range of implementable values, which is about 8 bits for modern chips [109]. In addition to errors caused by these design constraints, manufacturing process variations and environmental effects, some templates developed for use in ideal CNN structures can even produce different erroneous results for runs with the same input and initial conditions for a given chip. For instance, the results of consecutive averaging threshold operations shown in Figure 3.1 indicate that the source of many errors may be related to other types of post-manufacturing interference. The reason for this inconsistent behaviour may also be imperfect or noisy loading of the input and initial state from off-chip to on-chip memory prior to an operation. This may also contribute to the overall undesirable chip behaviour. According to these assumptions, the main reasons for erroneous behaviour observed in CNN-UM chips can be summarised as follows:

- Parameter variation introduced during the fabrication process;
- Noise in the electrical components of the cells;

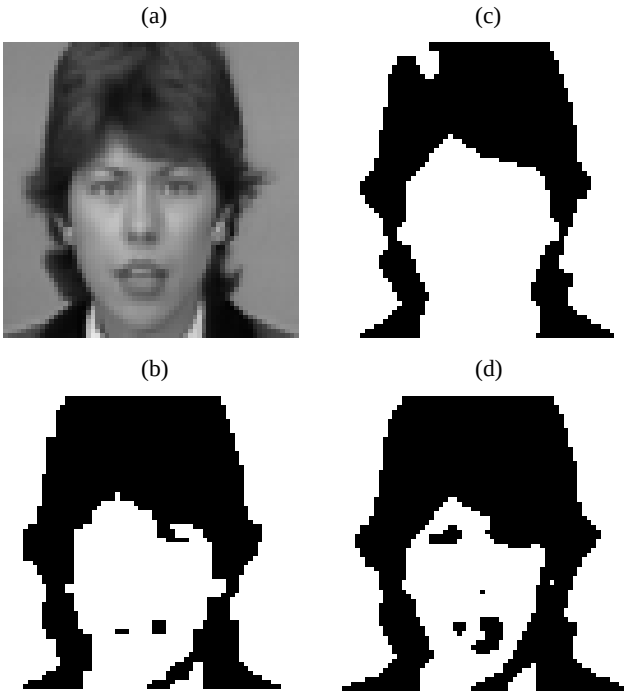


Figure 3.1: For the same input image (a), initial state, and template values, the same chip (an ACE4k) may produce different outputs (b,c, and d). These results were obtained for subsequent runs, what make temperature and noise interferences mostly stable. Therefore the different outputs are assumed to be due to imperfect loading from off-chip to on-chip memories.

- Imperfect or noisy loading of the input and initial state from off-chip to on-chip memory;

- Temperature variation.

These so far unavoidable and undesirable features make CNN-UM chips fairly unreliable for some simple template operations, such as edge detection, binary average, threshold, halftoning, and several others.

3.1.2 The effect of parameter deviations

Tetzlaff *et al.* have analysed and demonstrated by simulation the effect that parameter deviation can have on VLSI implementations of CNNs [137]. They used a universal simulation system called SCNN [75] to reproduce a CNN chip with template values affected by zero mean and randomly distributed stationary noise. The result of this noise addition is a space-variant CNN, whose template values are slightly different across its network according to the noise variance. The values of each connection in the network was therefore modified in the following manner:

$$\begin{aligned} A_{k,l} &\Rightarrow A_{i,j} = A_{k,l}(1 + \Delta A_{i,j}), \\ B_{k,l} &\Rightarrow B_{i,j} = B_{k,l}(1 + \Delta B_{i,j}), \\ z &\Rightarrow z_{i,j} = z + \Delta z_{i,j}, \end{aligned} \tag{3.1}$$

where $A_{k,l}$, $B_{k,l}$, and z are the original space-invariant templates and $A_{i,j}$, $B_{i,j}$, and $z_{i,j}$ are the resulting space-variant weights of the network, which incorporated the desired simulation of the parameter deviations.

Tetzlaff *et al.* applied this modelling to different templates from the CNN library [115]. The results of their simulations clearly show that even for very small variations the output can be very different from the original one. They also proposed a learning method to minimise the effect of the parameter deviations. This method uses the mean square error as the error function for the recurrent back-propagation algorithm to train ordinary cloning templates [137] and cloning templates with space-variant bias [139] to generate templates which are more robust to the deviations.

3.2 Chip-independent methods

Methods that consider the design of robust templates have been extensively developed in order to avoid or minimise the effects of the above mentioned obstacles for correct operation of analog VLSI implementations of CNN-UMs [21, 162, 53, 137, 138, 139, 124, 96, 52, 68, 90]. There are several definitions of robustness [96, 53] that in general define a measure for susceptibility of CNN templates to modifications in their values while still producing the correct output. Robust templates are generally expected to have their values in the middle of a correct operation interval so that tiny variations on these values remain within this interval. The size of this

interval defines how robust a template operation can be. The level of robustness is therefore dependent on the type of operation. The most robust operations often overlook parameter deviations and work correctly in chip implementations. These methods for design and learning of robust template operations consider ideal CNN models to assess these intervals without regard to specific implementations. For further clarity of this thesis, we state the following definition.

Chip-independent methods are design or learning approaches for generating robust CNN template operations that consider ideal CNNs models, rather than parameters measured on a specific chip, in order to define robustness.

3.3 Chip-specific methods

Although chip-independent methods are able to provide correct chip functionality for many robust templates, for less robust operations, these methods can not avoid erroneous behaviour. Methods that consider not only the working of ideal CNNs but also the specific differences of each chip implementation can further improve the functioning of these less robust templates. We present here a definition for such methods.

Chip-specific methods or chip-dependent methods are learning or tuning methodologies to generate CNN templates targeted to a specific chip. The correctness of the operation is defined experimentally with measurements of the given chip.

Földesy *et al.* [38] have developed a chip-specific approach to design fault-tolerant CNN templates. The approach is based on the well known least mean squares optimisation method.The general objective is to find templates and template sequences equivalent to existing ones that run correctly and reliably on specific CNN-UM chips. This method is restricted to uncoupled templates and uses decomposition of templates to ensure the correct desired operation when a single template fails to be optimised.

The decomposition happens in the following way. Upon a template optimisation failure,

- decompose the template into two child templates according to specific rules,

- optimise these child templates for the specific chip,
- If the child templates are successfully optimised, the procedure is finished, otherwise, repeat the decomposition process with child template that failed to optimise.

Although this approach may often lead to the correct functionality, the number of template decompositions may grow unnecessarily because the gradient descent optimisation cannot guarantee that the global optima can be found. The correct functionality thus comes at the cost of multiple template execution. The restriction to uncoupled templates only is another disadvantage of this method. Coupled templates are necessary to describe most of the complex and interesting behaviour observed to exist in CNNs, e.g. autowaves.

3.4 Template tuning

In this Section, we introduce a chip-specific optimisation approach to tune coupled and uncoupled CNN templates using global optimisation. This approach uses chip measurements to evaluate the template operation with *a priori* designed training sets of input-output images. It assumes as starting point values, a correctly working template designed to ideal CNNs, i.e. a template that works in simulation. The goal of this approach is to find a modified version of the initial template that allows for a correct on-chip template operation.

This approach has some advantages when compared with the method described in the previous Section. Földesy et al. use gradient based optimisation. This can lead the solution to a poor local minimum and force unnecessary template decomposition as suggested in their method. Besides, one of the most interesting features of CNNs, global interaction, is neglected because propagating, or coupled, templates cannot be optimised due to the lack of proper error derivative when including the feedback connections. On the other hand, global optimisation eliminates these restrictions but convergence is not as fast as in a local optimisation method. However, the choice for global optimisation methods has another advantage. Global optimisation can handle non-differentiable problems with many local minima, which may reflect well the characteristics of cost functions for CNN optimisation.

To test this approach, we have chosen to use the Adaptive Simulated Annealing (ASA) [60, 62] method to globally optimise our cost functions. For detailed information, see also Appendix A. ASA is known as a robust and fast method to search for a global optimum in non-linear complex problems with multiple local optima like CNN template optimisation.

To ensure correct optimisation, it is necessary to choose the training set wisely., Training sets are composed by a set of triplets θ containing the input $\mathbf{u}$, the initial state $\mathbf{x}$, and the desired output $\mathbf{y}^d$. Each individual element of a triplet entry has, in this approach, its values ranging from 0 to 1. In [38] the importance of this step is discussed and a good method to compose training sets for uncoupled operations is proposed. For more general and coupled templates, sample images and random images suitable for the given operations have to be considered. In Chapter 6, we present some roles for designing good training sets.

In order to evaluate each probing template, we use a normalised version of the cost function used in [72] for learning purposes. The normalised cost function is described in (3.2),

$$g(\mathbf{p}, \theta) = \frac{1}{\sqrt{k}} \sqrt{\sum_{i=1}^{k} (y_i^d - y_i(\infty))^2}, \tag{3.2}$$

where $\mathbf{p}$ denotes the parameter vector, i.e. the probing template, θ is the current training triplet, k is the number of cells, y_i^d is the value of the ith pixel of the desired output and $y_i(\infty)$ is the corresponding value of the steady-state output, whose values are acquired from direct chip measurements. Hence, the cost function $g(\mathbf{p}, \theta)$ of the probe template $\mathbf{p}$ for the input and initial state contained in the triplet θ gives the RMS value of the distance between the desired output vector $\mathbf{y}^d$ and the steady-state output $\mathbf{y}(\infty)$. The objective of the ASA algorithm is, therefore, to minimise $g(\mathbf{p}, \theta)$ given an initial template $\mathbf{p}_{init}$.

Imposing an initial approximation $\mathbf{p}_{init}$ seems to be less important when using a global optimisation method. However for this approach, this approximation is used to set the boundaries of the search since the objective here is tuning and not learning, where the whole parameter range would be used instead. Namely the boundaries for the search are $p_{min,i} = p_{init,i} - b$ for the lower bound and $p_{max,i} = p_{init,i} + b$ for the upper bound, where i is the index of each template parameter and b is a small value. Observe

that here two assumptions are made: the initial template is assumed to be a fully correctly working template on a simulator; and the parameter deviations that disturb its values on the chip are assumed to be smaller than b. Narrow search boundaries decrease the duration of the optimisation and allow the use of more minute search, resulting in a much more efficient optimisation. These boundaries are not rigorously strict as a mechanism of self-adjustment may be easily introduced with no significant loss for the algorithm, e.g. if any of the boundaries are close to and/or is constantly bounding its respective component it may be slightly extended.

Another advantage of using finite boundaries, besides modelling physical limitations, is the role played by recursive optimisation runs with relaxation of constraints. Applying constraints (such as symmetry, imposing zero values, or dependence between values) to the template under optimisation shrinks the search space and allows a faster search. The result of this search is then further reused in another search with less constraints, or more parameters, and narrower boundaries, or smaller parameter search space. This recursion is applied until no more constraints are left. The first searches, with more constrained templates and broader boundaries, serve to roughly localise the global optimum in the search space for further refinements using less constrained templates. An outline of this approach using ASA is demonstrated in Figure 3.2.

Once the training set and the search boundaries are defined, the optimisation can be performed. The procedure is finished and considered to be successful when the cost function becomes smaller than a certain end condition value or when the annealing temperatures decrease through a given limit of influence on the algorithm. If the cost for the best template is zero or smaller than the tolerance value then the tuning is finished. Otherwise, if the solution is restricted by any constraint, such as symmetry or imposed zero values, they are relaxed to the next level and another ASA optimisation is initiated. The result of the last optimisation is considered to be an optimal template.

The solution obtained by this method might not be unique, i.e. the optimum can be located inside a region of multiple optima. The next Section describes a solution proposal to search among a set of optimal templates in order to find the *best* template in terms of *robustness* for whenever there exists the possibility of multiple optima.

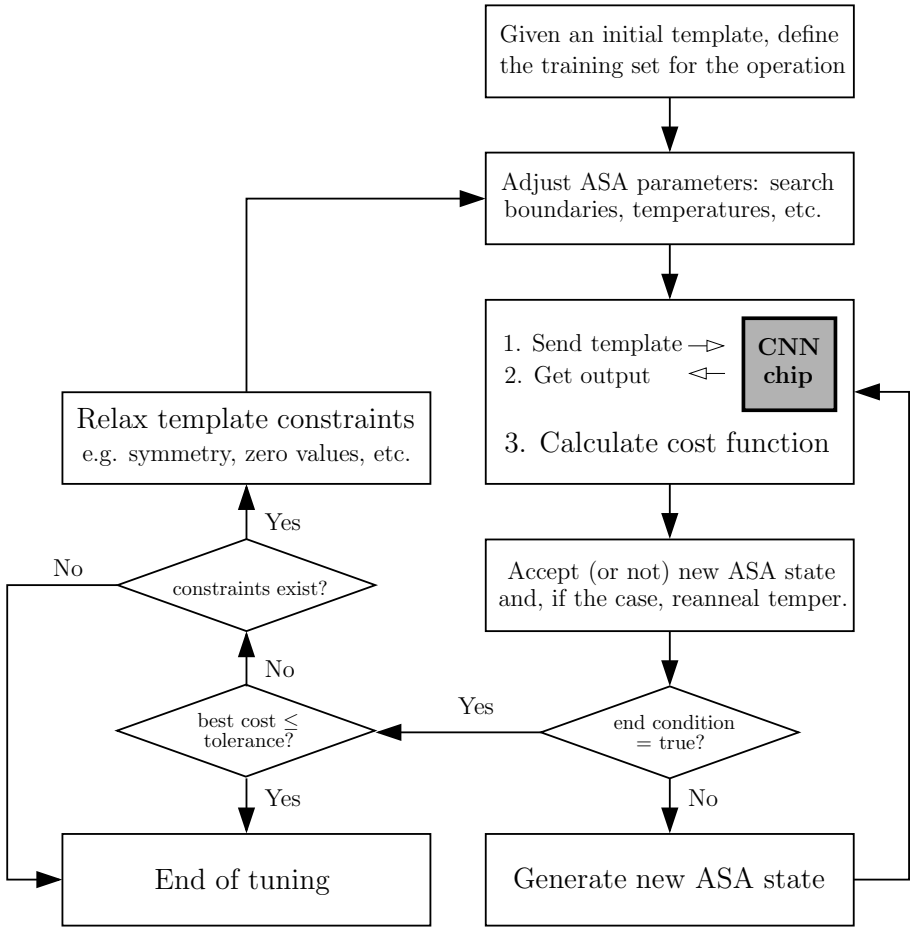


Figure 3.2: Outline of the tuning with Adaptive Simulated Annealing.

3.5 Chip-specific robustness

In contrast with design and learning, template optimisation, or template tuning, does not intend to create new CNN templates but to improve existent ones. The aim of the optimisation involves normally either robustness improvements [52], like in chip-independent methods, or error minimisation for a specific CNN chip implementation [38], like in chip-specific methods. In this approach both goals are pursued in combination. In order to in-

crease the error minimisation, robustness improvements are employed to an optimal template within the scope of a given chip. This procedure produces templates that are expected to be optimal, in the sense of error minimisation, and robust, with relation to variations on the optimal template values due to post-manufacturing interference. The concepts of robustness and optimality here are restricted to the specific chip, neither the term optimal nor robust can be employed with the tuned template for use in a different chip without proper repetition of the whole procedure.

As discussed in Section 3.1, robust template generation methods place the template values in the middle of a correct operation interval but these templates still present errors in VLSI implementations. A shift of this operation interval in VLSI implementations due to parameter deviations can be the cause of the persisting errors. If it is possible to compensate errors using chip-specific methods, it is reasonable to say that the correct operation interval can indeed be shifted in VLSI implementations. If the same interval that existed in ideal CNNs also continued to exist in their VLSI implementations, even that shifted, it is surely worthwhile to develop methodologies toward chip-specific robustness.

Chip-Specific Robustness is the robustness associated to the degree of susceptibility of a given CNN template, that had been tuned for a specific chip, to variations in its values caused by any post-manufacturing interference in the chip.

These methodologies can not only correct errors but also decrease the degree of sensitivity of template operations to variations in their values due to post-manufacturing disturbances. Chip-independent and chip-specific robustness can be directly associated with manufacturing and post-manufacturing error sensitivities respectively. Additionally, chip-specific robust templates can also attempt to correct manufacturing errors alike chip-independent ones.

Considering the class of binary CNN operations, namely operations with binary input and output, an optimal template is not necessarily unique. The error surface in the region of the optimum is often flat or very shallow instead of a deep isolated point (See Figure 3.5 and [38]). Therefore, from this observation one can conclude that these operations are asymptotically stable regarding template values as initial conditions, i.e. as for a stable operation with a given input and initial state, for small variations on the

initial conditions (template values), the output remains in the same fixed point. Due to the discrete nature of the results and to the continuous nature of the template values, asymptotic stability seems obvious for binary template operations. Nevertheless, it is less evident when considering real-valued (grey-scale) inputs operations.

Considering statistical circuit design [129], where techniques such as *design centring* attempt to find a centre for an acceptability region, an analogous formulation for the problem of finding chip-specific robust templates can be established, where a centre for an interval of correct operation needs to be estimated.

In statistical circuit design, *designable parameters* are used during circuit design as decision variables and will represent here the 19-dimensional vector $\mathbf{p} = (p_1, ..., p_D)$ of template parameters. *Random variables*, or *noise parameters*, will represent the parameter deviations, such as manufacturing parameter variation, temperature, and etc., denoted by the vector $\mathbf{e} = (e_1, ..., e_D)$ of random variables with zero mean. *Circuit variables*, which in statistical circuit design represent the variables used in circuit, process, or system simulation, will represent in this approach a noisy template parameter vector[1] denoted by $\mathbf{p}' = \mathbf{p} + \mathbf{e}$. And finally, the vector of *circuit performances* will be represented by a simple scalar denoting the value of a cost function $G(\mathbf{p}, \mathbf{e})$.

The acceptability region, which in statistical circuit design is defined as a region for which all inequality and equality constraints imposed on the vector of circuit performances are fulfilled, will be defined here in the $\mathbf{p}'$-space as such a set of $\mathbf{p}'$ vectors in the 19-dimensional space for which the inequality $G(\mathbf{p}') \leq s$ is fulfilled, where s denotes a tolerance imposed on the cost function.

The goal of this approach is therefore the same as in *design centring*, where the centre of the acceptability function is to be found. There are several methods in the literature that solve this problem with use of the derivatives of the circuit performances or their estimates. However, due to the difficulty to find a good estimate for the derivative of a cost function that uses chip measurements, we use again a global optimisation method to find the optimum for a noise corrupted cost function.

[1] $\mathbf{p}' = \mathbf{p} + \mathbf{e}$ models **absolute** parameter spreads and results in $\text{var}\{p_i'\} = \text{var}\{e_i\}$. Alternatively, one can assume $\mathbf{p}' = \mathbf{p}(\mathbf{1} + \mathbf{e})$, which models **relative** parameter spreads and results in $\text{var}\{p_i'\} = p_i^2 \text{var}\{e_i\}$. This last approach is equivalent to (3.1).

The addition of the noise in the cost function eliminates the flat regions of optima in the error surface and allows further improvements to an initial optimum. The cost function now contains several different embedded measurements instead of only one and the probing templates assimilate a small perturbation. The cost function is

$$G(\mathbf{p}, \mathbf{e}, \theta) = \frac{1}{r} \sum_{j=1}^{r} g(\mathbf{p} + \mathbf{e}_j, \theta), \tag{3.3}$$

where r denotes the number of runs executed for the triplet θ, and $\mathbf{e}$ is a vector where each element corresponds to Gaussian noise with zero mean and small variance.

The addition of different samples of $\mathbf{e}$ to the probe template $\mathbf{p}$ in (3.3) generates a *smoothed* cost function that will statistically make this function minimal when $\mathbf{p}$ has its elements in the middle of each corresponding dimensional range of optima, i.e. in the middle of the acceptability region. Figure 3.3 depicts the effect of this cost function in the final result for one component of the parameter vector.

The region of the error surface where the initial optimal template was located is no longer flat. With the addition of the perturbation $\mathbf{e}$ to the template values, it became noisy. The set of template values located in the middle of this noisy region has now stochastically more chances to generate the correct output than those closer to the borders. As a result, the final template will be very close to the most robust template for specific use in a given chip. Chip-specific robustness is thus the concept of robustness within a given chip.

3.6 Experiments

The experiments were performed using the Aladdin system [7] in connection with the Matlab environment. The main features of the ASA algorithm were written for Matlab and were triggered by a Analogic Macro Code (AMC) program running on the given CNN-UM chip. Two CNN-UM chips were used in the experiments: an ACE4k, a programmable CNN with 4096 cells disposed in a 64×64 regular grid; and an ACE16k with 16384 cells disposed in a 128×128 grid. All measurements were made *on-the-fly*. Figure 3.4 depicts an overview of the setup used during the optimisations.

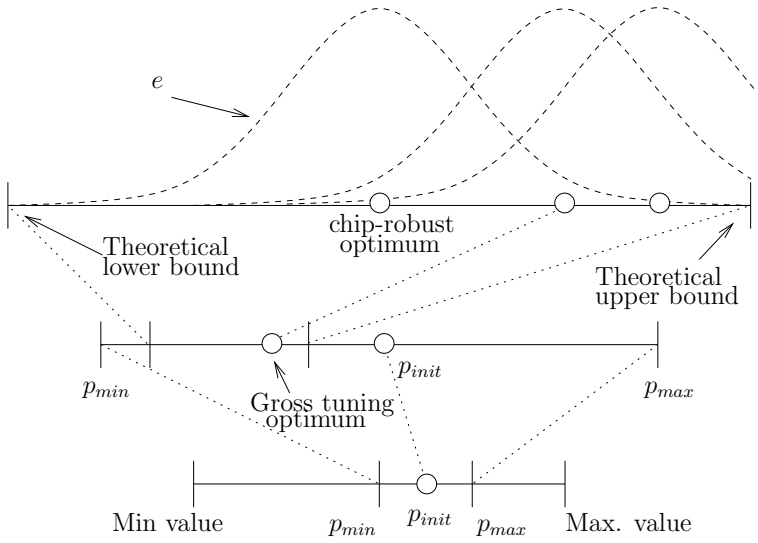


Figure 3.3: A illustration of how chip-specific robustness is achieved for one component of the parameter vector. In the first two levels of zoom, the region where the optimisation is performed is shown. In the top and last level of zoom, it is illustrated that if the probing parameter value is located in the middle of a theoretical working range, errors will be less likely to happen.

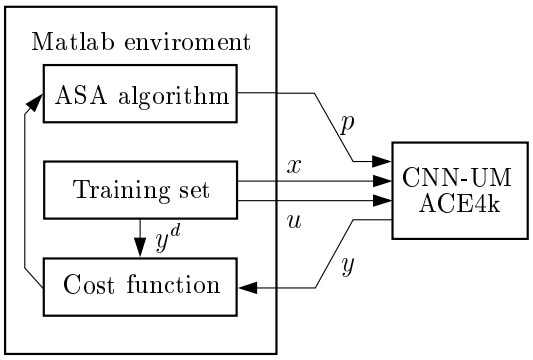


Figure 3.4: An overview of the structure used in the optimisation.

In order to ease the ASA search, each template optimisation might start

with constraints or structure in the template values such as symmetry or imposed zero values. The constraints are then softened for subsequent runs until the point where all 19 template values become free for optimisation, according to the main algorithm shown in Figure 3.2.

The input and initial state for each optimised template operation were in general random binary or grey-scale images, with some exceptions[2]. The respective desired output was obtained from simulators of ideal CNN-UM using robust templates available in the literature [116]. To avoid tiling, the size of the images were chosen according to the size of each chip, i.e. 64×64 pixels and 128×128 pixels for ACE4k and ACE16k respectively.

Owing to the speed of the chips, each annealing iteration consumed more time for the generation and acceptance of new probing templates in Matlab than for the evaluation of the cost function itself, which is done by sending the input and initial state images to the chip and acquiring its output. The total duration of a simple annealing iteration was about 50 milliseconds and the number of iterations for each optimisation was in average in the order of tens of thousand. The difference in size did not affect the duration of the measurements for the two different chips used here since CNN computations are totally parallel.

In what follows, a demonstrative example of chip-specific robustness and more precise explanations for each individual experiments are presented. Among the experiments performed in the lab, two optimisations on the ACE4k chip are described here: binary edge detection, and average with binary output. On the ACE16k chip, we performed three optimisations: average with binary output, thresholding to binary, and sobel edge detection.

3.6.1 An example of a chip-specific robust template

To demonstrate the concept of chip-specific robustness, Figure 3.5 shows the error surface measured with the chip for the logic difference template operation. For illustrative purposes, the template was optimised with only two free parameters. The structure of this template with the free parame-

[2]Template operation like binary edge detection do not optimise well with random images.

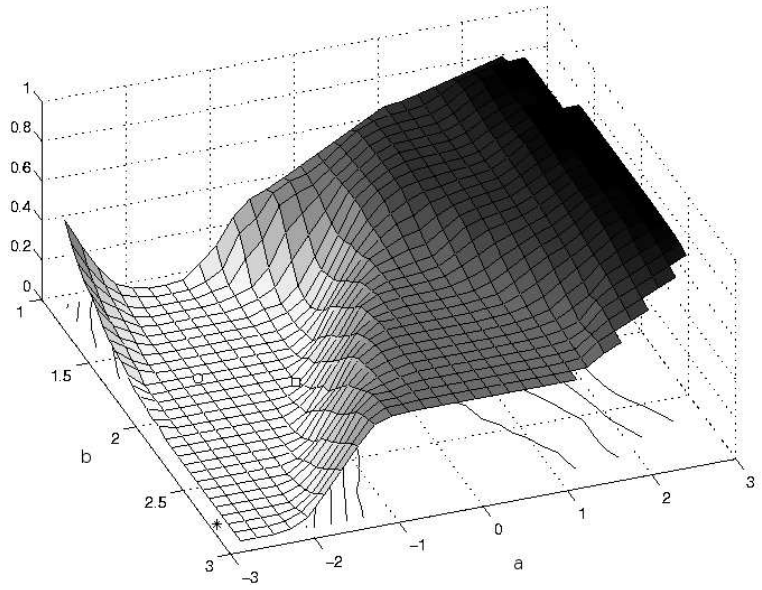


Figure 3.5: Logic Difference template error surface for the ACE4k chip. $\diamond$ - initial template; $\circ$ - chip-specific optimal template; $*$ - chip-specific robust template.

ters a and b is

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad z = -1. \tag{3.4}$$

Observe that, although the initial template values are located in an optimal point in the error surface, this point is near other non-optimal points. This proximity may result in errors as soon as any interference shifts the template values, even slightly. After the chip-specific optimisation, the resulting template is occasionally located in a better error neighbourhood. By optimising with target on chip-specific robustness, the template assumes the most robust position for the given error surface.

3.6.2 Binary edge detection on the ACE4k chip

Figure 3.6 shows the results for this template operation. The costs of the original and the final templates are presented in Table 3.1 together with the

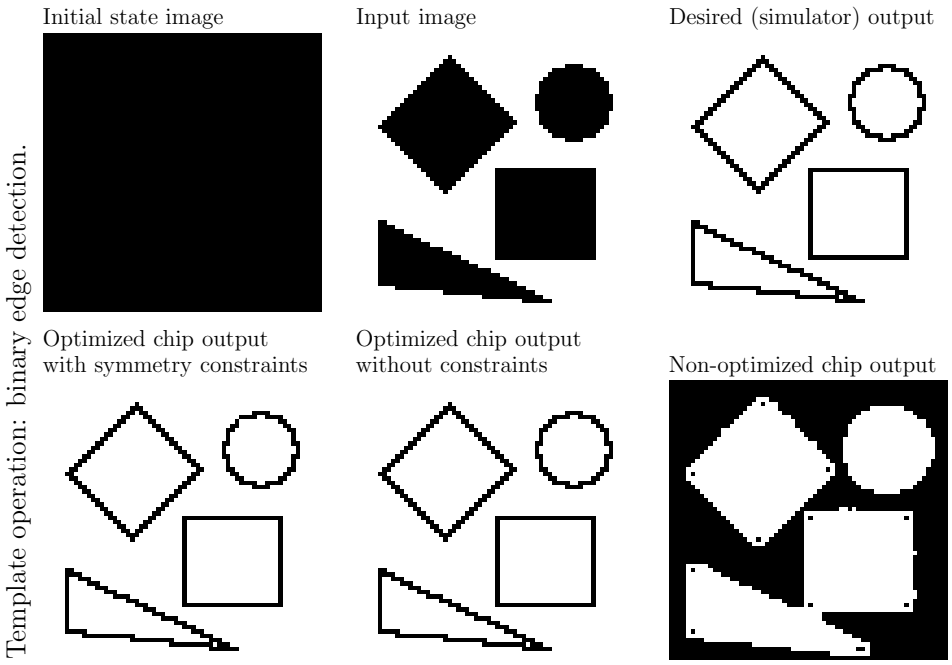


Figure 3.6: Results for binary edge detection template operation on the ACE4k chip.

respective template values. Figure 3.6 shows the output of the intermediate symmetric template

$$A = \begin{bmatrix} -0.24 & -0.24 & -0.22 \\ -0.17 & 1.89 & -0.17 \\ -0.22 & -0.24 & -0.24 \end{bmatrix};$$
$$B = \begin{bmatrix} -0.65 & 0.03 & -0.81 \\ -0.07 & 2.85 & -0.07 \\ -0.83 & 0.03 & -0.65 \end{bmatrix}; \quad z = -1.36,$$

which presented an average cost of 0.0050.

3.6.3 Average with binary output on the ACE4k chip

For this template operation, the chip reacted better using symmetric templates. The results can be seen in Figure 3.7 and the respective template values are in Table 3.1. The template values for the non-symmetric optimal

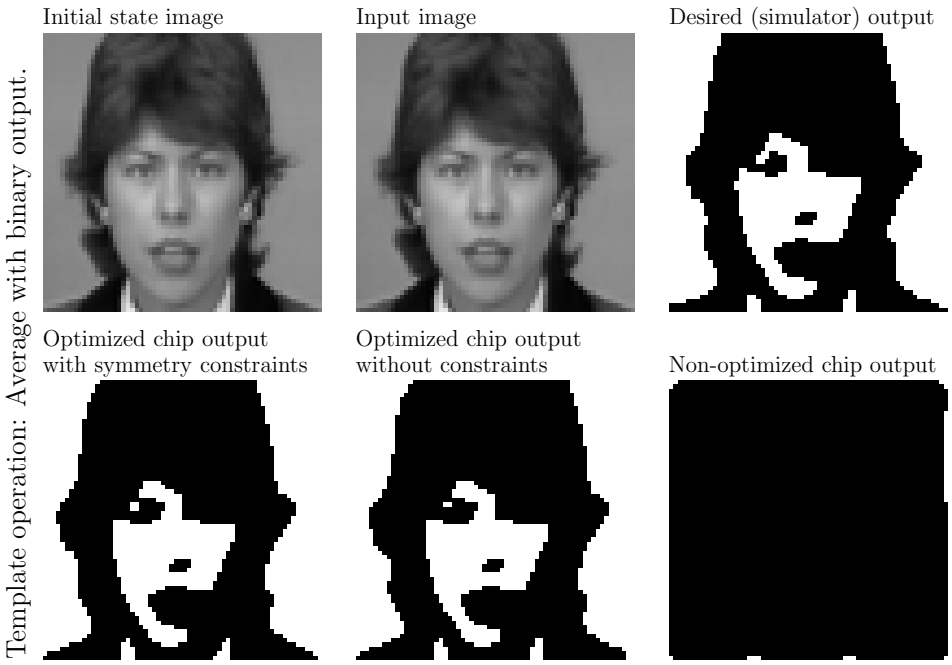


Figure 3.7: Results for average with binary output template operation on the ACE4k chip.

template are

$$A = \begin{bmatrix} -0.50 & 1.81 & -0.17 \\ 1.75 & 2.96 & 1.69 \\ -0.20 & 1.82 & -0.50 \end{bmatrix};$$
$$B = \begin{bmatrix} 0.34 & 0.78 & -0.17 \\ 0.62 & 0.88 & 0.77 \\ -0.25 & 0.57 & 0.41 \end{bmatrix}; \quad z = -2.51,$$

and its average cost was 0.1451. Its results can also be seen on Figure 3.7.

3.6.4 Average with binary output on the ACE16k chip

For this chip, in contrast with the ACE4k, the operation of average with binary output presented better results using the template without symmetric constraints. The best template is in Table 3.2. The intermediate

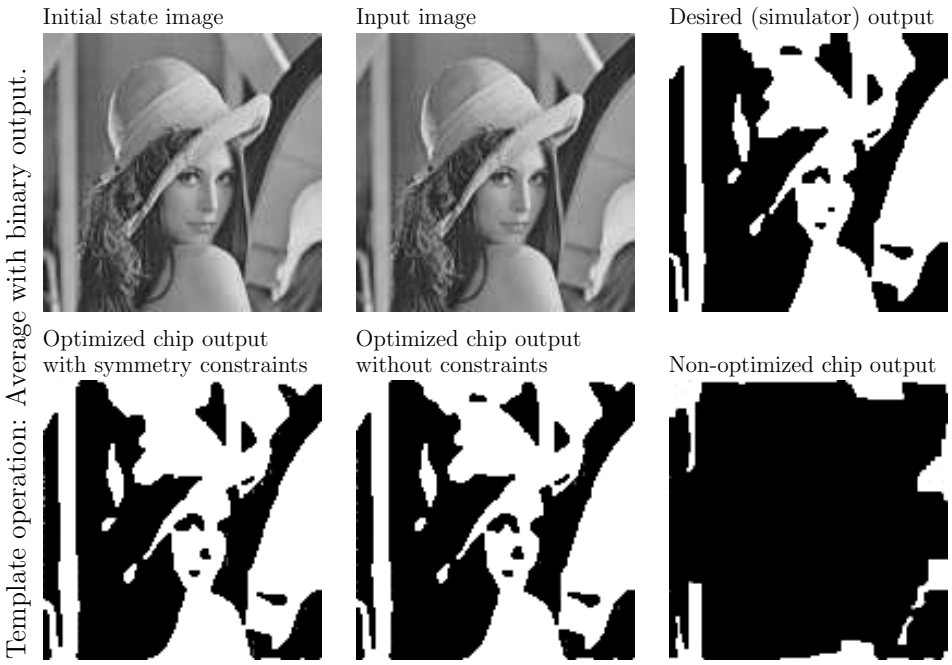


Figure 3.8: Results for average with binary output template operation on the ACE16k chip.

symmetric template

$$A = \begin{bmatrix} 0.38 & 2.68 & 0.32 \\ 2.99 & 5.38 & 2.99 \\ 0.32 & 2.69 & 0.38 \end{bmatrix};$$
$$B = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 4.13 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}; \quad z = -3.19,$$

which presented and average cost of 0.1734, has its results depicted in Figure 3.8 together with the other results.

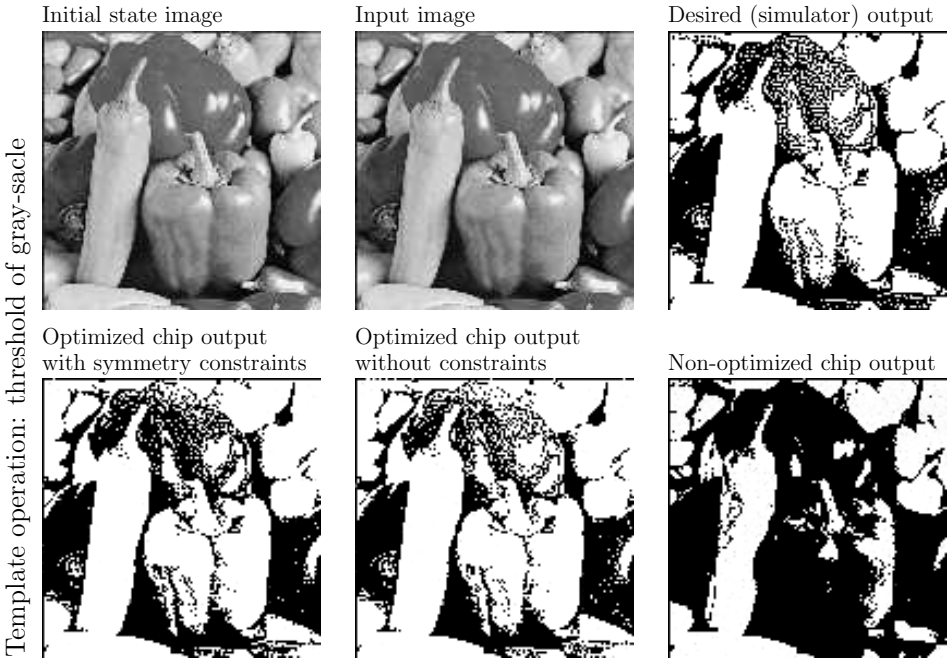


Figure 3.9: Results for thresholding to binary template operation on the ACE16k chip.

3.6.5 Thresholding to binary on the ACE16k chip

For this operation, symmetry constraints did not effect significantly the average cost. The symmetric template

$$A = \begin{bmatrix} -0.19 & -0.21 & -0.17 \\ -0.14 & 5.93 & -0.14 \\ -0.17 & -0.21 & -0.19 \end{bmatrix};$$
$$B = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 6.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}; \quad z = -2.46,$$

with average cost equal to 0.2188 was slightly worst than its non-symmetric version, which values and cost can be seen in Table 3.2. The results can be visualised in Figure 3.9.

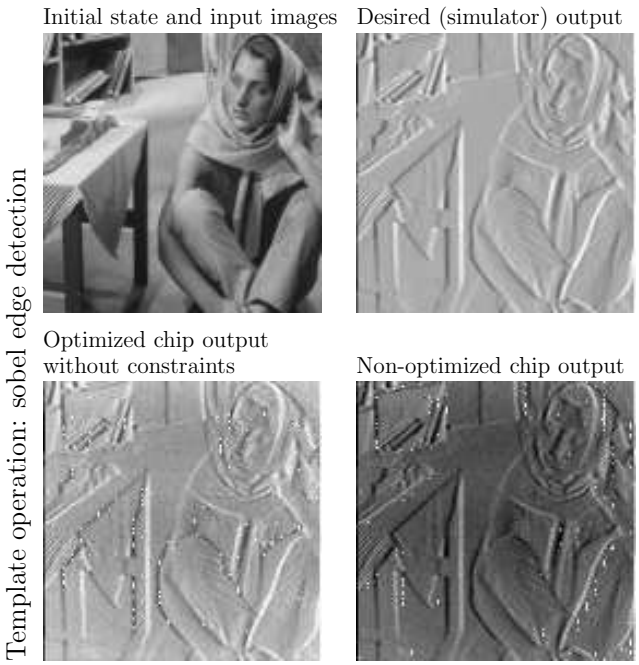


Figure 3.10: Results for sobel edge detection template operation on the ACE16k chip.

3.6.6 Sobel edge detection on the ACE16k chip

Although template operations with grey-scale output are difficult to optimise in CNN chips, it was possible to do that with sobel edge detection operation due to the stability of this operation. However for the general grey-scale case, a more complex approach, which takes into account desirable trajectories, like the one described in Chapter 4 needs to be considered. Figure 3.10 presents the results. The template values and respective costs for this operation can also be found in Table 3.2.

3.7 Conclusions

Despite the extraordinary speed performance of CNN-UM chips for image processing tasks, digital systems are still predominant in the field owing to their superior reliability. The development of a method toward chip-specific

Operation		Template values			Cost
binary edge detection	orig.	$A = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $B = \begin{bmatrix} -1.00 & -1.00 & -1.00 \\ -1.00 & 8.00 & -1.00 \\ -1.00 & -1.00 & -1.00 \end{bmatrix}$ $z = -1.00$			0.77
	final	$A = \begin{bmatrix} -0.23 & -0.07 & -0.19 \\ -0.18 & 2.08 & -0.16 \\ -0.26 & -0.18 & -0.14 \end{bmatrix}$ $B = \begin{bmatrix} -0.54 & -0.07 & -0.53 \\ 0.03 & 2.93 & -0.08 \\ -0.42 & -0.03 & -0.42 \end{bmatrix}$ $z = -1.94$			zero
Average with binary output	orig.	$A = \begin{bmatrix} 0.00 & 1.00 & 0.00 \\ 1.00 & 2.00 & 1.00 \\ 0.00 & 1.00 & 0.00 \end{bmatrix}$ $B = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $z = 0.00$			0.64
	final	$A = \begin{bmatrix} -0.35 & 1.62 & -0.24 \\ 1.58 & 2.95 & 1.58 \\ -0.24 & 1.62 & -0.35 \end{bmatrix}$ $B = \begin{bmatrix} 0.20 & 0.72 & 0.02 \\ 0.65 & 1.02 & 0.65 \\ 0.02 & 0.72 & 0.20 \end{bmatrix}$ $z = -2.39$			0.13

Table 3.1: Template values and the respective error costs for a specific ACE4k chip

Operation		Template values	Cost
Average with binary output	orig.	$A = \begin{bmatrix} 0.00 & 1.00 & 0.00 \\ 1.00 & 2.00 & 1.00 \\ 0.00 & 1.00 & 0.00 \end{bmatrix}$ $B = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $z = 0.00$	0.65
	final	$A = \begin{bmatrix} 0.44 & 2.88 & 0.45 \\ 2.93 & 5.67 & 2.82 \\ 0.61 & 2.78 & 0.82 \end{bmatrix}$ $B = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 4.91 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $z = -3.52$	0.16
Threshold	orig.	$A = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 2.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $B = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $z = 0.00$	0.5032
	final	$A = \begin{bmatrix} -0.14 & -0.25 & -0.22 \\ -0.10 & 5.85 & -0.23 \\ -0.15 & -0.12 & -0.22 \end{bmatrix}$ $B = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 5.89 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $z = -2.57$	0.22
Sobel	orig.	$A = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & -4.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $B = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ -3.00 & 0.00 & 3.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $z = -1.70$	0.31
	final	$A = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & -3.40 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$ $B = \begin{bmatrix} 0.24 & 0.01 & -0.07 \\ 2.59 & 0.31 & 2.81 \\ -0.10 & -0.04 & 0.17 \end{bmatrix}$ $z = -2.46$	0.07

Table 3.2: Template values and the respective error costs for a specific ACE16k chip

robustness contributes to weaken such superiority. The method described here works well for all tested stable binary output template operations and a grey-scale template operation. Using an optimisation method that does not rely on information about the gradient of the cost function allowed this approach to efficiently tune not only uncoupled templates but also coupled ones. For the case of grey-scale outputs, a more elaborated approach that takes into account transient time is desirable. Chip-specific robust tuning of templates provides a method to place parameter values in the middle of a correct operating range. This minimises the erroneous behaviour of CNN chips for optimised templates due to parameter variation caused by post-manufacturing disturbance, e.g. temperature and noise, which may cause the parameter values to fall outside the correctly working range. Chip-specific robustness exposes a trend of analog and mixed-signal self-test and self-tuning chips that might be explored using embedded implementation.

Chapter 4

Learning Dynamics in CNNs

Among other characteristics of cellular neural networks, the degree of parallelism, their suitability to VLSI implementation, and the ability to use these to perform many different image processing operations are already very remarkable features, but that is not all. Mapping traditional image processing tasks is merely just the tip of the iceberg. CNNs can exhibit a large number of extraordinary complex behaviours. In order to estimate how rich CNN behaviour can be, the reader simply need to look into the dynamics of a single cell and extrapolate this to an entire network. The temporal dynamics of a single cell can range from fixed-point asymptotic stability to limit cycles and chaotic behaviour. If the entire network is considered, besides temporal dynamics, spatial pattern formation can also emerge from topographically distributed arrays of cells. Many of these *spatiotemporal* phenomena observed in CNNs are also present in different disciplines of science, e.g. chemistry, biology, and physics. Recently, spatiotemporal CNN dynamics has been used to perform computation in the form of a new paradigm that has been called Active Wave Computing (AWC) [126].

In this chapter we present some examples of spatiotemporal dynamics that can be observed in both CNNs and in nature. We then describe a method for learning these phenomena in the framework of trajectory learning. To do so, we extend the theory for learning of trajectories with recurrent neural networks to support learning of spatiotemporal trajectories with CNNs. Some simulations and chip experiments with the resulting methodology are also presented here.

4.1 Spatiotemporal behaviour

Typically, in nature, spatiotemporal behaviour occurs in networks of dynamical systems. As a dynamical system, each node, or cell, of a network is responsible for generating temporal behaviour. In itself, each cell may present any type of dynamics, complex or not. The interconnections of a network are responsible for the interactions between cell dynamics. Depending on the degree of interconnection, spatial pattern formation may arise. In nature interconnections are usually restricted to a certain local neighbourhood that can be hierarchically observed in different scales. As an example of a node or cell, atoms interact locally with other atoms to form molecules, which form materials or substances. In biology, cells interact with neighbour cells to create organs that then compose an individual organ systems. Interactions between organ systems form an individual, animal or human. An individual interact with other individuals, enabling villages, societies, nations, etc. The spatiotemporal dynamics of such systems is the target of different studies in multidisciplinary fields, especially at the elementary level. Perhaps the most studied type of spatiotemporal behaviour in the past few years are these present in active media. An active or excitable medium is a spatially distributed system of autonomous elements, here called cells, with non-linear dynamical properties. Each of these cells interacts with the surrounding neighbours via diffusion [125]. Active media are characterised by the ability to propagate signals without decreasing strength. These signals often take the form of waves that emerge depending on different initial conditions. Such waves are commonly described as trigger waves or autowaves and can be widely observed in nature. Examples include waves in the ocean, combustion waves, phase transitions [76], waves of the hart tissue [6], electrical stimulus waves in the retina [47], etc.

Due to the principle of local activity, cellular neural networks can also behave as an active medium. With programmable parameters, CNN-UMs are able to simulate different types of homogeneous and heterogeneous active media. Besides simulating these complex natural phenomena, the CNN-UM can use waves to perform computation. Before CNN-UMs, elementary image processing tasks were already reported to be feasible with light-sensitive chemical waves [74].

In the following, we will present a few examples of 2-dimensional (2D) spatiotemporal dynamics that emerge from active media. We characterise them into two different types of behaviour: periodic and aperiodic.

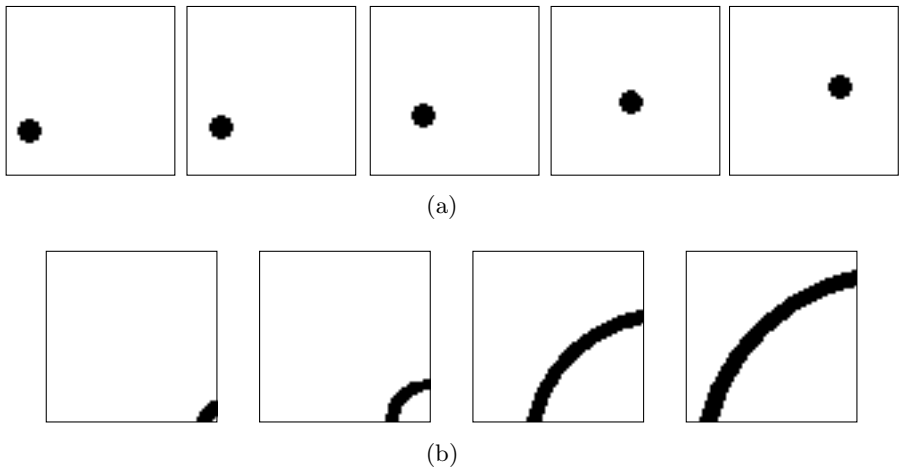


Figure 4.1: Initial conditions and snapshots in time of: (a) a dot travelling across the image at a given angle; and (b) a wave of given width propagating across the image in a specific direction.

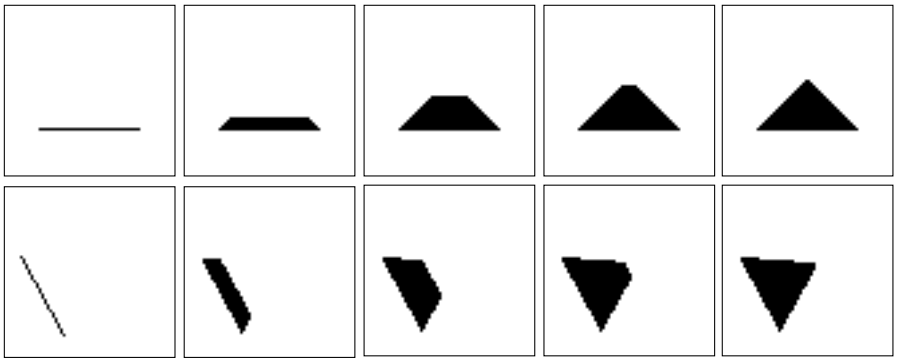
4.1.1 Aperiodic spatiotemporal behaviour

A simple example of aperiodic spatiotemporal behaviour can be seen of Figure 4.1(a) where a travelling dot can be observed which moves across the image at an arbitrary angle. In the same way as the dot, one might think of a travelling wave of a specific width also moving across the image, like in Figure 4.1(b).

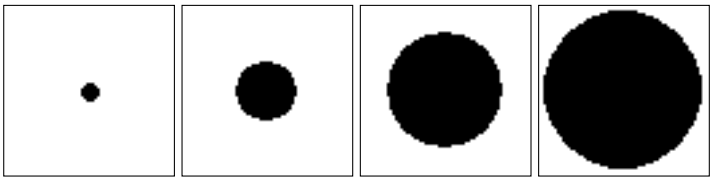
We define aperiodic travelling waves like the one in Figure 4.1(b) in two different kinds: "convergent" and "divergent"[1]. A typical case of a "convergent" wave is a wave that starts from a line and propagates with decreasing length in order to form a pyramid. This operation converges when the top of the pyramid is built like Figure 4.2(a). When aperiodic travelling waves do not converge to a fixed image[2], we will call it "divergent". An interesting aperiodic wave was used in [101] to compute the shortest path of flat and wrinkled labyrinths. This wave simulates e.g. the waves of combustion

[1] here, these concepts do not have a strict mathematical meaning but are rather limited to the geometric boundaries

[2] a fictitious image of infinite size is considered here to avoid issues with boundary condition



(a)



(b)

Figure 4.2: Initial conditions and snapshots in time of: (a) a wave that propagates to form a horizontal pyramid and a pyramid in a given angle and shape; and (b) a combustion wave which burns from the centre to the borders of the image.

where an active medium cannot return to the same state after the propagation of the wave. Figure 4.2(b) shows a typical combustion wave where the burning effect starts at the centre of the image.

4.1.2 Periodic spatiotemporal behaviour: autowaves

Another interesting class of spatiotemporal behaviour is autowaves. The term autowave is an abbreviation of "autonomous wave" commonly used to characterise self-sustained signals that induce a local release of stored energy in an active medium, and use this energy to trigger the same process in neighbour regions. This term has been often used in the CNN community to describe 2D periodic travelling waves that have the following properties [114]:

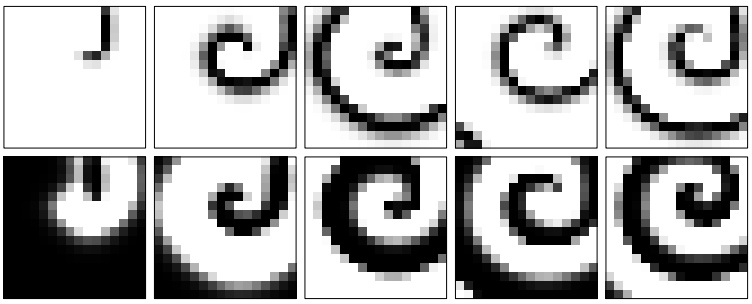


Figure 4.3: Spiral autowave phenomena in a 2-layer CNN simulator: initial conditions and time snapshots of the output in the two layers. Each row represents one layer.

- the waveform and the amplitude of wave remain constant during propagation;
- the waves do not reflect at obstacles;
- colliding waves are annihilated and thus no interference emerges;
- diffraction can be observed in the same way as for classical waves.

Many waves occurring in nature share the same properties. Typical examples include waves in the cerebral cortex, epidemic waves, combustion waves, and reaction-diffusion processes. In cellular neural networks, autowaves represent the means for the new AWC paradigm and have been used, for example, to guide robots along obstacles towards a target [5]. An example of autowaves, known as spiral wave, is shown in Figure 4.3.

In CNN systems, complex autowaves such as spiral waves have been observed to emerge in 2-D arrays of second or higher order cells [101, 92] and delayed type first order cells [114]. As arrays of regular first order cells cannot generate necessary active local dynamics, complex autowaves cannot be observed in these systems. However, in [157, 158] this type of wave was observed in a VLSI implementation that was designed to be a first order 2-D array of CNN cells [109]. Although internal sources of autowaves (chip-inherent) could not be avoided during the experiments, external sources could also be placed generating a competition between the sources. Explanations for this observation are not fully understood yet

although no important malfunction has been observed in this chip. It is possible that the physical system implemented in the chip behaves more like a second order or delayed type first order CNN system rather than as a first order system as intended in the original design.

4.2 Learning spatiotemporal behaviour

There exists a large number of methodologies for learning of ordinary template operations in CNNs. The output of these operations are generally composed of a single image. This image can either represent the output of a network when it converges to a fixed-point or a time-truncation snapshot of a temporally unstable operation. Our goal is to develop a methodology based on trajectory learning that can learn a sequence of images rather than a single one. In the sequel we define the CNN equations that we use in this Chapter in matrix form. CNN equations in matrix form are usually more convenient for image processing applications [23]. Here we use this representation as an intermediate step toward a vector representation that approximates the one used in trajectory learning. These equations are defined as follows

$$\begin{array}{rcl} \dot{\mathbf{X}} & = & -\mathbf{X} + A \otimes \mathbf{Y} + B \otimes \mathbf{U} + z\mathbf{J}, \\ \mathbf{Y} & = & F(\mathbf{X}), \end{array} \tag{4.1}$$

where

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M,1} & x_{M,2} & \cdots & x_{M,N} \end{bmatrix}, \mathbf{U} = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,N} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{M,1} & u_{M,2} & \cdots & u_{M,N} \end{bmatrix},$$

$$F(\mathbf{X}) = \begin{bmatrix} f(x_{1,1}) & f(x_{1,2}) & \cdots & f(x_{1,N}) \\ f(x_{2,1}) & f(x_{2,2}) & \cdots & f(x_{2,N}) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_{M,1}) & f(x_{M,2}) & \cdots & f(x_{M,N}) \end{bmatrix},$$

with $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{M \times N}$ denoting the time-dependent state and output of the whole network, respectively, and $\mathbf{U} \in \mathbb{R}^{M \times N}$ denoting its time-invariant input. $\mathbf{J} \in \mathbb{R}^{M \times N}$ is a matrix with all elements equal to 1. Time derivative

of $\mathbf{X}$ is denoted by $\dot{\mathbf{X}}$. The elements of $A, B \in \mathbb{R}^{(2r+1)\times(2r+1)}$, and bias value z denote template values, , where $r \in \mathbb{N}$ is the radius of the local neighbourhood scheme of the network, *e.g.* $r = 1$ results in matrices A and B with size 3×3. The operation $\otimes$ is the local template convolution applied to the whole array and can be defined as follows

$$\begin{aligned} &\otimes : \mathbb{R}^{(2r+1)\times(2r+1)} \times \mathbb{R}^{M\times N} \to \mathbb{R}^{M\times N} \\ &A \otimes \mathbf{Y} \mapsto \left\{ \mathbf{Q} | q_{i,j} = \sum_{|k,l| \leqslant r} a_{k,l} y_{i-k,j-l} \right\}, \end{aligned} \tag{4.2}$$

with $i, j \in \mathbb{N}$ and $k, l \in \mathbb{Z}$; $a_{k,l}$ is k,l-indexed element of A; $y_{i,j}$ and $q_{i,j}$ are the elements of $\mathbf{Y}$ and $\mathbf{Q}$ in the ith row, jth column of the array, respectively; and M and N are the number of cell rows and columns in the CNN array. Boundary conditions can be one of the three different types described in Chapter 2.

Without loss of generality, we can vectorise (4.1), resulting in the following equation:

$$\begin{aligned} \dot{\mathbf{x}} &= -\mathbf{x} + A \otimes \mathbf{y} + B \otimes \mathbf{u} + z\mathbf{j}, \\ \mathbf{y} &= f(\mathbf{x}), \end{aligned} \tag{4.3}$$

with the vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{u}$, and $\mathbf{j}$ being vectorised versions of $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{U}$, and $\mathbf{J}$, *e.g.*

$$\mathbf{x} = [x_{1,1}, x_{2,1}, \cdots, x_{M,1}, x_{1,2}, x_{2,2}, \cdots, x_{M,2}, \cdots, x_{1,N}, x_{2,N}, \cdots, x_{M,N}]^T.$$

Learning of ordinary template operations with this network can now be defined as the following error minimisation problem:

$$\min_{A,B,z} E(A, B, z), \tag{4.4}$$

with $E(A, B, z)$ being the squared error or cost function to be minimised,

$$E = \left|\left| \mathbf{y}^d - \mathbf{y}(A, B, z, T) \right|\right|_2^2 \tag{4.5}$$

$$= \sum_{i,j} (y^d_{i,j} - y_{i,j}(A, B, z, T))^2. \tag{4.6}$$

Observe that the minimisation of the norm would require a square root without effecting the minimisation problem. Hence we prefer the minimisation of the norm squared E, where $y^d_{i,j}$ is the i, j indexed element of the

desired output $\mathbf{y}^d$; equivalently, $\mathbf{y}(A, B, z, T)$ and $y_{i,j}(A, B, z, T)$ represent the actual output of the network according to (4.3) with template values A, B, and z, time-truncated at the time instant T.

For fixed-point convergent outputs, T must be sufficiently large to enable the complete evolution of the dynamics. For time-truncated snapshot outputs, T is either arbitrary or is used as an adjustable parameter for the template functionality.

Apart from intrinsic network characteristics, the problem of learning ordinary template operations is not much different from general supervised learning for neural networks related to input-output mappings. Nevertheless, the problem of learning spatiotemporal behaviour is somewhat distinct and is more closely related to trajectory learning.

In this Section we describe a methodology for learning 2D spatiotemporal behaviour with cellular neural networks. In order to do that, we extend the theory of trajectory learning to support the spatial character of spatiotemporal dynamics.

4.2.1 Trajectory learning and recurrent neural networks

Trajectory learning, the problem of modifying parameters of dynamical systems to ensure that their outputs follow a given function of time, has been analysed by many scientists [145, 94, 42, 165, 14, 12, 29, 8, 130]. Most frequently they have used Recurrent Neural Networks (RNN) as a model for such systems. Consider the following RNN model that describes the dynamics of these systems:

$$\begin{aligned} \dot{\mathbf{x}} &= -\mathbf{x} + W\mathbf{y} + W'\mathbf{u} + \mathbf{z}, \\ \mathbf{y} &= f(\mathbf{x}), \end{aligned} \tag{4.7}$$

where $\mathbf{y} \in \mathbb{R}^M$, $\mathbf{x} \in \mathbb{R}^N$, and $\mathbf{u} \in \mathbb{R}^P$ are column vectors representing output, state, and input, respectively, $\mathbf{x}$ and $\mathbf{y}$ being time-variant and $\mathbf{u}$ being time-invariant; the matrices $W \in \mathbb{R}^{N\times M}$, and $W' \in \mathbb{R}^{N\times P}$, are the weight matrices for the output $\mathbf{y}$ and the constant input $\mathbf{u}$, respectively. The term $\mathbf{z} \in \mathbb{R}^N$ denotes the bias of the network. The problem consists of minimising a cost function which is not defined at a fixed-point but rather is a function of the temporal behaviour of the model. The problem is often addressed in the literature by gradient descent methods [42, 41, 165, 80]. Pearlmutter's work [100] presents an earlier survey in this subject.

These methods are known to coexist with problems like local minima [14] and vanishing gradients with the evolution of the dynamics [12]. On the other hand, methods that do not use gradient information in the learning process [29, 8, 12, 130], which include global optimisation methodologies like simulated annealing, genetic algorithms, multi-grid random search, etc., are persistently slower, but are more frequently able to converge to a globally optimal solution. The learning problem can be described by the following equation:

$$\min_{W,W',\mathbf{z}} E(W, W', \mathbf{z}), \tag{4.8}$$

with $E(t, W, W', \mathbf{z})$ representing the error or cost function to be minimised,

$$E = \sum_{i=1}^{M} \int_{t_0}^{t_f} (y_i^d(t) - y_i(W, W', \mathbf{z}, t))^2 dt, \tag{4.9}$$

where the square difference between the desired trajectory function $y_i^d(t)$ and system's output function in time $y_i(W, W', \mathbf{z}, t)$ is integrated from time t_0 to t_f and then summed over all M output neurons. The cost function $E(W, W', \mathbf{z})$ may also assume different forms [29, 80, 130], but this one will be used here to devise a new cost function for the training of spatiotemporal behaviour on CNNs.

4.2.2 Trajectory learning and cellular neural networks

The mapping of trajectory learning with RNNs into learning of spatiotemporal behaviour with CNNs is straightforward due to the similarities between (4.3) and (4.7). Moreover, computational complexity can be considerably decreased for learning using CNNs.

Trajectory learning with RNNs can become increasingly complicated when using larger numbers of neurons due to the quadratic increase in the number of weight connections. This problem can be moderated by assuming zero elements in the weight matrices and therefore reducing computational burden. This is essentially what happens in (4.3) when compared to (4.7). More precisely, only weights of neighbouring cells are taken into account with the remaining values of the weight matrices equal to zero. Moreover, this CNN model has weights that are space-invariant. This means that the description of a cell is sufficient to describe the whole system. Indeed, templates are local and invariant equivalents of the matrices W, and W'

and bias term $\mathbf{z}$ in (4.7). Therefore, the description of a first order CNN system with local and invariant weights can be reduced to the description of the behaviour of a single cell:

$$\begin{aligned} \frac{dx_{i,j}}{dt} &= -x_{i,j}(t) + A \odot y_{i,j} + B \odot u_{i,j} + z, \\ y_{i,j} &= f(x_{i,j}), \end{aligned} \tag{4.10}$$

with A, B, and z being the local and space-invariant equivalent of W, W', and $\mathbf{z}$ in (4.7). The operation $\odot$ is the single-cell equivalent of the array convolution $\otimes$ defined in (4.2), such as

$$\begin{aligned} \odot : \mathbb{R}^{(2r+1)\times(2r+1)} \times \mathbb{R} \to \mathbb{R}, \\ A \odot y_{i,j} \mapsto q = \sum_{|k,l| \leqslant r} a_{k,l} y_{i-k,j-l}. \end{aligned} \tag{4.11}$$

In the case of $r = 1$, the operation $\odot$ defines the discrete convolution of two 3×3 matrices, which in this case are the A or B template and any 3×3 local slice of the whole output or input matrices, respectively.

Other CNN models that are especially important for the modelling of complex spatiotemporal behaviour are second order CNN systems. Consider one of these models with a regular array disposed in two layers, in which its dynamics can be represented by the following equations:

$$\begin{aligned} \frac{dx_{i,j;1}}{dt} &= -x_{i,j;1} + A_{1,1} \odot y_{i,j;1} + A_{1,2} \odot y_{i,j;2} + B_1 \odot u_{i,j;1} + z_1, \\ y_{i,j;1} &= f(x_{i,j;1}), \\ \frac{dx_{i,j;2}}{dt} &= -x_{i,j;2} + A_{2,2} \odot y_{i,j;2} + A_{2,1} \odot y_{i,j;1} + B_2 \odot u_{i,j;2} + z_2, \\ y_{i,j;2} &= f(x_{i,j;2}), \end{aligned} \tag{4.12}$$

where the index after the semi-column identifies the layer and the indices i and j locate the given cell within the layer. This equation can also be written in its condensed form:

$$\frac{d\mathbf{x}_{i,j}}{dt} = -\mathbf{x}_{i,j} + A \odot \mathbf{y}_{i,j} + B \odot \mathbf{u}_{i,j} + \mathbf{z}, \tag{4.13}$$

with

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2.1} & A_{2,2} \end{bmatrix}; \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}; \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix};$$

and

$$\mathbf{x}_{i,j} = \begin{bmatrix} x_{i,j;1} \\ x_{i,j;2} \end{bmatrix}; \quad \mathbf{y}_{i,j} = \begin{bmatrix} y_{i,j;1} \\ y_{i,j;2} \end{bmatrix}; \quad \mathbf{u}_{i,j} = \begin{bmatrix} u_{i,j;1} \\ u_{i,j;2} \end{bmatrix}.$$

These equations describe the behaviour of a second order cell, which when isolated from the neighbouring cells may behave as an oscillator [11] and when coupled to other cells is able to generate interesting complex spatiotemporal patterns. Moreover, these equations closely describe the model of the CACE1k chip [20] which is a major advance in VLSI implementation for generation of complex behaviour.

Given (4.3) and (4.7), due to locality and space invariance, trajectory learning with CNNs involves much less unknowns than with RNNs. Indeed, the number of unknowns remains constant despite the size of the cellular neural network. In spite of the condensed set of values that describes a CNN, a large variety of dynamical phenomena can be observed. Moreover, as is seen in Chapter 2, locality and invariance alone make these systems very suitable for VLSI implementation which is a trend that has been emerging in the past years. Today, high-end silicon versions of CNN Universal Machines (CNN-UM) are commercially available for the development of extremely high speed image processing applications [1].

So far, we presented the mapping of trajectory learning with RNNs into learning of temporal dynamics with CNNs simpler and straightforward. However, there are two important issues that should be well understood.

- The first issue concerns the type of optimisation method used to reduce the learning error. In spite of being very efficient and widely used, gradient descent techniques are hard to apply in the case of CNNs. The non-linear output function $y = f(x)$ often assumes the following piecewise linear non-differentiable form, which complicates the derivation of an analytical form for the gradient of the cost function and would require techniques from non-differentiable local optimisation:

$$f(x) = \frac{1}{2}(|x+1| - |x-1|). \tag{4.14}$$

  Differentiable approximations [139, 50] of this function have been proposed to allow utilisation of common gradient learning techniques like recurrent back propagation. However, for learning using chip measurements to calculate solution costs, analytically calculated gradients are merely approximations of the real system, which may lead

to convergence to false optima. Global optimisation methods can be used with reasonable confidence since the number of parameters to be optimised is not very high.

- The second issue needs more care and must be addressed in a case-by-case fashion. It concerns the generation of feasible training sets. While a trajectory can be described by a temporal sequence of values for each output, e.g. $\sin(t)$, spatiotemporal behaviour in a grid of cells needs to be described by continuous 2-D image sequences where the values of the pixels correspond to the output of a single cell. Due to coupling between cells, the desired values of every individual pixel trajectory cannot be derived independently and must be considered as a whole. Since cell interconnections are local and space-invariant, the specified desired spatiotemporal dynamics across the grid of cells must be consistent and coordinated in space and time. Manual generation of the desired image sequence can thus be tricky and may result in a behaviour that is physically impossible to learn.

In view of these two observations, we describe in the following Section an extension of the trajectory learning theory presented here in order to incorporate aspects that are relevant for the learning of spatiotemporal dynamics.

4.2.3 Learning sequences of images

In Chapter 3 we employed a global optimisation to adjust template parameters in order to minimise errors on chip results. For that, we described a cost function which uses chip responses in order to optimise these parameters. A similar approach is considered here for the spatiotemporal dynamics learning problem. The key differences between the two approaches lay on the extra requirements needed by latter.

First, while the optimisation described in Chapter 3 only concerns tuning, i.e. there exists a good initial approximation of the solution, this approach concerns learning, which means that an initial approximation may not exist. This emphasises the importance of using a global optimisation method and that convergence may take a longer period of time.

Second, the cost function for this problem involves more unknowns than in the tuning case. Namely, besides the templates values, time instants also need to be included in the set of parameter to be optimised. This

requirement is related to the issue concerning the feasibility of training sets for spatiotemporal dynamics on CNNs. In the following we illustrate this problem.

Feasible training sets

The appropriate design of training sets that incorporates all functionalities and constraints of a template operation is essential for successful chip-specific tuning of the operation, as seen in Chapter 3. Learning of spatiotemporal dynamics also requires equal effort. In addition, physical feasibility of the desired dynamics needs to be taken into account. We consider two distinct aspects that effect how feasible the desired spatiotemporal dynamics can be.

The first aspect relates to spatial feasibility. Considering the space invariance of the connections in CNNs, the desired dynamics must present equivalent behaviour across the network. Assuming each cell is identical and so are its set of local connections, every cell must present the same behaviour with the same speeds of dynamics. Figure 4.4 presents an example of an infeasible desired dynamics. Although the two waves in the image sequence present the same dynamics, a circular outward propagating front—their speeds of propagation are different. Such a difference would require an inhomogeneous distribution of template values across the array of cells, which conflicts with the principle of space-invariant CNNs. This aspect is also important when designing the training sets for tuning or learning of fixed-point ordinary template operations.

The second aspect concerns temporal feasibility and it is characteristic of trajectory learning. Assuming local and space-invariant CNNs and constant template values, the dynamical behaviour of the network trajectory may not change in time. Neither the speed of dynamics nor the dynamics itself may change along the trajectory. Figure 4.5 presents an illustrative example of such infeasible trajectory. The two waves in the image sequence are coherent spatially, but the temporal response can only be feasible with varying template values.

In summary, besides the need to ensure that all functionalities and constraints of the template operation are incorporated in the training set, spatial and temporal feasibility is also a concern. If these are not guaranteed, or at least improved to a certain level, the learning may fail even when all functionalities and constrains are present in the training set. Follow-

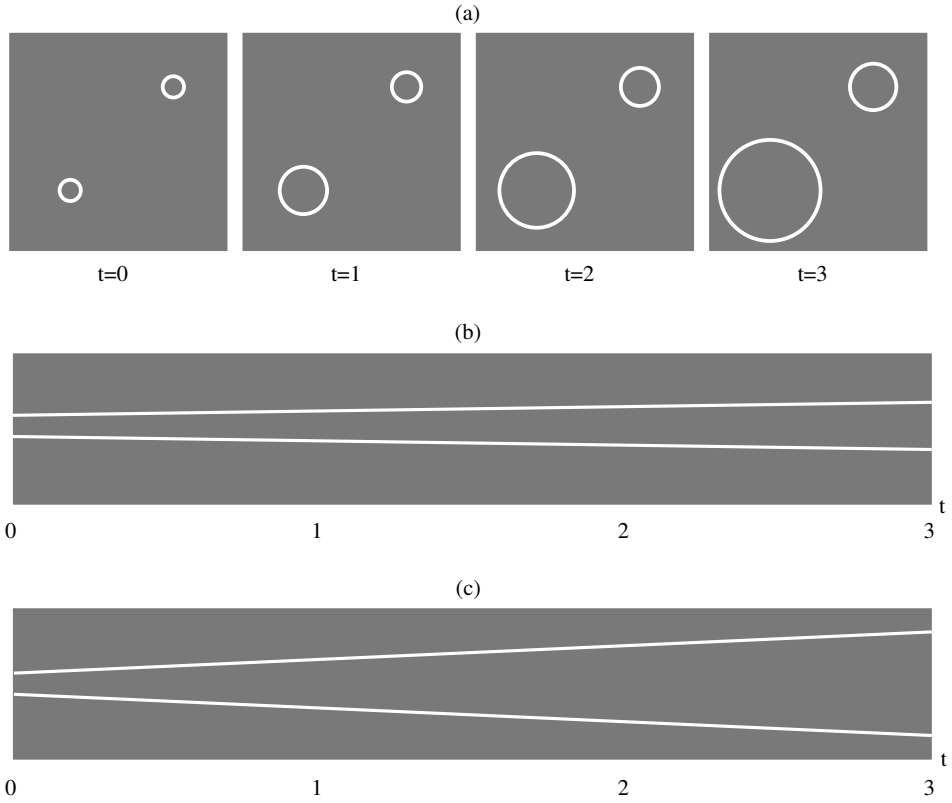


Figure 4.4: Example of infeasible spatiotemporal dynamics in local and space-invariant CNNs. Dynamics is not spatially homogeneous across the grid of cells. In (a), four snapshots of the network in different time instants. In (b) and (c), diagonal sections of the two waves shown in (a). Although both waves present, each, consistent temporal behaviour with constant growing rates individually, the spatial dynamics is not coherent because these rates, or the speeds of dynamics, are different.

ing, we define a cost function to tackle the problem of temporal feasibility. Spatially feasible training sets still need careful manual design.

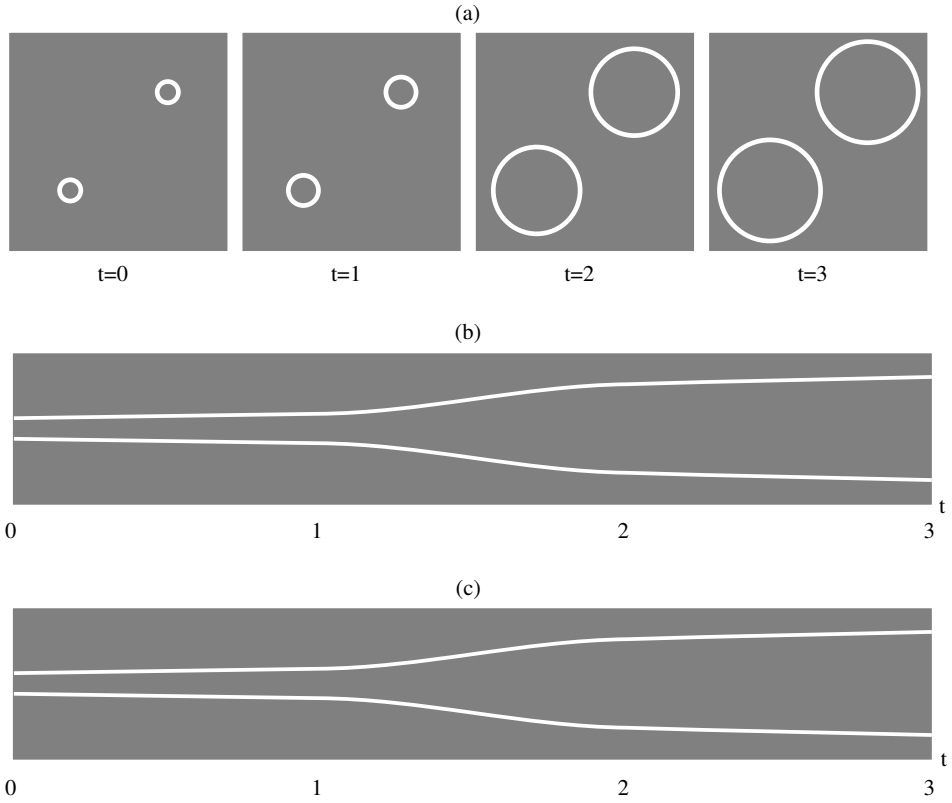


Figure 4.5: Example of infeasible spatiotemporal dynamics in local and space-invariant CNNs. Dynamics is not temporally homogeneous. In (a), four snapshots of the network in different time instants. In (b) and (c), diagonal sections of the two waves shown in (a). Both waves exhibit the same spatially distributed behaviour but the dynamics varies with time.

Cost function with relaxed temporal requirements

In order to improve temporal feasibility, we define here a cost function to avoid the necessity of a strict match between an irregular time evolution of the desired and resulting behaviour. This cost function assimilates the time intervals between snapshots of the system's output into the set of

parameters to be optimised. It takes the following form

$$E = \sum_{i,j} \sum_{k=1}^{N_T} (y^d_{i,j;k} - y_{i,j}(A, B, z, t_k))^2. \tag{4.15}$$

The problem of learning spatiotemporal behaviour with CNNs can be presented in this way as the minimisation of the cost function

$$\min_{A,B,z,\Delta t_1,\cdots,\Delta t_{N_T}} E(A, B, z, t_1, \cdots, t_{N_T}), \tag{4.16}$$

where $\Delta t_k = t_k - t_{k-1} \; \forall \; k = 0, \cdots, N_T$, representing the time interval between two output samples with N_T being a finite number of samples and $t_0 = 0$; $y^d_{i,j;k}$ denotes the desired output value of a pixel in the kth image of a given sequence of T images; the initial conditions $x(0)$ is defined in the training set; the value $y_{i,j}(A, B, z, t_k)$ denotes the output value of a pixel as the system has evolved to the time instant t_k with weight matrices A and B, and bias z.

This cost function has a few key differences with respect to (4.6) and (4.9) for allowing the learning of spatiotemporal dynamics and reduce temporal feasibility restrictions.

Compared to (4.6), (4.15) has an extra summation which accounts for the different snapshots of the temporal dynamics. Consequently, the period of the execution of the dynamics T is subdivided into N_T time instants $t_1, t_2, \cdots, t_{N_T}$. Each output $y_{i,j}$ correspond thus to the output of the network at time interval t_k, with $k = 1, 2, \cdots, N_T$. If the number of snapshots N_T is one, (4.15) is reduced to the case of fixed-point ordinary template learning of (4.6). Therefore, we conclude that (4.15) is a generalised cost function for learning of fixed-point or spatiotemporal dynamics with CNNs.

With respect to trajectory learning, when the cost function in (4.15) is compared to the cost function for trajectory learning with RNNs in (4.9), a time integral is replaced by a summation over N_T time instant snapshots. Additionally, the desired outputs $y^d_{i,j;k}$ in the CNN case are labelled by the index k of the N_T time instants rather than by the time instant t_k as it is for the measured outputs $y_{i,j}(t_k)$. These two measures are related to the desired relaxation of temporal requirements for physical feasibility. This relation becomes clear if we now look at the statement of the current learning problem, in (4.16). It can be noticed that the N_T time instants are not included in the set of parameters to be optimised but rather their local differences Δt_k are. The consequences of this are:

- Because Δt_k is optimised rather than t_k itself, only the order of appearance of the desired snapshots of the dynamics matters for the learning process.

- The desired outputs of the CNN network $y^d_{i,j;k}$ are not necessarily required to be regular in time because they do not depend on a matching time instant t_k.

- Because of these, the learning process also accepts temporally irregular image sequences as training set without compromising temporal feasibility. All the desired time instants t^d_k are irrelevant, only the order in which the images are learned matters, not at which specific time instants the images are generated by the CNN.

A way to exemplify the effect of these measures in view of trajectory learning is to think that if the trajectory to be trained is e.g. $\sin(\omega(t))$, with $\omega(t)$ being a monotonic increasing function of time. The resulting trajectory that is allowed to be learned is any frequency modulation of this behaviour, which also includes $\sin(t)$.

The relaxation of the schedule for the resulting spatiotemporal behaviour is an important issue for CNNs. Fixing a rigid schedule for the spatiotemporal trajectory to be learned would magnify the importance of a physically feasible training set. This needs to be avoided for a simple reason: in many cases nothing can guarantee that suggested time stamps for the desired spatiotemporal trajectory have a fixed relation between themselves in a real system.

The right choice for the number of time instants N_T to include in the optimisation depends on how difficult the learning problem is and on how much time and processing resources are available. Naturally, the higher the number of samples N_T of the dynamics, the more difficult the optimisation becomes. Yet, at the limit $N_T \to \infty$, RNN trajectory learning can be approximated by our approach for learning spatiotemporal dynamics with CNNs.

Learning spatiotemporal dynamics on multi-layer CNNs

The learning of complex behaviours in multi-layer CNNs, such as autowaves, can be done in the same way as for single-layer CNNs. Nevertheless, instead of one image sequence as output, multi-layer CNNs have multiple output

image sequences, which does not necessarily mean that all sequences need to be taken into account. It is possible to base the learning in one single desired output image sequence. This way no modifications are needed in the methodology described above. However, an extra output layer can serve as extra information that can be used by the optimisation process to find a global solution. In a 2-layer CNN for example, autowaves occur simultaneously in both layers but often with different waveforms. Frequently the outcome of what happens in one layer is sufficient for some applications and in this case only the output of this layer needs to be taken into account for the calculation of the cost function. However, the inclusion of the output of the second layer in the cost function calculation can sometimes bring more insight about the location of a globally optimal solution. For this case it is only necessary to include a summation over the number of layers in (4.15), and an index for the layers in the output and desired output values. A multi-layer version of (4.15) can therefore be represented as the following equation:

$$E = \sum_{i,j} \sum_{l} \sum_{k=1}^{N_T} (y^d_{i,j,l;k} - y_{i,j,l}(A, B, z, t_k))^2, \tag{4.17}$$

where the notation is identical to (4.15) with addition of l representing the layer index. In the case for the 2-layer model of (4.12), the layer indices assumes the values $l = 1, 2$.

Incremental learning

Learning of complex spatiotemporal dynamics can pose reasonably difficult problems. The more complex the behaviour the more intermediate trajectory steps need to be included in the training set. Since in this approach each extra desired dynamics snapshot in the training set leads to an extra parameter to be optimised, the more complex the behaviour is, the more unknowns there should be to be optimised. In order to avoid very long optimisation runs due to slow convergence, a few strategies are available. Generally these strategies concern different ways to perform *incremental learning*. In trajectory learning, this sort of learning is often used to gradually transform an existing and simpler trajectory into a different and more complex one by using intermediate target trajectories [130]. For an illustrative example of incremental learning applied to a given single-variable trajectory, see Figure 4.6.

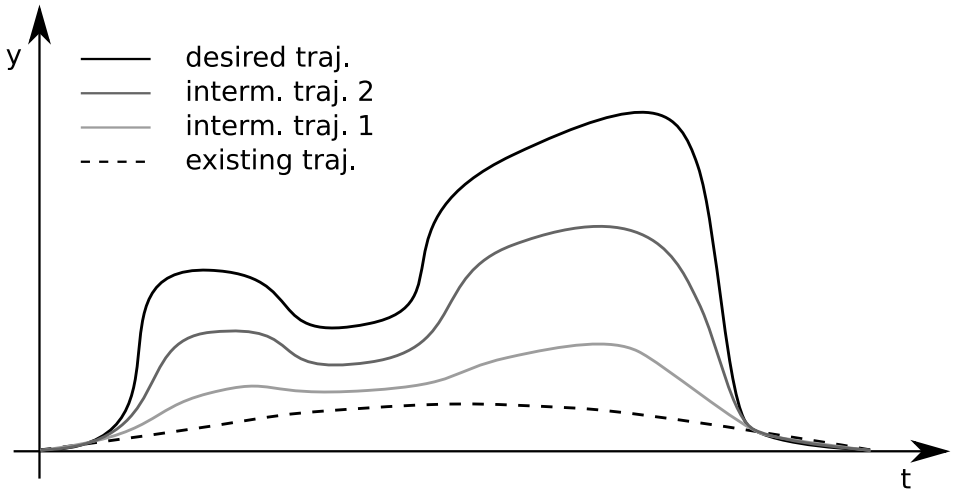


Figure 4.6: Example of incremental learning applied to a complete single-variable trajectory. An existing known trajectory is gradually transformed into the desired trajectory by using intermediate target desired trajectories.

For learning of spatiotemporal dynamics with CNNs, if there exist known template values for a dynamical behaviour that is similar to the desired dynamics, the same reasoning as in trajectory learning can be applied. If there exists no known similar behaviour however, other incremental learning strategies can still be used. As in Section 3.4 of Chapter 3, knowledge about the template form, e.g. symmetry, non-zero elements, etc., can be used to reduce the number of template elements to be optimised. For example, in the case of dynamics representing wave propagation in a given direction, one may wish to impose the assumption of symmetry along the direction of the propagation. In this way the number of parameters to be optimised is reduced and the optimisation process can converge faster. Once convergence is reached with this limited set of template values, the assumptions can gradually be removed in consecutive learning processes until all unknowns are included in the last process.

Another incremental learning strategy that can be applied here concerns temporal increments. While incremental learning using complete intermediate target trajectories is easy to visualise in the scalar case, when the trajectory is a state evolution of coupled cells disposed in a 2D array, the same principle does not become directly evident. If increments are given by

extending the trajectory gradually in time from the first snapshot until the last desired output, the strategy is not only evident for the single-variable trajectory case but it also is clearly the case for 2D spatiotemporal dynamics, *i.e.*, instead of the complete set of parameters, the learning process only has to consider the template values and the first time interval Δt_1 unknowns in the first optimisation. Subsequent processes gradually include more Δt's in the optimisation. By using this type of incremental approach to learn complex dynamics, we avoid clueless searches in very large search spaces by locating promising regions in smaller representations of these spaces. These promising regions serve then as clues for searching in wider spaces until we finally come to the complete search space representing the desired spatiotemporal dynamics.

4.3 Modifying speed of dynamics

The speed of dynamics or the amount of time necessary for a given dynamical system to reach a specified state will depend on its time constant. In trajectory learning, time constants are included in what are called *scale parameters* because of the nature of what happens in the system when these parameters are changed. The concept also holds for spatiotemporal dynamics.

In CNN systems, two systems with identical templates will reach a given state in a different amount of time depending on their time constants. The intrinsic time constants of CNN circuit implementations depends on the resistive, inductive, and capacitive values of the components. However, the time constant and consequently the speed of the dynamics of specific operations in programmable CNNs can also be modified by scaling the template parameters. Two CNN systems with the same intrinsic time constants τ can still present similar dynamical behaviour with different speeds if their template values are modified accordingly, without changing τ.

The methodology described in Section 4.2 can also be applied in combination with incremental learning strategies in order to modify the speed of dynamics of spatiotemporal behaviour with known template values. Varying the speed of existing CNN dynamics can be important for improving the efficiency of a large number of existing applications without performing any structural changes. Moreover, Varying the speed of these dynamics can also be crucial for the development of new AWC applications which

make use of CNN operations with accurate time evolution requirements, e.g. wave metrics [133].

Given existing CNN spatiotemporal dynamics, constraints can be applied in an incremental optimisation process to force the dynamical behaviour to gradually evolve faster, or slower. In order to formalise the problem of modifying the speed of dynamics of existing spatiotemporal behaviour, we describe the following problem

$$\begin{aligned} &\min_{A,B,z} E = \sum_{i,j} \sum_{k=1}^{N_T} (y^d_{i,j;k} - y_{i,j}(A, B, z, \omega t_k))^2 \\ &\text{subject to} \quad 0 < \omega \leq 1 - \Delta\tau, \end{aligned} \tag{4.18}$$

where the desired outputs $y^d_{i,j;k}$ for $k = 1, \cdots, N_T$ are simply generated from the existing template dynamics, i.e. $y^d_{i,j;k} = y_{i,j}(t_k)$. $\Delta\tau$ is the desired proportional increasing step on the speed of dynamics. In the case of a desired decrease on the speed of dynamics, the constraint on ω becomes $\omega \geq 1 + \Delta\tau$.

Incremental learning can best be applied here in case of large desired $\Delta\tau$, where the increment is applied to $\Delta\tau$ itself.

4.4 Simulations and on-chip experiments

A variety of experiments were performed to evaluate the method proposed here. The optimisation method used to minimise the cost function in (4.16) was Adaptive Simulated Annealing (ASA) [61]. We also have used this method for tuning fixed output templates for VLSI implementations as it is seen in Chapter 3. It can be observed that if we make $T = 1$ in (4.16) and t_1 is removed from the optimisation and made sufficiently long, this cost function is reduced to the fixed output case. The same approach of relaxation of constraints and search boundaries that we used in Chapter 3 can also be made useful for learning in the following way: (a) in the beginning of the learning process no limits are imposed to the weight values and thus the maximum range of values are available; (b) after this process converges, better solutions are obtained by limiting the weight values to values that are close to the first solution and/or incrementally relaxing existing constraints, e.g. symmetry, non-zero values, etc; (c) the last step is then repeated until any stopping criteria is reached.

We performed the experiments on the same set of examples presented in Section 4.1. Learning was performed on-chip, using the ACE4k CNN-UM [85], a chip with a grid of 64×64 cells; and in simulations. On-chip learning has two main advantages, which concern superior speed and the fact that no extra chip-specific optimisation is necessary to ensure the right functionality in the given chip. Simulations of the type of complex behaviour that CNN can exhibit are very expensive in respect to computational resources. Learning processes which requires a large number of function evaluation can become impractical to simulate even in modern digital computers. With on-chip cost function evaluations, the same processes only take a fraction of the time due to the high level of parallelism present in those chips.

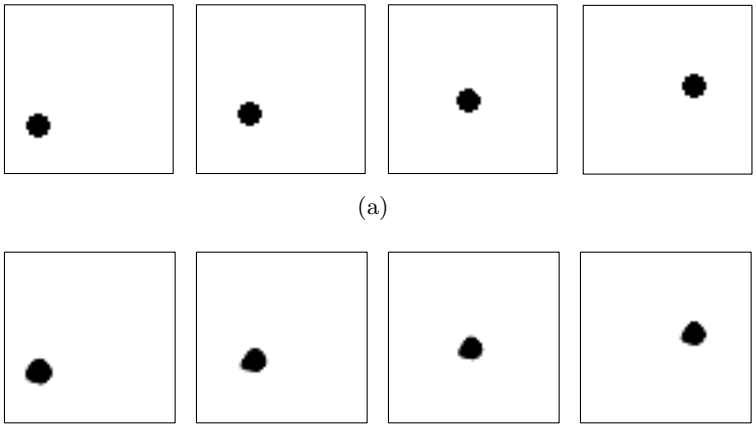
For all experiments realised here, only immediate neighbour cells are assumed to have a non-zero weight, which makes the matrices A,$B \in \mathbb{R}^{3\times3}$. We have used prior knowledge about the template matrices to reduce the number of parameters that actually need to be optimised. For example, when symmetry is considered, the number parameter to be optimised can go from 9 to 5 parameters for each matrix. When there existed no prior knowledge, the number of parameters to be optimised remained 19, respective to full template matrices, plus the number of time intervals N_T.

For simplification of the illustrative examples presented here, the input images $\mathbf{u}$ in (4.10) and (4.13) were set to zero, and therefore the input weight matrix B were assumed zero. The inclusion of input images and input weights in the optimisation process is straightforward but brings little or no complementary clarity to these examples.

In the following Sections we present the results of learning experiments realised on a VLSI CNN-UM chip, simulation learning of complex behaviour, and observations w.r.t. change in the speed of dynamics during the learning process.

4.4.1 On-chip learning experiments

We performed five learning experiments with aperiodic spatiotemporal behaviour in the ACE4k CNN-UM chip [85], among which three were divergent and two convergent spatiotemporal dynamics. Next, we present the results for the divergent dynamics experiments, namely a travel dot, a travelling wave and a combustion wave; then we present the results for the two convergent dynamics: two pyramiding waves.



(a)

(b)

Figure 4.7: Results of learning on a ACE4k (64×64 cells) chip for a dot travelling at a given angle, where (a) is the desired dynamics and (b) is the result of the learning. Although the the travelling angle was learned, the shape of the circular dot was not preserved.

Travelling dot

In this experiment we tried to train the chip with the spatiotemporal behaviour described by a dot travelling in an image sequence. We used the manually generated images from Figure 4.1(a) as initial condition and training set. The first image in the sequence from Figure 4.1(a) was used as the constant input $\mathbf{u}$ and initial state $\mathbf{x}(t = 0)$. Although very simple to picture, this travelling is relatively difficult to train. Due to the angle of the shifting, no assumptions, e.g. symmetry, could be taken to reduce the number of template values to be optimised. Temporal incremental learning was applied here to reduce the initial search space. The resulting spatiotemporal dynamical behaviour after learning can be visually compared to the original one in Figure 4.7. It can be observed that although the direction of the movement could be accurately learned, the contour of the dot is not fully accurate. This can indicate a physical limitation of the chip or the CNN model to shift objects in arbitrary directions while maintaining their shapes.

(a)

(b)

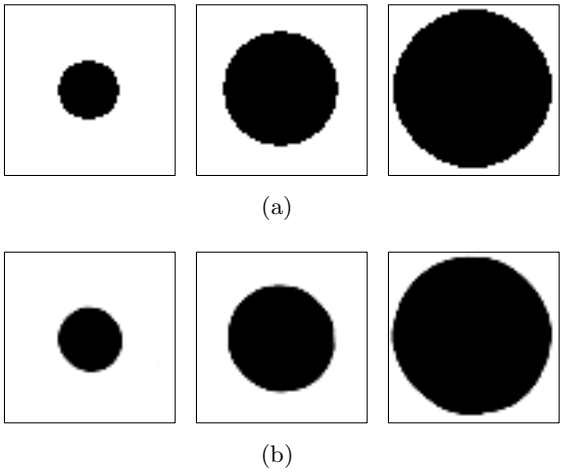
Figure 4.8: Results of learning on a ACE4k (64×64 cells) chip for a wave travelling in the second diagonal direction, where (a) is the desired dynamics and (b) is the result of the learning. The network was able to learn the travelling wave but with irregular width.

Travelling wave

The objective of this experiment was to train the chip to propagate a wave with fixed length across the array of cells in a diagonal direction. We used the symmetry along the second diagonal of the array to reduce the number of initial unknown template values. Although the network was able to learn a travelling wave, it failed to match the shape of the desired wave. The resulting behaviour is a travelling wave with varying width along the wave front. Figure 4.8 presents the results of the learned behaviour and the respective training set of images, originated from Figure 4.1(b). Initial condition, input, and initial state images were set in the same way as for the travelling dot.

Combustion wave

Combustion waves are quite common natural phenomena. Here, we trained the ACE4k chip to simulate a combustion wave which starts in the centre of the array of cells and "burns" outward homogeneously. The resulting



(a)

(b)

Figure 4.9: Results for the learning of an outward-burning combustion wave on a ACE4k (64×64 cells) chip, where (a) is the desired dynamics and (b) is the result of the learning.

behaviour is very similar to the desired dynamics. Figure 4.9 presents the resulting dynamics and the desired images used in the training set. Initial state and input images were set to the first image in the sequence in Figure 4.2(b).

Pyramiding waves

We now present the resulting dynamics delivered by the learning of two "convergent" propagating waves. The dynamics are described by pyramiding waves: one which propagates vertically to form a standing pyramid, see Figure 4.10; and another which propagated at a given angle to form a rotated pyramid, see Figure 4.11. Both dynamics were reasonably well learned by the network.

4.4.2 Learning of complex dynamics: a spiral autowave

We also applied the methodology described in this Chapter to train complex spatiotemporal behaviour. For that, we simulated a second order two-layer CNN described in (4.13). In this experiment, we used the spiral autowave

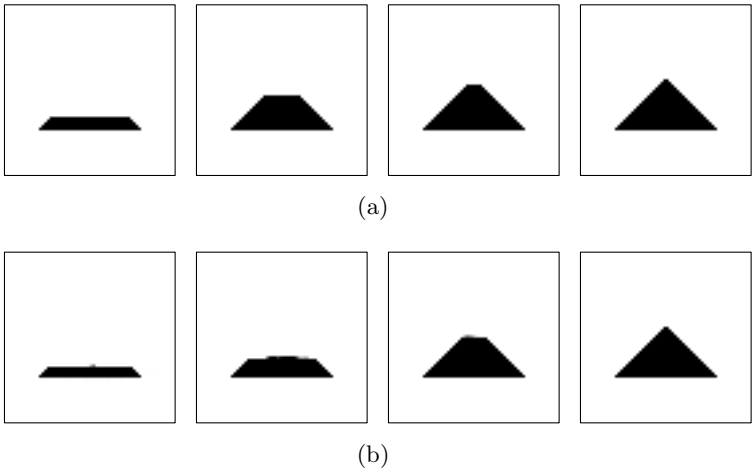


Figure 4.10: Resulting dynamics for the learning on chip of a "convergent" propagating wave, where (a) is the desired dynamics and (b) is the result of the learning. The wave grows to form a standing pyramid from a horizontal line.

example.

Initial conditions are an important aspect to consider when trying to generate autowaves. For the 2-layer system used here, the initial state of the second layer was set to be an inverted image of the first layer shifted one or two pixels in the direction of the desired propagation. This is a very simple way to generate initial conditions for autowaves but is not the only one, see e.g. [11] for another procedure.

The images in Figure 4.3 were used as training set for the experiment with autowaves. These images were obtained by a CNN simulator to avoid manual generation, which could lead to impractical and physically infeasible behaviour. Although the results shown in Figure 4.12 were also obtained by performing the training in simulation, the final template was obtained without any prior knowledge of the original. The same procedure could also be used to train a CACE1k CNN-UM chip [20]. The objective here is to demonstrate the effectiveness of the methodology described in this Chapter for learning of autowaves in CNNs. It can be seen that there is a good generalisation of outputs further in time that were not used for the

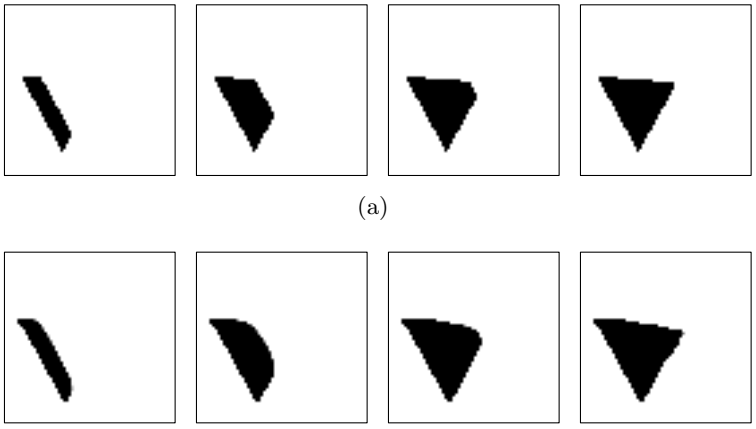


Figure 4.11: Resulting dynamics for the learning on chip of a "convergent" propagating wave, where (a) is the desired dynamics and (b) is the result of the learning. The wave grows from a line of pixels to form a rotated pyramid.

learning process.

4.4.3 Change on the speed of dynamics

In Section 4.3 we presented a method to change the speed of existing dynamics. Although we have not performed a specific experiment using this method, we can demonstrate that changing the speed of dynamics is possible by only modifying the template values. For that, we show in Figure 4.13 the evolution further in time of the same desired and learned dynamics presented in Figure 4.12. To improve visualisation we also execute the dynamics on a larger grid of cells, 64×64. The learned dynamics is about 20% faster than the desired one. It is important to notice that no constraints regarding a desired speed increase were applied to the learning procedure. The observed increase emerged naturally from the learning process itself in the same way that a speed decrease could have emerged.

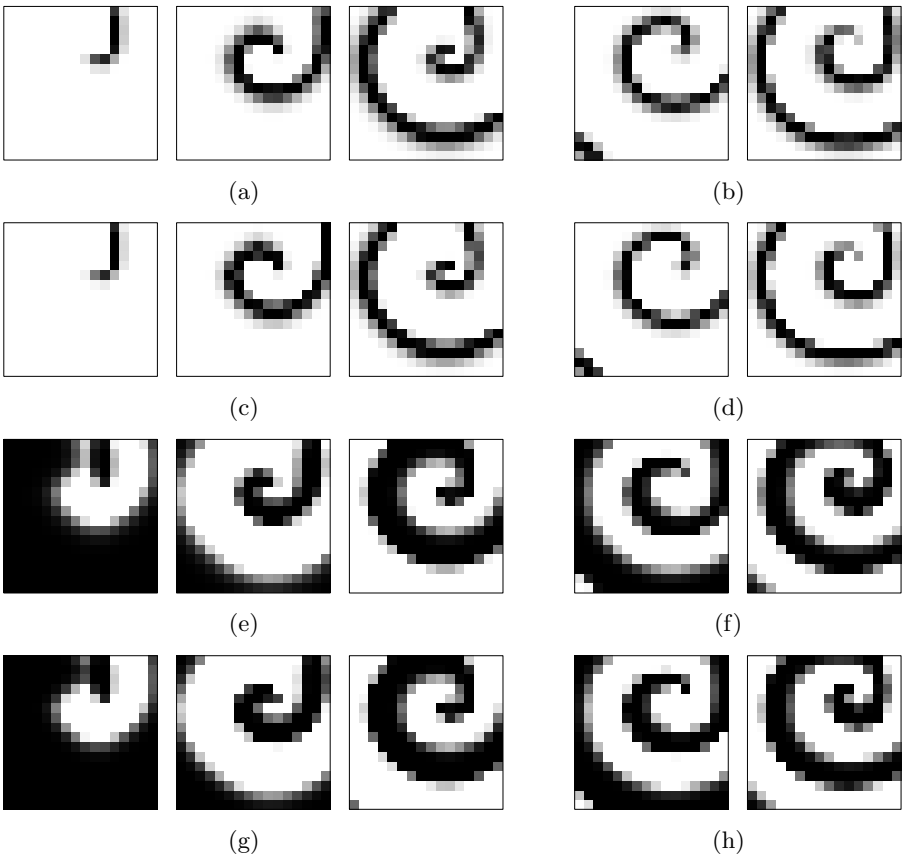
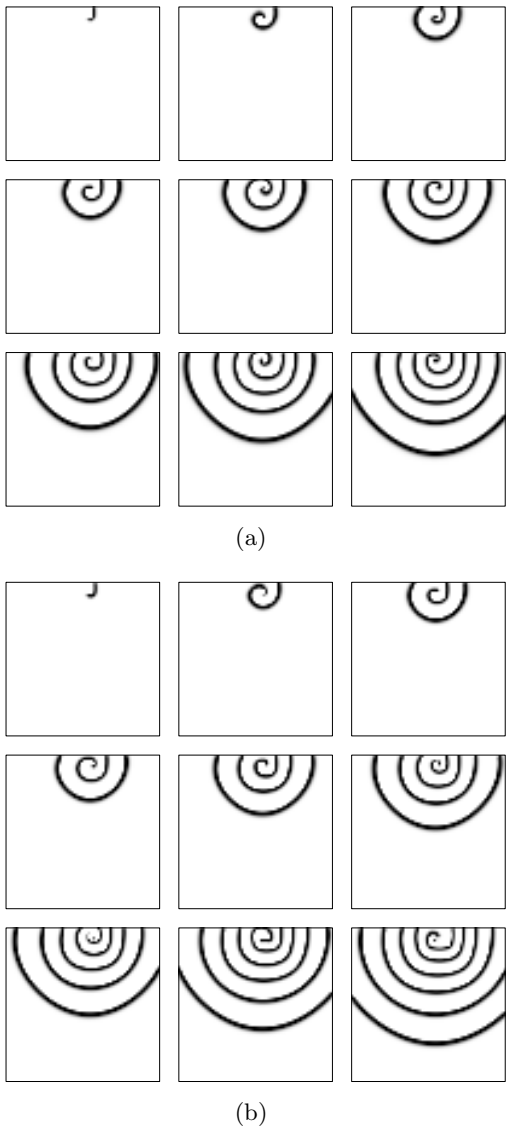


Figure 4.12: Results for the learning of a spiral autowave in a 16×16 array of cells in simulation. The training set is presented in (a) for the first and (b) for the second layer; snapshots (c) and (g) present the resulting behaviour. Images (d) and (h) present the generalisation of the learned behaviour in the future for the snapshots (b) and (f), which were not used in the training set.



(a)

(b)

Figure 4.13: Evolution spiral autowave for the original (a) and trained (b) templates on a 64×64 CNN grid. The images represent snapshots taken in equivalent time instants. Generalisation further in time and change in speed can be clearly seen.

4.5 Conclusion

This Chapter approached the problem of learning 2-dimensional spatiotemporal behaviour with cellular neural networks. Trajectory learning with recurrent neural networks can be seen as a starting point for the formulation of the problem. It was shown that although trajectory learning with RNNs and learning of spatiotemporal behaviour with CNNs have many elements in common, two key points distinct the two problems: (a) due to the locality and space invariance of CNN weights, the number of parameters to be optimised in these networks is much smaller and thus learning is considerably easier favouring the use of global optimisation methods to avoid the need of an expression for the gradient of the cost function; and (b) the generation of an efficient training set for the CNN problem is not straightforward as in classical trajectory learning and thus customised solutions need to be devised to ensure a feasible training. Taking into account these two points, a cost function and methodology was presented for learning of spatiotemporal behaviour in CNNs. This cost function also assimilates time intervals as parameters to be optimised. This reduces the importance of generating temporally feasible training sets. Another method presented here concerns the optimisation of existing template values to ensure that the desired behaviour has an increase or decrease in speed of the dynamics. With this method, existing CNN applications can benefit from faster execution by moving the speed of template operations to its limit. Results for the learning of different examples of spatiotemporal behaviour were presented with experiments made in simulations and CNN chip implementations. These results show that qualitatively good operations can be learned on-chip as well as in simulation.

Chapter 5

Coupled Simulated Annealing

In the two Chapters preceding this one we have presented two general CNN optimisation methodologies that are inherently dependent on an optimisation core. This core is responsible for generating probing solutions and selecting the promising ones accordingly with their cost evaluation. Although we have used what perhaps is the most robust Simulated Annealing (SA) algorithm existing at the time as the core for these methodologies, convergence to reasonable results have not been trivial and reruns have had to happen very often. At a certain point, we have realised that instead of running and rerunning an optimisation several times, perhaps it would be better to cast several optimisation runs at once. In fact, this idea evolved to a concept that is in the very heart of cellular neural networks: coupling. Coupling had already been used in an approach called coupled local minimisers (CLM) to steer gradient based methods in order to outperform multi-start approaches. CLM can be seen as a CNN where cells correspond to local optimisation algorithms and the couplings correspond to synchronisation constrains.

In this Chapter we define the class of global optimisation methods called Coupled Simulated Annealing (CSA). As CLM, CSA also uses coupling to create cooperative behaviour among parallel optimisation processes in order to reach the global optimum more efficiently. Instead of local gradient based optimisers like in CLM, in CSA we have global SA optimisers working in parallel and exchanging information through coupling. Although this

approach was developed to become the optimisation core of our CNN optimisation approaches, it can also be seen as a general purpose distributed global optimisation approach. Therefore it is suitable to other optimisation problems that share the same features from CNN optimisation problems, *i.e.* hard problems that are multi-modal and multidimensional with many local optima. For the sake of generality, this Chapter does not pertain the specifics of CNN optimisation but is rather focused on describing a general purpose distributed global optimisation class of methods.

5.1 Diversity in optimisation

Optimisation is a very active field of research. Different techniques exist for different purposes. Problems in optimisation can be characterised by several different classes. Roughly, we can group problems in two main classes, numerical and combinatorial optimisation, which can be associated with optimisation of continuous and discrete sets, respectively. Protein structure prediction, travelling salesman problem, spin glasses, quadratic assignment problem, etc. are examples of problems in the domain of combinatorics. Safety engineering, aerodynamics, control, circuit design, CNN optimisation, etc. are typical examples of problems that are often defined in the continuous space. Numerical optimisation problems are spread among many fields and include both the academic and industrial world. They are characterised by a continuous search space, which can be constrained or not, where every solution is represented by a point in this space. The best solution-method for such problems in general depend persistently on the problem itself. In this thesis we are concerned only with numerical optimisation.

5.1.1 Numerical optimisation

Within the class of numerical optimisation problems, we can make a clear distinction between convex and non-convex problems. In fact, most of the studies in convex optimisation are nowadays concerned with the formulation of the given problem itself [16]. The goal of optimisation modelling in the study of convex optimisation is to recast given non-convex problems into well defined convex ones. Given that a problem is convex, its solution is trivial to be found using one of the many existing solvers, or simply using

a gradient descent method. The problems that are not convex or cannot be transformed or translated into a convex framework are characterised as non-convex.

Numerical optimisation problems can also be divided into constrained or unconstrained. This classification is not mutually exclusive with the division between convex and non-convex problems. In fact, convex problems can be constrained or unconstrained as well as the non-convex ones. It is possible to keep subdividing non-convex problems into subclasses; however the boundaries are many times much more subjective than in the previous cases.

5.1.2 Defining the target problems

In this Chapter, we define a class of methods that can be used to solve unconstrained numerical optimisation problems that are naturally non-convex. The existing methods used to solve non-convex problems are vast. When a problem is differentiable, gradient descent can be applied like in convex optimisation, but in this case, no guarantees can be given in general about global optimality. Nevertheless, for some classes of non-convex problems, variations of these methods can perform surprisingly good. Coupled Local Minimisers (CLM), for example, is a technique inspired by CNNs which has multiple gradient descent optimisers as the cells of a network that are coupled by synchronisation constraints [132, 131]. Nevertheless, for a large part of real-life non-convex problems, gradient based procedures can not be applied due to the lack of cost function derivatives or to the huge computational cost of some large scale problems. Moreover, for some challenging problems, gradient techniques are less suitable to be applied due to multi-modality, multidimensionality and/or presence of many local minima.

Many global optimisation techniques, which are often based on heuristics, were developed in order to provide alternative solution-methods to solve multi-modal and multidimensional problems. Examples includes: simulated annealing, genetic algorithms, and particle swarm optimisation. While being able to escape from multiple local minima, the strongest drawback of these methods is the large number of cost function evaluations required to reach the basin of the global optimum solution. This issue has been undertaken by the development of many fast global optimisation techniques where local and global procedures are often combined in order

to obtain a faster convergence [60, 160]. Not surprisingly, faster convergence introduces a trade-off between speed and quality of solutions. Fast global techniques become more frequently trapped in poor local minima. A challenge of this field is to find a good trade-off for the given optimisation problem. The methodologies defined in this Chapter to solve unconstrained non-convex numerical optimisation problems is, therefore, targeted at the portion of these problems that are considered to be difficult, namely, multi-modal and multidimensional problems with many local optima. Additionally, we look into the problem of robust initialisation conditions. The methods described here are based on the idea of joining together simulated annealing and CLM.

5.2 Simulated annealing

Simulated Annealing (SA) is a technique among several other global optimisation approaches designed to solve difficult non-convex problems. It is originally based on the thermodynamic annealing process, which consists on heating up a metal, glass, or crystal, holding its temperature and then cooling it in a very slow rate. This physical-chemical process gives as result high quality materials [70]. The analogy with an optimisation procedure comes by the following relations:

$$\begin{aligned} \text{Physical material states} &\rightarrow \text{Problem solutions} \\ \text{Energy of a state} &\rightarrow \text{Cost of a solution} \\ \text{Temperature} &\rightarrow \text{Control parameter.} \end{aligned}$$

Physical annealing is modelled or simulated in software by Monte Carlo techniques resulting in an efficient computational algorithm that is widely used nowadays to optimise many different problems [77, 156, 93]. This algorithm is basically composed of two stochastic processes. One is responsible for the generation and the other for the acceptance of solutions. Both processes are controlled by a temperature value. The temperature can be the same for both processes, but usually there are independent temperatures for the generation and the acceptance process. As in physical annealing, this temperature must follow an annealing schedule.

There are two temperature schedules to consider when designing a SA algorithm, which are the generation and the acceptance schedule. The generation temperature is responsible for the correlation between generated

probing solutions and the current, or original, one. Generally, probing solutions are obtained by the addition of a random vector ε, of the same size as the solution vectors, to the vector representing the current solution. A variation in the generation temperature modifies the distribution from where ε is obtained. Namely, an increase in this temperature represents a widening of the distribution, whereas a decrease causes the distribution to become narrower. The right distribution is the one which fits best with the generation temperature schedule. There exist many convergence proofs which pair different temperature schedules with the right distribution. Table 5.1 presents many of these pairs.

The temperature schedules used for generation can also be used for acceptance. The acceptance temperature weights the difference between a probing solution and the current one. Fixing this difference, the higher this temperature, the larger the probability that an uphill move is accepted. When this temperature becomes lower, the probability becomes smaller. Therefore, early in the optimisation, many uphill moves are accepted, and with the evolution of the process, less and less uphill moves are allowed, while close to the end, almost no uphill moves are accepted. Such approach permits an extensive exploration of the cost function at the beginning and a gradually more localised search when the end is approaching.

There exist many forms for the acceptance function. Among the most common are the Metropolis rule:

$$A(x \to y) = \exp\left(\frac{E(y) - E(x)}{T_k^{ac}}\right), \tag{5.1}$$

and the following expression:

$$A(x \to y) = \frac{1}{1 + \exp\left(\dfrac{E(y) - E(x)}{T_k^{ac}}\right)}, \tag{5.2}$$

where $A(x \to y)$ is the probability that the probing solution y is accepted considering that x is the current solution. $E(\cdot)$ is the energy of the given solution, while T_k^{ac} denotes the acceptance temperature at iteration k.

In Figure 5.1 the reader can find a simplified flowchart of a classical SA algorithm depicting data and program flows. Observe that the temperatures are only updated once the *equilibrium criterion* is met. The objective

Reference	Generation Distribution	Temperature Schedule
[45]	$g_k(\varepsilon, T_k) = (2\pi T_k)^{-\frac{D}{2}} \exp\left[\frac{-\varepsilon^2}{2T_k}\right]$	$T_k = \frac{T_0}{\ln(k+1)}$
[136]	$g_k(\varepsilon, T_k) = \frac{T_k}{\left(\varepsilon^2 + T_k^2\right)^{\frac{D+1}{2}}}$	$T_k = \frac{T_0}{k+1}$
[60]	$g_k(z_n, T_{k,n}) = \frac{1}{2\left(\lvert z_n\rvert + T_{k,n}\right)\ln\left(1 + 1/T_{k,n}\right)}$	$T_{k,n} = \frac{T_{0,n}}{\exp(b_n k^{\frac{1}{D}})}$
[160]	$g_k(z_n, T_{k,n}) = \frac{1}{2\left(\lvert z_n\rvert + \frac{1}{\ln\left(1/T_{k,n}\right)}\right)\ln\left(1 + \ln\left(1/T_{k,n}\right)\right)}$	$T_{k,n} = \frac{T_{0,n}}{\exp(\exp(b_n k^{\frac{1}{D}}))}$

Table 5.1: Overview of pairs of generation distribution and temperature schedule with convergence proofs for SA algorithms. For [45] and [136], the probing solution $\mathbf{y} = [y_1, y_2, \cdots, y_n, \cdots, y_D]$ is obtained by adding ε to the current solution $\mathbf{x}$, where ε is chosen randomly from the respective generation distribution $g_k(\varepsilon, T_k)$, *i.e.* $y_n = x_n + \varepsilon_n$. For [60] and [160], the probing solution $\mathbf{y}$ is obtained by generating each of its vector components individually by $y_n = x_n + z_n(B_n - A_n)$, where A_n and B_n are the individual lower and upper bounds of each vector component, and z_n is obtained from the corresponding generation distribution $g_k(z_n, T_{k,n})$, with $b_n > 0$ denoting a constant parameter.

of such criterion is to wait enough iterations until there is no or little variation in the energy of the accepted solutions, which means that the equilibrium was reached. An example of a straightforward criterion is to wait a fixed number of iterations N. Theoretically, every proof of convergence for SA assumes $N \to \infty$, which is practically impossible to reach. Therefore, in practice a reasonable N is chosen by more elaborated techniques which take into account the variance of the accepted solutions; or simply by setting a maximum N according to the available physical resources, like time and computational power.

Figure 5.1 can be also explained by the following algorithm:

Algorithm 1

1. *Initialisation:* assign a random initial solution to x; assess its cost $E(x)$; set the initial temperatures $T_k = T_0$ and $T_k^{ac} = T_0^{ac}$; set the time index $k = 0$.

2. *Generate* a probing solution y according to $y = x + \varepsilon$, where ε is a random variable sampled from a given distribution $g(\varepsilon, T_k)$; assess the cost for the new probing solution $E(y)$.

3. *Accept* solution y with probability 1 if $E(y) \leqslant E(x)$, otherwise with probability $A(x \to y)$, *i.e.* make $x := y$ only if $A > r$, where r is a random variable sampled from a uniform distribution [0,1]; *go to step 2* for N inner iterations (equilibrium criterion).

4. *Decrease temperatures* according to schedules $U(T_k, k)$, and $V(T_k^{ac}, k)$; increment k.

5. *Stop* if stopping criterion is met, *otherwise go to step 2*,

where T_k and T_k^{ac} are the generation and acceptance temperature parameters at time instant k, respectively. The function $g(\varepsilon, T_k)$ is the generation distribution and $A(x \to y)$ is the probability of accepting the solution y, given that the current solution of the system is x, with $x, y \in \Omega$, where Ω denotes the set of all possible solutions.

In past years, several SA versions were developed [105, 60, 127, 160]. Every version introduces a different level of trade-off between speed of convergence and quality of solution. Some methods achieve extra levels of speed of convergence by introducing parallelism into the originally strictly

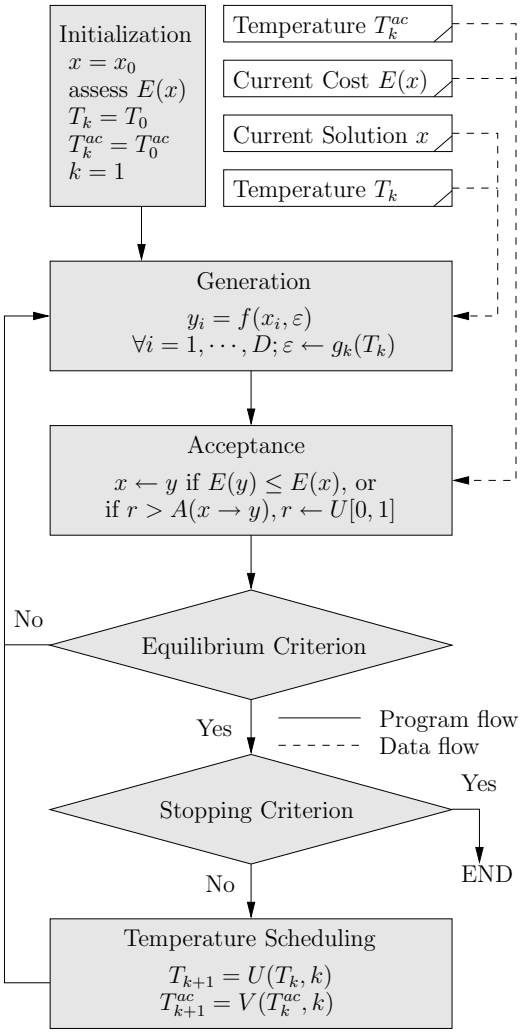
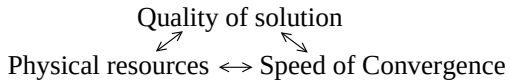


Figure 5.1: Flowchart of a typical SA process. Full lines represent the program flow, whereas dashed lines represent the data flow, with x denoting the current solution, T_k and T_k^{ac} denoting the temperatures at iteration k, and T_0 and T_0^{ac} denoting the respective initial temperatures. $E(\cdot)$ is the energy function, D is the problem dimension, and $U(T_k, k)$ and $V(T_k^{ac}, k)$ are the generation and acceptance temperature schedules, respectively.

sequential SA algorithm [89, 13, 22, 77, 121, 141]. This of course at the cost of more physical resources. The two-side trade-off then becomes a three-side one:



There are many classifications made for the different types of Parallel SA algorithms [79, 97, 13, 141]. It is not the purpose of this thesis to classify the many existing versions. The reader may refer to [141] for a recent discussion. However, as far as it is relevant for this scope, it can be said that there exists a more general classification concerning *non-sequential* algorithms which categorise Ensemble [121] and Parallel SA algorithms.

In **Ensemble SA**, a group of SA processes move in the search space without interaction between solutions. The ensemble moves as a whole and the decision to make a move is taken based on the average cost of the group.

In **Parallel SA**, the concurrent processes move independently with the objective of minimising the cost of the best solution, which is shared in one way or another among the processes either synchronously or asynchronously.

In this Chapter, we present a new class of algorithms, called Coupled Simulated Annealing (CSA), that can not be fully described by Ensemble SA neither by Parallel SA. In fact, CSA may be considered as a mixture of Ensemble and Parallel SA. The main principles are discussed in the next Section.

5.3 Cooperative behaviour and global optimality

Global optimisation methods originally are very slow. For many difficult problems, ensuring convergence to a global optimum might mean impractical running times, even for state-of-the-art digital computers. For such problems, a reasonable solution might be enough in exchange for a faster convergence. Precisely for this reason, many SA algorithms [105, 60, 127, 160] and other techniques based on heuristics have been developed. However, due to speed-up procedures, these methods often get trapped in poor optima.

The focus of developing accelerated global optimisation methods mainly seems to have been on increasing the speed of convergence, rather than improving the quality of the final solution. For this reason, optimising a certain cost function requires many times multiple attempts with a variety of different initialisation conditions. Consequently, what *a priori* seemed to be a decrease of convergence speed, might be counterbalanced by possible excessive number of different initialisation attempts that are necessary in order to achieve a certain level of quality.

The class of CSA methods presented in this chapter is designed to be able to easily escape from local optima and thus improve the quality of solution without compromising too much the speed of convergence. To better understand the underlying principles of the class of methods presented in this Chapter, consider the work of Suykens *et al.* [131]. They have shown that coupling among local optimisation processes can be used to help gradient optimisation methods to escape from local optima in non-convex problems. Here, with the objective of increasing the quality of the final solution, we present the use of coupling in a global optimisation method like SA. Additionally, by designing a coupling mechanism with minimal communication, these coupling algorithms can be implemented very efficiently in parallel computer architectures, making them very appealing to the multi-core trend in the new generation of computer architectures.

CSA introduces a new formalism for the acceptance probability functions. The idea is to allow several concurrent SA processes with acceptance probability defined by a function that depends on a coupling term. This term is defined by a function of all energies of the current states in each concurrent process. The form of the acceptance function and the coupling term attached to it define the type of coupling. Like in Ensemble SA, there exist no iterations between the solutions; furthermore, like in Parallel SA, the different optimisation processes move independently, but under the influence of coupling in the acceptance probability function.

5.4 CSA: general principles

Importance Sampling, the main principle underlying classical SA, has been used in statistical physics to selectively, rather than randomly, choose sample states of a particle system model in order to efficiently estimate some physical quantity related to the system. Random sampling of these states

turned out to be very inefficient because in these systems only a few low-energy states carry most of the relevant information. Importance sampling cares for a rejection/acceptance mechanism of sampled states in order to probabilistically favour states with lower energies. The well known Metropolis algorithm was the first to use the idea to estimate these quantities efficiently. Also, this algorithm complies with the principle of *Detailed Balance*[1], which gives a sufficient condition to test the validity of Monte Carlo schemes. In terms of a master equation of a thermodynamic system, it states that

$$\frac{P(x \to y)}{P(y \to x)} = \frac{\exp(-E(y)/T)}{\exp(-E(x)/T)}, \tag{5.3}$$

where $P(x \to y)$ is the transition probability for the system to go from the current state x to a candidate state y, $\forall\ P(y \to x) \neq 0$, with T being a fixed temperature and the quantities $E(x)$ and $E(y)$ denoting the energy of the states x and y, respectively. Transition probabilities can be subdivided into the product of a generation or selection probability and an acceptance probability, i.e. $P(x \to y) = G(x \to y)A(x \to y)$ with G and A denoting generation and acceptance probabilities, respectively. If G is chosen to be equally alike for all states, *i.e.* $G = 1/n$, with n denoting the number of all possible states, (5.3) can be reduced to

$$\frac{A(x \to y)}{A(y \to x)} = \frac{\exp(-E(y)/T)}{\exp(-E(x)/T)}. \tag{5.4}$$

Many acceptance probability functions for SA were derived according to (5.4), including the Metropolis rule (5.1) and (5.2). We use the latter one in Section 5.4.3 to exemplify the class of CSA methods. Observe that since the probing state y is randomly chosen, the only information about the status of the system that is taken into account when deciding whether to accept or not the new state with (5.2) is the energy of the current state $E(x)$.

CSA features a new form of acceptance probabilities functions that can be applied to an ensemble of optimisers. This approach considers several current states which are coupled together by their energies in their acceptance function. Also, as opposite of classical SA techniques, parallelism is

[1]Intuitively, Detailed Balance ensures that the balance between the probability of 'leaving' a given state and arriving in it from another state holds overall and individually for any pair of states.

an inherent characteristic of this class of methods. The motivation to create coupled acceptance functions which comprise the energy of many current states, or solutions, is in fact to generate more information when deciding to accept less favourable solutions in a global optimisation process. Moreover, it can also be observed that this class of acceptance functions can be in fact a generalisation of existing SA acceptance functions.

5.4.1 A formal definition for CSA

In CSA, each optimisation process, *i.e.* the algorithmic steps involving generation and acceptance of a single current state, is performed separately. This process behaves for each current state as a single classical SA process. In fact, the only difference between such a process and a SA process is held on the acceptance probability. While in SA this probability is a scalar function, $0 \leq A(x \rightarrow y) \leq 1$, for every $x, y \in \Omega$, with Ω denoting the set of all possible states, in CSA it is a scalar function according to

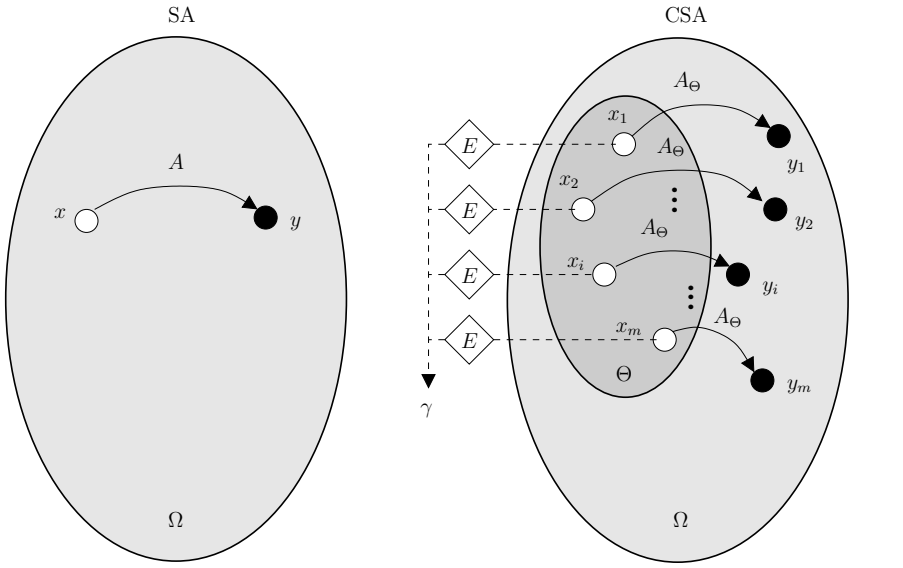
$$0 \leq A_{\Theta}(\gamma, x_i \rightarrow y_i) \leq 1, \tag{5.5}$$

for every $x_i \in \Theta$, $y_i \in \Omega$, and $\Theta \subset \Omega$, with x_i and y_i being current and probing states, respectively, for every $i = 1, \cdots, m$, with m being the number of elements in Θ. The set Θ, is presented as the set of current states and is defined as $\Theta \equiv \{x_i, i = 1, \cdots, m\}$ throughout the Chapter. Given one of its elements, the decision about swapping the element with a probing state which is outside of the set Θ depends on the given element, on the probing state, and also on the coupling term γ, which is a function of the energy of the elements in Θ,

$$\gamma = f\left[E(x_1), E(x_2), \cdots, E(x_m)\right]. \tag{5.6}$$

In summary, in order to identify a method belonging to the CSA class, this method need to comply with both (5.5) and (5.6). The general difference between classical SA and CSA acceptance processes is illustrated in Figure 5.2.

Like in classical SA, in CSA an acceptance probability function can assume different forms. Besides what was mentioned above, these functions also need to inherit specific properties where the most desirable one is the steering of the states to low-energy regions. Compliance to detailed balance is also one of the most common desired features for SA acceptance functions,



$0 \leq A(x \to y) \leq 1,$
$\forall\, x, y \in \Omega.$

$0 \leq A_\Theta(\gamma, x_i \to y_i) \leq 1,\ \forall\, x \in \Theta, y \in \Omega,$ with
$\Theta \subset \Omega,$ and $\gamma = f\left[E(x_1), E(x_2), \cdots, E(x_m)\right].$

Figure 5.2: The general difference between SA and CSA lies in the acceptance process. While SA only considers the current solution x for the acceptance decision of the probing state y, CSA considers many current states in the set Θ, which is a subset of all possible solutions Ω, and accepts each probing state y_i based not only on the corresponding current state x_i but by considering also the coupling term γ, which depends on the energy of all other elements of Θ.

especially when the target of the simulation is the evaluation of statistical properties of particle systems. For optimisation, however, this property may not be essential.

5.4.2 The role of the acceptance temperature in CSA

Good solutions in CSA methods are more likely to be preserved, while poor solutions are easily swapped, resulting in a safe and intense exploration of the energy surface. However, like in any other SA-like method, appropriate temperature schedules are fundamental. The challenge is to find the right schedule for the problem at hand. Temperature in CSA methods has a

slightly different functionality compared to SA. In classical SA processes the acceptance temperature is responsible for weighting the difference between the probe and the current solutions in order to affect the decision of accepting uphill moves. Initially, these moves happen more often. Gradually, according to a certain temperature schedule, they are more seldomly accepted. In CSA, the temperature does not serve to weight this difference, but to weight the proportion that each acceptance probability has in the overall sum of probabilities. Very high temperatures cause all the acceptance probabilities to be equal, while very low temperatures result in only one process with probability equating 1 with all the other probabilities equating 0. Both cases are obviously of little help, which makes an appropriate temperature schedule as important as for classical SA.

For the purpose of this thesis, we have used the temperature schedule in [136] for the generation and acceptance processes. However, for the acceptance temperature, we found out that the coupling applied to CSA-M permits the use of an efficient approach for the schedule of this temperature. Namely, this temperature is used to control the variance of the acceptance probabilities. More details on this on Section 5.6.

5.4.3 A CSA generalisation of SA

In this Section we show how a specific SA acceptance probability function, precisely (5.2), can be generalised as a CSA acceptance function. Since SA and CSA processes only differ in the acceptance function, we omit here any unnecessary reference to other parts of the algorithms in order to prove the generalisation of this specific SA algorithm by CSA.

If we multiply both numerator and denominator of the right hand side in (5.2) with $\exp(\frac{-E(y)}{T_k})$, it results in the following equivalent equation:

$$A(x \to y) = \frac{\exp\left(\frac{-E(y)}{T_k}\right)}{\exp\left(\frac{-E(y)}{T_k}\right) + \exp\left(\frac{-E(x)}{T_k}\right)}. \tag{5.7}$$

Consider now a heat-bath thermodynamic particle system with only two states. The probability of this system being in each of its two states is given by the Boltzmann factor

$$P_i = \frac{\exp\left(\frac{-E_i}{k_b T}\right)}{Z},$$

where Z is called the partition function of the system and is given by

$$Z = \sum_{i=1}^{2} \exp\left(\frac{-E_i}{k_b T}\right),$$

where k_b is the Boltzmann constant, and E_i denotes the energy of the i-th state. It can be observed that the acceptance probability function (5.7), for a given probing solution y and fixed temperature, can in fact be approximated by the Boltzmann probability for the two-states system. Therefore, a Simulated Annealing process can be approximated by the modelling of a two-state particle system with a variable probing state energy.

In CSA, in order to couple many SA processes, we can for instance model the ensemble of process by a particle system with many states. Being x_i one of the many states and y_i the corresponding probing state, we can achieve this modelling by inserting more current states to (5.7) by considering the sum over a set of current states $x \in \{x_1, x_2, \cdots, x_m\}$ within the term $\exp\left(\frac{-E(x)}{T_k}\right)$,

$$A(x_i \to y_i) = \frac{\exp\left(\frac{-E(y_i)}{T_k}\right)}{\exp\left(\frac{-E(y_i)}{T_k}\right) + \sum\limits_{x \in \{x_1, x_2, \cdots, x_m\}} \exp\left(\frac{-E(x)}{T_k}\right)}. \tag{5.8}$$

This is a typical example of an acceptance function for CSA since it satisfies (5.5) and (5.6) with

$$\gamma = \sum_{x \in \{x_1, x_2, \cdots, x_m\}} \exp\left(\frac{-E(x)}{T_k}\right).$$

Observe that if $m = 1$, this generalised equation is reduced again to (5.7). With this example we show that (5.8) is a generalisation of (5.7). However, this is not a general proof that CSA is an extended class for SA since other SA acceptance functions exist.

5.5 Three instances of the CSA class of methods

In this Section, three CSA example methods are shown. It is clarifying to mention that many others may exist and that these examples do not stand

alone. What follows is a description of a general algorithm that is the basis for the three different coupling schemes illustrated next. Let Θ be a set containing m current solutions, let x_i be the ith element of this set, and y_i a corresponding probing solution. Let E be the cost, or energy to be minimised, associated to a given solution, and let γ be the coupling term as a function of the energy of the current states. T_k and T_k^{ac} denote the temperatures of the generation and acceptance processes, respectively, at the iteration k. The following algorithm can now be formulated.

Algorithm 2

1. *Initialisation:* assign random initial solutions to Θ; assess the costs $E(x_i)$, $\forall\ x_i \in \Theta$, and evaluate the coupling term γ; set initial temperatures $T_k = T_0$ and $T_k^{ac} = T_0^{ac}$; set the time index $k = 0$.

2. *Generate* a probing solution y_i for each element of Θ according to $y_i = x_i + \varepsilon_i$, $\forall\ x_i \in \Theta$, where ε_i is a random variable sampled from a given distribution $g(\varepsilon_i, T_k)$; assess the costs for all probing solutions: $E(y_i)$, $\forall\ i = 1, \cdots, m$.

3. *Accept* solution y_i with probability 1 if $E(y_i) \leqslant E(x_i)$, otherwise with probability $A_\Theta(\gamma, x_i \rightarrow y_i)$, $\forall\ x_i \in \Theta$, *i.e.* make $x_i := y_i$ only if $A_\Theta > r$, where r is a random variable sampled from a uniform distribution [0,1]; evaluate γ again; and go to step 2 for N inner iterations (equilibrium criterion).

4. *Decrease temperatures* according to schedules $U(T_k, k)$, and $V(T_k^{ac}, k)$. Increment k.

5. *Stop* if stopping criterion is met, *otherwise go to step 2.*

Many SA convergence proofs [45, 136, 60, 160] were established by associating a given generating distribution $g(\varepsilon, T_k)$ with a generating temperature schedule $U(T_k, k)$. Therefore, the choice for the distribution $g(\varepsilon, T_k)$ depends on the schedule $U(T_k, k)$. We have not investigated which $[g(\varepsilon, T_k), U(T_k, k)]$ pair is the best for CSA. In Section 5.8, we present our choice of $[g(\varepsilon, T_k),\ U(T_k, k)]$, and $V(T_k^{ac}, k)$ for the experiments performed in this Chapter.

5.5.1 Multi-state Simulated Annealing (CSA-MuSA)

This method is a direct generalisation of the classical SA with acceptance probability driven by (5.2), or (5.7). As mentioned before in Section 5.4.3, these acceptance functions can be approximated by the modelling of a particle system with only two states. In order to generate the acceptance function for this CSA method, we add more states to the original function. Hence the name Multi-state Simulated Annealing (CSA-MuSA). When accepting a probing solution, the whole collection of current states is taken into account. The following equation illustrates this acceptance function, which is essentially (5.8) with the actual CSA notation:

$$A_\Theta(\gamma, x_i \to y_i) = \frac{\exp\left(\dfrac{-E(y_i)}{T_k^{ac}}\right)}{\exp\left(\dfrac{-E(y^\star)}{T_k^{ac}}\right) + \gamma}, \tag{5.9}$$

where γ is given by

$$\gamma = \sum_{x_j \in \Theta} \exp\left(\frac{-E(x_j)}{T_k^{ac}}\right), \tag{5.10}$$

with $y^\star = y_i$. This acceptance function makes the probability of accepting a probing solution inversely proportional to its energy. Observe that the coupling term γ here is given by the only term in A_Θ that is shared among all current states.

Analysis of the coupling

The coupling here makes the probability equal to the Boltzmann factor in a system with many states, *i.e.* the sum of the probabilities for the current states Θ and the probing solution y_i equals 1. We make the difference between y_i and $y^\star$ here for purpose of normalisation. The probabilities will sum to 1 only if $y^\star = y_j$, for any fixed $j = 1, 2, \cdots, m$. An overview of the formulas is shown in Table 5.2, row 2.

5.5.2 Blind Acceptance (CSA-BA)

In this CSA method, the Boltzmann factor describes the probability of a system to *stay* in the current state upon generation of a probing state with

Method	Alg.	Acceptance function	Coupling term γ	Normalisation
CSA-MuSA	2	$\dfrac{\exp\left(\dfrac{-E(y_i)}{T_k^{ac}}\right)}{\exp\left(\dfrac{-E(y^\star)}{T_k^{ac}}\right)+\gamma}$	$\sum\limits_{x_j\in\Theta}\exp\left(\dfrac{-E(x_j)}{T_k^{ac}}\right)$	$\forall y^\star = y_j, j=1,2,\cdots,m:$ $\sum_{x_j\in\Theta} A_\Theta(\gamma, x_i\to y_j)$ $+A_\Theta(\gamma, x_i\to y^\star)=1$
CSA-BA	2	$1-\dfrac{\exp\left(\dfrac{-E(x_i)}{T_k^{ac}}\right)}{\gamma}$	$\sum\limits_{x_j\in\Theta}\exp\left(\dfrac{-E(x_j)}{T_k^{ac}}\right)$	$\sum\limits_{x_i\in\Theta} A_\Theta(\gamma, x_i\to y_i)=1$
CSA-M	2	$\dfrac{\exp\left(\dfrac{E(x_i)}{T_k^{ac}}\right)}{\gamma}$	$\sum\limits_{x_j\in\Theta}\exp\left(\dfrac{E(x_j)}{T_k^{ac}}\right)$	$\sum\limits_{x_i\in\Theta} A_\Theta(\gamma, x_i\to y_i)=1$
CSA-MwVC	3	$\dfrac{\exp\left(\dfrac{E(x_i)}{T_k^{ac}}\right)}{\gamma}$	$\sum\limits_{x_j\in\Theta}\exp\left(\dfrac{E(x_j)}{T_k^{ac}}\right)$	$\sum\limits_{x_i\in\Theta} A_\Theta(\gamma, x_i\to y_i)=1$
SA	1	$\dfrac{1}{1+\exp\left(\dfrac{E(y)-E(x)}{T_k^{ac}}\right)}$	–	–

Table 5.2: Overview of the studied methods. Multi-state SA (CSA-MuSA); Blind Acceptance (CSA-BA); Coupled Simulated Annealing-Modified (CSA-M); CSA-M with Variance Control (CSA-MwVC); and the classical SA algorithm.

higher energy. All lower energy states are accepted with probability equal to 1. The probability of accepting a less favourable state is proportional to the energy of the current state, *i.e.* a higher energy probing state is more frequently accepted at high energy current states while low energy current states are more preserved. The acceptance function is chosen as follows

$$A_{\Theta}(\gamma, x_i \to y_i) = 1 - \frac{\exp\left(\frac{-E(x_i)}{T_k^{ac}}\right)}{\gamma}, \tag{5.11}$$

where γ is given by (5.10). The probability of accepting state y_i and therefore leaving state x_i is the probability of *not staying* in state x_i, and it does not depend on y_i itself. Hence this method is called CSA Blind Acceptance (CSA-BA).

Analysis of the coupling

Low energy states are accepting fewer uphill moves than high energy states. This causes a localised search on low energy states and a more global exploration on high energy states. Observe that the acceptance probability for higher energy solutions is independent from the energy of the probing solution; therefore the name of the method. Also note that although this method presents a considerably different approach to CSA when compared to the previous one, its acceptance function has the same coupling term as in CSA-MuSA. The coupling here ensures that the sum of the probabilities of the system to *stay* in any of the current solutions equals 1. Observe that detailed balance is also not satisfied here. An overview of the formulae can be seen in Table 5.2, row 3.

5.5.3 CSA Modified (CSA-M)

Both previously described methods incorporate two distinct search features. The first one manages to hold more knowledge of low energy regions of the cost function, whereas the second explores better unknown regions. In Coupled Simulated Annealing Modified (CSA-M), we combine both search strategies. In both previous examples of CSA, the Boltzmann factor composes the acceptance function. Here a similar function is used, representing

the probability of *leaving* the current states upon an uphill move:

$$A_\Theta(\gamma, x_i \to y_i) = \frac{\exp\left(\frac{E(x_i)}{T_k^{ac}}\right)}{\gamma}. \tag{5.12}$$

Therefore, the sum of the probabilities of *leaving* any of the current states equals 1. Like in the second example method, this one also performs *blind acceptance* because its acceptance probability is independent of the energy of the probing solution.

The coupling term γ here is given by

$$\gamma = \sum_{x_j \in \Theta} \exp\left(\frac{E(x_j)}{T_k^{ac}}\right). \tag{5.13}$$

Observe that a the energy of the states here has a positive signal. This may cause numerical overflow instabilities in the evaluation of the acceptance functions. Fortunately, for many cost functions this problem can be easily solved by a simple cost function normalisation. The problem is discussed in more detail further in Section 5.5.3. Refer to Table 5.2, row 4, for an overview of the formulae related to this method.

Analysis of the coupling

The effect of the coupling in this method has clear advantages w.r.t. the other two. Probabilistically, at least one current state is likely to change at each iteration of the method. This ensures global search even at very low temperatures. Additionally many current states are allowed to have acceptance probabilities very close to zero, which generates knowledge via the coupling term for deciding better if it is worth to accept or not uphill moves. Similar to both previous examples, the coupling here is given by a sum of probabilities. However, here the probabilities considered are of *leaving* any of the current solutions.

Numerical Considerations

The use of a positive sign for the energies in (5.12) and (5.13) instead of a negative one like in (5.2) may result in numerical overflow instabilities for some cost functions with unknown output bounds. When these bounds are

known, a simple pre-scaling of its output is sufficient to suppress instability. However, with unknown cost function bounds, it is necessary to use other approaches. In this case, we suggest that all energies in (5.12) and (5.13) are subtracted by the maximum current energy, as follows

$$A^{\star}_{\Theta}(\gamma^{\star}, x_i \rightarrow y_i) = \frac{\exp\left(\dfrac{E(x_i) - \max\limits_{x_i \in \Theta}(E_{x_i})}{T^{ac}_k}\right)}{\gamma^{\star}}, \tag{5.14}$$

and

$$\gamma^{\star} = \sum_{\forall x \in \Theta} \exp\left(\frac{E(x) - \max\limits_{x_i \in \Theta}(E_{x_i})}{T^{ac}_k}\right). \tag{5.15}$$

The result of such a transformation is that now (5.14) is numerically stable and yet equivalent to (5.12). This can easily be seen because if we multiply both the numerator and denominator of (5.12) by $\exp\left(-\max_{x_i \in \Theta}(E_{x_i})/T^{ac}_k\right)$ we obtain (5.14). The resulting equation is numerically more attractive because all the exponential evaluations are of negative values.

5.6 Controlling variance of acceptance probabilities

As mentioned above, the acceptance temperature in CSA is not responsible for weighting the difference between the energy of the probe and current solutions but rather it is responsible for weighting the proportion that each acceptance probability has to the overall sum of the probabilities, which in any case must be equal to 1. In the case of CSA-M, this sum expresses that the probability of any probing solution being accepted equals 1. However, individually, the contribution of each process to this sum is given by (5.12). The value of this contribution depends of course on the energy of all current solutions, but individually, its proportion is mainly determined by the energy of the own current solution and the acceptance temperature. Each contribution is exponentially proportional to the energy of the current solution. The higher this energy, the larger the probability that the process accepts a probing solution. On the other hand, the acceptance temperature has a mixed role to this contribution. It appears in the numerator as well

as in the denominator of (5.12). Because of that, a change in this temperature affects the acceptance probability of individual processes differently. A temperature increase causes the probability of the process of the lowest energy to increase, while it decreases for the process with the highest energy. This effect spreads gradually along the energies in the intermediate range.

Besides the number of processes m, the acceptance temperature is the only parameter that can control the overall distribution of the contributions. One variable that we can control with this temperature is the variance of the m acceptance probabilities at a certain iteration. Since the sum of the probabilities is bounded above by 1, this variance is also bounded. Therefore, knowing that

$$\sum_{\forall x_i \in \Theta} A_\Theta(\gamma, x_i \to y_i) \equiv \sum_{\forall x_i \in \Theta} A_\Theta = 1,$$

the variance for A_Θ assumes the following form:

$$\begin{aligned} \sigma^2 &= \frac{1}{m} \sum_{\forall x_i \in \Theta} A_\Theta^2 - \left(\frac{1}{m} \sum_{\forall x_i \in \Theta} A_\Theta \right)^2 \\ &= \frac{1}{m} \sum_{\forall x_i \in \Theta} A_\Theta^2 - \frac{1}{m}. \end{aligned} \tag{5.16}$$

By using the fact that

$$\frac{1}{m} \leq \sum_{\forall x_i \in \Theta} A_\Theta^2 \leq 1,$$

it can be concluded that

$$0 \leq \sigma^2 \leq \frac{m-1}{m^2}.$$

This variance plays a significant role in the optimisation. A good variance value is the one which gives the right balance between global exploration and localised search. From (5.12) it is easy to see that at a high enough temperature, regardless of the energy of the current solutions, all the acceptance probabilities approach $1/m$, whereas for a low enough temperature, all but one solution approaches 0 while the one with the highest energy approximates 1. These two cases correspond to the two bounds for the

variance and are obviously to be avoided. In short, the acceptance temperature can be used to control the variance of the probabilities regardless of the current energies. Although the ideal variance value is unknown to us, our experiments with different cost functions show that values in the neighbourhood of the maximum variance deliver the best results. Typically, we recommend 99% of the maximum variance value.

An analytical relation between the acceptance temperature and the variance of the probabilities could not be found. Such a shortcoming prevents us from devising directly the appropriate temperature value for a desired variance. However, a very simple control rule can be used to steer this variance to the desired value. It can be done in the following manner:

$$\begin{aligned} &\text{if } \sigma^2 < \ \sigma_D^2, \ T_k^{ac} = T_{k-1}^{ac}\,(1-\alpha)\,, \\ &\text{if } \sigma^2 > \ \sigma_D^2, \ T_k^{ac} = T_{k-1}^{ac}\,(1+\alpha)\,, \end{aligned}$$

where σ_D^2 is the desired variance value and α is the rate for the increase or decrease of the temperature, typically in the range of $(0, 0.1]$. If the value of the acceptance variance is below its desired value, the acceptance temperature is decreased by a factor of $1-\alpha$, otherwise, it is increased by a factor of $1+\alpha$.

Such simple variance control can be applied only due to the coupling in the acceptance probability function. It substitutes a schedule for the acceptance temperature and more importantly, it works for any initial acceptance temperature. This is important because the setup of initial parameters in SA is most of the time a very cautious work. With this approach, we eliminate two initialisation aspects at once, which are the choices for an acceptance schedule and an initial acceptance temperature. In return, two other parameters are introduced, α and σ_D^2, but these have a well defined operating range and are much less dependent on the optimisation problem at hand. The complete algorithm with the variance control can now be stated as follows

Algorithm 3

1. *Initialisation:* assign random initial solutions to Θ; assess the costs $E(x_i)$, $\forall\ x_i \in \Theta$, and evaluate the coupling term γ; set initial temperatures $T_k = T_0$ and $T_k^{ac} = T_0^{ac}$; set the time index $k = 0$; set σ_D^2, *e.g.* $\sigma_D^2 = 0.99\left(\frac{m-1}{m^2}\right)$, and *e.g.* $\alpha = 0.05$;

2. *Generate* a probing solution y_i for each element of Θ according to $y_i = x_i + \varepsilon_i$, $\forall\ x_i \in \Theta$, where ε is a random variable sampled from a given distribution $g(\varepsilon_i, T_k)$; assess the costs for all probing solutions: $E(y_i)$, $\forall\ i = 1, \cdots, m$;

3. *Accept* solution y_i with probability 1 if $E(y_i) \leqslant E(x_i)$, otherwise with probability $A_\Theta(\gamma, x_i \rightarrow y_i)$, $\forall\ x_i \in \Theta$, as in (5.12), *i.e.* make $x_i := y_i$ only if $A_\Theta > r$, where r is a random variable sampled from a uniform distribution $[0, 1]$; evaluate γ again; and go to step 2 for N inner iterations (equilibrium criterion);

4. *Adjust* acceptance temperature T_k^{ac} according to the following rules: if $\sigma^2 < \ \sigma_D^2$, $T_k^{ac} = T_{k-1}^{ac}(1-\alpha)$; if $\sigma^2 > \ \sigma_D^2$, $T_k^{ac} = T_{k-1}^{ac}(1+\alpha)$;

5. *Decrease generation temperature* according to a schedule $U(T_k, k)$, increment k;

6. *Stop* if stopping criterion is met, *otherwise go to step 2.*

5.7 Parallel implementation

One of the most commonly used arguments by engineers and researchers to choose Genetic Algorithms (GA) rather than SA is the limited parallelisation capabilities of SA algorithms. In fact, there exist many parallel SA versions [77, 89, 13, 22]. However, their effective gain with parallelism can only be seen in low and very low temperature regions of the annealing schedules. This is mostly due to the fact that SA is inherently a sequential algorithm. Moreover, many parallel versions of SA seem to focus on accelerating convergence, rather than distributing processing while ensuring global search. There exists a trade-off between faster convergence and quality of solution, *i.e.* a trade-off between speed and global optimality. In CSA, we mainly aim at improving the quality of the final solution. This approach also happens to be highly suitable for parallelisation.

5.7.1 Examples of parallel architectures for CSA

While SA is inherently sequential, CSA is inherently parallel in every aspect. In CSA, every generation and acceptance process can be run separately in different CPUs. The effective necessary communication is due to

the coupling term. This involves very limited communication when compared with other parallel approaches which communicate entire solution vectors. Besides, transferring only the cost, or energy, does not scale with the dimension of the optimisation problem. Finally, since each acceptance process accepts new current solutions not necessarily at the same time, the type of communication can be asynchronous. However, synchronous communication is clearly also not discarded.

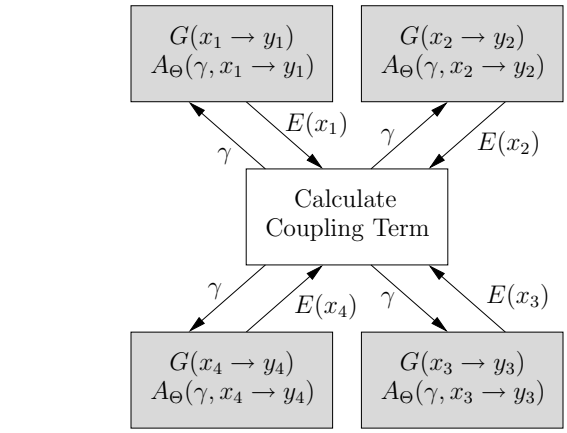
The actual parallel implementation can be elaborated in many different ways, from distributed systems to multiprocessor machines. For example, CSA can be implemented in a master-slave manner, where the energies of all current states are sent to a single node, the master, which is responsible to calculate and distribute the current coupling term to the other nodes, the slaves. Another example implementation involves a fully connected architecture, where each node receives the values of all current states energies and then calculate its own coupling term. Figure 5.3 shows these two typical examples of possible CSA parallel implementations.

In order to test the efficiency of the CSA methods described here, we have implemented these methods in a number of ways. We have used for instance Matlab to implement a sequentialised version of the algorithms. We have also tested the algorithms on a distributed implementation via local Ethernet network. Finally, most of our experiments were performed on the High Performance Computing (HPC) Linux cluster of our university, the K.U.Leuven VIC supercomputer [4]. Details about this implementation are given in the next Section.

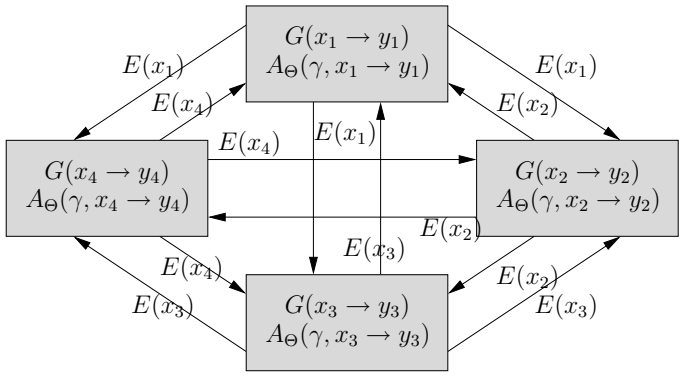
5.7.2 CSA on the VIC supercomputer

VIC is a HPC Linux cluster with nearly 900 processing nodes and more than 1 Tera-byte RAM with a theoretical peak computing performance of about 4 Tera flop/s [4]. We have analysed all the algorithms presented here with experiments performed on VIC. For that, we coded the algorithms in the *C* programming language using as Inter-Process Communication (IPC) a message passing method called Message Passing Interface (MPI) [3]. MPI is a *de facto* standard for communication among the parallel processes. Most MPI implementation consist of a specific set of routines written for different programming languages, including *C*.

The implementation of the CSA algorithms presented here on VIC for testing purposes is motivated by a series of facts. CSA algorithms as well as



(a)



(b)

Figure 5.3: Two typical examples of CSA parallel implementations: (a) Master-slave architecture; (b) Full connectivity. Each box in the figure can be implemented in an individual processing node. In the master-slave case, the energies of all current states are sent to a single node, which is responsible to calculate and distribute the current coupling term. In the fully connected architecture, each node receives the values of all current states energies and then calculates its own coupling term.

most global optimisation methods have an inherent stochastic nature. This means that in order to obtain concluding results, it is necessary to perform numerous experiments with the objective to generate reliable statistics on the results. In addition, it is known that according to the *no free lunch theorem* [147], global optimisation methods can only perform well on a limited number of problems, *i.e.* no method is universally better than all the others. Hence, we needed to test the CSA methods in a wide and variate type of problems in order to identify their stronger and weaker points for specific problems. In summary, it is important to perform a large number of experiments in order to validate each method presented here. Parallel implementation is therefore very helpful here. Moreover, the VIC supercomputer permits queueing of several batch jobs, which enables a more automatic experimentation, reducing eventual mistakes due to human interaction.

5.8 Experiments and results

We performed several experiments in order to assess the optimisation potential of the class of CSA algorithms. More specifically we tested the three CSA example algorithms described in Section 5.5 with 14 functions from 3 different problem groups. We used the classical SA algorithm described in Section 5.2 as reference for most of the experiments. In order to generate a fair comparison, we have used the same generation and acceptance schedules for the three CSA and the classical SA algorithms, with exception of the experiments where the variance control is evaluated. In this case, the acceptance temperature schedule for the CSA algorithm does not exist because this temperature is used as the manipulated variable for the control problem. In addition, in order to balance the algorithms w.r.t. parallelism and number of cost function evaluations, we compared the results of multiple runs of the classical SA algorithms in such a way that the number of different SA runs is equivalent to the number of parallel CSA processes, with the same number of cost function evaluations. Table 5.2 presents an overview of the algorithms used in the experiments.

5.8.1 Test problems

According to the *no free lunch theorem* [147], no global optimisation method can be universally better than all the others. Therefore, although the CSA methods presented here were developed with hard multi-modal cost functions as main problem target, we have tested these methods with three different problem groups, including one group with a unimodal and a simple multi-modal function. The remaining functions are from moderate to hard difficulty. In total, we have tested the algorithms with 14 functions with very different characteristics. These 14 functions, or part of them, appear often in optimisation research papers [142, 161, 98, 87].

Unimodal and simple multi-modal functions: group 1

The first problem of this group, function no. 1, is an easy unimodal sphere function. The second problem is the ubiquitous Rosenbrock's function, very often used for testing optimisation algorithms. The reader may refer to Table 5.3 for the equations.

Multi-modal functions: group 2

A collection of multidimensional and multi-modal continuous functions were chosen from the literature to be used as test cases. These functions feature many local minima and therefore are regarded as being difficult to optimise [161, 142]. The six multi-modal test functions belonging to this group are presented in Table 5.4.

Function no. 3 is called the Ackley's function and is probably the easiest in the group with one narrow global optimum basin and many minor shallow local optima. Function no. 4 is the Griewank's function. Its cosine term causes linkages among variables, making this function difficult to optimise. However, interestingly enough, this function is more difficult to optimise in lower than higher dimensions [146]. Function no. 5, the Weierstrass' function, is continuous but non-differentiable in several of points. Function no. 6 is the Rastrigin's function. It is a complex multi-modal problem with many persistent local optima. Function no. 7 is a non-continuous version of the Rastrigin's function with the same number of local optima. Schwefel's function is function no. 8 and the last of this group. Its complexity is due to deep local optima that are far from the global optimum. A common characteristic of most of the functions in this group is that they can be

No.	Function	minimum value	Input range
1	$f_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	0	$[-100, 100]$
2	$f_2(\mathbf{x}) = \sum_{i=1}^{D-1} \left((1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2\right)$	0	$[-2.048, 2.048]$

Table 5.3: Unimodal and simple multi-modal functions: test problems, group 1. These are simple functions that can easy to minimise. The dimensionality of these functions can be adjusted with the term D.

No.	Function	Input range
3	$f_3(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left[\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right] + 20 + e$	$[-32.768, 32.768]$
4	$f_4(\mathbf{x}) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$
5	$f_5(\mathbf{x}) = \sum_{i=1}^{D}\left\{\sum_{k=0}^{20}\left[(0.5)^k\cos\left(2\pi 3^k(x_i+0.5)\right)\right]\right\} - D\sum_{k=0}^{20}\left[(0.5)^k\cos\left(\pi 3^k\right)\right]$	$[-0.5, 0.5]$
6	$f_6(\mathbf{x}) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]$
7	$f_7(\mathbf{x}) = f_6(\mathbf{y}), y_i = \begin{cases} x_i & \lvert x_i\rvert < \frac{1}{2} \\ \frac{\text{round}(2x_i)}{2} & \lvert x_i\rvert \geqslant \frac{1}{2}. \end{cases} \quad \forall i = 1, \cdots, D$	$[-5.12, 5.12]$
8	$f_8(\mathbf{x}) = \sum_{i=1}^{D} x_i \sin\left(\lvert x_i\rvert^{\frac{1}{2}}\right)$	$[-500, 500]$

Table 5.4: Multi-modal functions: test problems, group 2. These functions present many local minima and are considered hard problems to optimise, especially in large dimensions. Zero is the minimum of all these functions but function no. 8, which minimum value is $-418.9829 \times D$. The dimensionality of these functions can be adjusted with the term D.

also minimised by multiple unidimensional searches, which often does not reflect well the characteristics of real-life problems.

Rotated multi-modal functions: group 3

Although the functions in group 2 are considered to be hard multi-modal problems, they can possibly be separable. This means that the minimisation problem can be solve using D unidimensional searches, where D is the dimension of the problem. A large variety of real-life optimisation problems are non separable. Therefore, in order to approximate these problems, in this group to test problems, we use rotated version of the functions in group 2. The rotated functions preserve the same shape characteristics of the original functions but cannot be solved by D unidimensional searches.

In order to rotate a function, we multiply the argument $\mathbf{x}$ of the original function by an orthogonal rotation matrix $\mathbf{M}$ to obtain the new argument $\mathbf{z}$ for the rotated function. This rotation matrix is obtained using the Salomon's method [122].

Finally, we can define the functions in this group as described in the next equations:

$$f_n(\mathbf{x}) = f_{n-6}(\mathbf{z}), \forall n = 9, \cdots, 13; \tag{5.17}$$

with

$$\mathbf{z} = \mathbf{M}\mathbf{x}, \tag{5.18}$$

for the first 5 test problems of this group. The last function is defined as follows

$$f_{14}(\mathbf{x}) = f_8(\mathbf{z}), \tag{5.19}$$

with

$$z_i = \begin{cases} y_i \sin\left(|y_i|^{\frac{1}{2}}\right) & |y_i| \leqslant 500, \\ 0.001\left(|y_i| - 500\right)^2 & |y_i| > 500, \end{cases} \quad \forall i = 1, \cdots, D; \tag{5.20}$$

$$\mathbf{y} = \mathbf{y}' + 420.96, \tag{5.21}$$

$$\mathbf{y}' = \mathbf{M}(\mathbf{x} - 420, 96). \tag{5.22}$$

This is necessary in order to keep the global optimum of original Schwefel's function, located at $[420.96, 420.96, \cdots, 420.96]$, within the search range after rotation.

5.8.2 Initialisation and temperature schedules

SA algorithms tend to be very sensitive to different initial parameters such as temperature. Therefore, in order to avoid too much tuning of these parameters, we have predefined the following set of initial test values for the acceptance and generation temperatures for all algorithms:

$$T_0 \in [0.001, 0.01, 0.1, 1, 10, 100],$$

$$T_0^{ac} \in [0.0001, 0.001, 0.01, 0.1, 1, 10, 100].$$

For each of the experiments presented here, all involved algorithms were then first tested with all possible combination of these temperatures values, with 5 runs per combination. The combination that had the best average result for each method was then used as initial temperatures in the specific experiment. The remaining initialisation parameter, *i.e.* the number of steps per fixed temperature, N, was kept fixed because its effect in the performance of a SA algorithm is related to the value of the initial generation temperature. In this way we hoped that our final generation temperature value would suit the value we fixed for N. The values that we fixed for N were $D^2 = 100$ and $D^2 = 900$ steps, for $D = 10$ and $D = 30$, respectively. These values were chosen to be conveniently small to not delay the experiments too much.

All tested algorithms were subjected to the same generation and acceptance schedules [136]. *i.e.*

$$U(T_k, k) \Rightarrow T_{k+1} = \frac{T_0}{k+1},$$

and

$$V(T_k^{ac}, k) \Rightarrow T_{k+1}^{ac} = \frac{T_0^{ac}}{k+1};$$

hence, according to Szu *et al.* [136], the resulting generation distribution is the Cauchy distribution

$$g(\varepsilon, T_k) = \frac{T_k}{(\varepsilon^2 + T_k^2)^{(D+1)/2}}.$$

By choosing the same schedules for all algorithms, inclusive classical SA, we hope to be able to better demonstrate the effect the different coupling schemes have in the optimisation. There was no study to establish which schedule would suit best each algorithm. Therefore, it is most probably that the performance of the algorithms presented here are yet not optimal.

5.8.3 Results for CSA versus multi-start SA

In order to establish a reference for comparison with the CSA algorithms proposed here, we performed experiments with all 14 test functions and compared the results of these algorithms with the results of the best performance of multiple runs of the classical SA algorithm of Section 5.2. Details about the choice of the initialisation conditions and temperature schedules were given in Section 5.8.2.

For each function, we performed 100 optimisation runs in 10 and 30 dimensions. Each optimisation run was composed of 10 parallel processes with 40,000 iterations per process, for $D = 10$, and 200,000, for $D = 30$. For CSA methods, the parallel processes are coupled, *i.e.* $\Theta = 10$, whereas for the classical SA method, each process consists of an independent optimisation run. In this case, at the end of all 10 processes, only the best result was used as the output of the multi-start SA.

Next we present box plots of the results for each group of test problems that were obtained for all four methods tested here in 10 dimensions. The same plots for dimension 30 are presented in Appendix B.

Unimodal and simple multi-modal functions: group 1

The results for these functions are presented in Figure 5.4. Clearly from the figure we can see that CSA-MuSA presented the worst performance for function no. 1, mainly due to the excess of outliers. Although the performance of the other three methods were similar, the two CSA ones can be distinguished for being slightly better. The same performance pattern can also be observed for the same function in 30 dimensions. For the Rosenbrock's function, multi-start SA performed better in 10 dimensions. In 30 dimensions, we consider the performance of CSA-MuSA and SA to be equivalent. The performances of CSA-M and CSA-BA were less good than multi-start SA and CSA-MuSA for this function in both 10 and 30 dimensions.

Multi-modal functions: group 2

The results of the experiments with the functions in this group in 10 dimensions are presented in Figure 5.5 and Figure 5.6. Although the performance of CSA-MuSA was considerably poorer for a couple of functions, *e.g.* func-

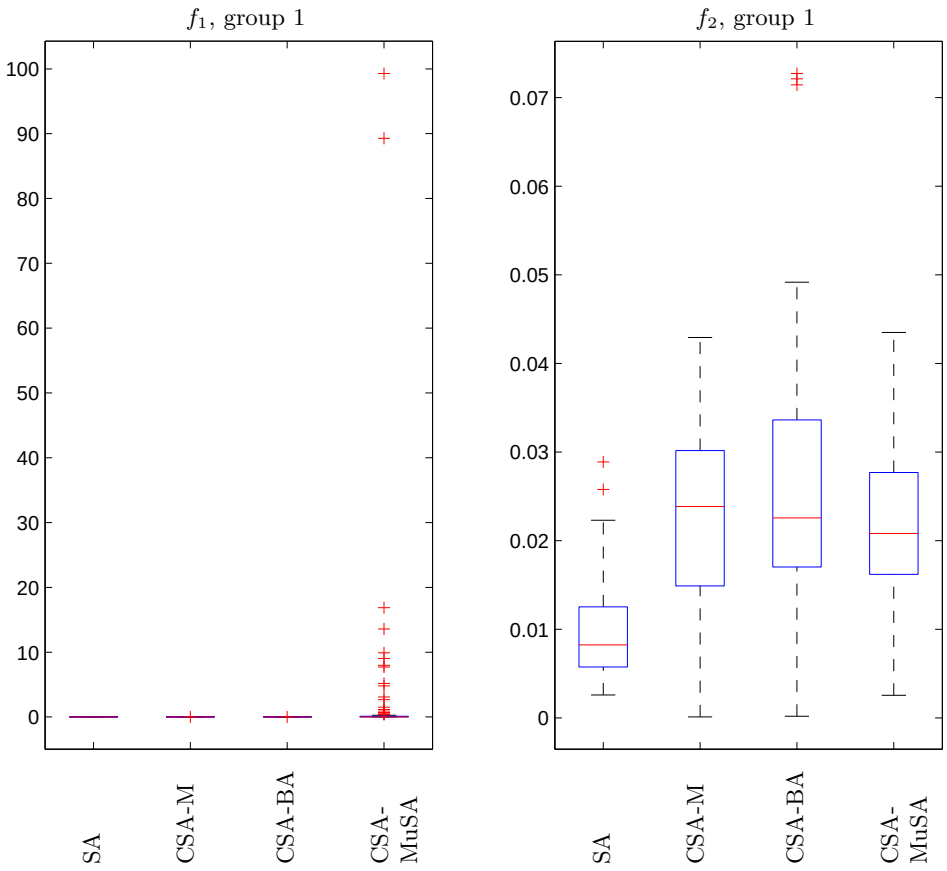


Figure 5.4: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 1 in 10 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 per parallel process with 100 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

tion no. 3,6, and 7, we consider the general performance of the other three algorithms to be very similar to each other.

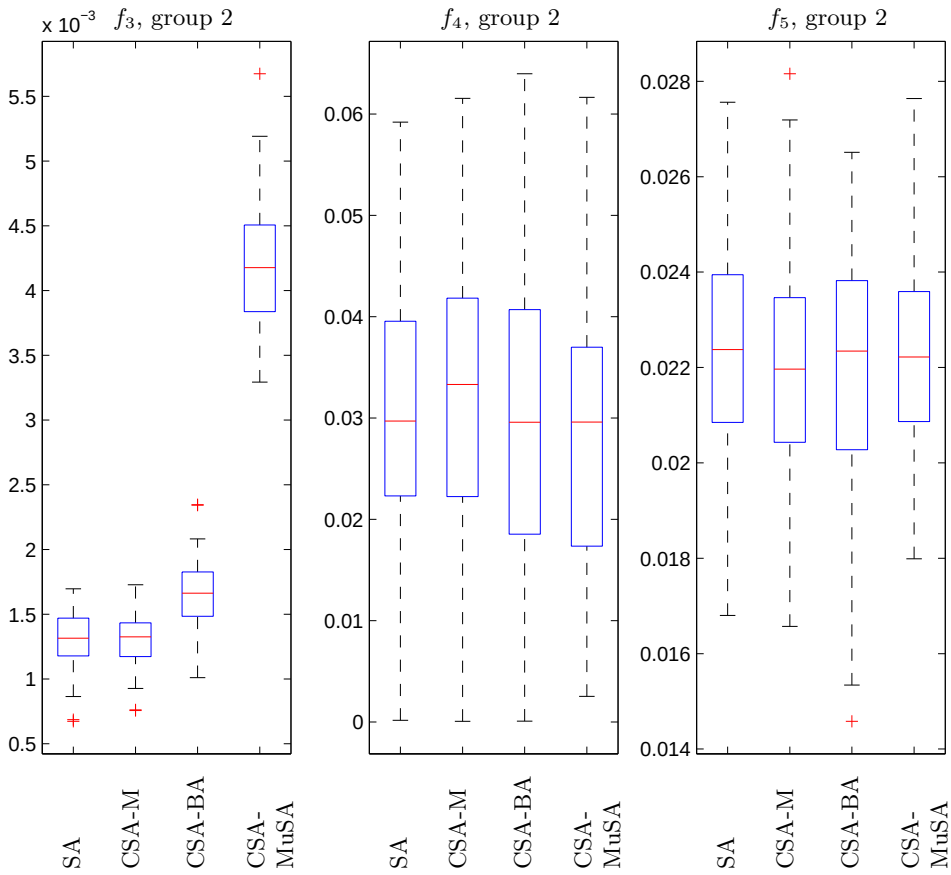


Figure 5.5: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 2 in 10 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 per parallel process with 100 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

Rotated multi-modal functions: group 3

Results for these experiments are presented in Figure 5.7 and Figure 5.8 for $D = 10$. Here, all the methods present considerably similar performance. However, multi-start SA presented some superiority, especially in 30 dimensions.

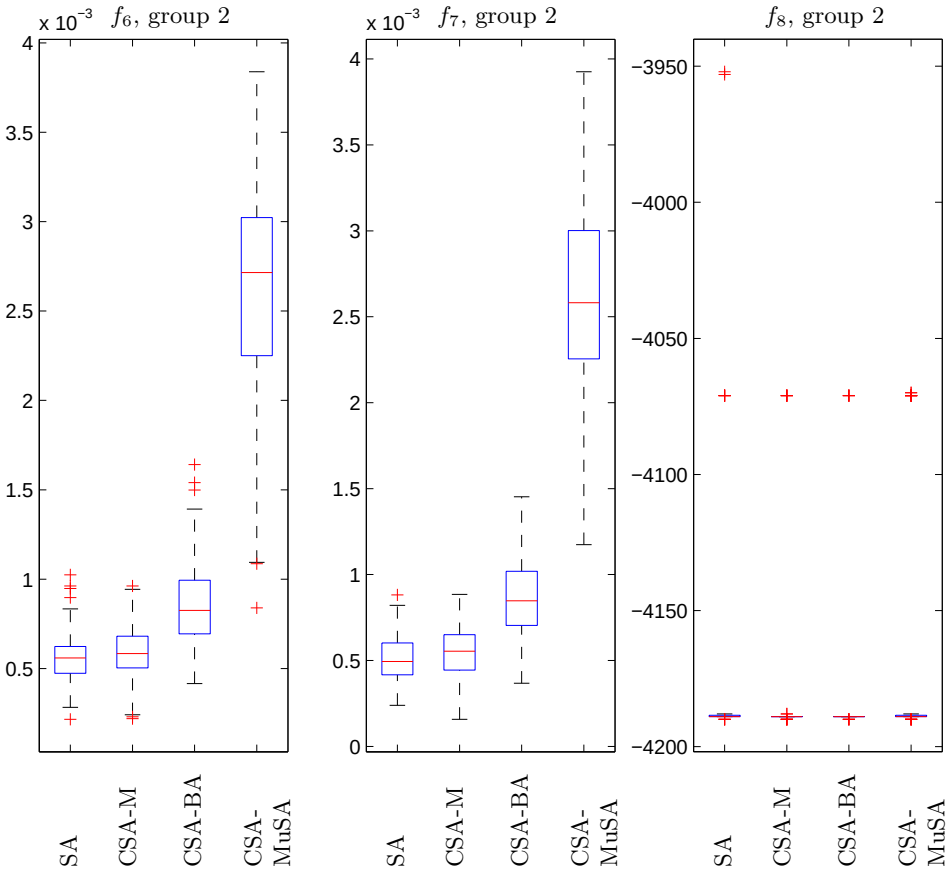


Figure 5.6: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 2 in 10 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 per parallel process with 100 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

Discussion of the results

Tables 5.5 and 5.6 present mean and variance summaries of Figure 5.4-5.8 and Appendix B, for 10 and 30 dimensions, respectively. The best results for each function are shown in bold. Analysing these results for each group we have that for the group of unimodal and simple multi-modal functions,

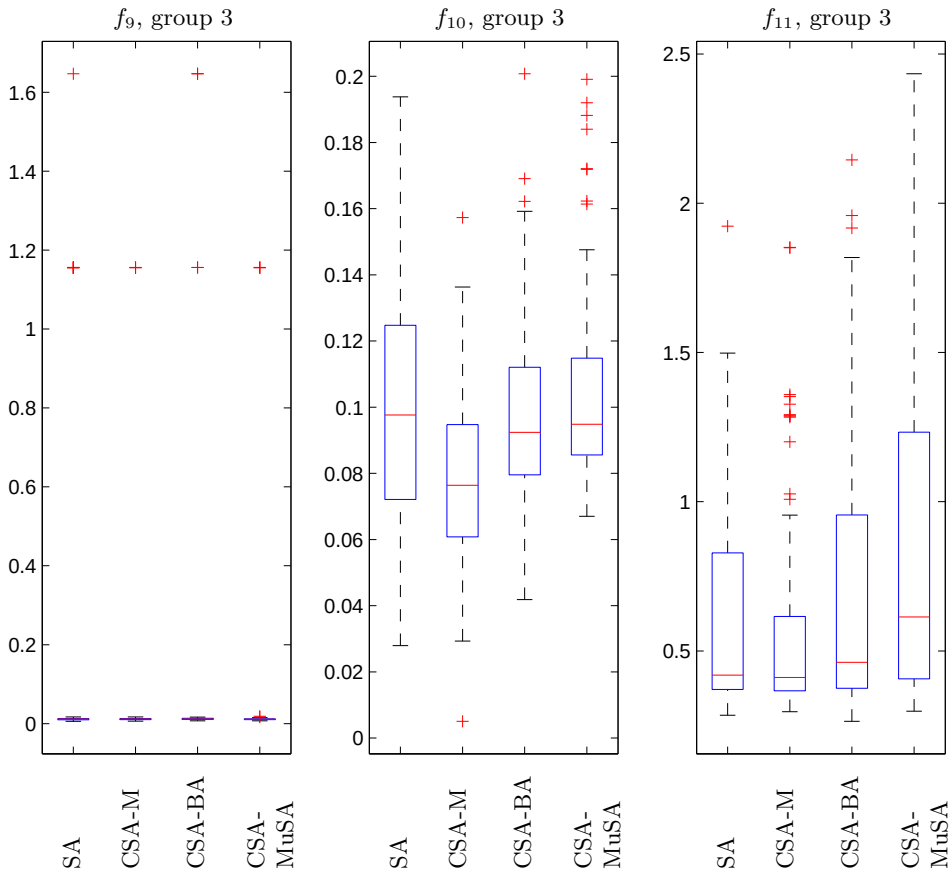


Figure 5.7: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 3 in 10 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 per parallel process with 100 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

multi-start SA performed always better with the Rosenbrock's function whereas CSA-M was always the best with the other group 1 function. The best performance for the functions in group 2 was equally divided among all methods in 10 dimensions. For $D = 30$, CSA-M performed best in 3 out of the 6 multi-modal function. Multi-start SA, CSA-MuSA, and CSA-

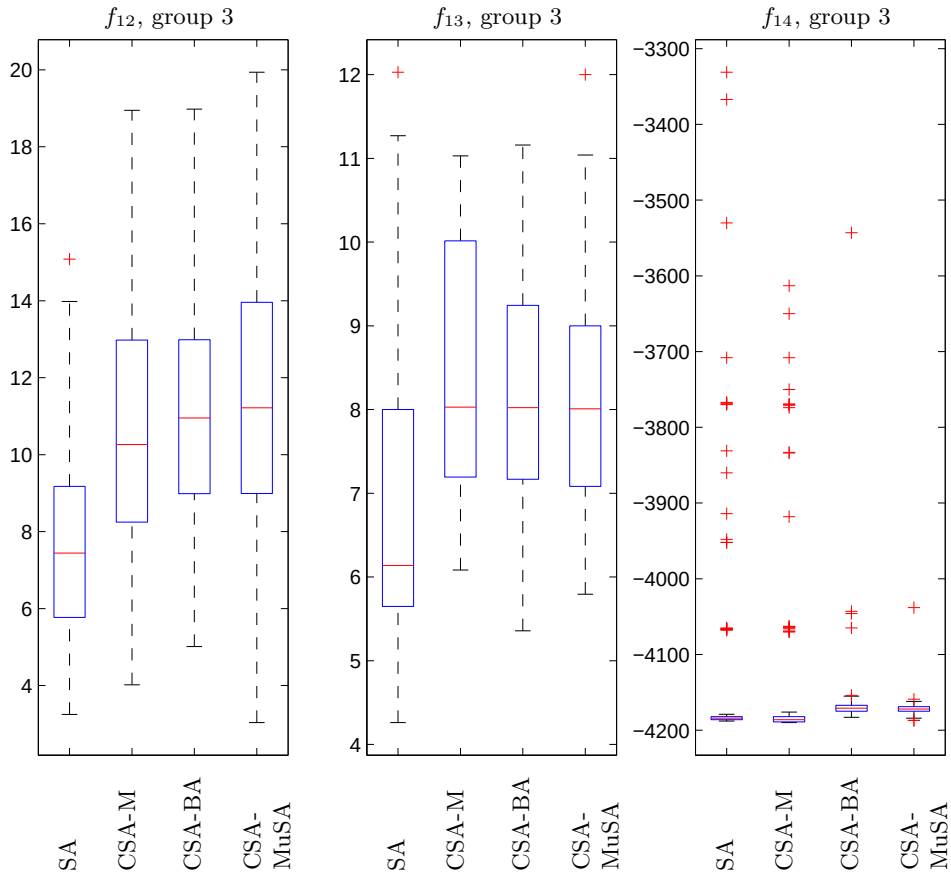


Figure 5.8: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 3 in 10 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 per parallel process with 100 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

M shared the same number of best results for the rotated function in 10 dimension. As for $D = 30$, multi-start SA performed almost always better that all the others for group 3 functions.

We can see that multi-start SA and CSA-M perform generally better than the other two CSA algorithms. Particularly, in 30 dimension,

	Multi-start SA	CSA-MuSA	CSA-BA	CSA-M
f_1	3.268e-06/6.911e-13	2.778e+00/1.791e+02	1.646e-07/2.610e-15	**1.015e-07/6.911e-16**
f_2	**9.778e-03/2.833e-05**	2.154e-02/9.149e-05	2.537e-02/2.139e-04	2.201e-02/1.266e-04
f_3	1.311e-03/4.229e-08	4.186e-03/2.005e-07	1.660e-03/5.888e-08	**1.303e-03/3.809e-08**
f_4	3.131e-02/1.584e-04	**2.865e-02/1.704e-04**	2.953e-02/1.815e-04	3.246e-02/1.680e-04
f_5	2.239e-02/4.332e-06	2.226e-02/3.548e-06	2.200e-02/6.006e-06	**2.190e-02/5.044e-06**
f_6	**5.646e-04/2.140e-08**	2.605e-03/3.889e-07	8.509e-04/5.806e-08	5.881e-04/2.102e-08
f_7	**5.083e-04/1.665e-08**	2.624e-03/3.051e-07	8.604e-04/4.885e-08	5.530e-04/2.057e-08
f_8	-4.164e+03/2.870e+03	-4.167e+03/2.147e+03	**-4.182e+03/7.871e+02**	-4.169e+03/1.967e+03
f_9	9.620e-02/9.813e-02	**5.699e-02/5.031e-02**	1.014e-01/1.109e-01	7.993e-02/7.389e-02
f_{10}	9.977e-02/1.241e-03	1.041e-01/8.163e-04	9.811e-02/8.364e-04	**7.798e-02/7.082e-04**
f_{11}	6.073e-01/1.304e-01	8.485e-01/3.364e-01	7.181e-01/2.095e-01	**5.739e-01/1.150e-01**
f_{12}	**7.709e+00/6.855e+00**	1.136e+01/1.089e+01	1.087e+01/9.796e+00	1.073e+01/1.047e+01
f_{13}	**6.800e+00/2.657e+00**	8.145e+00/2.069e+00	8.263e+00/1.962e+00	8.376e+00/1.852e+00
f_{14}	-4.115e+03/2.831e+04	**-4.171e+03/2.043e+02**	-4.161e+03/4.307e+03	-4.129e+03/1.852e+04

Table 5.5: Summary results for the 14 test functions with $D = 10$. The results are given in the format [mean/variance] for 100 runs of each experiment. The best mean values are presented in bold font.

	Multi-start SA	CSA-MuSA	CSA-BA	CSA-M
f_1	3.282e-05/8.856e-12	3.209e-04/1.690e-08	1.956e-07/9.110e-16	**1.494e-07/3.444e-16**
f_2	**7.348e-01/2.674e+00**	7.604e-01/2.619e+00	1.017e+00/2.865e+00	1.082e+00/3.665e+00
f_3	9.350e-04/4.454e-09	4.968e-03/7.027e-08	2.254e-01/2.483e+00	**9.332e-04/3.298e-09**
f_4	**1.372e-04/5.423e-07**	5.013e-03/4.483e-06	3.315e-04/2.099e-06	2.043e-04/1.490e-06
f_5	**9.219e-02/4.121e-02**	1.039e-01/4.980e-02	1.266e-01/7.643e-02	1.170e-01/6.978e-02
f_6	8.218e-04/1.226e-08	4.468e-03/1.122e-07	9.855e-04/2.624e-08	**8.112e-04/1.067e-08**
f_7	8.127e-04/1.126e-08	4.319e-03/1.689e-07	9.854e-04/2.604e-08	**8.067e-04/1.168e-08**
f_8	-1.248e+04/6.293e+03	**-1.249e+04/5.288e+03**	-1.248e+04/6.114e+03	-1.249e+04/6.078e+03
f_9	**9.854e-02/8.100e-04**	2.275e-01/3.030e-02	4.901e-01/1.966e-01	2.613e-01/4.098e-02
f_{10}	**1.509e-02/2.051e-04**	3.233e-02/6.172e-04	3.236e-02/6.236e-04	3.282e-02/6.013e-04
f_{11}	**3.879e+00/1.467e+00**	6.886e+00/1.325e+00	6.731e+00/1.782e+00	6.857e+00/2.083e+00
f_{12}	6.613e+01/8.437e+01	6.928e+01/1.512e+02	**6.500e+01/8.460e+01**	7.012e+01/8.960e+01
f_{13}	**4.454e+01/4.808e+01**	5.822e+01/6.737e+01	5.916e+01/6.925e+01	5.821e+01/7.535e+01
f_{14}	**-1.029e+04/1.699e+05**	-1.019e+04/2.035e+05	-1.021e+04/2.141e+05	-1.019e+04/1.885e+05

Table 5.6: Summary results for the 14 test functions with $D = 30$. The results are given in the format [mean/variance] for 100 runs of each experiment. The best mean values are presented in bold font.

multi-start SA performed best for most functions in group 3. However, the difference in performance between these two algorithms was always small. Although this makes it difficult to judge if one method is considerably better than the other for a specific function of group of functions, we feel that CSA methods have at least the potential to be better than multi-start SA. The reason for that is that although we only presented three examples of CSA methods here, many other possibly more efficient instances might be possible to be designed. Moreover, considering that the coupling is an exclusive feature of CSA methods, it opens possible improvement paths like the one described in Section 5.6, whose related experiments appear in the sequel.

5.8.4 Results for CSA with variance control

We performed experiments with all 14 test functions using the method described in 5.6. We compared the results of these experiments with the results of the best results of multiple runs of the classical SA algorithm of Section 5.2. The initialisation parameters were defined according to Section 5.8.2, except for the initial acceptance temperatures. These temperatures were chosen randomly from the predefined set for each optimisation run, for both CSA-MwVC and multi-start SA. This way, we expected to demonstrate the advantages of using CSA-MwVC without bothering to tune this temperature.

The configuration of the experiments were exactly the same as in Section 5.8.3, *i.e.* for each function, 100 optimisation runs in 10 and 30 dimensions, each run composed of 10 parallel processes with 40,000 and 200,000 iteration, for $D = 10$ and $D = 30$, respectively. The box plots of the results for each group of test problems are presented in Figure 5.9-5.15.

Discussion of the results

Tables 5.7 and 5.8 present mean and variance summaries of Figure 5.9-5.15. The best results for each function are shown in bold. The results show that CSA-MwVC was only once worse than multi-start SA. More precisely, multi-start SA was better with function no. 8 in 30 dimensions. For a few other experiments, multi-start SA was slightly worst than CSA-MwVC, *e.g.* functions no. 13, and 14 in 30 dimensions. For all the other experiments, CSA-MwVC was better or much better than its counterpart.

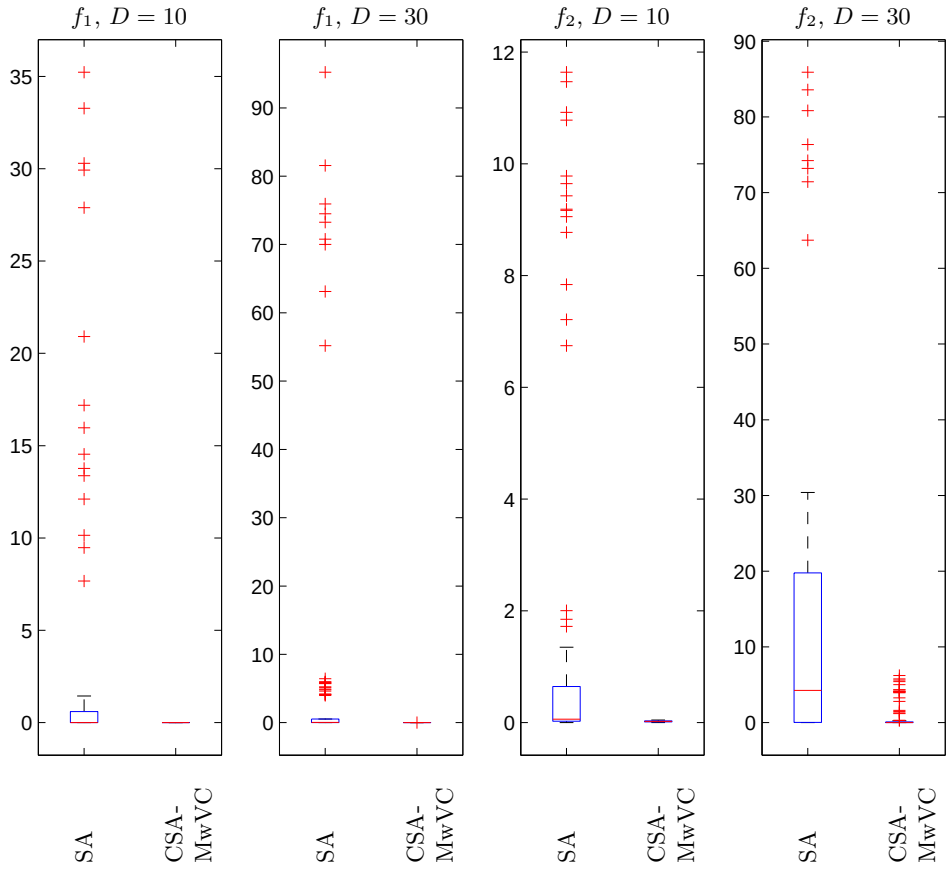


Figure 5.9: Box plots for experiments with CSA-M with variance control (CSA-MwVC) and Multi-start SA (MSA) for the functions in test group 1 in 10 and 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 and 200,000 per parallel process with 100 and 900 steps per fixed temperature for 10 and 30 dimensions, respectively. Initial generation temperatures were chosen from a predefined set after exhaustive search whereas the acceptance temperatures where chosen randomly from a predefined set of temperatures.

In summary, the variance control applied to the acceptance probabilities was able to steer optimisation runs to better regions of the search space. As

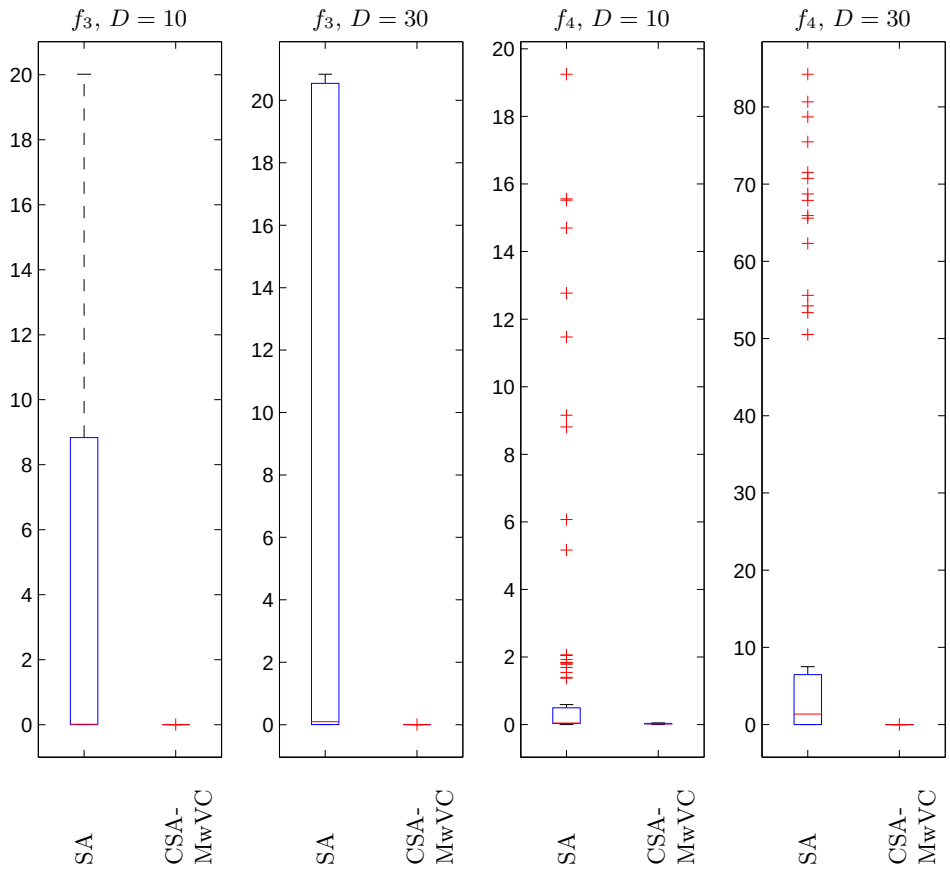


Figure 5.10: Box plots for experiments with CSA-M with variance control (CSA-MwVC) and multi-start SA (MSA) for the functions in test group 2 in 10 and 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 and 200,000 per parallel process with 100 and 900 steps per fixed temperature for 10 and 30 dimensions, respectively. Initial generation temperatures were chosen from a predefined set after exhaustive search whereas the acceptance temperatures where chosen randomly from a predefined set of temperatures.

a result, almost all optimisation experiments with CSA-MwVC were better, or much better than multi-start SA for the same random initialisation

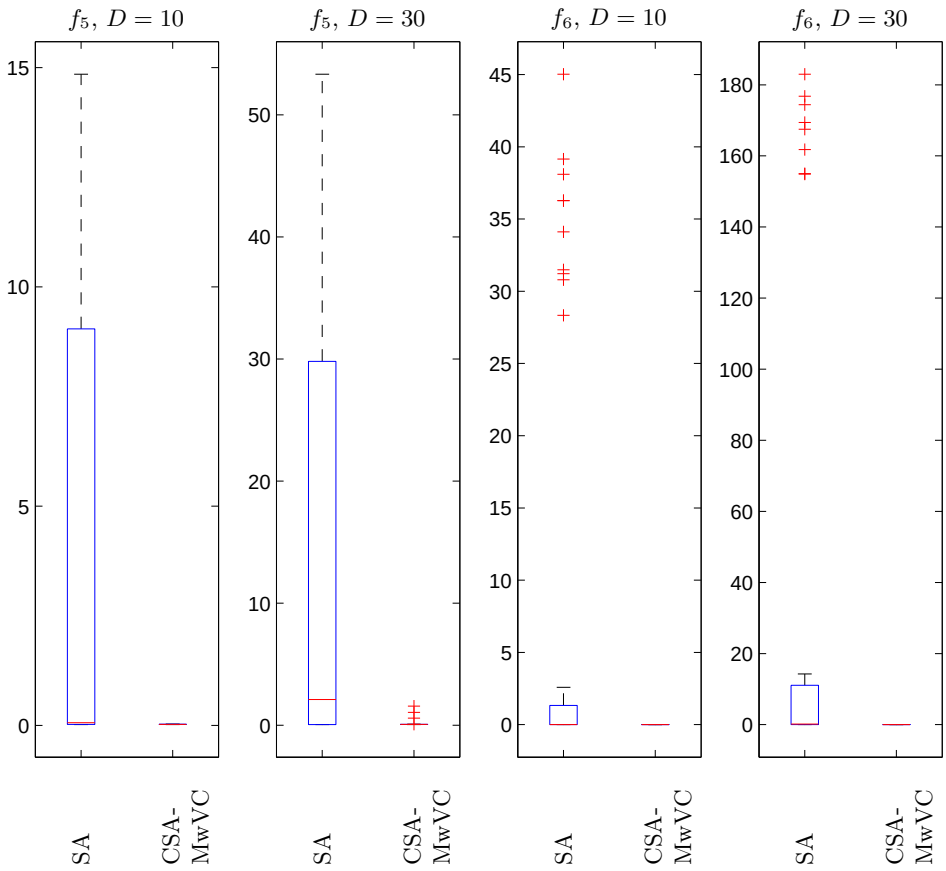


Figure 5.11: Box plots for experiments with CSA-M with variance control (CSA-MwVC) and multi-start SA (MSA) for the functions in test group 2 in 10 and 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 and 200,000 per parallel process with 100 and 900 steps per fixed temperature for 10 and 30 dimensions, respectively. Initial generation temperatures were chosen from a predefined set after exhaustive search whereas the acceptance temperatures where chosen randomly from a predefined set of temperatures.

parameter. This means that without tuning, CSA-M is almost always better than multi-start SA. However, it does not say much about the quality

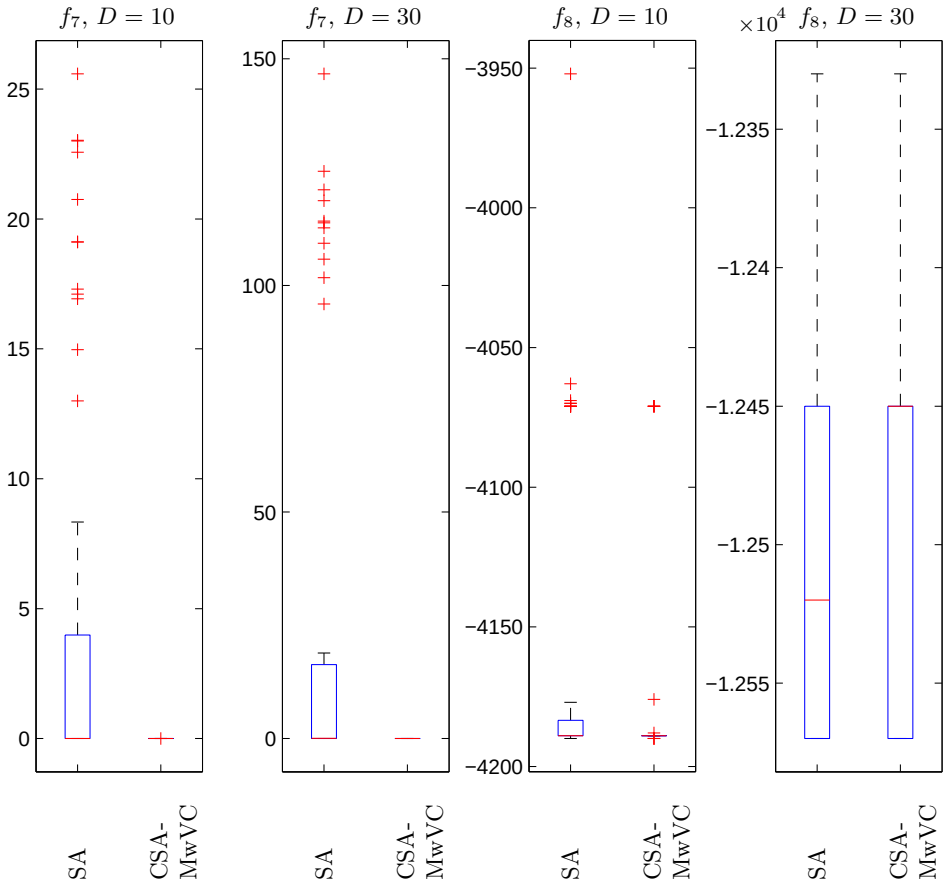


Figure 5.12: Box plots for experiments with CSA-M with variance control (CSA-MwVC) and multi-start SA (MSA) for the functions in test group 2 in 10 and 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 and 200,000 per parallel process with 100 and 900 steps per fixed temperature for 10 and 30 dimensions, respectively. Initial generation temperatures were chosen from a predefined set after exhaustive search whereas the acceptance temperatures where chosen randomly from a predefined set of temperatures.

of the results of the variance control w.r.t. to the best possible results in the presence of tuning. In the sequel we clarify this point with a few more

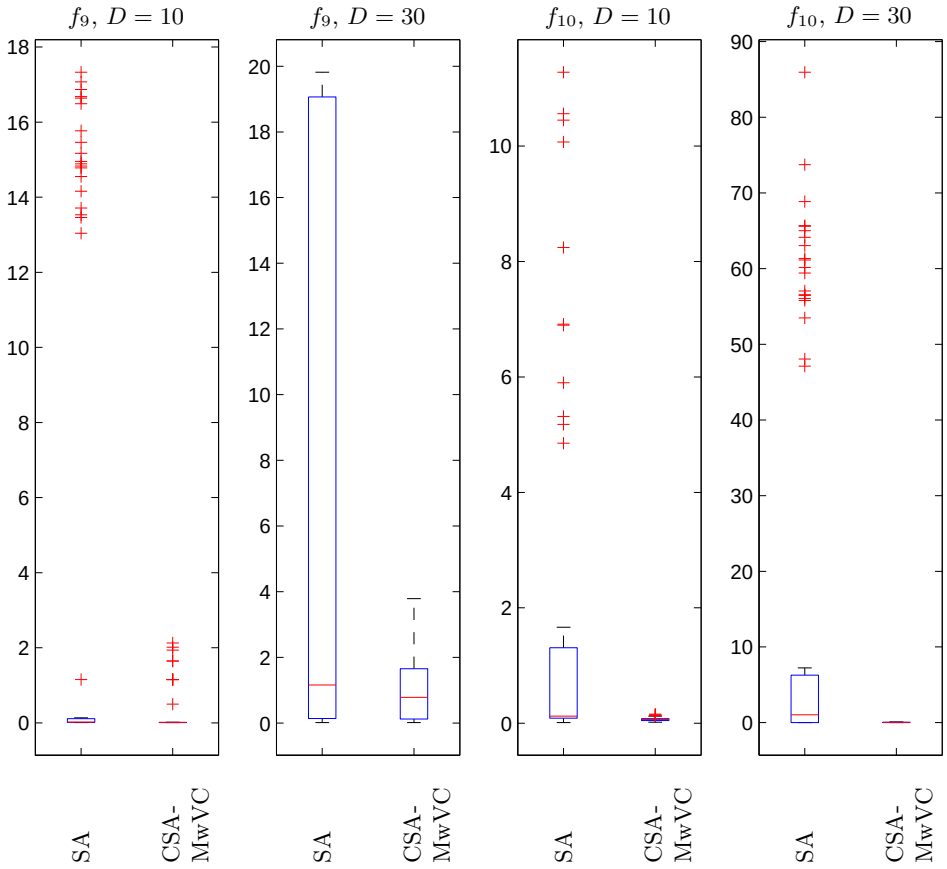


Figure 5.13: Box plots for experiments with CSA-M with variance control (CSA-MwVC) and multi-start SA (MSA) for the functions in test group 3 in 10 and 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 and 200,000 per parallel process with 100 and 900 steps per fixed temperature for 10 and 30 dimensions, respectively. Initial generation temperatures were chosen from a predefined set after exhaustive search whereas the acceptance temperatures where chosen randomly from a predefined set of temperatures.

experiments that compare CSA-MwVC with two different configuration of CSA-M without the variance control.

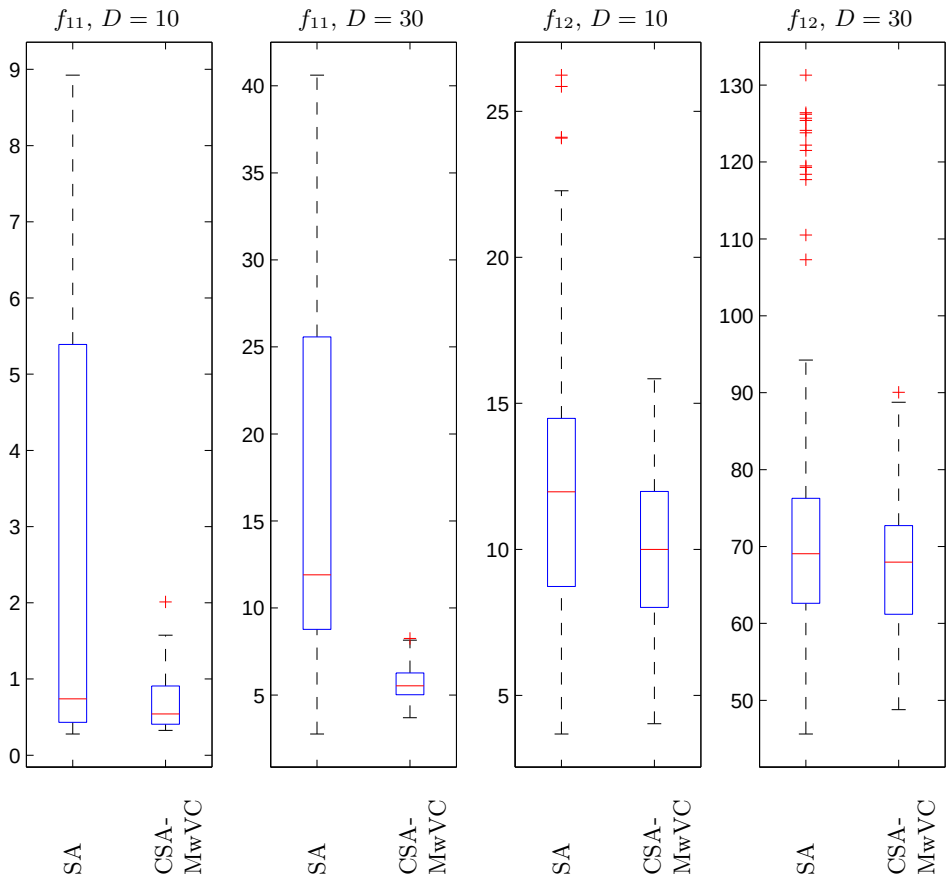


Figure 5.14: Box plots for experiments with CSA-M with variance control (CSA-MwVC) and multi-start SA (MSA) for the functions in test group 3 in 10 and 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 and 200,000 per parallel process with 100 and 900 steps per fixed temperature for 10 and 30 dimensions, respectively. Initial generation temperatures were chosen from a predefined set after exhaustive search whereas the acceptance temperatures where chosen randomly from a predefined set of temperatures.

5.8.5 Variance control versus best run

The effects of the variance control described in Section 5.6 were analysed using function no. 6. To better demonstrate this effect, results were gen-

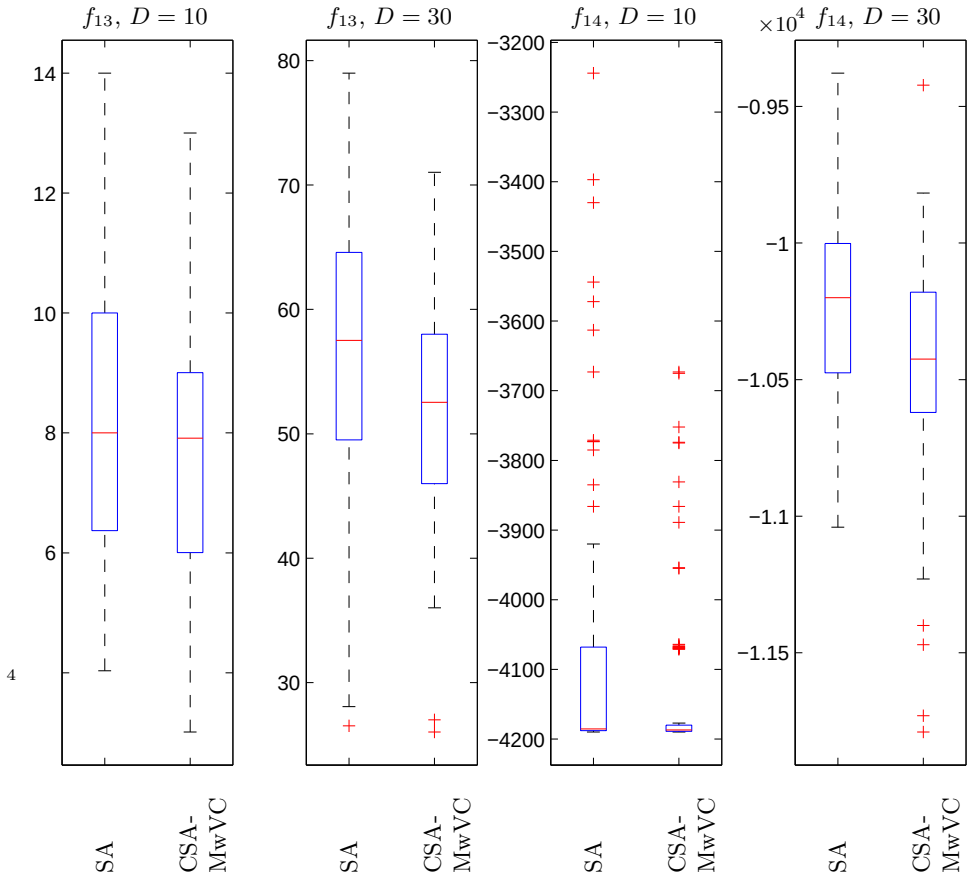


Figure 5.15: Box plots for experiments with CSA-M with variance control (CSA-MwVC) and multi-start SA (MSA) for the functions in test group 3 in 10 and 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 40,000 and 200,000 per parallel process with 100 and 900 steps per fixed temperature for 10 and 30 dimensions, respectively. Initial generation temperatures were chosen from a predefined set after exhaustive search whereas the acceptance temperatures where chosen randomly from a predefined set of temperatures.

erated for different dimensions and number of optimisers. The number of optimisers were chosen to be equal to the problem dimension with values

	Multi-start SA	CSA-MwVC
f_1	3.038e+00/5.993e+01	**1.115e-07/7.279e-16**
f_2	1.513e+00/1.057e+01	**2.183e-02/1.388e-04**
f_3	4.844e+00/6.984e+01	**1.359e-03/3.887e-08**
f_4	1.472e+00/1.411e+01	**2.942e-02/1.636e-04**
f_5	3.757e+00/2.972e+01	**2.328e-02/6.106e-06**
f_6	3.858e+00/1.111e+02	**6.860e-04/3.810e-08**
f_7	3.164e+00/4.044e+01	**6.538e-04/3.043e-08**
f_8	-4.167e+03/2.332e+03	**-4.174e+03/1.574e+03**
f_9	2.932e+00/3.581e+01	**1.674e-01/2.193e-01**
f_{10}	1.208e+00/6.169e+00	**6.729e-02/7.843e-04**
f_{11}	2.638e+00/8.419e+00	**6.991e-01/1.421e-01**
f_{12}	1.230e+01/2.607e+01	**9.969e+00/7.477e+00**
f_{13}	8.296e+00/4.593e+00	**7.560e+00/3.776e+00**
f_{14}	-4.088e+03/3.860e+04	**-4.136e+03/1.381e+04**

Table 5.7: Summary results for the 14 test functions with $D = 10$. The results are given in the format [mean/variance] for 100 runs of each experiment. The best mean values are presented in bold font.

ranging from 5 to 30. Three different configurations were used, two of them using CSA-M without the variance control and one with it.

The first configuration is a CSA-M algorithm using the best initialisation parameters obtained after exhaustive search. The second one uses the same algorithm but with random values for T_0^{ac}, uniformly distributed between 0 and twice the mean value of the cost function. In the third configuration we have the variance control approach. For this configuration, we use the same values for T_0^{ac} as used in the second one, *i.e.* random values. All configurations featured a maximum number of iterations per optimiser equal to 100,000, with 500 steps per temperature and $T_0 = 0.075$. For configuration with the variance control, the value for the desired variance was $\sigma_D^2 = 0.99\left(\frac{m-1}{m^2}\right)$ and $\alpha = 0.05$. The results of these experiments can be seen in Figure 5.16.

It can be noticed in Figure 5.16 that configuration (c) follows very closely (a). Although (c) was initialised with the same T_0^{ac} random values used for (b), the variance control was able to steer the optimisation process to quasi-optimal runs. In fact, configurations (a) and (c), unlike (b), were

	Multi-start SA	CSA-MwVC
f_1	7.447e+00/4.421e+02	**2.173e-07/7.925e-16**
f_2	1.362e+01/4.349e+02	**7.133e-01/2.543e+00**
f_3	9.983e+00/1.034e+02	**1.004e-03/8.340e-09**
f_4	1.181e+01/5.597e+02	**1.091e-03/8.092e-06**
f_5	1.204e+01/3.199e+02	**9.787e-02/3.408e-02**
f_6	1.630e+01/2.030e+03	**1.263e-03/4.018e-08**
f_7	1.549e+01/1.282e+03	**1.245e-03/3.600e-08**
f_8	**-1.250e+04/5.058e+03**	-1.250e+04/5.985e+03
f_9	6.380e+00/7.268e+01	**9.881e-01/8.881e-01**
f_{10}	1.333e+01/5.916e+02	**3.270e-02/7.308e-04**
f_{11}	1.719e+01/1.374e+02	**5.642e+00/8.382e-01**
f_{12}	7.498e+01/4.484e+02	**6.758e+01/6.675e+01**
f_{13}	5.735e+01/1.226e+02	**5.184e+01/6.484e+01**
f_{14}	-1.022e+04/1.162e+05	**-1.045e+04/1.572e+05**

Table 5.8: Summary results for the 14 test functions with $D = 30$. The results are given in the format [mean/variance] for 100 runs of each experiment. The best mean values are presented in bold font.

able to find at the end of the optimisation the basin of the global optimum, *i.e.* a location in the problem space from where the global optimum can easily be found with a simple local optimisation method. In conclusion, the coupling can also be used to approximate the ideal annealing temperature, thus reducing the influence of this initial parameter on the final search result, independently of the number of optimisers, Θ.

In Figure 5.17, the reader can find a plot of a typical run of the CSA-M algorithm with variance control for the test function no. 6. For this plot, $D = 5$ and also the number of optimisation processes $\Theta = 5$. The values for the initial parameters are the same as for configuration (c) in Figure 5.16. Although it might not be clear in the figure, the overall best solution is not retained by only a single optimiser, but it switches many times from one to another. This is a clear effect of the coupling on the balance of the acceptance probabilities, which also provides an equilibrium between global and local search by avoiding concentrations of optimisers in both low and high energy regions.

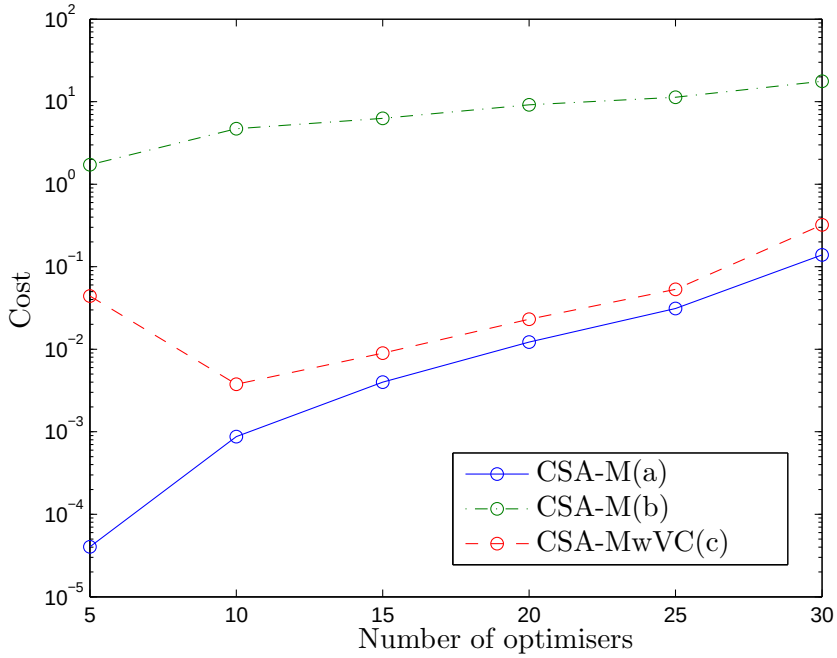


Figure 5.16: Plot of the average of 50 runs for function no. 6 with $D = \Theta$ for 3 different configurations: (a) CSA-M using best found initial parameters; (b) CSA-M using T_0^{ac} equal to random values uniformly distributed between 0 and twice the mean value of the cost function; (c) CSA-M using variance control with the same values for T_0^{ac} as used in (b). All configurations featured a maximum number of iterations per optimiser equal to 100,000, with 500 steps per temperature and $T_0 = 0.075$. For configuration (c), the value for $\sigma_D^2 = 0.99\left(\frac{m-1}{m^2}\right)$ and $\alpha = 0.05$.

5.8.6 Scaling with dimensionality

At last, we performed experiments with CSA to check the scaling of the necessary number of iterations to reach a given minimum energy tolerance, with an increase in the number of optimisers. These tests were executed for function no. 6 with several different values for D. The results can be seen in Figure 5.18, which is presented with a logarithmic scale in the vertical axis for better visualisation. This figure suggests that an increase in

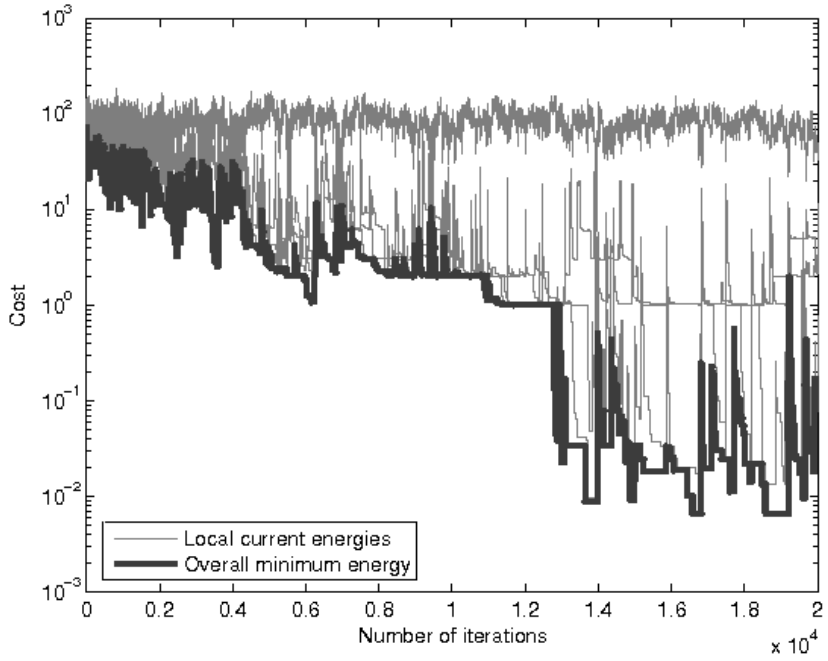


Figure 5.17: Plot of a typical CSA-M run with variance control for function no. 6 with $D \equiv \Theta = 5$. Initial parameters are: $T_0^{ac} = 64$; 500 steps per temperature; $T_0 = 0.075$; $\sigma_D^2 = 0.99\left(\frac{m-1}{m^2}\right)$; and $\alpha = 0.05$.

the number of optimisers decreases exponentially the number of necessary iterations to reach a given energy tolerance, regardless of the dimension D of the problem.

5.9 Conclusions

Coupling of SA processes has been introduced in this work as a new approach to solve hard continuous global optimisation problems. The coupling is applied within the acceptance probability function. Coupled Simulated Annealing is not considered as a single algorithm but rather as a class of algorithms defined by the form of the acceptance function and the coupling term. Supposedly, this class is also an extended class of SA algorithms.

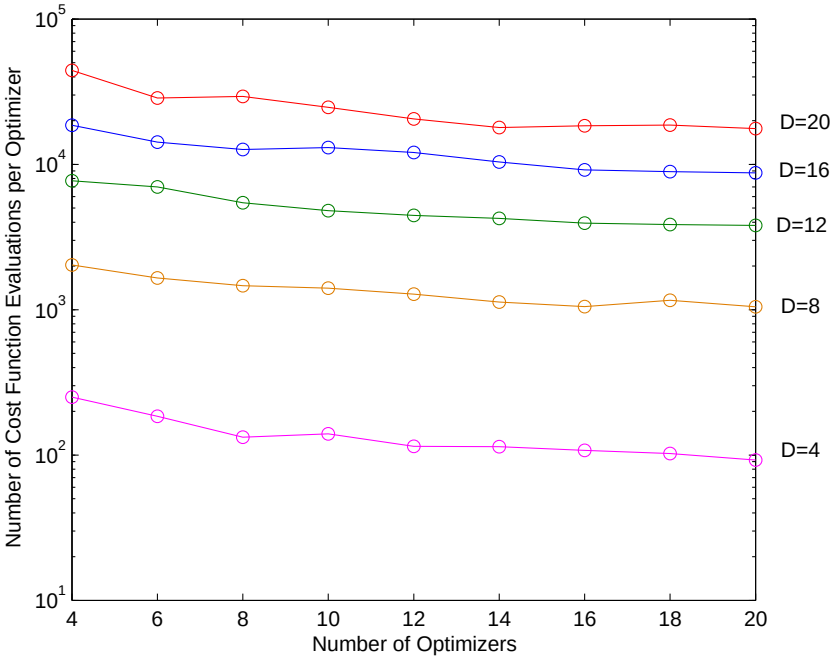


Figure 5.18: Plots of performance curves of CSA for different dimensions D of f_6. Every curve was obtained using the same initial temperatures. The vertical axis represents the number of necessary cost function evaluations per optimiser in order to reach a minimum energy tolerance. For all dimensions, this number decreases approximately exponentially with the number of optimisers, i.e. the number of elements in Θ. Note that the values in the vertical axis are on a logarithmic scale.

By analysing the results of the methods presented here, we conclude that the coupling of SA processes increases their exploration capabilities. We can also conclude that although there might exist many possible coupling schemes, the right design can be essential to boost such capabilities. In addition, CSA processes can run on parallel systems with little communication overhead, since only the coupling term needs to be distributed or evaluated among the processing nodes. Besides improving convergence rates, coupling is also useful to reduce the influence of the initial parameters and steer the

overall optimisation process to quasi-optimal runs. This is possible due to a simple but efficient variance control that uses the acceptance temperature as the control parameter to keep the variance of the acceptance probabilities away from undesired values. Finally, the algorithms described here are based on classical SA, the addition of many features developed along the past years to improve SA, like individual parameter *sensitivity* as in [60], can possibly improve even further the results shown here.

Chapter 6

CNNOPT: CSA Applied to CNN Optimisation

In Chapters 3 and 4 we presented two CNN optimisation methodologies for chip-specific tuning and learning, respectively. The global optimisation method that was common on these two methodologies and used to minimise each of the cost functions has been kept generic. We have used therefore a off-the-shelf global optimisation method known as Adaptive Simulated Annealing (ASA). In the previous Chapter, we introduced our own optimisation method inspired by the benefits of coupling in CNNs. The motivation for developing Coupled Simulated Annealing (CSA) was the difficulty in finding the right initialising parameters for ASA and similar methods. Through coupling, CSA was designed to solve multidimensional multi-modal problems with many local optima, which reflect well the characteristics of CNN optimisation problems. The coupling also gives CSA the necessary robustness w.r.t. initialisation parameters that we were looking for.

In this Chapter we present the application of *coupled simulated annealing* to *cellular neural networks*, the two main subjects of this thesis. We use CSA as the optimisation core for a unified framework that incorporates the CNN optimisation approaches described in Chapter 3 and 4. Therefore, we discuss here aspects that are relevant to this unification and features in addition to those already described. Additionally, we present a Matlab toolbox that implements the ideas described by this new unified framework.

6.1 A generalised approach for CNN optimisation

In Chapter 3 we described a CNN optimisation approach based on a global optimisation method for designing more robust templates targeting an individual chip instance. In order to find the template values that make a chip react as an ideal CNN structure, the output of template executions on the actual CNN chip was used to calculate the cost of a certain solution. The result of the optimisation process is a template that is optimal w.r.t. the imperfections of the chip. We define this approach as *tuning* rather than *learning* because there exists a good initial approximation of the final template. However, in absence of an initial template the same methodology can be considered as *learning of fixed-point templates*. We presented another learning approach in Chapter 4 for finding template values when the trajectory of each cell is specified at different time intervals as a sequence of images. This way, besides learning of steady-state behaviour, the network can also learn spatiotemporal dynamics. An additional complexity for this task w.r.t. the approach of Chapter 3 is that the cost function must assimilate the time instants of the evolution of the output into the set of parameters to be optimised.

For both CNN optimisation approaches described in this thesis, it is necessary to make use of a global optimisation technique for either tuning, learning of fixed-point or spatiotemporal behaviour. In Chapter 5, we presented a global optimisation approach which, inspired by the effect of coupling in CNNs, uses multiple simulated annealing processes coupled by their acceptance functions in order to efficiently optimise hard multidimensional numerical problems. Such problems reflect well the characteristics of the CNN optimisation problems. Therefore, as a consequent and natural sequel, we present in this Chapter a methodology that unifies tuning and learning of fixed-point and spatiotemporal templates using as our own global optimisation method, coupled simulated annealing, as the core of such methodology. This way, we unify here the contributions of the main three Chapters of this thesis.

Before we describe our unified framework, let us recast a few equations from previous chapters. Consider a full-signal range model cellular neural network organised in a regular rectangular grid of size $M \times N$ whose the

state of each cell obeys the following equation:

$$\begin{aligned}\frac{dx_{i,j}(t)}{dt} &= -\hat{g}(x_{i,j}(t),\Phi_{Sig}) \\ &\quad + \sum_{|k,l|\leqslant r} a_{k,l}(\Phi_{Tem})x_{i-k,j-l}(t) \\ &\quad + \sum_{|k,l|\leqslant r} b_{k,j}(\Phi_{Tem})u_{i-k,j-l}(\Phi_{Opt},\Phi_{Sig}) + z \quad (6.1)\end{aligned}$$

$$\hat{g}(x_{i,j}(t),\Phi_{Sig}) = \lim_{m\to\infty}\begin{cases} m, & \forall\ x_{i,j}(t) > g_{max}(\Phi_{Sig}), \\ x_{i,j}(t) & \text{otherwise}, \\ -m, & \forall\ x_{i,j}(t) < g_{min}(\Phi_{Sig}). \end{cases} \quad (6.2)$$

Fundamentally, this equation is a combination of (2.9),(2.12) and (2.16), which results in a space-invariant full-signal range CNN model. The notation here is mainly borrowed from the original equations. There are, however, additional features included here, namely the vector of hardware reference values Φ appear in the equation. These parameters are common to the family of ACE chips and are subdivided into *signal*, *template*, and *optical* reference values, or Φ_{Sig}, Φ_{Tem} and Φ_{Opt}, respectively.

For CNN optimisation, the inclusion of hardware parameters Φ in the CNN model can improve the degree of optimality of template operations. It is known that a template operation with the same input image and initial state can result in different output images depending on the actual voltage values of black pixels and white pixels [46]. These voltage values are directly affected by Φ_{Sig}. Therefore the inclusion of these and the other hardware parameters in the optimisation may promote the correct functioning of the template.

In (6.2), the hardware parameters are included as dependences for the corresponding affected variables, *e.g.* Φ_{Sig} may affect not only the $\hat{g}(\cdot)$ but also the range of u. The exact working of these parameters is not specified by the makers of the chips, nor by the makers of the computational setup interface of the chips. They are heuristically included here as the result of our own experience with their usage. In any case, a precise description of the working of these parameters is not essential for the approach described here. However, it is important to notice that they exist in actual VLSI CNN-UM implementation and that they can be used to modify the working of these chips in a way that is similar to how template values can be used.

It is also useful to stress that due to the properties of the FSR model defined in Section 2.2.2, the nonlinearity disappears from the output. Therefore, the output and the state of the network in this model are equivalent and any distinguishable mention of any of these terms hereafter is purely conceptual.

Finally, the unified framework that combines both previous CNN optimisation approaches with a target CNN model described by (6.2) can be formalised by the systematic minimisation of the following cost function:

$$E = \frac{1}{N_S N_R N_T} \sum_{s=1}^{N_S} \sum_{n=1}^{N_R} \sum_{k=1}^{N_T} \sum_{i,j} (x^d_{i,j;s,k,n} - x_{i,j}(A, B, z, \Phi, t_k))^2 \qquad (6.3)$$

where $\Delta t_k = t_k - t_{k-1} \; \forall \; k = 1, \cdots, N_T$ represent the time intervals between two output samples of the spatiotemporal dynamics, with $t_0 = 0$ and N_T being the number of samples. $x^d_{i,j;s,k,n}$ denotes the desired output value of a pixel in the kth image of a given sequence of N_T images. The value $x_{i,j}(A, B, z, \Phi, t_k)$ denotes the actual output value of a pixel as the system has evolved to the time instant t_k with weight matrices A and B, current z and hardware reference parameters Φ. N_S denotes the number of instances of initial states, inputs, desired-outputs in the training set. N_R denotes the number of repetitions a template is executed on chip. Repeating the execution of a template while adding noise to the template values or to the training set is important in order to generate chip-specific robust templates, as described in Chapter 3.

The actual optimisation problem of minimising (6.3) is given by

$$\min_{A,B,z,\Phi,\Delta t_1,\cdots,\Delta t_T} E(A, B, z, \Phi, t_1, \cdots, t_{N_T}), \qquad (6.4)$$

where, as in (4.16), the time intervals Δt_k between samples are used instead of the actual time instants t_k.

For every evaluation of the cost function in (6.3), the initial conditions, *i.e.* initial states $x_{i,j}(0)$ and inputs $u_{i,j}$ as well as desired-outputs $x^d_{i,j;s,k,n}$ are defined within a training set instance. To understand better the description of our systematic unified approach, we redefine here the following terms.

Training Set (TS) is a collection of images organised in TS instances.

TS instance is a set of images needed to train a template operation in order to perform the desired functionality. This set of images usually contains only the essential images involved in a template operation, namely input, initial state and outputs, which in this case are desired time-evolution outputs. To avoid clutter, we only consider these essential images. The incorporation of other images such as fixed-state map, bias map, and output mask, into the concept of TS instance is straightforward. These optional images are sometimes necessary to implement specific template functionalities on CNN chips. Formally, we define a TS instance as the set $\{\mathbf{u}, \mathbf{x_0}, \mathbf{x_1^d}, \cdots, \mathbf{x_{N_T}^d}\}$, whose elements are images with pixel values having a one-to-one correspondence with the input, initial state, or desired outputs of cells in the network grid.

Template is a term often used in the CNN community to denote the variable set A, B, z for space-invariant CNNs. For CNN hardware implementations, however, these variables do not uniquely specify the functioning of the array. For the ACE family for instance, additional variables, which include *e.g.* the hardware references Φ, also need to be specified. The time given for the evolution of the dynamics is another parameter that needs to be set in order to execute a template operation on chip. Therefore, in this approach, we characterise a template as the set of variable given by $\{A, B, z, \Phi, t_k\}$, where t_k is the time given for the evolution of the dynamics, and $\Phi = \{\Phi_{Sig}, \Phi_{Tem}, \Phi_{Opt}\}$ includes all hardware parameters. Hence, here we *extend* the template definition of Chapter 2 to a set of tunable parameters that are included completely or partially in the set of values to be optimised. It may or may not have an initial approximation. In the case of existence of a good initial approximation, we call the optimisation process *tuning*, otherwise, we call it *learning*, regardless of the nature of the problem, which can be either fixed-point or spatiotemporal dynamics.

6.2 Systematic CNN optimisation

Our main objective with the establishment of a unified approach for CNN optimisation is to create a standard procedure for tuning and learning for

different kinds of template operations. Starting from an initial specification sheet that describes the details and options of the CNN optimisation, we want to define a self-sustained systematic procedure that can find the most robust and optimal chip-specific template. Although there are many ways in which such a procedure can be defined, we want to avoid especially the trivial way, which means proceed with the optimisation assuming all unknowns as parameters to be optimised, neglecting all assumptions. Although this unsophisticated procedure might work within a not so impracticable time-frame for some simple template operations, our experience shows that for other moderate and complex operations it is necessary to make some reasonable assumptions in order to reduce the dimensionality of the search space. Therefore, we define here a systematic methodology to general CNN optimisation that is intended to be self-sustained and efficient w.r.t. time to convergence and chip-specific robustness and optimality. This methodology is defined under the terms of the problem stated in the previous Section.

There are many details specific to tuning or learning that need to be incorporated in the general CNN optimisation approach. For example, as CSA is inherently a parallel optimisation method, we can request multiple cost function evaluations from the chip. This feature must be incorporated into the unified framework. In order to enrich the descriptions of each of these details, we present in Figure 6.1 an overview of the framework that shows the flow of data and control variables for a general CNN optimisation algorithm.

As depicted in Figure 6.1, our unified approach is composed of two preparation stages and the main self-sustained optimisation module. The latter is composed of several sub blocks, including generation of probe solutions, evaluation of these solutions, and cost accumulation and normalisation. The two first blocks are the most important ones since they serve as the basis for rest of the optimisation procedure. Failure to define these initial modules well would most probably result in the failure of reaching the objective of the CNN optimisation. In Sections 6.2.1 and 6.2.2 we present detailed insights of these two steps and define some guidelines. What follows is a description of the submodules within the optimisation module.

Generate new probe solutions: this module receives from the option sheet the first initial approximation of the set of parameter to be optimised. From then on, each input flow comes from the *accept*

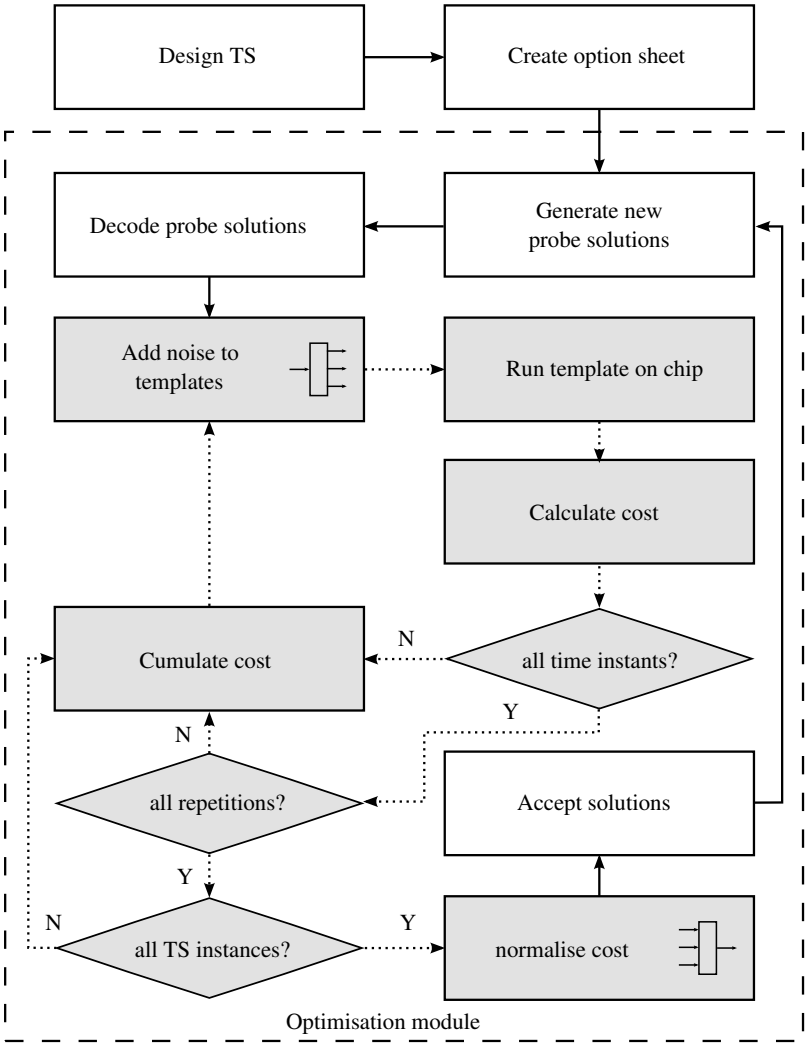


Figure 6.1: Block diagram of the unified CNN optimisation framework. The shaded elements in the diagram are part of the cost function (6.3). Full lines represent flow of batch solutions whereas doted lines represent flow of single template operations. The blocks within the dashed rectangle are part of the self-sustained optimisation module of the unified framework.

solutions module. The currently accepted solutions are then used to generate new probe solutions. This step is part of the CSA algorithm.

Decode probe solutions: in this step, the solution generated in the previous module are converted into the template form. That means that any assumption used to reduce the parameter search space must now be unfolded, *e.g.* in the case of symmetry assumptions, the template values omitted due to symmetry are now restored from their symmetric copy.

Add noise to templates: here we add noise to the decoded templates. The addition of noise is used in combination with repeated execution of the same probe template in order to achieve robustness, as explained in Chapter 3. In case no noise addition is specified in the option sheet, this stage is bypassed. This state also acts as a multiplexer, *i.e.* template solutions comes in batches and are fed to the next stage one by one.

Run template on chip: a probe template, with or without noise, is finally executed in this module, which correspond to the chip itself. The cell model in the chip is assumed to be similar to (6.2).

Calculate cost: After the chip executes the probe template, its output is then used in this module to calculated the cost of this solution.

Test completeness of time instants, repetition, and TS instances: These are the three diamonds in Figure 6.1 that represent tests for completeness of each of the summations in (6.4). Upon an incomplete summation, the flow of the algorithm is directed to the *add noise to templates* module. Upon completeness, we perform the next test until no test is left. We then proceed forward.

Normalise cost: After all summations are completed, we then normalise the cost in this module and pass it to the *accept solutions* module. It contrast to the *add noise to template* module, it acts as a demutiplexer, *i.e.* the flow of the algorithm is only passed forward once all solutions of each CSA process are evaluated.

Accept solutions: This module, like the *generate new probe solutions*, is also part of the CSA optimisation core. Based on CSA rules, it accept

or not the just probing solutions. The set of currently accepted solutions are then passed through for generation of new probe solutions.

Although we decided to use our own optimisation core, it can be easily replaced by any other core due to the modularity of the approach. In Figure 6.1, our core is represented by the modules *generate new probe solutions* and *accept solutions*.

6.2.1 Defining a training set

The topology of the cost surface is affected by the actual TS used in the optimisation. Thus, for learning or for tuning of fixed-point or spatiotemporal dynamics, constructing a comprehensive TS is of key importance for obtaining desired template functionality. The TS design needs to account for all desired functionalities while avoiding undesired ones. In addition, the metric used to compare the desired behaviour and the actual output can also modify the shape of the cost surface. It is important to design an efficient TS in order to smooth the cost surface with an approach for removing undesired plateaus and deep local optima. In other words, the designer must have a good understanding of the operation to be solved by the template in order to be able to create a correct TS.

Failure to design the right TS may result in a cost surface where the desired operation corresponds a local optimum, whereas the actual global optimum corresponds to an undesired operation. In such a case, the learned template may work for the TS instances but might fail to generalise to other untrained instances, causing an effect similar to *overfitting*. A good illustration of this is the grey-scale-constrained isotropic trigger wave template. In Figure 6.2, we present a few examples of incomplete training sets and an example of a comprehensive TS for the template. Another case of poor TS design is when the desired operation corresponds to the global optimum, but the difference between the cost of the global optimum and a number of local optima is not high or the basin of attraction of the global optimum is too narrow. A more careful TS design can alleviate this problem. The use of an adequate or more discriminative metric can also provide a similar effect. More about that on Section 6.2.3.

In order to minimise the effects of poorly designed training sets and maximise convergence rate and performance of the template optimisation, we define here a few rules of thumb:

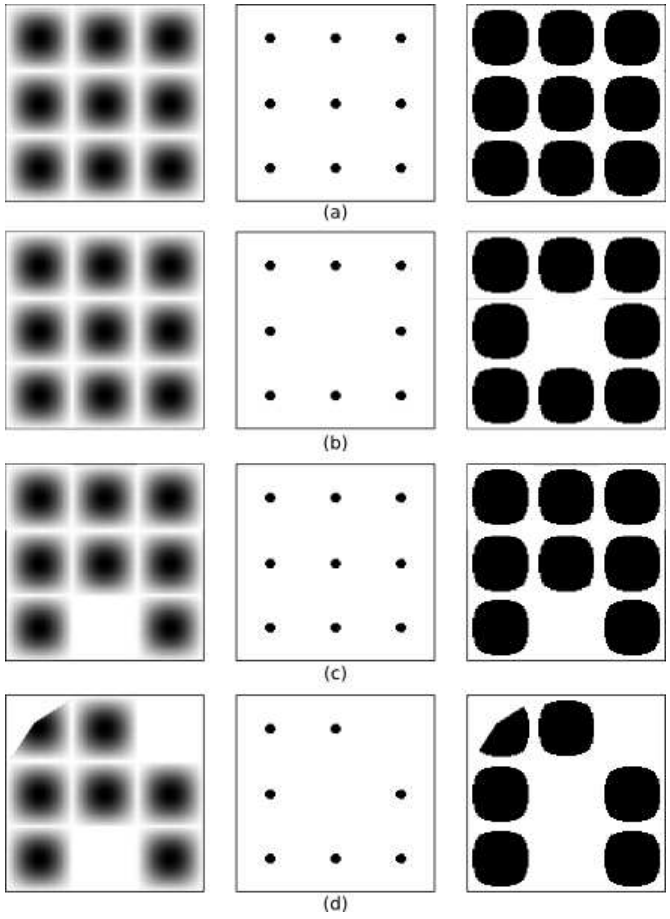


Figure 6.2: A collection of TS instances for the grey-scale constrained isotropic trigger wave. The objective of this operation is to create a propagating wave in the desired output (right) from marks in the initial state image (centre) that is constrained to travel only in regions covered by a certain grey level in the input image (left). (a,b,c) are examples of incomplete TS. In (a) and (c), a simple thresholding operation on the input would generate the specified desired output. (b) is incomplete because it omits the case of the existence of a marker without a constraining grey-scale region. (d) presents a more complete TS which assesses all desired functionalities and excludes the undesired ones.

- Analyse the functionality that the template operation has to learn and ensure that all essential input-output mappings are encompassed by your training set.
- For many functionalities, besides the input image, other images are also important in order to achieve what is desired. Analyse how the input, initial state, bias map, and fixed state map can alter the desired output.
- Make sure the desired output is well balanced. In the case of the metric used in (6.4), the average pixel value should be about zero in order to avoid the optimisation getting trapped into fully black (all pixels $\rightarrow +1$) or fully white (all pixels $\rightarrow$ -1) images.
- It is not necessary to create one TS instance for each input-output functionality mapping of the operator to be learned. A single TS instance can carry most of the relevant information embedding as many input-output mappings as possible into a single TS instance. This can be done in a spatially distributed way (see Figure 6.2). In addition, encoding various input-output mappings can improve the balance of black and white pixels.
- For learning, when the desired template functionality requires both input and initial state, it is difficult to be sure about the contents of the input and initial state. Equivalent functionalities can be achieved with different templates by exchanging the input and initial state images. For instance, for the shadow template, the object to be shadowed can be put either in the input or in the initial state or in both. In the simulator, it is enough to load the object as initial state and define B with all 0. When implementing templates on chip, often the robustness, speed, and/or generalisation are stronger in one of the cases. Also, for single input functionalities, input or initial state images can assume -1 or $+1$ globally, or be purely arbitrary. It is better to try different configuration as often, it is not clear at first which configuration is best.
- For the learning of spatiotemporal behaviour, apply the same rules above to each time step and make sure that the desired dynamics are clearly visible and as continuous as possible between consecutive images.

6.2.2 Reducing the search space

The main disadvantage of global optimisation methods is that they need a large number of function evaluations before they can reach global optimum convergence. For many problems, the time that is necessary to find the global optimum using any known type of global optimisation method is likely to grow exponentially with the dimensionality of the search space. Therefore, even for low dimensionality problems like CNN optimisation, *i.e.* in the order of dozens of parameters at most, it is worth attempting to reduce this dimensionality as much as it is possible.

There are several ways to reduce CNN optimisation search space. The appropriate means depends on the operation being optimised. Whatever reduction is used, it must be specified in the option sheet. In fact, the main objective of creating an option sheet is the precise definition of the search space. Other information present in the option sheet are mainly initialisation values. The precise module in Figure 6.1 that uses the search space reduction information from the option sheet to "decompress" the parameters being optimised into an effective set of template values is the *decode probe solutions* module.

In order to define the rules in the option sheet for reducing the search space, it is essential to gather relevant knowledge about the operation to be optimised. Knowledge about the spatial structure of the template is especially important. For example, template stability constraints [99], fixed template values, or template symmetry are information that can seriously reduce search space. The trigger-wave template [106], for instance, consists of an A template where the surrounding elements are the same. In the B template, only the central element is not zero. By exploiting these two a priori known constraints, 15 unnecessary parameter values can be removed from the optimisation. Additionally, chip implementations may involve hard-wired limitations of the CNN model. For instance, the ACE16k does not fully implement all the template values at once. Either the surrounding template elements of the A template or those of the B template can be used at one time. This is a hardware related constraint that limits the range of template types and can be automatically imposed.

Search space can be cut down further by setting bounds on the range of each optimised variable. Firstly, in accordance with the actual VLSI implementation, hardware bounds apply to all template values. Further truncation of the template value range can be derived mainly in the case

of chip-specific tuning, *i.e.* either when the simulator version of a template is known or when a first optimisation run had already found a working template and subsequent optimisation is to be launched in order to improve the robustness of the solution.

6.2.3 The influence of the metric

In (6.3) we consider a Euclidean metric, which is a measure of the symmetrical difference between two images and can be considered as the degree of coincidence of two point sets $\mathcal{P}$ and $\mathcal{Q}$. In the binary case this is the same as counting the number of different points between images, which is often referred to as calculating the Hamming distance. Another commonly used distance is the Hausdorff distance that measures the distance of a point on $\mathcal{P}$ that is farthest from any point on $\mathcal{Q}$.

Although the Hamming and Hausdorff distances are commonly used in image processing applications for object comparison and classification, they have several disadvantages. Hamming distance measures only the area difference, but does not reveal anything about shape difference. In addition, it is sensitive to object shift and noise. Hausdorff metric, on the other hand, measures the mismatch between two shapes but also cannot tell anything about shape properties. A single noisy pixel can drastically modify the Hausdorff distance.

In [134] the so-called non-linear wave metric was introduced that inherently measures both area and shape differences between two binary objects. Let a binary wave be started from $\mathcal{P} \bigcap \mathcal{Q}$ and spreading only to the points of $\mathcal{P} \bigcup \mathcal{Q}$. The time required for the wave to occupy $\mathcal{P} \bigcup \mathcal{Q}$ measures the difference between the shapes $\mathcal{P}$ and $\mathcal{Q}$. The result is a grey-scale map where values are related to the time required for the wave to reach a given position. If these so-called local Hausdorff distances are summarised then the wave-type metric takes both Hamming and Hausdorff distances into account. In addition to capturing both area and shape differences, this technique has parallel implementation with about $10\mu s$ running time [109] and may be implemented in the CNN-UM itself. We believe that the use of this metric would improve considerably the shape of the cost surface in favour of enhancing the location of the global optima. However, we have not implemented this metric with our approach and therefore we yet make use of the Euclidean metric.

6.3 CNN optimisation cases

Depending on the particular choice of Φ and N_R in (6.4), learning of dynamical operations or learning of steady-state behaviour of an operation or tuning for chip-specific robustness of an existing template can be performed. The choice of N_S is important in a different aspect. Setting it to a number greater than 1, *i.e.* using more than one training set instance, is beneficial to ensure the final template is not depending on our choice of particular images in the training set.

6.3.1 Tuning for chip-specific robustness

Tuning or learning of a steady-state behaviour operation is performed when Φ is set to 1. Robustness of a template is ensured if N_R is set higher than 1, and a proper amount of noise is added to images in the TS and to each probed template.

In order to minimise the time needed to find the robustly optimal template, it is better to split optimisations into two epochs. First, a rough optimum is searched, ensuring the template performs the desired functionality. Then, a second optimisation starting from the result of the first epoch introduces noise into template and/or image values in order to increase robustness. The definition of the variance of the noise involves some experimentation. Variance set too high can corrupt the correct functioning of the template whereas too low variance does not improve robustness enough. In theory, the higher the number of repetitions the more robust the final template. Note however that optimisation time increases linearly with the number of repetitions.

6.3.2 Learning of fixed-point dynamics

In our context, learning means the identification of specific values for all variables in the extended template so that the desired functionality, *e.g.* an image processing operation, is performed on the CNN array. This task differs from tuning in that we do not have an initial guess of the values of the extended template. This implies in many cases that the whole parameter space must be searched.

6.3.3 Spatiotemporal learning

In the case of learning of spatiotemporal dynamics, there several time steps which the array must perform in order to transform the initial state into the final output.

6.4 Implementation - Matlab toolbox

The unified framework for learning and tuning CNN templates has been implemented as a Matlab Toolbox. Simple analogic routines can be designed using the Bi-i Vision System [164] via its own low-level programming language called Analogic Macro Code (AMC) [73]. More complex algorithms which also need digital routines can be designed in a relatively easy way via the software development kit (SDK). However, additional software and knowledge is required involving the purchase of Texas Code Composer Studio and DSP programming experience. Matlab is considered as a reasonable trade-off between required programming knowledge and flexibility. The benefit of the proposed Matlab Toolbox is that it relieves the user from all the intricate programming issues needed to design algorithms via the Bi-i SDK.

Although chip robustness will most probably improve in future VLSI implementations, this issue may have crucial importance when targeting industrial applications. For application fields where environmental conditions are stable (as in many surveillance tasks), a robust CNN based algorithm can be used with one-time chip-specific tuning of the templates used. Where conditions are varying in an *a priori* known range and one single template tuning cannot ensure correct operation in the whole range, a viable solution could be the re-calibration of the system in constant intervals or when change in environmental conditions is detected.

This toolbox can be downloaded from

http://www.esat.kuleuven.be/sista/chaoslab/cnnopt/.

6.5 Conclusions

We presented a unified framework for template learning and chip specific tuning using CSA as optimisation core. Due to the variance control of

the acceptance probabilities, CSA reduces user interaction during optimisation by avoiding re-tuning of initialisation parameters. The inclusion of hardware reference values in the optimisation allows the tuning of template operations that were otherwise impossible be performed in the ACE16k_v2 CNN chip [58].

Definition of a training set remains the key issue for successful template learning and tuning. Designing a good TS for an unknown operation is an iterative process during which the user gets more and more insights into how the set of all possible input-output mappings of an operation can be condensed into the TS. The guidelines for design a training set presented here can help the CNN algorithm designer to reduce the number of iterative steps necessary to generate a final and correct TS. Although a good solution is not guaranteed when using global optimisation in a limited number of cost function evaluations, with a reasonable amount of time, the proposed method can be a valuable tool to implement new fixed-point and active wave operators on CNNs.

The proposed Matlab toolbox is presented to the CNN community in order to relieve the analogic algorithm designer from as much hassle as possible related to code interfacing and hardware hazards so that the CNN algorithm design can be put in focus.

Chapter 7

Robust VLSI CNN-UM Applications

7.1 Real-time object tracking

In this Section we describe a *analogic* and visual algorithm for tracking of an object immersed in an image sequence. Object tracking allows the analysis of objects and persons' movements in video images, making it possible to instantaneously calculate their position, the direction and speed of their movements, and whether or not they will meet or collide. The objects that could be considered for tracking are innumerable including airplanes, missiles, vehicles, people, animals, insects, microorganisms, etc. Typically, there exist more than one object in the scene, which introduces problems due to the fact that the objects can touch and occlude each other, move inside or outside the image boundaries, etc. Besides multiple object scene problem, tracking algorithms also need to account for usual single-object tracking issues such as deformation, occlusion, illumination, and time-varying background. In summary, due to all these issues, object tracking is complex task. The objective of the algorithm described in this Section is thus to tackle some of these these issues rather than solving all of them.

In this section we consider object tracking with locking on a given object, which permits collecting information from the locked object regardless of the presence of ambiguous similar objects in the scene. The applications are vast and range from the field of security and traffic analysis to sport events examination.

The advantages of using a CNN system to perform object tracking are mainly related to the speed that such systems can deliver. This feature is especially necessary for processing image sequences in real-time. Images often carry a lot of redundant information making sequential architectures like Digital Signal Processor (DSP) unsuitable to achieve very high frame rates. CNN systems on the other hand are inherently parallel, which is an immediate advantage for image-processing; and analog, which avoids delays of analog-to-digital (A/D) conversions, necessary when working with any digital technology in image-processing like DSPs. Many CNN systems even support direct optical input, which also eliminates the step of storing the image between the acquisition and the processing, which drastically shortens the input acquisition latency. In these systems, unlike DSP systems, the acquired image can be directly processed. Real-time constraints for object tracking can be met by using a CNN-UM VLSI implementation [113, 85, 83]. Such devices are able to deliver ultra-high processing speeds owing to its highly parallel and *analogic* array processing.

After a detailed description of our CNN-based object tracing algorithm in Section 7.1.1, we make an analysis of performance and speed of the proposed algorithm in Section 7.1.3.

7.1.1 A visual/analogic algorithm for tracking with locking

The objective of this algorithm is to identify the coordinates of an object in a sequence of images containing one or more objects, given the initial information about which object to lock on. The presented algorithm contains two main steps that need to be executed for each frame of the sequence. The first step is the isolation of the object that has to be tracked. This step denotes the locking feature of the algorithm. The second step is related to the calculation of the coordinates of the chosen object within the image. At each frame, the algorithm define the position of the centre of the object with relation to the upper left corner of the image.

Locking on the chosen object

In order to provide the tracking with locking, it is necessary to isolate the chosen object from other objects in the current frame of the image sequence. This step has as input the current frame and the processed previous frame containing only the isolated chosen object. It gives as output the isolated

object in the current frame. An important remark for this step is that the shifting of the object between two consecutive frames can not exceed its transverse length in the given shifting direction. This remark yields a very hard constraint for the algorithm since its performance scales proportionally with the transverse lengths of the object. Such a constraint may require extremely high processing speed and an abridged image acquisition latency. These features are natural characteristics of some VLSI CNN implementations. Further discussions about this remark are given in Section 7.1.3.

Consider an image frame F_i as being the ith frame in a sequence $F_i, \forall i = 1, 2, 3, ...$, where F_i is a represented by a matrix with binary elements (pixels) with value 0 (*false/white*) or 1 (*true/black*). The sets of adjacent *true* elements represent objects in the image. The image frame O_i will be the image of the chosen object alone in the same position as in the image F_i. Given a new frame F_i, and the previous image of the chosen object O_{i-1}, the calculation of O_i is given by the following equations:

$$O_i = F_i \text{ AND } M_i,$$

where M_i is the marked absolute difference between F_i and O_{i-1} calculated by

$$M_i = \text{recall}(A_i, O_{i-1}),$$

where $recall(R, S)$ is an operation that reconstructs only the objects of R that are marked with elements of S, and A_i is the union (OR logic operation) of F_i and O_{i-1} calculated by

$$A_i = F_i \text{ OR } O_{i-1}.$$

During the initialisation of the algorithm, when O_{i-1} does not exist yet, O_{i-1} needs to be initialised with an image containing a mark, denoted by one or more elements set to *true*, to be placed in the exact correspondence with any element of the chosen object in F_i, and with all other elements set to *false*. An illustrative example of this step of the algorithm is shown in Figure 7.1.

Obtaining the coordinates of the object

The objective of this step is to find the coordinates of the centre of the object with relation to the upper left corner of the image. The input is

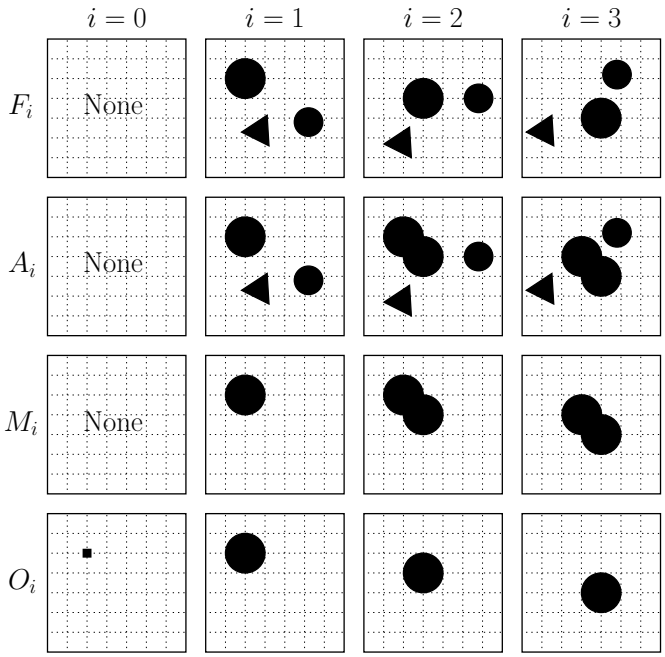


Figure 7.1: Locking on the object

the image of the chosen object alone, O_i, given by the previous step. The concept of centre here coincides with the mass centre of the object growing to a rectangle[1]. The projections of the object to the bottom and left side of the image, representing the length and height of the rectangle, are used to obtain the horizontal and vertical coordinates respectively. The centre of each projection is found by pyramiding[2] the projection and then erasing everything but the top. The top is then shadowed horizontally or vertically, according to the dimension of the coordinate. The result is used as a mask for a massive diffusion CNN template operation [40] having as initial state a real-valued image with degrading pixel values in the orthogonal direction to the given projection, namely -90 degrees related to the projection. At the end of the diffusion, every element of the resulting image (now real-valued) is expected to be proportional to the given coordinate. Figure 7.2

[1]This approach was used instead of the well known recursive rotating peeling templates due to the smaller number of operations necessary to find the centre.

[2]Pyramiding up: $A = [2.10\ -0.35\ 2.10\ 0.13\ 2.95\ 0.13\ 0.66\ 0.67\ 0.66]; B = 0; z = 4.7$.

shows the evolution of this step for the frame O_3 of the example of Figure 7.1.

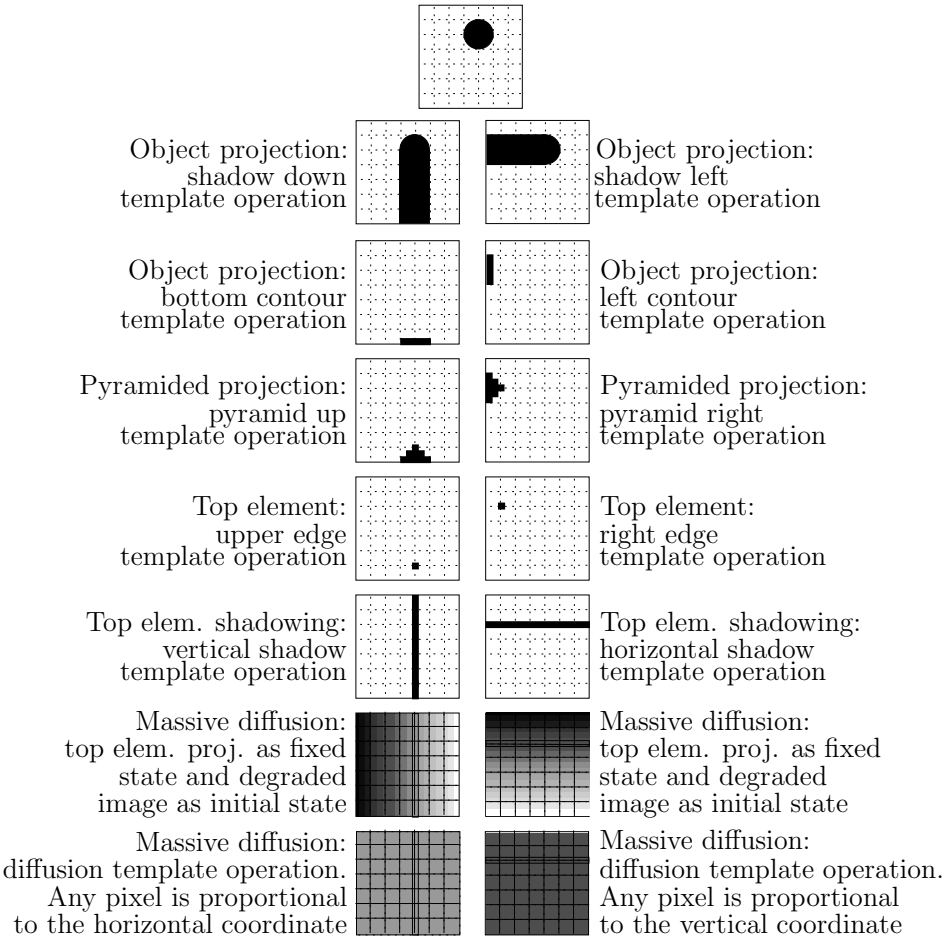


Figure 7.2: Obtaining the coordinates of the object

Owing to difficulties to implement the *diffusion* template in VLSI CNN-UM chips, an alternative algorithm is also proposed to replace this template operation. After the top of the pyramided projection is found, it will be now shadowed in the same direction of the projection and the result shadowed in the orthogonal direction, namely -90 degrees related to the projection. A search in the resulting image is then performed to find the position of the

last element with value *true* of the resulting image, starting from the less significant position value in the row or column containing the projection. The complexity of this search is $O(\log N)$, where N is the size in pixels of the related image dimension. Figure 7.3 shows the evolution of these last operations for the example of Figure 7.2.

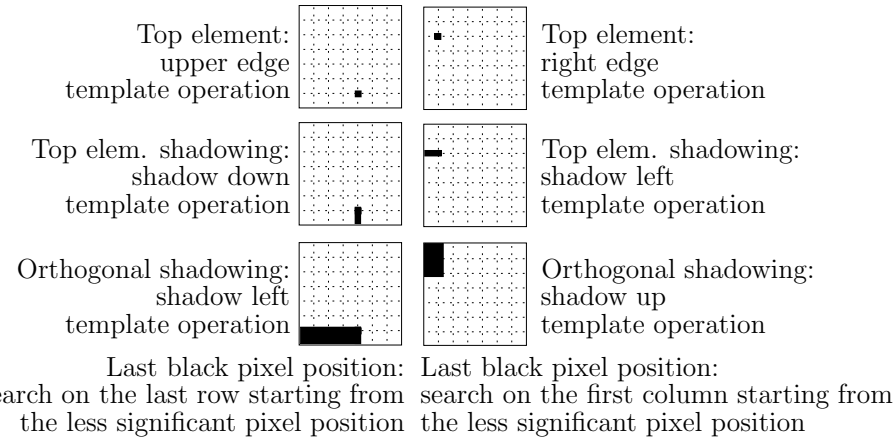


Figure 7.3: Obtaining the coordinates of the object without the diffusion template operation

In order to apply this two-steps approach to lock on and track an object, one may need to pre-process the input images (e.g. with adaptive thresholding, edge detection, hole filling, etc.). Such preprocessing can eventually be done with the CNN-UM architecture too, but is not described here. Another constraint for the algorithm is that the object needs to remain within the image frame in order to be tracked. However, by using the resulting coordinates information one can make sure the object remains within the view area of the camera [39].

7.1.2 Chip-specific robust templates

Despite of the existence of a large range of designed CNN template operations, a reasonable amount of them does not work correctly when executed on VLSI CNN-UM implementations [155]. There are several reasons for that but it is mainly due to manufacturing failure to reproduce in silicon the exact model of the cellular neural network. There exist also in the

literature many robust versions of these templates that, independently of any implementation, increase the chances of correct functioning of CNN devices. However, even the most chip-independent robust template can not always provide correct behaviour on VLSI CNNs.

In order to generate robust templates for the ACE4K device [85] used in our experiments, all templates values used in the proposed algorithm for object tracking with locking were optimised as described in Chapter 3 in order to find their chip-specific robust versions.

7.1.3 Speed and performance analysis

In order to estimate performance and speed, an experimental setup was established with a ACE4K CNN-UM chip (64x64 cells) installed in a DSP board hosted in a digital computer. The images were acquired on-the-fly by a video camera connected to the computer. The algorithm described here has three main parts: image acquisition, locking on the object, and calculation of the coordinates. Being the last part subdivided in horizontal and vertical coordinate calculations. Due to limitations of the camera used in the setup, the maximum frame rate reached was approximately 35 frames per second. Assuming that a faster camera is used or that the images are acquired by an optical input in the chip, which input latency is around 50 microseconds, the algorithm proposed here could reach up to 370 frames per second. The table below shows the average time delays for each part of the algorithm.

Algorithm step	Average delay
Reading from camera	25.20ms
Locking on object	0.50ms
Vertical coordinate calculation	1.00ms
Search for last black pixel in leftest column	0.17ms
Horizontal coordinate calculation	0.82ms
Search for last black pixel in bottom row	0.17ms
Total without reading from camera	2.66ms
Total	27.86ms

As mentioned before, fast frame rate is fundamental to the performance of the algorithm. This has specially effect in the locking step. The maximum object speed which the algorithm can follow is proportional to the

length of the object's transverse cut in the same direction of the movement, l, and to the delivered frame rate, r. This speed is given by the following equation

$$S_{max} = (l-1)r \quad \text{pixel/second},$$

Thus, the worst case scenario is when the object is moving in the same direction of its smallest transverse cut.

7.1.4 Conclusion

In this section we provided a detailed algorithm to lock on and track an object in a video image. The locking allows the object to be tracked even with the presence of other similar objects in the scene. The method may be used in a wide range of applications requiring real-time constraints as hard as 370 frames per second with the use of direct optical image acquisition. The generation of chip-specific robust templates avoided much of the misbehaviour of the applied templates on the CNN-UM chip and made possible the implementation of the proposed algorithm in such device.

7.2 Hands-free wheelchair driving

In this Section, we present an algorithm for tracking features of the human face using a CNN-UM chip implementation. This algorithm is then applied to a hands-free mechanism to drive a wheelchair.

Face tracking is a problem that has been actively studied in the recent years [17, 33, 91]. As part of a larger and more ambitious goal, face tracking has been intended to help on the realisation of perceptual user interfaces. Other parts of such system include face detection, face recognition, gaze point estimation, and finally the translation of information into computer actions like mousing. While the aim of such system can be reasonably broad, it is mainly intended to aid people with a handicap to be able to use the computer. Face tracking can also help these people with their mobility.

CNN[110] technology seems a natural choice to implement on-board wheelchair face tracking due to its high parallelism and reduced size. By placing a CNN visual system on board of a wheelchair and using an object tracking algorithm to track face features, the control of the driving of the wheelchair can be easily translated into movements of the face.

We present a CNN visual algorithm that is fast and robust to track face features in order to control the driving of a wheelchair. For test purposes, we created a three-dimensional model of a wheelchair which reacts upon face movements of the user, who can also interrupt and restart the hands-free driving at anytime.

7.2.1 Face tracking

We describe here the relevant aspects for our face tracking algorithm. We do not present any solution in the direction of face detection nor face recognition. Our algorithm relies on extra information at the initialisation phase in order to locate the face in the initial image and start the tracking.

Tracking alternatives

The orientation of the human face can be calculated in real-time by tracking face features as individual objects. These objects must be clearly identified in the face. The most common choices are:

- eyes,
- nose breaches,
- mouth,
- hair, and
- eyebrows.

All these features stand out reasonably well in a frontal face image. The eyes have the advantage of being rather large. They are long visible as the head rotates and are rather similar among different people. Their disadvantages are the fact that they blink, that some people have to wear eyeglasses and that eyes and eyebrows lay sometimes very close to each other.

Besides having mostly the same advantages as the eyes, the eyebrows do not blink. Their disadvantages are that the colour differs from people to people and that sometimes they are very near the hair.

An alternative are the nose breaches. They are very similar among different people and with a camera placement under the head they become

clearly visible. Additionally, they lay rather isolated of other possible interfering tracking objects. Their disadvantages though are that they become invisible by rotating the head completely down, and that they are rather small. Moreover, they become invisible by people with a moustache.

The advantage of the mouth is that when open it gives a large object that is good for tracking. However, it changes of form and a closed mouth is not a good tracking object. Further on, the mouth does not work as a good tracking object with people with beard and/or moustache.

At last, the hair is a very large object and so it is easy to find. The problem is that it borders to the background, which may interfere with the tracking algorithm. Moreover, it does not change of place in function of the rotation of the head.

The right choice for a tracking object depends thus mostly on the kind of face that needs to be tracked. By considering a face feature as a general object independently of its type, it is possible to develop a general method that can track these different features seamlessly. The algorithm we present here was developed with this objective.

Tracking window: a visual algorithm

In order to efficiently track one of the face features described above, we have developed a visual algorithm whose main principle is to make sure that the object being tracked is always in the centre of a window which floats along a larger streaming input picture. This approach is especially suitable for use with VLSI CNN-UM implementations because such a window can be made equivalent to the chip size.

This algorithm consists of two basic steps.

- First, at each current frame, the object to be tracked must be isolated from the rest of the image. After an image containing only this object is retrieved, it is calculated if the window has to be shifted, in which direction, and how much this shifting is.
- The second and novel step of this algorithm is to move the sliding window accordingly with the position of the object in the current frame. The objective is to make sure that the extracted object always stays within the tracking window.

Isolating the object to be tracked in the current frame can be easily performed as explained in 7.1.1. The part of the object that overlaps in

the current frame with the isolated previous frame can be used to retrieve the object alone in the current frame. This operation can be performed using the recall template by using the current frame as input image and the previous frame with the isolated object as initial state image. At initialisation, when a previous frame does not yet exist, it is necessary to locate the object and define an image with a marker at its location that serves as the previous frame for the first frame.

The object can be kept within the tracking window by a procedure that uses the result of the intersection of 4 pixels with the shadow of the object. The same procedure is executed twice, once for the horizontal direction and once for the vertical one, at each current frame. What follows is a description of the procedure for horizontal component. It is analogous to the vertical one.

First, the shadow operation needs to be applied to the object. The shadow of the object is checked on the bottom line in four pixel locations. Two of these pixels are located on each end of the line and the two other in the centre of the line at equal distance from each other and from the other pixels. The intersections of the shadow with these four pixels are used to define the shifting. The objective here is to make sure that the object's shadow overlaps always with the two centre pixels and avoid the overlapping with the other two. We devise two simple rules to define the necessary shifting:

- If the shadow of the object intersects with only one of the centre pixels, the window is shifted r pixels in the direction of this pixel.
- If the shadow touches one of the edge pixels, the window is shifted s pixels to the pixel direction.

See Figure 7.4 for an illustration of this rules. The values for r and s are determined heuristically. r is obviously directly related to the actual speed of the object, which can only be instantaneously estimated. s is dependent on this speed but also on the size of the object and of the sliding window. Although the ideal values of r and s can be adaptively estimated accordingly to the speed of the object, constant values delivered sufficient performance for our implementation. See Section 7.2.3 for details on these values. Note that despite the sequential reading of maximum 7 pixels, the other only two operations of this method can be fully implemented in parallel with CNN templates.

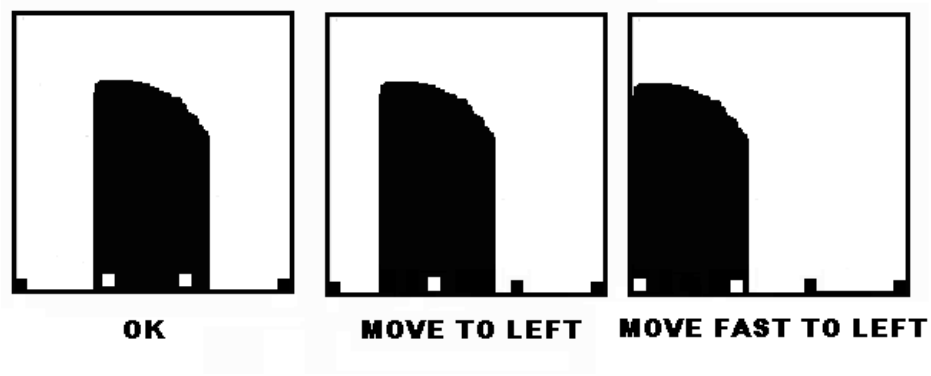


Figure 7.4: The tracking window is shifted if necessary with two different values according to the intersection of four pixels with the shadow of the object. Here the two situations are presented for the horizontal shifting.

This simple procedure produces a very efficient way to track the given object by keeping the object in the centre of the tracking window. Nevertheless, there are two fundamental issues to the algorithm as it was presented so far. First, the use of the recall operation to isolate the object in the current frame only works if the object always overlaps itself in consecutive frames. Fortunately, because this algorithm is designed for a CNN-UM and make use its high parallelism and fast speed, we can achieve sufficient frame rate to ensure the overlapping for most applications.

The second issue is related to the size of the object. In the method described above, the object must always fit in the tracking window and should not be smaller than the distance between the two centre pixels in order to guarantee the correct shifting. In real-life applications, the object to be tracked varies constantly in size and form due to depth movements, lighting, rotation, etc. The following Section describes a methodology to overcome this problem.

Adaptive object resizing

Tracking an object that is constantly changing in size and form presents a problem to the method described above when it does not fit in the tracking window or when it becomes smaller than the distance between the two centre pixels. In both cases, the direction of the shifting becomes impossible

to be calculated. In order to solve this problem, it is necessary first to monitor the size of the object and then apply some action to the image so that the object can be resized to an acceptable size. Such action can be e.g. either zooming, a change in lighting, adjusting the threshold level, adjusting the objective shutter speed, or a combination of them. We have chosen to adjust the shutter speed and the threshold value of the grey-scale to binary conversion.

These values are adapted using the readings of the pixels from the window shifting principle, i.e. no extra reading operation is required. If the object touches both edges of the tracking window, the two outer pixels are overlapped by the object's shadow, which means that the object has increased in size, e.g. because of too little illumination, and it no longer fits in the tracking window. In this case, the algorithm adjusts first the shutter speed. If the speed cannot be decreased further, the threshold value is then adjusted. On the other hand, when the object is being tracked and it decreases too much in the size such that it becomes smaller than the distance between the two centre pixels, the object's shadow does not cover any of these pixels and thus the threshold and shutter speed values need to be adjusted in such a way that the object appears larger in the image. See Figure 7.5 for an illustration of the two cases.

With such a procedure to complement the method described in the previous Section, we have an efficient algorithm to track objects which are subject to constant size variation in its image projection. Although such a procedure already increases significantly the robustness of the method, in the next Section we present a strategy which can deal with other occasional problems, e.g. partial or complete occlusion of the object.

Robust face tracking

In order to increase the robustness of face tracking method presented above, we propose the tracking of multiple face features instead of only one. This boosts the reliability of the whole process because if one of these objects gets lost from the tracking, it is often possible to retrieve it by the relatively fixed geometry of the face. For that, the positions of the objects that are still being tracked are used to estimate a marker for the lost object. It is clear that the more objects that are being tracked, the larger the robustness of the tracking system will be. Nevertheless, this is only true up to a certain level. Although the tracking of one single object can be done in parallel

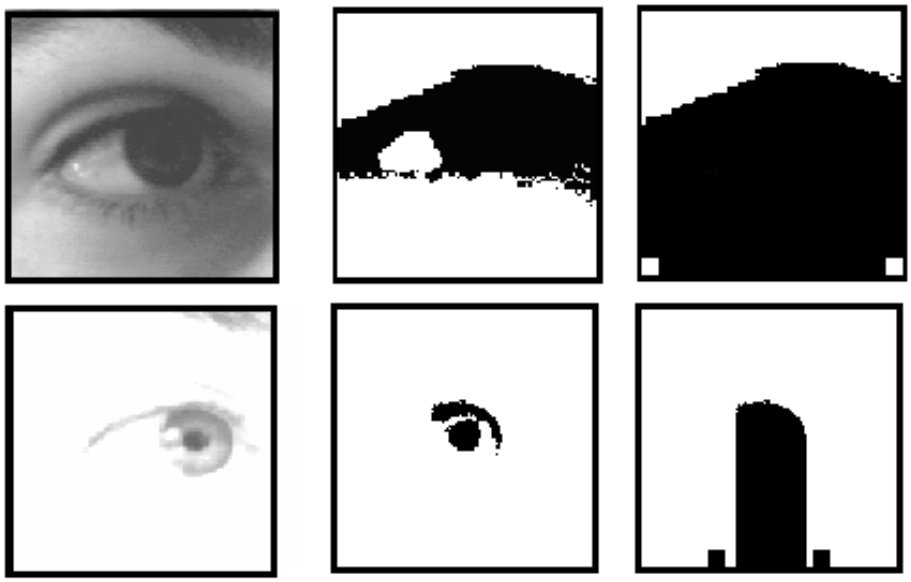


Figure 7.5: The resizing algorithm changes the parameters the control the object's size according to the intersection of its shadow with four pixels. On the top row of the figure, there is too little light and a longer exposure time is needed; On the bottom row, there is too much light and a shorter exposure time is needed.

in the CNN-UM, the tracking of the different face features is sequential. The addition of more objects to the tracking results in a reduction of the maximum frame rate of the whole system. Consequently, it results in a reduction in the maximum speed in which the object being tracked can still be followed. Therefore, the number of objects to track simultaneously must be wisely traded off with the frame rate of the system so that the overall robustness prevails.

The combination of the window shifting and adaptive resizing algorithms with multiple tracking results in a very simple and efficient algorithm for face tracking. An overview flowchart of the tracking process which involves these three features is presented in Figure 7.6.

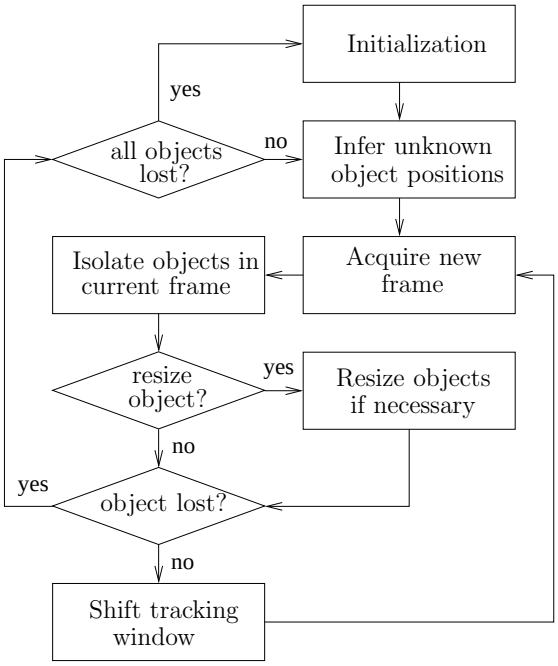


Figure 7.6: An overview of the different stages of the multiple object tracking system.

7.2.2 Driving of a wheelchair

The challenge here is to implement a hands-free driving mechanism for an automated wheelchair in order to give the user the ability to move the chair forward, backwards, turn left, and right. Additionally, the user should be able to start or interrupt the driving at any time also without the use of hands. Finally, this mechanism should be robust enough to work in different environments with different illumination types.

In order to realise such a mechanism, we propose the use of the movements of the user's face to initialise, move, and interrupt the movements of the wheelchair. A procedure for the initialisation and interruption of the movements needs to be created. This way, while driving the chair, the user can at any time stop the controlling, move his/her head freely, and restart the driving. The mechanism needs thus two distinct states corresponding to active and passive tracking. When in passive tracking, the face of the

user is being tracked but its movements do not result in any motion for the wheelchair. The system must be waiting for a start command which can be e.g. a sequence of predefined movements. The active tracking is the state which most of the face movements yield in motion to the wheelchair. At this stage, the system must be also aware of a predefined interruption command.

We have created a protocol to translate the face tracking into commands to the wheelchair. In the passive state, the system must wait for a fast motion of the head in the horizontal direction. While in active driving, the commands to go forward, go backwards, turn left, and turn right were associated with the movements of the head look up, down, to the left, and to the right, respectively. In order to stop the driving and go into the passive mode, the user must again wave his/her head in the horizontal direction. Note that this could be ambiguous with the turning commands, but there is a practical solution. The turning commands can be delayed in such a way that these fast horizontal movements would not be actually translated into wheelchair movements because the passive mode would have already taken oven. In general, this is a very simple and efficient protocol for testing purposes. Nevertheless, in a final implementation more sophisticated protocols might be more interesting to deal with circumstances that were uncovered here, e.g. when the user expresses a 'no' by moving his/her head in passive mode, without the intention to switch the state of the system to the active mode.

Another aspect that should be covered in a final implementation is the initialisation of the tracking itself. This must be a fast and straightforward procedure in order to rapidly recover from a tracking failure. Although there are many options for such a procedure, we propose one here which we believe fits well many requirements. Aligned with the camera which captures the image of the user's face, we propose to place a light beam, e.g. oriented LED light, which shines every time a failure occurs, i.e. all the face objects being tracked are lost. Upon a failure, the user must thus place his eye in front of the light beam. The place in the image where this light shines is the place where the initial marker for the first frame is. After the eye is successfully tracked, the light beam goes off. In the sequence the positions of the other objects are calculated relatively to the position of the eye. Besides providing an efficient way to recover from failure, this method also provides the user with the information about the status of the tracking

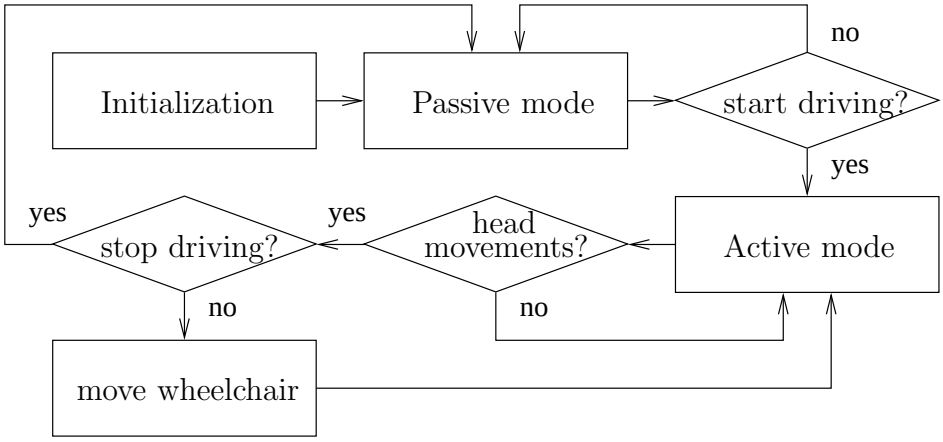


Figure 7.7: Overview of the wheelchair driving mechanism.

through the status of the light beam.

Figure 7.7 presents and overview of the main features of the wheelchair driving mechanism.

7.2.3 Implementation and practical considerations

We implemented both face tracking visual algorithm and wheelchair driving protocol in a Bi-I system [164] with an ACE16k_v2 [109] CNN-UM chip placed at one of the optical inputs, and a higher definition CCD camera placed at the other input. Besides the ACE16k_v2 and the CCD camera, the Bi-I also embeds a DSP.

During the implementation we have encountered a number of problems related to the CNN chip and the CCD camera. A description of these problems follows in the next Sections together with the solutions that were applied.

Tuning of CNN templates and chip-specific robustness

Unfortunately, templates designed to work in ideal CNNs are not guaranteed to work on analog VLSI CNN implementations [156]. The reason for that lies mainly on manufacturing mismatches, which for analog VLSI are around 10%. Therefore, in order to make use of these templates in a Bi-I, it is necessary to tune the template values for the specific chip to be used.

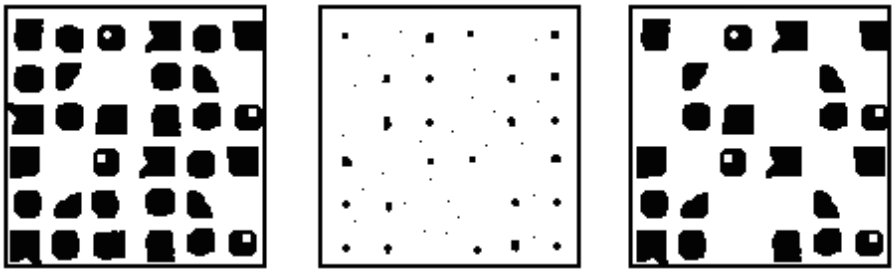


Figure 7.8: Training set for the recall template. Input, initial state, and desired output are shown in the picture.

With this purpose, we used the framework in Chapter 6 in order to tune the recall and shadow templates.

For the recall template, the images in Figure 7.8 were used as training set. Only the tuning of the hardware parameters was necessary to achieve a good working template. Observe that many single dots were added to the marker image. This was necessary to make sure that noise in the initial state image would not recall undesirable objects from the input image.

The same training set design strategy was used for the shadow template, see Figure 7.9. The addition of noise to the input and initial state images in the training phase avoided that resulting output images would also shadow noisy pixels. Moreover, in the tuning it is possible to optimise the time necessary to perform the operation. This way, our shadow operation can be performed in the shortest time possible for the chip, improving the overall speed of the tracking algorithm.

Maximal shifting and minimal frame rate

The window shifting algorithm described in Section 7.2.1 relies on the recall operation to precisely re-position the window in such a way that the object is always in the centre of the image. On the other hand, the recall operation also relies on the right window shifting in order to ensure the overlapping of the object in the previous and current frame, see Figure 7.10. Therefore, the values for the two different shifting values r and s need to be defined with precaution.

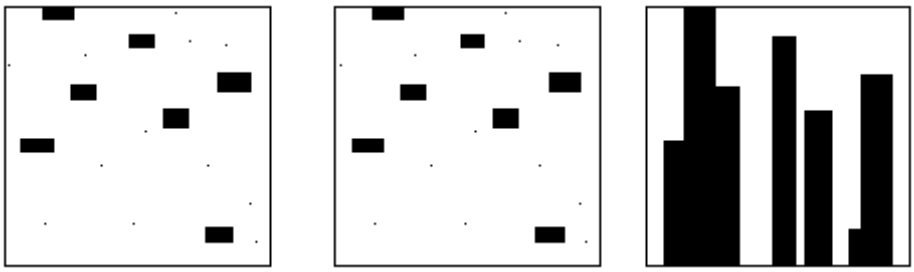


Figure 7.9: Training set for the shadow south operation. Input, initial state, and desired output are shown in the picture.

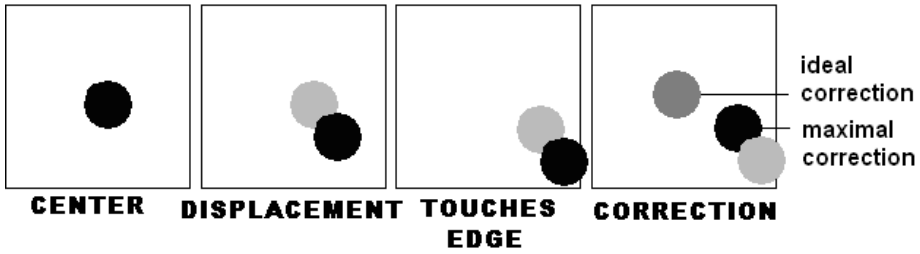


Figure 7.10: Defining the maximal correction. Black: current frame, grey: previous frame. The ideal shifting cannot be applied due to the recall operation that always needs overlapping.

The value of r obviously needs to be sufficiently small such that the object is stabilised in the centre of the window upon object stop or even little movement. We have used a value of two pixels for the r shifting. When the object is moving with a larger speed than the r shifting can follow, it eventually reaches the border of the image. The s shifting is applied at this point. The ideal value for s would be to bring the object to centre of the window. However, depending on the size of the object, the recall operation would promptly fail. This shifting thus must not be larger than the object diameter plus is actual speed. Empirically, good results were achieved with $s = 24$ for a window of size 128 pixels.

The recall operation also can cause the loss of the object when the frame rate becomes too slow. It can happen that wrong objects become also recalled. Upon an abrupt movement it can occur that the overlapping

region of the object in two successive frames does not only contain the object being tracked but also another object nearby, e.g. by a fast movement of the head in the vertical direction, it can happen that the eye in the previous frame does not only overlap the eye in the current frame but also the eyebrow. The eyebrow and the eye become member of the initial state marker and are tracked together. This in itself is not yet a problem for this application. However, if this eyebrow overlaps with the hair, now the eye, the eyebrow, and the hair become tracked. All these components do not fit in the tracking window and the tracking system would fail to work properly. In this situation, our system assumes that the objects is lost by checking if it cannot be resized back into the window.

Illumination issues

One of the most important aspects of processing images is that they need to be of good quality. There are two important aspects in order to acquire good images. The object being photographed must be sharp and the lighting must be appropriate. It may not be too dark nor too bright. Sharpness is generally no problem for the wheelchair case. It is assumed that the wheelchair user always sits on approximately the same distance of the camera. The lens must be adjusted only once to make the tracking objects look sharp. On the other hand, solving the illumination problem is more complex. Since the intention is that the wheelchair drives around, the light intensity might change with that. The parameters that modify the picture illumination must change according to the quantity of the light that is available at that moment. These parameters can be either the opening of the lens, the objective shutter speed, the sensitivity of the CCD or the threshold value for the conversion of grey-scale to binary pictures. The most obvious choice of a parameter to compensate different illumination is the adjustment of the shutter speed or, illumination time, i.e. the time which the CCD sensor is exposed to the light. In the Bi-I, the waiting for this time can be parallel to the CNN template operation for the previous frame. Since the CNN template operations for the object tracking are normally faster than the standard Bi-I illumination time, this becomes a bottleneck for the whole system and therefore must be optimised. Our first step in this direction was to set the value for the objective opening up to the maximum. This causes loss in depth sharpness, which imposes almost no problem since the user of the wheelchair sits always approximately at

the same distance of the camera. The sensitivity of the CCD sensor and the binary threshold are other two additional values that we have optimised in order to decrease the illumination time. When close to extreme values, these parameters bring noise into the image and therefore must be carefully used.

Testing with a 3D wheelchair model

In order to test the driving system in real-time, we have created in Virtual Reality Modelling Language (VRML) a three-dimensional (3D) environment with a 3D model of a wheelchair. The goal was to make the 3D wheelchair move as an actual wheelchair according to our face tracking mechanism. The commands from the Bi-I were transmitted to the 3D wheelchair via the Matlab VRML toolbox. Figure 7.11 depicts the VRML environment and an overview of our setup.

In order to increase robustness, we followed not one but two face features, the left and right eyes. The initialisation procedure was performed by positioning the left eye in the middle of the initial tracking window, where the marker for the first frame was. After the left eye was found and locked for tracking, the right eye is immediately inferred and another marker is created at its inferred position.

We obtained a frame rate of 92 frames/s for the tracking of the two eyes. This frame rate decreases to an average of 35 frames/s when poor illumination is present. With ideal illumination, optimised memory management, and further tuning of the templates involved, we believe frame rates over 300 frames/s could be reached for a single object.

7.2.4 Conclusions

In this Section, we have presented a visual algorithm for face tracking which is fast and robust enough to be applied to a hands-free wheelchair driving system. The wheelchair user can drive the chair forward and backwards and steer it to the left or to the right accordingly to his/her head movements. Additionally, the user can interrupt or restart the driving at any moment by predefined head movements. The system also adapts to a different range of illumination intensities. Additional robustness could be achieved by tracking multiple face features at the same time. This way, if one of these features is lost, its position can be inferred accordingly with the position

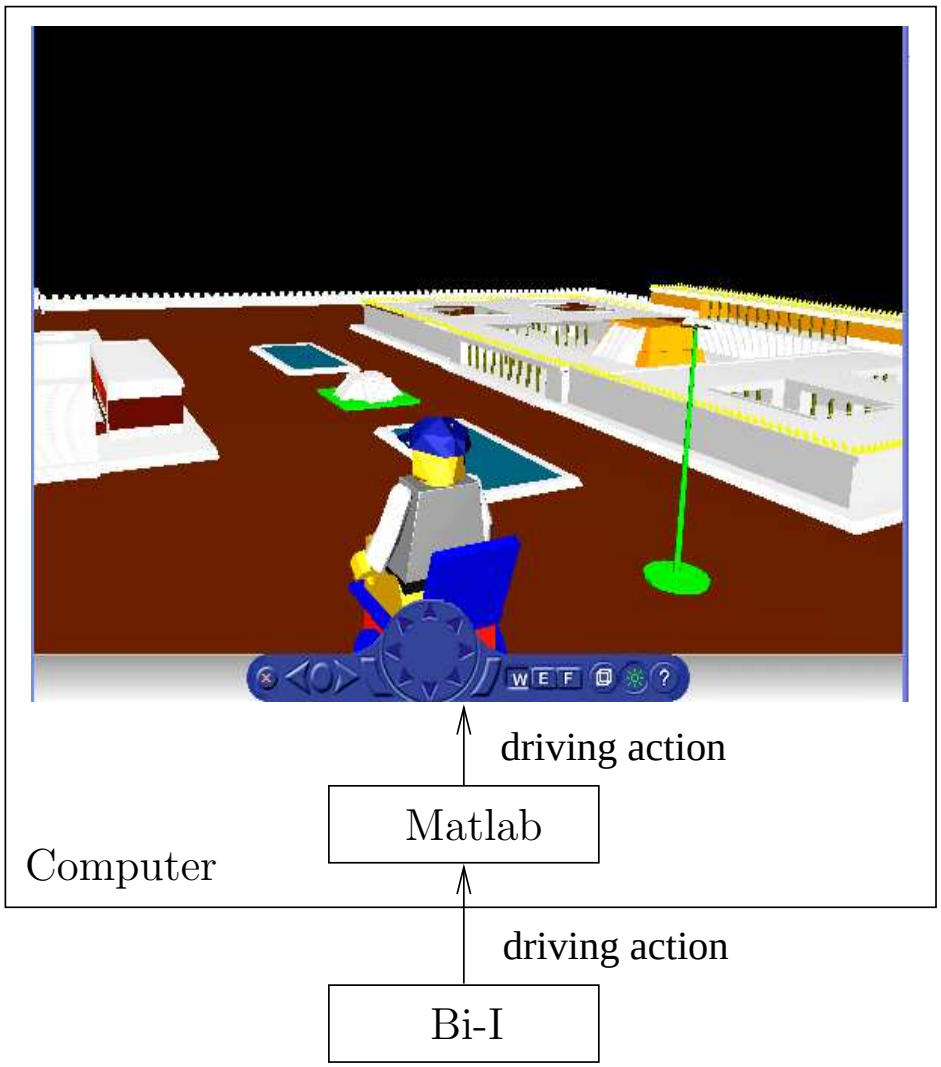


Figure 7.11: Test setup for the real-time driving of a 3D model wheelchair using the head movements. Matlab is merely a communication channel between the Bi-I and the 3D environment.

of the features that are still being tracked, then the lost feature can be reinserted into to the tracking algorithm. Tests of this system in a 3D simulation environment have shown that an actual physical implementation is feasible.

Chapter 8

General Conclusions

8.1 General conclusions

The methodologies described throughout the Chapters of this thesis have as main objective to improve the robustness of existing and forthcoming analog VLSI implementations of CNNs. As seen in Chapter 2, CNNs are not only a paradigm for complexity but they can also be used as a platform for computation. Since the time the CNN-UM, a programmable CNN, was invented, many *analogic* algorithms have been developed. With the modern VLSI CNN-UM circuit implementation, these algorithms can be executed in extraordinary computational speeds, only comparable to today's supercomputers. The one-to-one correspondence between pixel and cells, provides a perfect architecture for image processing applications because it allows full network parallelism. This without mentioning the reduced size and power consumption of these devices, which make them easy to be embedded in portable appliances.

Although virtually any SIMD image processing operator can be implemented in modern CNN-based processors, and despite all advantages mentioned above, these programmable CNNs are still far from replacing even small portions of today's DSP systems. While the majority of today's computation relies on well established digital VLSI technology, CNN-based processors rely on analog VLSI technology, which despite the remarkable recent advances, still cannot guarantee the accuracy that CNNs requires to function reliably. Templates that were designed to be robust against analog VLSI parameter deviations still present erroneous behaviour when

finally executed in real CNN VLSI chips. The methodologies described in Chapter 3, proved to be very useful in reducing and even eliminating these errors. The reason why it works is that we search not only for optimal CNN template values but focus on robustness, which means that we search for the most robustly optimal template values. This is possible because we optimise each operation targeting a specific chip, which means that for this chip the final template values are in the middle of the correct operating range. This makes the template not only optimal against the parameter deviations introduced during the manufacturing process, but also robust against post-manufacturing disturbances, such as temperature variations and electrical noise.

Optimising the template values using direct chip measurements can also be useful for other purposes other than tuning. If instead of a simulator in a digital computer, we use a CNN chip to evaluate the cost function for a CNN learning process, we can learn new operations much faster. This is especially true for spatiotemporal dynamics, which demands the evaluation of many different output snapshots. In Chapter 4, we present a methodology based on trajectory learning with RNNs to learn CNN spatiotemporal behaviour. This is especially suitable for learning of new active wave operation or for chip-specific tuning of existing ones. The same methodology can also be proven to be useful for adjusting the speed of existing CNN template operations.

Both methodologies described in the Chapters 3 and 4 depend on an optimisation core. For that, we have used in the beginning of our research an existing global optimisation method called Adaptive SA. Although considered to be very robust and fast, we experienced that often the optimisation must be stopped and re-initiated with different parameters due to premature convergence to a local minimum. Inspired by the working of the coupling in CNNs, in Chapter 5 we present a class of global optimisation methods called CSA which uses coupling to guide the optimisation toward global optima. We have shown with an instance method of this class that it is possible to steer the optimisation by controlling the variance of the acceptance probabilities with the acceptance temperature. This leads to quasi-optimal runs, which are much less sensitive to initialisation parameters than in classical SA.

In Chapter 6 we finally presented a unification framework of all methodologies described previously in the thesis. In this framework, we use CSA

as the optimisation core for tuning and learning of fixed-point and spatiotemporal operations in CNN-based processors. This framework is also presented as a Matlab toolbox which can be used by the analogic CNN algorithm designer in order to optimise template operators. One of the only remaining manual tasks for the designer is also one of the most important ones. The design of the training set of the input, state and desired images still remains a task for the designer. Given the importance of this task for the correct operation of the template to be optimised, we have defined a set of rules to ease the task of creating a correct training set. However, a deep understanding of the desired operation is still required.

In Chapter 7 we presented a couple of CNN-based applications which serves as proof-of-concept for our chip-specific CNN optimisation methodologies.

The techniques for post-manufacturing enhancement of CNN-based processors described here can help the establishment of CNN technology in areas that are dominated by classical digital computation. Ultra-fast visual processors have a large potential application area. Robust CNN-based processors could be applied for example on visual quality control in agricultural, semiconductors, textile and other industries; on surveillance and traffic analysis; on microbial process inspection; on intelligent airbag systems; on aid to people with limited mobility; on visual gaming interaction, and on many other visually based applications. However, the problem of parameter spread limits this gamut of applications to only a few simpler applications where robustness is not of primary importance. The methodologies described in this thesis have proven that it is possible to reduce or even eliminate this problem. The further development of these post-manufacturing enhancement methodologies is therefore essential to make these analog systems more competitive. The alternative solution to solve this robustness problem would be to switch from analog to digital technology. Although such a twist is possible, most of the advantages of the original analog system, w.r.t. power, silicon area, and speed, would be considerably reduced. At this point it is important to emphasise again that because of the analog nature of these systems they are able do what only supercomputers would be able to, in the digital world. The only advantage that a fully digital CNN chip would keep after morphing from its analog counterpart would be the high parallelism. In fact, the trend in the digital processor industry toward multi-core shows that parallelism is indeed

necessary in the future of general purpose computation. However, w.r.t. power, silicon area, and computation speed, such advanced and mature multi-core systems are still pretty much behind the relatively recent and almost experimental CNN-based systems. Indeed the only remaining challenge for the latter is ensuring robustness against parameter spread. This thesis shows that this too is possible to the solved.

8.2 Challenges for future work

Although the maturity of the techniques presented here in combination with the recent and forthcoming advances in analog VLSI technology have the potential to bring CNN-based computation toward the mainstream of computation, this path is still not paved and therefore needs all the efforts that can be spent from both fronts. With respect to chip-specific post-manufacturing enhancement methodologies, we present here some ideas that can serve as a road-map for forthcoming research:

- For some template operations, the maximum robustness that can be obtained is still not sufficient to ensure correct operation in analog VLSI CNN chips. For these extreme cases, a methodology for decomposing these templates into sub-templates that can allow larger robustness might be a viable alternative. Földesy *et al.* presented such a decomposition approach for a limited set of templates, namely the non-propagating ones. The development of an automatic and unrestricted decomposition methodology that can be incorporated with the chip-specific CNN optimisation approaches described in this thesis can enlarge the gamut of highly robust CNN template operations.

- The CNN-UM is a very flexible computer architecture. By using cascading templates, it might be possible to generate a solution for emulating multi-layer CNNs with a single-layer CNN-UM chip. This solution can possibly be found with help of an approach based on the spatiotemporal learning methodologies described in Chapter 4.

- Modern CNN-based processors have a set of hardware reference values that can be adjusted by software. In Chapter 6 we presented a framework that can use these values to optimise specific template operators. With the right training set, it might be possible to identify

the best general purpose values for this set of hardware references. The challenge here is to devise a training set that is general enough to tune these value in an unbiased way.

- Instead of tuning of single template operations, it might be interesting attempting to tune entire analogic algorithms.

- The method described in this thesis uses a host computer with Matlab to optimise the given CNN operation. The part of the system handled in this host PC is related to the optimisation procedure. If an optimisation procedure can be developed to run in the CNN-based processor itself, it would mean a step closer toward self-tuning CNN systems. Such systems could make the difference in many uncontrolled, unstable, or variable application environments.

Appendix A

Adaptive Simulated Annealing

The Adaptive Simulated Annealing algorithm is a very robust, yet flexible, optimisation method that allows different parameters to have distinct finite ranges. Each parameter also have distinct sensitivities. These are measured by the immediate gradient at a local minimum and are dependent on the annealing time. The probing parameters p_i, with $i = (1, ..., D)$, are randomly generated from the cumulative probability distribution

$$P_i^T(\gamma_i) = \frac{1}{2} + \frac{\mathrm{sgn}(\gamma_i)}{2} \frac{\ln\left(1 + \frac{|\gamma_i|}{T_i}\right)}{1 + \frac{1}{T_i}},$$

where

$$\gamma_i = \mathrm{sgn}\left(u_i - \frac{1}{2}\right) T_i \left[\left(1 + \frac{1}{T_i}\right)^{|2u_i - 1|} - 1\right]$$

is generated from the uniform distribution $u_i \in U[0, 1]$, with $p_i^{j+1} = p_i^j + \gamma_i(p_{max,i} - p_{min,i})$. The annealing temperatures are scheduled according to

$$T^j = T^0 \exp(-cj^{\frac{1}{D}}), \tag{A.1}$$

where T^0 represents the initial temperatures, and c is an ASA adjust parameter. The acceptance temperature is analogously scheduled at each accepted point. The temperatures are re-annealed after a given number of

accepted points, e.g. 100, according to the formula

$$T_{new}^{j} = T^{j} \left(\frac{s_{max}}{s} \right), \tag{A.2}$$

where the sensitivities $s = \frac{\partial g}{\partial p_i}$ are calculated at the most current minimum value of the cost function (3.2). The indices j are updated isolating them from (A.1) and substituting (A.2).

Appendix B

CSA and MSA Results for Test Functions in Higher Dimensions

The following figures are related to the experiments realised for the three groups of test functions in Chapter 5.

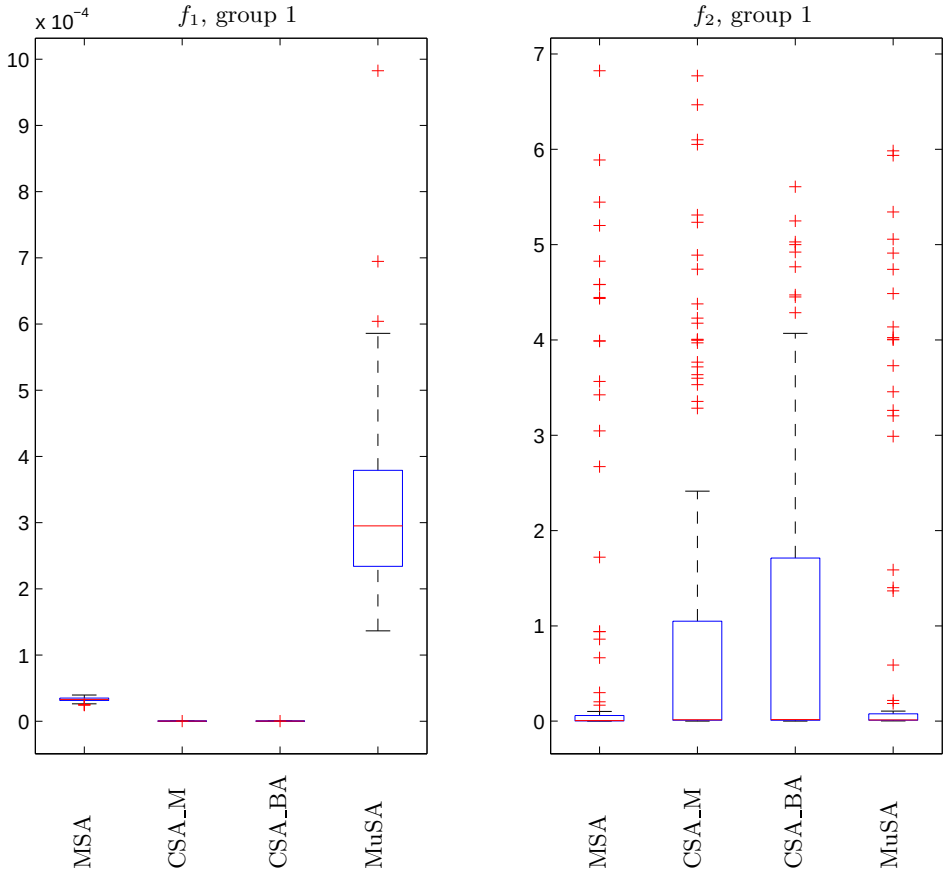


Figure B.1: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 1 in 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 200,000 per parallel process with 900 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

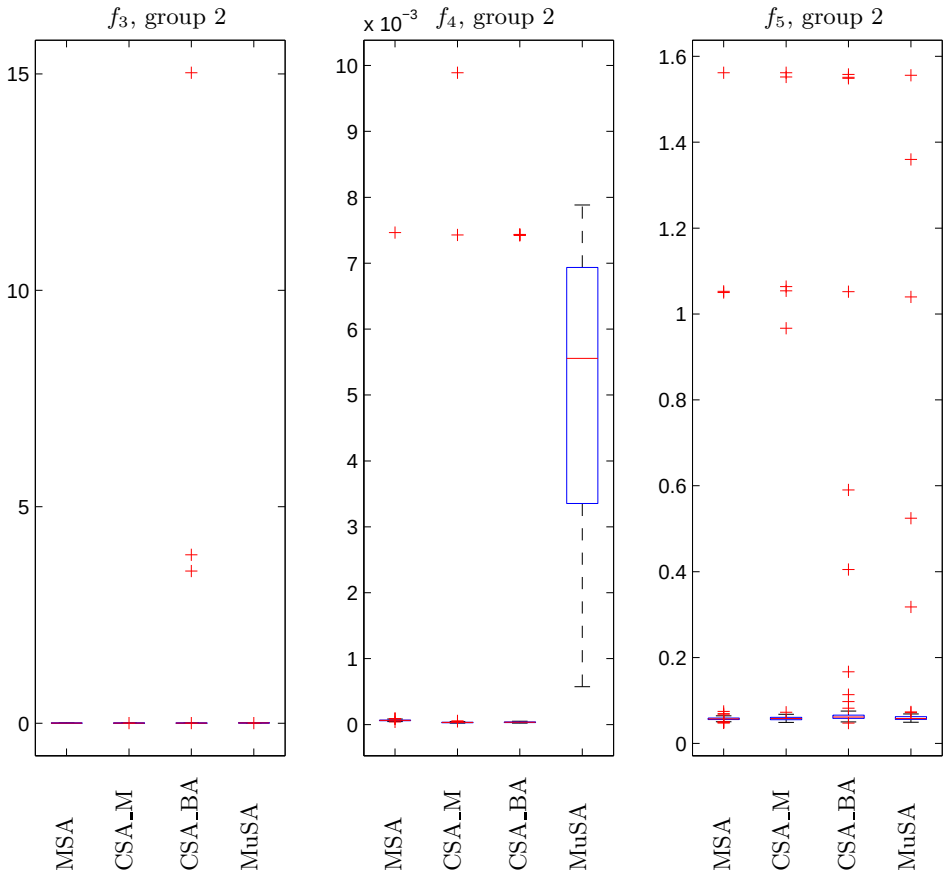


Figure B.2: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 2 in 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 200,000 per parallel process with 900 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

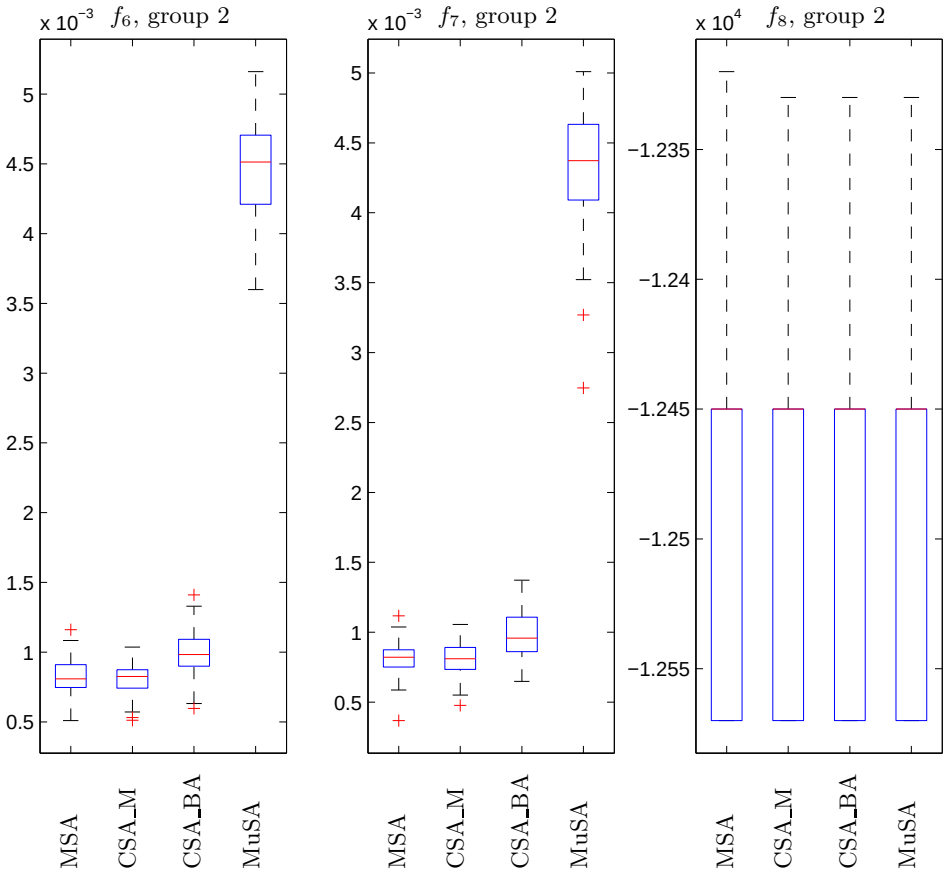


Figure B.3: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 2 in 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 200,000 per parallel process with 900 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

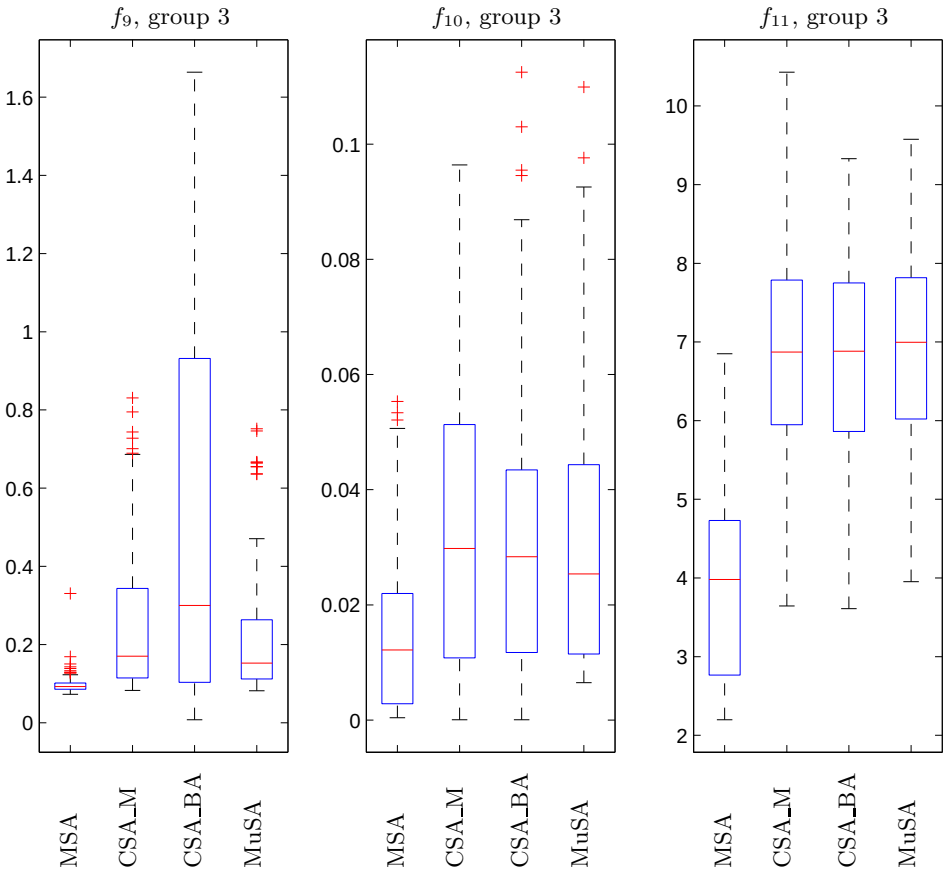


Figure B.4: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 3 in 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 200,000 per parallel process with 900 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

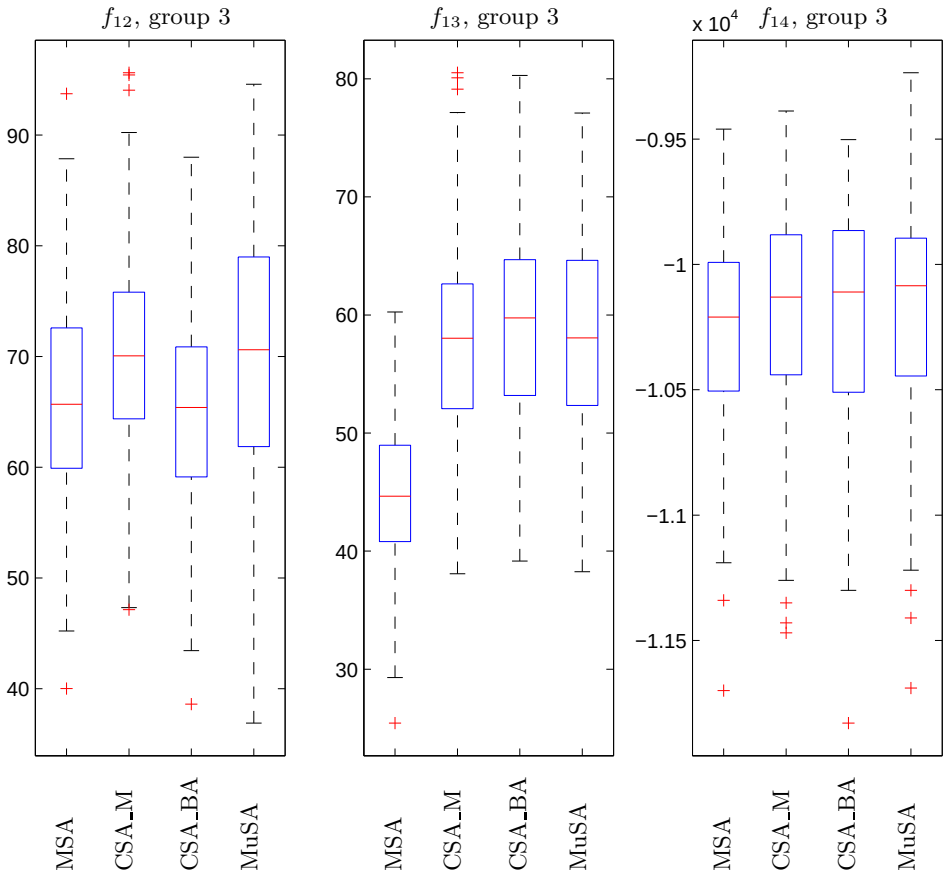


Figure B.5: Box plots for experiments with four algorithms (horizontal axis) for the functions in test group 3 in 30 dimensions. In the vertical axis we have the final costs of each cost function. Each method used a maximum number of function evaluations equals to 200,000 per parallel process with 900 steps per fixed temperature. Initial generation and acceptance temperatures were chosen from a predefined set after exhaustive search.

Bibliography

[1] AnaFocus. http://www.anafocus.com/.

[2] EU research project: Dynamic image computing via teraops analogic machines. http://istresults.cordis.europa.eu/index.cfm/section-/news/tpl/article/BrowsingType/Features/ID/62925/highlights-/DICTAM.

[3] Message Passing Interface (MPI) Standard. http://www.mpi-forum.-org/.

[4] VIC - High-performance Linux Cluster. http://ludit.kuleuven.be-/hpc/.

[5] A. Adamatzky, P. Arena, A. Basile, R. Carmona-Galan, B. D. L. Costello, L. Fortuna, M. Frasca, and A. Rodríguez-Vázquez. Reaction-diffusion navigation robot control: from chemical to VLSI analogic processors. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 51(5):926–938, May 2004.

[6] M. A. Allesie, F. I. M. Bonke, and F. J. G. Schopman. Circus movement in rabbit atrial muscle as a mechanism of tachycardia. *Circulation Research*, 14:54, 1973.

[7] Analogic Computers. Aladdin Visual Computer. http://www.-analogic-computers.com, 2003.

[8] P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. Neural Networks*, 5(1):54–65, Jan 1994.

[9] P. Arena, S. Baglio, L. Fortuna, and G. Manganaro. Chua's circuit can be generated by CNN cells. *IEEE Trans. on Circuit and Systems-I*, 42:123–125, 1995.

[10] P. Arena, S. Baglio, L. Fortuna, and G. Manganaro. Generation of n-double scrolls via cellular neural networks. *International Journal of Circuit Theory and Applications*, 24:241–252, 1996.

[11] P. Arena, S. Baglio, L. Fortuna, and G. Manganaro. Self-organization in a two-layer CNN. *IEEE Trans. on Circuits and Systems-I*, 45(2):157–162, Feb 1998.

[12] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, Mar 1994.

[13] A. Bevilacqua. A Methodological Approach to Parallel Simulated Annealing on an SMP System. *J. Parallel and Distributed Computing*, 62:1548–1570, 2002.

[14] M. Bianchini, M. Gori, and M. Maggini. On the problem of local minima in recurrent neural networks. *IEEE Trans. Neural Networks*, 5(2):167–172, Mar 1994.

[15] N. Blachford. Cell architecture explained, version 2. http://www.blachford.info/computer/Cell/Cell0_v2.html.

[16] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. http://www.stanford.edu/ boyd/cvxbook/.

[17] G. R. Bradski. Computer Vision Face Tracking For Use in a Perceptual User Interface. *Intel Technology Journal*, Q2:15, 1998.

[18] V. M. Brea, D. L. Vilarino, and D. Cabello. A mixed-signal CMOS DTCNN chip for pixel-level snakes. In *IEEE International Symposium on Circuits and Systems (ISCAS 2004)*, volume 5, pages V–465–V–468, 2004.

[19] R. Carmona-Galán, F. Jimênez-Garrido, R. Domínguez-Castro, S. Espejo, T. Roska, Cs. Rekeczky, I. Petrás, and A. Rodríguez-Vázquez. A bio-inspired two-layer mixed-signal flexible pro-

grammable chip for early vision. *IEEE Transactions on Neural Networks*, 14(5):1313 – 1336, Sept 2003.

[20] R. Carmona-Galán, F. Jimenez-Garrido, C. M. Domínguez-Mata, R. Domínguez-Castro, S. E. Meana, I. Petras, and A. Rodríguez-Vázquez. Second-order neural core for bioinspired focal-plane dynamic image processing in CMOS. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 51(5):913–925, May 2004.

[21] B. Chandler, Cs. Rekeczky, Y. Nishio, and A. Ushida. Adaptive Simulated Annealing in CNN Template Learning. *IEICE Trans. Fundamentals*, E82(2):398–402, Feb. 1999.

[22] H. Chen, N. S. Flann, and D. W. Watson. Parallel Genetic Simulated Annealing: A Massively Parallel SIMD Algorithm. *IEEE Trans. Parallel and Distributed Systems*, 9(2):126–136, Feb 1998.

[23] L. O. Chua. *CNN: A Paradigm for Complexity*, volume 31 of *World Scientific Series on Nonlinear Science, Series A*. World Scientific, Jun. 1998.

[24] L. O. Chua. Local activity is the origin of complexity. *International Journal of Bifurcation and Chaos*, 15(11):3435–3456, 2005.

[25] L. O. Chua, M. Hasler, G. S. Moschytz, and J. Neirynck. Autonomous cellular neural networks: a unified paradigm for patternformation and active wave propagation. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 42(10):559–577, Oct 1995.

[26] L. O. Chua, T. Roska, T. Kozek, and Á. Zarády. CNN Universal chips crank up computing power. *IEEE Circuits and Devices*, 12(4):18–28, 1996.

[27] L. O. Chua and L. Yang. Cellular Neural Networks: Applications. *IEEE Trans. Circuits and Systems*, 35:1273–1290, 1988.

[28] L. O. Chua and L. Yang. Cellular Neural Networks: Theory. *IEEE Trans. Circuits and Systems*, 35:1257–1272, 1988.

[29] B. Cohen, D. Saad, and E. Marom. Efficient training of recurrent neural network with time delays. *Neural Networks*, 10(1):51–59, Jan 1997.

[30] K. R. Crounse and L. O. Chua. Methods for image processing and pattern formation in Cellular Neural Networks: a tutorial. *IEEE Trans. Circuits and Systems*, 42(10):583–601, Oct. 1995.

[31] J. M. Cruz and L. O. Chua. A CNN chip for connected component detection. *IEEE Transactions on Circuits and Systems*, 38(7):812–817, Jul 1992.

[32] J. M. Cruz, L. O. Chua, and T. Roska. A fast, complex, and efficient test implementation of the CNN universal machine. In *International Workshop on Cellular Neural Networks and Their Applications (CNNA 1994)*, pages 61–66, Rome, Italy, 1994.

[33] T. Darrell, B. Moghaddam, and A. Pentland. Active face tracking and pose estimation in an interactive room. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'96)*, pages 67–72, San Francisco, CA, June 1996.

[34] R. Domínguez-Castro, S. Espejo, A. Rodríguez-Vázquez, and R Carmona. A CNN universal chip in CMOS technology. In *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and their Applications CNNA-94*, pages 91–96, Dec 1994.

[35] R. Domínguez-Castro, S. Espejo, A. Rodríguez-Vázquez, R. A. Carmona, P. Földesy, P. Zarándy, Á.; Szolgay, T. Sziranyi, and T. Roska. A 0.8-μm CMOS two-dimensional programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage. *IEEE Journal of Solid-State Circuits*, 32(7):1013 – 1026, Jul 1997.

[36] S. Espejo, R. Carmona, R. Domínguez-Castro, and A. Rodríguez-Vázquez A. A VLSI-oriented continuous-time CNN model. *Int. J. of Circuit Theory and Applications*, 24(3):341–356, May-Jun 1996.

[37] S. Espejo, R. Domínguez-Castro, R. Carmona, and A. Rodríguez-Vázquez. A CNN Universal Chip in CMOS Technology. *International Journal of Circuit Theory and Applications*, 24:93–109, Jan-Feb 1996.

[38] P. Földesy, L. Kék, Á. Zarándy, and G. Bártfai T. Roska. Fault-Tolerant Design of Analogic CNN Templates and Algorithms—Part

I: The Binary Output Case. *IEEE Trans. on Circuits and Systems-I*, 46(2):312–322, February 1999.

[39] A. Gacsádi and P. Szolgay. An anologic CNN algorithm for following continious moving objects. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'2000)*, pages 99–104, Catania, Italy, May 2000.

[40] V. Gál and T. Roska. Collision Prediction via the CNN Universal Machine. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'2000)*, pages 105–110, Catania, Italy, May 2000.

[41] M. Galicki, L. Leistritz, and H. Witte. Learning continuous trajectories in recurrent neural networks with time-dependent weights. *IEEE Trans. Neural Networks*, 10(4):741–756, Jul 1999.

[42] M. Galicki, L. Leistritz, E. B. Zwick, and H. Witte. Improving generalization capabilities of dynamic neural networks. *Neural Computation*, 16(6):1253–1282, Jun 2004.

[43] V. Gellens and M. Deman. Sturing van de computermuis als toepassing van real-time CNN-UM Object Tracking. Master's thesis, K.U.Leuven. Faculteit Ingenieurswetenschappen. Departement Elektrotechniek (ESAT), 2005.

[44] P. P. Gelsinger, P. A. Gargini, G. H. Parker, and A. Y. C. Yu. Microprocessors circa 2000. *IEEE Spectrum*, 26(10):43–47, 1989.

[45] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6(6):721–741, Nov 1984.

[46] M. Gilli and P. Paolo-Civalleri. Template design methods for binary stable cellular neural networks. *International Journal of Circuit Theory and Applications*, 30(2-3):211–230, 2002.

[47] N. A. Gorelova and J. Bureš. Spiral waves of spreading depression in the isolated chicken retina. *Journal of Neurobiology*, 14(5):341–409, Sep 1983.

[48] C. Güzeliş. Chaotic cellular neural networks made of Chua's circuits. In r. N. Madan, editor, *Chua's circuit: A Paradigm for Chaos*, volume 1 of *World Scientific Series on Nonlinear Science, Series B*, pages 952–961. World Scientific, 1993.

[49] C. Güzeliş and L. O. Chua. Stability analysis of generalized cellular neural networks. *International Journal of Circuit Theory and Applications*, 21:1–33, 1993.

[50] C. Güzeliş, S. Karamahmut, and I. Genç. A recurrent perceptron learning algorithm for cellular neural networks. *ARI - Interdiscip. Journal of Phys. and Eng. Sciences*, 51(4):296–309, 1999.

[51] K. Halonen, V. Porra, T. Roska, and L. O. Chua. Programmable analog VLSI cnn chip with local digital logic. *International Journal of Circuit Theory and Applications*, 20(5):573–582, Sep-Oct 1992.

[52] M. Hanggi and G. S. Moschytz. Stochastic and Hybrid Approaches Toward Robust Templates. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'98)*, pages 366–371, London, UK, Apr. 1998.

[53] M. Hanggi and G. S. Moschytz. An Exact and Direct Analytical Method for the Design of Optimally Robust CNN Templates. *IEEE Trans. Circuits and Systems*, 46(2):304–311, Feb. 1999.

[54] H. Harrer and J. A. Nossek. Discrete-Time Cellular Neural Networks. *Int. J. of Circuit Theory and Applications*, 20(5):453–467, Sep-Oct 1992.

[55] H. Harrer, J. A. Nossek, T. Roska, and L.O. Chua. A current-mode dtcnn universal chip. In *IEEE International Symposium on Circuits and Systems (ISCAS'94)*, volume 4, pages 135–138, May-Jun 1994.

[56] D. Hillier, V. Binzberger, D. L. Vilarino, and Cs. Rekeczky. Topographic cellular active contour techniques: Theory, implementations and comparisons. *Int. J. of Circuit Theory and Applications*, 34(2):183–216, Mar-Apr 2006.

[57] D. Hillier, S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle. CNNOPT Matlab toolbox of on-chip CNN learning and optimization. http://www.esat.kuleuven.be/sista/chaoslab/cnnopt.

[58] D. Hillier, S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle. CNNOPT: Learning dynamics and CNN chip-specific robustness. In *Proceedings of the 10th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'06)*, pages 114–119, Istanbul, Turkey, Aug 2006.

[59] D. Hillier, S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle. Live demo summary - CNNOPT: Learning dynamics and CNN chip-specific robustness. In *Proceedings of the 10th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'06)*, page 14, Istanbul, Turkey, Aug 2006.

[60] L. Ingber. Very Fast Simulated Re-Annealing. *Journal of Mathematical Computer Modelling*, 12:967–973, 1989.

[61] L. Ingber. Adaptive simulated annealing (ASA): lessons learned. *J. Control and Cybernetics*, 25(1):33–54, 1996.

[62] L. Ingber. Adaptive simulated annealing (ASA). version 24.1 source code at http://www.ingber.com, 2002.

[63] M. Itoh and L. O. Chua. Star cellular neural networks for associative and dynamic memories. *Int. J. of Bifurcation and Chaos*, 14(5):1725–1772, May 2004.

[64] L. Kék and Á. Zarándy. Implementation of large-neighbourhood non-linear templates in the CNN universal machine. *International Journal of Circuit Theory and Applications*, 26(6):551–566, 1998.

[65] P. Kinget and M. Steyaert. A programmable analogue CMOS chip for high speed image processing based on cellular neural networks. In *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, pages 570–573, May 1994.

[66] P. Kinget and M. Steyaert. Analogue CMOS VLSI implementation of cellular neural networks with continuously programmable templates. In *IEEE International Symposium on Circuits and Systems (ISCAS'94)*, volume 6, pages 367–370, May-Jun 1994.

[67] P. Kinget and M. Steyaert. An analog parallel array processor for real-time sensor signal processing. In *IEEE International Solid-State Circuits Conference (ISSCC'96)*, pages 92–93, Feb 1996.

[68] P. Kinget and M. Steyaert. Evaluation of CNN Template Robustness Toward VLSI Implementation. *International Journal of Circuit Theory and Applications*, 24(1):93–110, 1996.

[69] P. Kinget and M. S. J. Steyaert. A programmable analog cellular neural network CMOS chip for high speed image processing. *IEEE Journal of Solid-State Circuits*, 30(3):235–243, Mar 1995.

[70] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[71] G. Koch. Discovering Multi-Core: Extending the Benefits of Moore's Law. In Technology@intel Magazine. Technical report, Intel Corporation, Jul 2005.

[72] T. Kozek, T. Roska, and L. O. Chua. Genetic Algorithm for CNN Template Learning. *IEEE Trans. Circuits and Systems*, 40(I):392–402, March 1993.

[73] T. Kozek, A. Zarándy, S. Zöld, T. Roska, and P. Szolgay. Analogic Macro Code (AMC) Extended Assembly Language for CNN Computers. Technical Report DNS-10, MTA SzTAKI, 1998.

[74] L. Kuhnert, K. I. Agladze, and V. I. Krinsky. Image-processing using light-sensitive chemical waves. *Nature*, 337(6204):244–247, Jan 1989.

[75] R. Kunz, R. Tefzlaff, and D. Wolf. SCNN: a Universal Simulator for Cellular Neural Networks. In *Proceedings of The Fourth IEEE International Workshop on Cellular Neural Networks and Their Applications*, pages 255–260, Sevilla, 1996.

[76] Y. Kuramoto. *Chemical Oscillations, Waves, and Turbulence.* Springer, Berlin, 1984.

[77] K. Kurbel, B. Schneider, and K. Singh. Solving Optimization Problems by Parallel Recombinative Simulated Annealing on a Parallel Computer - An Application to Standard Cell Placement in VLSI Design. *IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics*, 28(3):454–461, Jun 1998.

[78] L. M. Lederman and D. N. Schramm. *From Quarks to the Cosmos: Tools of Discovery*, volume 28 of *Scientific American Library Series*. W.H. Freeman & Company, 1995.

[79] S.-Y. Lee and K. G. Lee. Synchronous and asynchronous parallel simulated annealing with multiple Markov chains. *IEEE Trans. on Parallel and Distributed Systems*, 7(10):993–1008, Oct 1996.

[80] L. Leistritz, M. Galicki, H. Witte, and E. Kochs. Training trajectories by continuous recurrent multilayer networks. *IEEE Trans. Neural Networks*, 13(2):283–291, Mar 2002.

[81] G. Liñán-Cembrano, R. Domínguez-Castro, S. Espejo, and A. Rodríguez-Vázquez. ACE16K: A programmable focal-plane vision processor with 128×128 resolution. In *Proceedings of the European Conference on circuit Theory and Design (ECCTD 2001)*, volume I, pages 345–348, Espoo, Finland, Aug 2001.

[82] G. Liñán-Cembrano, R. Domínguez-Castro, S. Espejo, and A. Rodríguez-Vázquez. ACE16K: An advanced focal-plane analog programmable array processor. In *Proceedings of the 27th European Solid-State Circuits Conference (ESSCIRC 2001)*, pages 201–204, Sept 2001.

[83] G. Liñán-Cembrano, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez. Architectural and Basic Circuit Considerations for a Flexible 128×128 Mixed-Signal SIMD Vision Chip. *Analog Integrated Circuits and Signal Processing*, 33:179–190, Nov. 2002.

[84] G. Liñán-Cembrano, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez. ACE16k: A 128x128 Focal Plane Analog Processor with Digital I/O. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'02)*, pages 132–138, Frankfurt, Germany, Jul. 2002.

[85] G. Liñán-Cembrano, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez. ACE4k: An analog I/O 64x64 visual microprocessor chip with 7-bit analog accuracy. *Int. J. of Circuit Theory and Applications*, 30(2-3):89–116, 2002.

[86] G. Liñán-Cembrano, A. Rodríguez-Vázquez, R. C. Galán, F. Jimênez-Garrido, S. Espejo, and R. Domínguez-Castro. A 1000 FPS at 128/spl times/128 vision processor with 8-bit digitized I/O. *IEEE Journal of Solid-State Circuits*, 39(7):1044 – 1055, Jul 2004.

[87] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. on Evolutionary Computation*, 10(3):281–295, Jun 2006.

[88] C.-T. Lin and C.-H. Huang. Cellular neural networks for hexagonal image processing. In *9th International Workshop on Cellular Neural Networks and Their Applications (CNNA2005)*, pages 81–84, Hsinchu, Taiwan, May 2005.

[89] S. W. Mahfoud and D. Goldberg. Parallel Recombinative Simulated Annealing: A Genetic Algorithm. *Parallel Computing*, 21:1–28, 1995.

[90] B. Mirzai, D. Lim, and G. S. Moschytz. Robust CNN Templates: Theory and Simulations. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'96)*, pages 393–398, Sevilla, 1996.

[91] L.-P. Morency and T. Darrell. Head gesture recognition in intelligent interfaces: the role of context in improving recognition. In *Intelligent User Interfaces*, pages 32–38, 2006.

[92] A. P. Muñuzuri, V. Perez-Muñuzuri, M. Gomez-Gesteira, L. O. Chua, and V. Perez-Villar. Spatiotemporal structures in discretely-coupled arrays of nonlinear circuits: a review. *Int. J. of Bifurcation and Chaos*, 5(1):17–50, Feb 1995.

[93] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1518–1524, Dec. 1996.

[94] K. S. Narendra and K. Parthasarathy. Gradient methods for the optimization of dynamical systems containing neural networks. *IEEE Trans. Neural Networks*, 2(2):252–262, 1991.

[95] C. Niederhofer and R. Tetzlaff. Recent results on the prediction of EEG signals in epilepsy by discrete-time cellular neural networks (DTCNN). In *IEEE International Symposium on Circuits and Systems (ISCAS 2005)*, volume 5, pages 5218– 5221, 2005.

[96] J. A. Nossek. Design and Learning with Cellular Neural Networks. *International Journal of Circuit Theory and Applications*, 24:15–24, 1996.

[97] E. Onbaşoğlu and L. Özdamar. Parallel Simulated Annealing Algorithms in Global Optimization. *J. Global Optimization*, 19(1):27–50, Jan 2001.

[98] L. Özdamar and M. Demirhan. Experiments with new stochastic global optimization search techniques. *Computers & Operations Research*, 27:841–865, 2000.

[99] P. Paolo-Civalleri and M. Gilli. On stability of cellular neural networks. *The Journal of VLSI Signal Processing*, 23(2-3):429 – 435, 1999.

[100] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks - a survey. *IEEE Trans. Neural Networks*, 6(5):1212–1228, Sep 1995.

[101] V. Pérez-Muñuzuri, V. Pérez-Villar, and L. O. Chua. Autowaves for image processing on a two-dimensional cnn array of excitable nonlinear circuits: flat and wrinkled labyrinths. *IEEE Trans. on Circuits and Systems-I*, 40(3):174–181, Mar 1993.

[102] D.C. Pham, T. Aipperspach, D. Boerstler, M. Bolliger, R. Chaudhry, D. Cox, P. Harvey, P.M. Harvey, H.P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Pham, J. Pille, S. Posluszny, M. Riley, D.L. Stasiak, M. Suzuoki, O. Takahashi, J. Warnock, S. Weitzel, D. Wendel, and K. Yazawa. Overview of the architecture, circuit design, and physical implementation of a first-generation Cell processor. *IEEE Journal of Solid-State Circuits*, 41(1):179–196, Jan 2006.

[103] A. Pogromsky, T. Glad, and H. Nijmeijer. On diffusion driven oscillations in coupled dynamical systems. *International Journal of Bifurcation and Chaos*, 9(4):629–644, 1999.

[104] I. Prigogine. *From being to becoming*. Freeman, San Francisco, 1980.

[105] S. Rajasekaran. On Simulated Annealing and Nested Annealing. *J. Global Optimization*, 16:43–56, 2000.

[106] Cs. Rekeczky and L. O. Chua. Computing with Front Propagation: Active Contour And Skeleton Models In Continuous-Time CNN. *The Journal of VLSI Signal Processing*, 23(2-3):373 – 402, 1999.

[107] A. Rodríguez-Vázquez, S. Espejo, and R. Domínguez-Castro. 32×32 CCD. In *Proceedings of the IEICE Symposium in Nonlinear Theory and its Applications (NOLTA'93)*, pages 5–8, 1993.

[108] A. Rodríguez-Vázquez, S. Espejo, R. Domínguez-Castro, J. L. Huertas, and E. Sanchez-Sinencio. Current-mode techniques for the implementation of continuous- anddiscrete-time cellular neural networks. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(3):132–146, Mar 1993.

[109] A. Rodríguez-Vázquez, G. Liñán-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Gálan, F. Jiménez-Garrido, R. Domínguez-Castro, and S.E. Meana. ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 51(5):851–863, May 2004.

[110] T. Roska. Computational and computer complexity of analogic cellular wave computers. *J. Circuits, Systems, and Computers*, 12(4):539–56, 2003.

[111] T. Roska, T. Boros, A. Radvenyi, P. Thiran, and L. O. Chua. Detecting moving and standing objects using cellular neural network. *International Journal of Circuit Theory and Applications*, 20(5):613–628, 1992.

[112] T. Roska and L. O. Chua. Cellular neural networks with nonlinear and delay-type template elements and non-uniform grids. *International Journal of Circuit Theory and Applications*, 20(5):469–482, 1992.

[113] T. Roska and L. O. Chua. The CNN Universal Machine: an Analogic Array Computer. *IEEE Trans. Circuits and Systems*, 40(II):163–173, March 1993.

[114] T. Roska, L. O. Chua, D. Wolf, T. Kozek, R. Tetzlaff, and F. Puffer. Simulating nonlinear waves and partial differential equations via CNN—part I: Basic techniques. *IEEE Trans. on Circuits and Systems-I*, 42(10):807–815, Oct 1995.

[115] T. Roska, L. Kék, L. Nemes, Á. Zarándy, M. Brendel, and P. Szolgay. *CADETWin*. Computer and Automation Institute of the Hungarian Academy of Sciences, Budapest, 1998.

[116] T. Roska, L. Kék, L. Nemes, Á. Zarándy, M. Brendel, and P. Szolgay. *"CNN Software Library" in CADETWin*. Computer and Automation Institute of the Hungarian Academy of Sciences, Budapest, 1998.

[117] T. Roska and A. Rodríuez-Váquez. Towards Visual Microprocessors. *Proceedings of the IEEE*, 90:1244–1257, Jul. 2002.

[118] T. Roska, P. Szolgay, Á. Zarándy, P. L. Venetianer, A. Radvanyi, and T. Sziranyi. On a CNN chip-prototyping system. In *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA-94)*, pages 375–380, Rome, Italy, Dec 1994.

[119] T. Roska and J. Vandewalle. *Cellular Neural Networks*. John Wiley & Sons, Inc., 1994.

[120] T. Roska, Á. Zarándy, S. Zöld, P. Földesy, and P. Szolgay. The computational infrastructure of analogic CNN computing. I. The CNN-UM chip prototyping system. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 46(2):261 – 268, Feb 1999.

[121] G. Ruppeiner, J. M. Pedersen, and P. Salamon. Ensemble approach to simulated annealing. *J. Phys. I*, 1:455–470, 1991.

[122] R. Salomon. Reevaluating Genetic Algorithm Performance Under Coordinate Rotation of Benchmark Functions: A Survey of Some Theoretical and Practical Aspects of Genetic Algorithms. *BioSystems*, 39(3):263–278, 1996.

[123] E. Schrödinger. *What is life*. Cambridge University Press, London, 1945.

[124] A. J. Schuler, M. Brabec, D. Schubel, and J. A. Nossek. Hardware-Oriented Learning for Cellular Neural Networks. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'94)*, pages 183–188, Rome, Dec. 1994.

[125] I. Sendiña-Nadal. *Patterns on Active Media under the Effect of Periodical and Fluctuating Fields*. PhD thesis, Faculty of Physics, University of Santiago de Compostela, Spain, 2001.

[126] B. E. Shi, P. Arena, and Á. Zarándy, editors. *IEEE Transactions on Circuits and Systems-I: Regular Papers - Special issue on CNN technology and active wave computing*, volume 51 (5). May 2004.

[127] P. Siarry, G. Berthiau, F. Durbin, and J. Haussy. Enhanced Simulated Annealing for Globally Minimizing Functions of Many-Continuous Variables. *ACM Trans. Mathematical Software*, 23(2):209–228, Jun 1997.

[128] S. Smale. A mathematical model of two cells via Turing's equation. *Lecture in Applied Mathematics (American Mathematical Society)*, 6:15–26, 1974.

[129] M. A. Styblinski. Statistical circuit design. In *Computer-Aided Design and Optimisation*, chapter 55, pages 1453–1486. CRC Press, 1995.

[130] M. K. Sundareshan and T. A. Condarcure. Recurrent neural-network training by a learning automaton approach for trajectory learning and control system design. *IEEE Transactions on Neural Networks*, 9(3):354–368, Jan 1998.

[131] J. A. K. Suykens, J. Vandewalle, and B. De Moor. Intelligence and cooperative search by coupled local minimizers. *Int. J. of Bifurcation and Chaos*, 11(8):2133–2144, Aug 2001.

[132] J. A. K. Suykens, M. E. Yalcin, and J. Vandewalle. Coupled Chaotic Simulated Annealing Processes. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 582–585, Bangkok, Thailand, May 2003.

[133] I. Szatmári, Cs. Rekeczky, and T. Roska. A Nonlinear Wave Metric and its CNN Implementation for Object Classification. *The Journal of VLSI Signal Processing*, 23(2-3):437–447, Nov. 1999.

[134] I. Szatmári, A. Schultz, Cs. Rekeczky, T. Kozek, T. Roska, and L. O. Chua. Morphology and autowave metric on CNN applied to bubble-debris classification. *IEEE Trans. Neural. Networks*, 11(6):1385–1393, 2000.

[135] P. Szolgay, Á. Zarándy, S. Zöld, T. Roska, P. Földesy, L. Kék, T. Kozek, K. Laszlo, I. Petras, Cs. Rekeczky, I. Szatmari, and D. Balya. The computational infrastructure for cellular visual microprocessors. In *Proceedings of the Seventh International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems (MicroNeuro'99)*, pages 54–60, Apr 1999.

[136] H. H. Szu and R. L. Hartley. Fast Simulated Annealing. *Physics Letters A*, 122:157–162, 1987.

[137] R. Tetzlaff, R. Kunz, and G. Geis. Analysis of cellular neural networks with parameter deviations. In *Proc. IEEE ECCTD 97*, pages 650–654, 1997.

[138] R. Tetzlaff, R. Kunz, G. Geis, and D. Wolf. Minimizing the effects of tolerance faults on hardware realizationsof cellular neural networks. In *Proceedings of The Fifth IEEE International Workshop on Cellular Neural Networks and Their Applications*, pages 385–390, London, UK, Apr 1998.

[139] R. Tetzlaff, R. Kunz, and D. Wolf. Minimizing the effects of parameter deviations on cellular neural networks. *Int. J. of Circuit Theory and Applications*, 27(1):77–86, Jan-Feb 1999.

[140] P. Thiran. Influence of Boundary Conditions on the Behavior of Cellular Neural Networks. *IEEE Trans. on Circuits and Systems—I: Fundamental Theory and Applications*, 40(3):207–212, Mar 1993.

[141] D. R. Thompson and G. L. Bilbro. Sample-sort simulated annealing. *IEEE Trans. on Systems, Man and Cybernetics, Part B*, 35(3):625–632, Jun 2005.

[142] Z. Tu and Y. Lu. A Robust Stochastic Genetic Algorithm (StGA) for Global Numerical Optimization. *IEEE Trans. on Evolutionary Computation*, 8(5):456–470, Oct 2004.

[143] A. M. Turing. The chemical basis of morphogenesis. *Philos. Trans. Roy. Soc. London*, B(237):37–72, 1952.

[144] J. E. Varrientos, E. Sanchez-Sinencio, and J. Ramirez-Angulo. A current-mode cellular neural network implementation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(3):147–155, Mar 1993.

[145] P. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1990.

[146] D. Whitley, S. B. Rana, J. Dzubera, and K. E. Mathias. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85(1-2):245–276, 1996.

[147] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr 1997.

[148] S. Xavier-de-Souza, M. Van Dyck, J. A. K. Suykens, and J. Vandewalle. Fast and Robust Face Tracking for CNN chips: application to wheelchair driving. In *Proceedings of the 10th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'06)*, pages 200–205, Istanbul, Turkey, Aug 2006.

[149] S. Xavier-de-Souza, M. Van Dyck, J. A. K. Suykens, and J. Vandewalle. Live demo summary - Fast and Robust Face Tracking applied to wheelchair driving. In *Proceedings of the 10th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'06)*, page 20, Istanbul, Turkey, Aug 2006.

[150] S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle. Real-time tracking algorithm with locking on a given object for VLSI CNN-UM implementations. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and their applications*, pages 291–296, Budapest, Hungary, Sep 2004.

[151] S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle. Learning wave phenomena on the CNN universal machine. In *The 2006 International Symposium on Nonlinear Theory and its Applications (NOLTA2005)*, Bruges, Belgium, Oct 2005.

[152] S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle. Learning of Spatiotemporal Behavior in Cellular Neural Networks. *International Journal of Circuit Theory and Applications - Special Issue on CNN Technology (Part 1)*, 34:127–140, Jan 2006.

[153] S. Xavier-de-Souza, J. A. K. Suykens, J. Vandewalle, and D. Bollé. Cooperative behavior in coupled simulated annealing processes with variance control. In *The 2006 International Symposium on Nonlinear Theory and its Applications (NOLTA2006)*, Bologna, Italy, Sep 2006.

[154] S. Xavier-de-Souza, J. A. K. Suykens, J. Vandewalle, and D. Bollé. Coupled Simulated Annealing for Continuous Global Optimization. Technical report, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2006. URL: ftp.esat.kuleuven.be/pub/SISTA/sdesouza-/papers/CSA2006submitted.pdf.

[155] S. Xavier-de-Souza, M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle. Automatic Chip-Specific CNN Template Optimization using Adaptive Simulated Annealing. In *Proceedings of European Conference on Circuit Theory and Design (ECCTD'03)*, pages 329–332, Krakow, Poland, Sep 2003.

[156] S. Xavier-de-Souza, M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle. Toward CNN Chip-specific robustness. *IEEE Transactions on Circuits and Systems-I: Regular Papers*, 51(5):892–902, May 2004.

[157] M. E. Yalcin and J. A. K. Suykens. Spatiotemporal pattern formation on the ACE16k CNN chip. *Int. J. of Bifurcation and Chaos*, 16(5):1537–1546, May 2006.

[158] M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle. Spatiotemporal pattern formation in the ACE16k CNN chip. In *IEEE International Symposium on Circuits and Systsems (ISCAS 2005)*, pages 5814–5817, Kope, Japan, May 2005.

[159] T. Yang, K. R. Crounse, and L. O. Chua. Spherical cellular nonlinear networks. *International Journal of Bifurcation and Chaos*, 11(1):241–257, 2001.

[160] X. Yao. A New Simulated Annealing Algorithm. *Intern. J. Computer Math.*, 56:161–168, 1995.

[161] X. Yao, Y. Liu, and G. Lin. Evolutionary Programming Made Faster. *IEEE Trans. on Evolutionary Computation*, 3(2):82–102, Jul 1999.

[162] Á. Zarándy. The Art of CNN Template Design. *International Journal of Circuit Theory and Applications*, 27(1):5–23, Jan-Feb 1999.

[163] Á. Zarándy. ACE Box: high-performance visual computer based on the ACE4k analogic array processor. In *Proceedings of the European Conference on circuit Theory and Design (ECCTD 2001)*, volume I, pages 361–364, Espoo, Finland, Aug 2001.

[164] Á. Zarándy and Cs. Rekeczky. Bi-i: a standalone ultra high speed cellular vision system. *IEEE Circuits and Systems Magazine*, 5(2):36–45, 2005.

[165] P. Zegers and M. K. Sundareshan. Trajectory generation and modulation using dynamic neural networks. *IEEE Trans. Neural Networks*, 14(3):520–533, may 2003.

Publications by the Author

Journal papers

- **S. Xavier-de-Souza**, J. A. K. Suykens, and J. Vandewalle. Learning of Spatiotemporal Behavior in Cellular Neural Networks. *International Journal of Circuit Theory and Applications - Special Issue on CNN Technology (Part 1)*, 34:127–140, Jan 2006.

- **S. Xavier-de-Souza**, M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle. Toward CNN Chip-specific robustness. *IEEE Trans. on Circuits and Systems-I - Special issue on CNN technology and active wave computing*, 51(5):892–902, May 2004.

- P. Marchal, M. Jayapala, **S. Xavier-de-Souza**, P. Yang, F. Catthoor And G. Deconinck, Matador: An Exploration Environment for System-Design. *Journal of Circuits, Systems and Computers*, 11(5):503–535, 2002.

Conference papers

- **S. Xavier-de-Souza**, J. A. K. Suykens, J. Vandewalle, and D. Bollé. Cooperative behavior in coupled simulated annealing processes with variance control. In *The 2006 International Symposium on Nonlinear Theory and its Applications (NOLTA2006)*, Bologna, Italy, Sep 2006.

- **S. Xavier-de-Souza**, M. Van Dyck, J. A. K. Suykens, and J. Vandewalle. Fast and Robust Face Tracking for CNN chips: application to wheelchair driving. In *Proceedings of the 10th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'06)*, pages 200–205, Istanbul, Turkey, Aug 2006.

- D. Hillier, **S. Xavier-de-Souza**, J. A. K. Suykens, and J. Vandewalle. CNNOPT: Learning dynamics and CNN chip-specific robustness. In *Proceedings of the 10th IEEE Int. Workshop on Cellular Neural Net- works and Their Applications, (CNNA'06)*, pages 114–119, Istanbul, Turkey, Aug 2006.

- **S. Xavier-de-Souza**, J. A. K. Suykens, and J. Vandewalle. Learning wave phenomena on the CNN universal machine. In *The 2006 International Symposium on Nonlinear Theory and its Applications (NOLTA2005)*, Bruges, Belgium, Oct 2005.

- **S. Xavier-de-Souza**, J. A. K. Suykens, and J. Vandewalle. Real-time tracking algorithm with locking on a given object for VLSI CNN-UM implementations. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and their applications*, pages 291–296, Budapest, Hungary, Sep 2004.

- **S. Xavier-de-Souza**, M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle. Automatic Chip-Specific CNN Template Optimization using Adaptive Simulated Annealing. In *Proceedings of European Conference on Circuit Theory and Design (ECCTD'03)*, pages 329–332, Krakow, Poland, Sep 2003.

- **S. X. Souza**, A. D. DÓria-Neto, J. A. F Costa, M. L. Andrade-Neto. A Neural Hybrid System for Large Memory Association. In *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN'01)*, Washington DC, USA, 2:1174–1179, July 2001.

- **S. X. Souza**, A. D. DÓria-Neto, J. B. Bezerra, J. D. Melo. Parallel Architecture for Hopfield Neural Networks. In *Proc. of IEEE International Conference on Engineering and Computer Education (ICECE'2000)*, São Paulo, Brazil, 2000.

- **S. X. Souza**, A. D. DÓria-Neto, J. B. Bezerra, J. D. Melo. Uma Configuração Paralela para Ampliar a Capacidade de Armazenamento da Rede de Hopfield. In *Proc. of XII Congresso Brasileiro de Automática (CBA2000)*, Florianópolis, Brazil, 2000. (In Portuguese).

Technical reports and other publications

- **S. Xavier-de-Souza**, J. A. K. Suykens, J. Vandewalle, and D. Bollé. Coupled Simulated Annealing for Continuous Global Optimization. Technical report, 2006. URL: ftp.esat.kuleuven.be/pub/SISTA/sdesouza/papers/CSA2006submitted.pdf.

- **S. Xavier-de-Souza**, M. Van Dyck, J. A. K. Suykens, and J. Vandewalle. Live demo summary - Fast and Robust Face Tracking applied to wheelchair driving. In *Proceedings of the 10th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'06)*, page 20, Istanbul, Turkey, Aug 2006.

- D. Hillier, **S. Xavier-de-Souza**, J. A. K. Suykens, and J. Vandewalle. Live demo summary - CNNOPT: Learning dynamics and CNN chipspecific robustness. In *Proceedings of the 10th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, (CNNA'06)*, page 14, Istanbul, Turkey, Aug 2006.

Curriculum Vitae

Personal information

Date of Birth: 4th November 1976
Place of Birth: Natal - Brazil
Nationalities: Brazilian, Belgian
Address:
Molenstraat, 15
B-3010 Kessel-lo BELGIUM
Tel. Home: +32 (0) 16.65.2112
Tel. Work: +32 (0) 16.32.8662

Diplomas

- **Pre-Doctoral Degree**
  ESAT - Electrical Department
  Katholieke Universiteit Leuven, BELGIUM
  Advisers: Prof. Joos Vandewalle and Prof. Johan Suykens.
  Period: from July 2002 to June 2003.

- **5 year-Degree in Computer Engineering**
  Department of Computer and Automation Engineering
  Federal University of Rio Grande do Norte, Natal, BRAZIL.
  Thesis: *Distribution of patterns among Hopfield Networks in parallel using SOM algorithm.*
  Advisers: Prof. Adrião D. Doria Neto and Prof. José Alfredo F. Costa.
  Period: from February 1996 to December 2000.

Final Percentage Score: 78,3%
Last Year Percentage Score: 87,2%

- **4 year-Diploma in Electro-techniques**
  Federal Technical School of Rio Grande do Norte, Natal, BRAZIL.
  Period: from Feb'1991 to July 1995.

Professional experience

- **Research Assistant July 2002 - Currently**.
  Team: ESAT/SCD - SISTA
  Location: Katholieke Universiteit Leuven, BELGIUM.

- **IMEC Internship - August 2001 to June 2002**.
  Team: Advanced Design Technologies DESICS/ADT
  Responsibilities: Development of a multi-processor simulator environment to support research of novel design techniques.
  Location: IMEC vzw, BELGIUM.

- **Software Specialist - April 2001 to July 2001**.
  Team: Power Systems Stability.
  Responsibilities: Development of a software for estimation of the load of existing complex power systems.
  Location: COSERN/DCA-UFRN Research Project, BRAZIL.

Scholarships/Awards/Other achievements

- **Best Technological Report 2000**.
  Title: Parallel Configuration for Hopfield Network Storage Capacity Increasing.
  In: XI Scientific Initiation Conf. of Federal Univ. of Rio Grande do Norte (UFRN). Brazil.

- **Scholarship August 2000 - December 2000**.
  Title: Distribution of patterns among Hopfield Networks in parallel using SOM algorithm.
  From: National Council for Scientific and Technological Development (CNPq), Brazil.

- **Scholarship August 1999 - July 2000**.
  Title: Parallel Configuration of Hopfield Network for Increasing the Storage Capacity.
  From: National Council for Scientific and Technological Development (CNPq), Brazil.

- **Scholarship March 1998 - July 1999**.
  Title: Parallel Neural Decoder for Convolutional Codes.
  From: National Council for Scientific and Technological Development (CNPq), Brazil.

- **Scholarship March 1997 - February 1998**.
  Title: Cold Plasma Chamber Automatization.
  From: National Council for Scientific and Technological Development (CNPq), Brazil.