Analysis of Parallel Local Methods in Image Segmentation, Image Compression, and Motion Analysis

Ph.D. Thesis of

Czúni László

Supervisor: Szirányi Tamás

MI 1 Ph.D. Program

University of Veszprém

2000

Párhuzamos, lokális módszerek vizsgálata a képszegmentálásban, tömörítésben és optikai mozgásanalízisben

(Analysis of Parallel Local Methods in Image Segmentation, Image Compression, and Motion Analysis)

Értekezés doktori (Ph.D.) fokozat elnyerése érdekében

Írta: Czúni László
informatikus mérnök

Készült a Veszprémi Egyetem Műszaki Informatika doktori program 1-es (MI 1 jelű) alprogramja keretében.
Témavezető: Dr. Szirányi Tamás
Elfogadásra javaslom: igen / nem .

……
dátum, aláírás

A jelölt a doktori szigorlaton ……%-ot ért el.
Veszprém, 200……

……...
a Szigorlati Bizottság elnöke

Az értekezést bírálóként elfogadásra javaslom: igen / nem .
Első bíráló: ……. igen / nem

……
aláírás

Második bíráló: ……. igen / nem

……
aláírás

A jelölt az értekezés nyilvános vitáján ……%-ot ért el.
Veszprém, 200……

……...
a Bíráló Bizottság elnöke

A doktori (Ph.D.) oklevél minősítése…….

……...
az EDT elnöke

TABLE OF CONTENTS

CHAPTER I

MARKOV RANDOM FIELD-BASED IMAGE SEGMENTATION ON ANALOG PROCESSOR ARRAYS

CHAPTER II

IMAGE COMPRESSION WITH NOISY 2D PROCESSOR ARRAYS

CHAPTER III

SPATIO-TEMPORAL SEGMENTATION OF VIDEO SEQUENCES WITH 2D PROCESSOR ARRAYS

Kivonat

A jelen doktori értekezés a képfeldolgozás három különböző területének három speciális részterületével foglalkozik:

- statisztikai képszegmentálás - képanalízis,
- DCT (Diszkrét Koszinusz Transzformáció) alapú képtömörítés – képkódolás,
- mozgás szegmentálás (sűrű vektormezőkön) – mozgás analízis.

A dolgozatban tárgyalt algoritmusok mindegyike alapvetően abban különbözik az irodalomban eddig található eljárásoktól, hogy ezek párhuzamos processzortömbökre lettek kidolgozva. Míg az ilyen platformoknak nagyon jó az ár/teljesítmény mutatója, addig számottevő az a hátrányuk, hogy csak meghatározott típusú és számú műveletekre képesek, korlátozott a lokális memóriák száma és nem túl nagy a számítási pontosságuk (~8bit).

Az első fejezetben egy Markov Valószínűségi Mező-n (Markov Random Field - MRF) alapuló szegmentálási eljárás a vizsgálat tárgya, tekintettel a többfelbontásos megvalósításra. A fejezet külön kitér a számítási zaj hatására, ami – egy bizonyos érték alatt - a sztochasztikus optimalizálási módnak köszönhetően nem rontja, hanem javítja a szegmentálási teljesítményt.

A számítási zaj szintén fontos szerepet játszik a második fejezetben is, ahol egy ortogonális transzformációt végző architektúra a számítások alapja. Itt egy optimalizáló kódolási stratégia biztosítja a hatékony tömörítést a zajos áramkörökön, akár több transzformációs eljárás segítségével (Koszinusz és Hadamard transzformációk).

Mozgás- és téridőbeli szegmentálási eljárásokkal foglalkozik a harmadik fejezet. A mozgásvektor szegmentálási eljárások MRF alapú statisztikai módszerekkel működnek, míg a téridőbeli eljárás pixel szintű követést, térbeli intenzitásinformációt és az előzőleg becsült mozgásmezőt használja, a mozgó foltok, objektumok jobb szegmentálása érdekében.

Abstract

The Thesis deals with three topics of three main areas of image processing:

- statistical image segmentation (image analysis)
- DCT (Discrete Cosine Transformation) based image compression (image coding)
- motion segmentation (based on dense vector field) (motion analysis).

All the discussed methods differ from most of the preceding algorithms found in literature by the fact that they are supposed to run on fully parallel processor arrays. While these platforms, when realized in VLSI chips, have superior computational speed at relatively low cost compared to other architectures, have the disadvantage of restricted number and type of operations, number of memory and precision of computations.

In Chapter I a Markov Random Field (MRF) based image segmentation method is investigated also in view of multiscale implementation. Some important observations of this chapter come from simulations of computation noise, which show that a certain amount of noise even enhances the segmentation performance, thanks to the stochastic optimization technique.

The effect of computation noise also plays important role in Chapter II, where compression architecture, based on orthogonal decomposition, is described. Furthermore a bit allocation strategy is proposed based on the competition of Hadamard and Discrete Cosine Transformations.

Motion and spatio-temporal segmentation algorithms are discussed in Chapter III. Motion vector segmentation methods are based on MRF statistical methods, while the spatio-temporal segmentation algorithm uses a pixel-level tracking algorithm, the spatial intensity information and the estimated motion field for a better segmentation of moving blobs in an image sequence.

Zusammenfassung

Die Arbeit beschäftigt sich mit drei Teilgebieten der Bildvearbeitung:

- Statistische Bildsegmentierung (Bildanalyse)
- Einfluß der DKT (Diskret Kosinus Transformation) auf die Bildkompression – Bildcodierung
- Bewegung-Segmentierung (basierend auf dichten Vektorfeldern) – Bewegungsanalyse

Alle Algorithmen dieser Arbeit unterscheiden sich von den Algorithmen die in der Literatur beschrieben wurden dadurch, dass diese Algorithmen für Paralellmatrixrechner ausgearbeitet worden sind. Diese Hardwerbasis besitzt ein sehr gutes Kosten/Rechenegeschwindigkeitverhältsnis, aber es hat auch mehrere Nachteile. Das eine ist die eingeschränkte Zahl und Art von Operationen, das zweite ist die begrenzte Zahl der Arbeitsspeicher und Rechengenauigkeit.

Im ersten Kapitel wird eine auf dem Markov Random Field beruhende Segmentierungsmethode untersucht, es wird auch die multiscale Implementation beabsichtig. Einige wichtige Beobachtungen dieses Kapitels kommen von der Simulation des Rechnerrauschens. Es wird gezeigt, dass ein gewisses Mass von Rechnerrauschen sogar zur Erhöhung der Segmentationsgüte beiträgt. Dies ist der stochastischen Optimierungstechnik zu danken.

Das Rechnenrauschen hat eine wichtige Rolle auch im zweiten Kapitel, wo eine Kompressionsarchitektur beschrieben wird, die auf eine orthogonale Zerlegung basiert. Hier wird auch eine Strategie der Bit-Zuordnung vorgeschlagen, die auf Hadamard und Diskret Kosinus Transformation beruht.

Bewegungs- und RaumzeitSegmentierungsalgoritmen werden im dritten Kapitel untersucht. Der Algoritmus der Bewegungsvektorsegmentierung basiert auf einen MRF statistischen Algoritmus, während der Raumzeitalgoritmus das Nachführen des Bildpunkt-Niveaus benutzt, es benutzt die Helltasten der Information in dem Raum und dem Bewegungsfeld für bessere Segmentierung der Bewegung Flecken in einer Bildsequenz.

Related Publications

While the Thesis has three main chapters dealing with different areas of image processing, they are all common that they are related to image coding and use the same architecture for computations. This platform can be described as a fully parallel architecture consisting of cell elements generally organized in matrix form. These elements are connected to each other, contain memory and processing units and usually represent image pixels. One typical example for these platforms is the analog Cellular Neural/Nonlinear Network (CNN) [14,77], while other digital devices also exist such as the Mitshubishi's artificial retina [99] or other smart pixel arrays [24]. Currently, according to its physical parameters and complexity [21,57] it seems that CNN is capable of the most complex computations, however, it is also true that for certain applications other (even 1D) architectures are also satisfactory [82].

Chapter I contains algorithms for image segmentation based on Markov Random Fields (MRFs). General reviews of image segmentation can be found in [34,64], however, the application of MRF techniques in image analysis and segmentation are in [4,8,9,10, 20,30,32,35,47,49,50,51,104]. According to a large number of publications in this field, algorithms described in the Thesis differ in the basic interpretation of data of observations made at pixel locations [96]. This is necessary for the localization of statistical features (expected value and variance are measured at the pixel-level instead of using global classes) and makes the CNN adaptation possible. Also none of the known literature deals with the realization of MRF related statistical algorithms on noisy hardware. The described models of the Thesis also seem to be good applications of the so-called Modified Metropolis Dynamics (MMD) optimization method [50], what is a special type of Simulated Annealing [30,52], while multiscale MRF segmentation methods have been tested according to [32].

Chapter II deals with the possible use of CNN in image compression. While there are a lot of papers about the application of CNN in image processing and a lot of papers in the area of general image coding, only a very few articles deal with image coding/compression in the CNN environment. In [100,101] an architecture for DCT (Discrete Cosine Transform) and subband-based compression are given, however, the Thesis proposes a different architecture capable of simultaneous coding and decoding of the image [94]. This feature is necessary to implement Shannon's rate-distortion theory [83], a framework that is used to choose an optimal bit allocation strategy. It is also important to mention that while CNN seems to be quite robust in MRF-based segmentation, it is a bigger problem in image coding where the effect of computational noise can decrease image quality seriously. While [100,101] do not deal with the effects of noise, it is discussed in the current paper in details. Many other

bit allocation strategies exist (e.g. [15,59]), the currently most used one (based on DCT) is JPEG [105]. In the Thesis a different algorithm for optimal bit allocation is proposed, it is similar to the Dynamic Video Coding (DVC) approach discussed in [22,72]. [25] describes the algorithmic solution for converting constraint problems to unconstrained problems, also used in the proposed optimization algorithm.

In Chapter III motion segmentation algorithms and a spatio-temporal segmentation algorithm is proposed. These algorithms can serve as bases for second generation image/video coding. [98] contains details about this approach considered as a successor of today's compression technology [43,44,46], while MPEG-4 [45] is already one main application of this technique.

Several optical flow estimation and segmentation techniques have been investigated from the aspects of parallel implementation. [6,7,55,58,88] serve as good starting points for comparisons of computation complexity and performance of several methods.

Correlation-based techniques are widely used in MPEG applications but their serial implementation is very time consuming. In case of motion field optimization, iterative or relaxation algorithms are used to obtain reliable results [40,65]. However, the iterative reevaluation of the displaced frame difference increases time complexity and may not fit our computation model based on local interactions. Generally, correlation techniques seem to be rejected in the Simple Instruction Multiple Data (SIMD) model we addressed, but now we show that if no motion model based optimization is needed then our fully parallel architecture can compute vector estimates with the correlation technique.

According to [6] gradient-based approaches have good accuracy, although, it is also well-known that these methods are very sensitive to temporal aliasing and need large temporal support. In [6] best results needed 15 image frames to be stored in memory simultaneously for computations (see the next section). In [28] a recursive filter is designed to avoid the need for large image memories and long delays, and they report similar results to the original model in accuracy.

In [84] filter banks had been built for spatio-temporal filtering. Their method is well suited for parallel implementation and has good accuracy as reported, but its application could be limited due to the large number of filters needed for general image flows.

[49] describes motion compensated vector classification in the MRF framework. The proposed model in [49] applies 3D cliques to exploit inter-frame information, but this method still leads to color classification instead of real spatio-temporal segmentation, which is closer to the recognition of video objects.

Many probabilistic approaches use region labeling algorithms for spatio-temporal segmentation, e.g. [11,31]. These models, however, can not be implemented in our parallel framework, since the proposed region indexing and labeling methods are not well suited to the pixel array with many similar local processing elements and memories. That is why a different contour-based segmentation method is employed in this work.

Besides this overview other observations and comparisons are present in the following three Chapters.

PREFACE: PARALLEL ARCHITECTURES AND CELLULAR ARRAYS

While in the past a lot of effort has been made to design real-time hardware systems, today many of those problems can be solved with high performance general-purpose personal workstations at low cost. On the other hand there is an increasing demand for a new generation of image analysis tasks, such as second-generation video coding [62,98], virtual reality, etc. While up-to-date Complex Instruction Set Computers (CISC), such as PCs based on the Pentium MMX family, speed up multimedia applications tremendously, these tasks are still far from being in the range of the computing power of conventional CISC machines in the close future [58,70].

One possible breakthrough is the use of cellular processor arrays (analog or digital) which assign a simple processor to each pixel resulting in fully parallel operation [14,77]. Contrary to the general CISC class of processors, which can also show some parallel capabilities as well, cellular processor arrays can utilize only simple pixel based functions defining a relatively narrow instruction set – although, at a very high speed.

The typical example for the higher cell-complexity of cellular processor arrays is the family of the Cellular Neural/Nonlinear Network (CNN) chips [21,57]. CNN can process images locally on the pixel-level with small neighborhood connectivity. It can perform convolution, nonlinear (sigmoid) dynamics, etc. in a feed-forward/feed-back operation mode. CNN Universal Machine (CNN-UM) [77] is a programmable computer based on the basic CNN architecture with many additional features like those that conventional computers have: local and global memories, pixel-level logical and arithmetic functions, digital memories etc., all in one single chip.

Investigating the success of general-purpose high performance processors in the image processing area, it seems to be reasonable to develop new, low-level algorithms for this new class of cellular processor arrays, since they can be considered as widely usable real-time image processing architectures with superior speed, integrated into VLSI. The crucial question is how to develop a successful cellular image processing system for certain purposes containing local connections and reduced instruction set, considering the limitations of the cell-complexity of VLSI implementation

CHAPTER I

MARKOV RANDOM FIELD-BASED IMAGE SEGMENTATION ON ANALOG PROCESSOR ARRAYS

1 IMAGE SEGMENTATION

Image segmentation can be considered as a special kind of image classification. In dense pixel classification pixels are represented in the feature space and in many cases the task is to find the most important feature vector elements that are required to group the pixels into classes. A very important aspect in segmentation is that the spatially neighboring pixels should belong to the same class with high probability.

Image segmentation is an operation very often used in image processing. Its output can be directly useful to a large class of applications such as medical image processing, agricultural applications, etc. High level applications, such as robot vision and second generation image coding also use segmentation as an essential low-level preprocessing step.

There are a large number of different image segmentation algorithms [34,64]. The most important criteria for good segmentation are the following:

- The segmented image should contain connected and disjoint regions.
- All pixels should belong to one class.
- Regions should be homogenous with well-defined contours separating them.

However, it is always a question how to evaluate an image segmentation algorithm. In most cases artificially generated images serve as ground truth. In cases of natural data the image is usually segmented manually, based on auxiliary information, and the result obtained is used for performance evaluation of the automatic segmentation process.

Many of the segmentation methods, e.g. split and merge, region growing, etc., construct a graph (or similar high-level) representation of the image content. Unfortunately, this kind of high-level representation is not possible in our framework where images are represented densely on a lattice of elements. The most important feature of the Markov Random Field (MRF)-based segmentation models discussed in this chapter is that they are based on low level observations and operations. On the

other hand, a very important aspect of the development of the following algorithms was to use simple operations that are realizable in analog VLSI circuits.

The proposed low-level processes can be used for the segmentation of grayscale, non-textured images. Such kinds of images are common in medical image processing (e.g. Computer Tomography (CT) and Magnetic Resonance Imaging (MRI) data) or in the area of processing spot or air-born images.

A very important feature of the presented algorithms that they can be executed parallel promising a fast VLSI implementation on cell array computers such as the CNN-UM.

2 MRF IMAGE SEGMENTATION MODELS WITH MAXIMUM A POSTERIORI OPTIMIZATION

Since the work of Geman and Geman [30] there are a lot of examples where stochastic optimization and Bayesian approaches are used for solving image labeling problems. However, the idea of parallel, low-level cooperating processes is much older, many basic ideas were already reviewed in [20]. Examples for edge detection, image segmentation, restoration, stereovision, motion segmentation, etc. can be found in [4,8,9,10,30,32,35,50, 51,104].

For many of the early vision processes, where the image is represented on a lattice, the problem is posed as a labeling problem. In our case, when our aim is to segment the input image to a given number of classes, we will determine the possible classes by the grayscale information of the input image, but usually MRF based methods can be extended to the analysis of color images as well [47,49].

To find an optimal labeling, the minimization of a cost function - which is constructed from the observed data, a priori information and constraints - is needed. The obtained cost function is usually non-convex and several relaxation techniques have been proposed to reach an optimum labeling. One class of methods deals with stochastic relaxation and is based on Simulated Annealing (SA) [30,52]. These algorithms converge asymptotically towards the global minimum but require a great deal of computation. The second group of methods is related to deterministic relaxation. These techniques are sub-optimal but require less computational time than the previous ones. That is why so many deterministic (or pseudo stochastic) relaxation algorithms have been recently investigated (Graduated Non Convexity (GNC) [9], Iterated Conditional Mode (ICM) [8], Mean Field Annealing (MFA) [104], Modified Metropolis Dynamics (MMD) [50]). In the proposed framework the MMD will be implemented.

2.1 Basic Definitions

First, I give a brief introduction to the theory of MRF [3,74,96], and then a general image model, used in the following sections, is described.

Let $\mathbf{S} = (s_1, s_2, s_3, ...s_N)$ be a set of sites (or pixels). Two points s_i and s_j are neighbors, if there is an edge connecting them. The set of points that are neighbors of site s is denoted Γ_s. $\Gamma = \{\Gamma_s \mid s \in \mathbf{S}\}$ is a neighborhood system for $\mathbf{S}$ if:

1. $s \notin \Gamma_s$
2. $s \in \Gamma_r \Leftrightarrow r \in \Gamma_s$.

A subset $C \subseteq \mathbf{S}$ is a clique if every pair of distinct sites in C is composed of two neighbors. $\boldsymbol{C}$ denotes the set of all cliques. In image processing the most commonly used neighborhood systems are the homogeneous systems. In this case, we consider $\mathbf{S}$ as a lattice and define these neighborhoods as

$$\Gamma^n = \{\Gamma^n_{(i,j)} : (i,j) \in \mathbf{S}\},$$

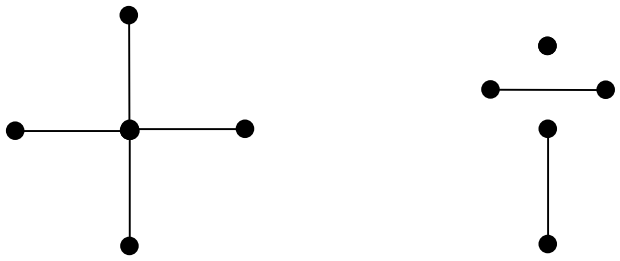
$$\Gamma^n_{(i,j)} = \{(k,l) \in \mathbf{S} : (k-i)^2 + (l-j)^2 \le n\}.$$

Obviously, sites near the boundary have fewer neighbors than interior ones. Furthermore, $\Gamma^0 \equiv \mathbf{S}$ and for all $n \ge 0 : \Gamma^n \subset \Gamma^{n+1}$. Fig. 1 shows a first-order neighborhood corresponding to *n*=1. The cliques are *{(i,j)}, {(i,j), (i,j+1)}, {(i,j), (i+1,j)}.*

Let $X = \{X_s \mid s \in \mathbf{S}\}$ denote any family of random variables so that $\forall s \in \mathbf{S} : X_s \in \Lambda$, where $\Lambda = \{1,.....,L\}$ is a common state space. In our case they will mean the labels. Furthermore, let $\Omega = \{\omega = (\omega_{s_1}, \omega_{s_2}, \omega_{s_3}, ..., \omega_{s_N}) : \omega_{s_i} \in \Lambda,\ 1 \le i \le N\}$ be the set of all possible configurations.

X is a MRF with respect to Γ if

1. $\forall \omega \in \Omega : P(X = \omega) > 0$,
2. $\forall s \in \mathbf{S}, \forall \omega \in \Omega : P(X_s = \omega_s \mid X_r = \omega_r, r \ne s) = P(X_s = \omega_s \mid X_r = \omega_r, r \in \Gamma_s)$.



A central site and its 4 neighbors *Cliques*

Figure 1.
First order neighborhood-system and cliques.

The functions in point 2 are called the *local characteristics* of the MRF, and the probability distribution $P(X=\omega)$ of any process, satisfying point 1, is uniquely determined by these conditional probabilities. However, it is extremely difficult to determine these characteristics in practice. Gibbs distribution and the Hammersley-Clifford theorem provide us a simple way to solve this problem.

A Gibbs distribution, relative to the neighborhood system $\mathcal{G}$, is a probability measure π on Ω with the following representation:

$$\pi(\omega) = \frac{1}{Z}\exp\big(-E(\omega)\big), \tag{1}$$

where Z is the normalizing constant or partition function:

$$Z = \sum_{\omega}\exp\big(-E(\omega)\big), \tag{2}$$

and the energy function E is of the form:

$$E(\omega) = \sum_{C\in\boldsymbol{C}} E_C(\omega)\,. \tag{3}$$

Each E_C is a function defined on Ω depending only on those ω_s elements of ω for which $s\in C$. The restriction of ω to the sites of a given clique C is denoted by ω_C. Such a function is called a potential. A very important theorem is the Hammersley-Clifford theorem, which points out the relation between the MRF and the Gibbs distribution:

- X is a MRF with respect to the neighborhood system $\mathcal{G}$ if and only if $\pi(\omega) = P(X=\omega)$ is a Gibbs distribution with respect to $\mathcal{G}$.

Using the above theorem the definition of the MRF is completed by the knowledge of the clique potentials $E_C(\omega_C)$ for every C in $\boldsymbol{C}$ and every ω in Ω.

2.2 The Segmentation Model: Bayesian Model with Maximum A Posteriori (MAP) Estimation

During the labeling segmentation process we would like to choose the most probable labels, i.e. our estimation should be based on the Bayesian estimation model. We can start from the following probabilities:

$$P(\omega \mid F) = \frac{P(F \mid \omega)P(\omega)}{P(F)}, \tag{4}$$

where F is the observed grey-level image data, $F = \{f_s\}_{s \in \mathbf{S}}$. Since the MAP estimation techniques are quite effective in image processing, we can use this model to define the Bayes risk. The cost function in MAP estimation is defined as:

$$R(\omega, \omega') = 1 - \Delta_{\omega'}(\omega), \tag{5}$$

where $\Delta_{\omega'}(\omega)$ is the Dirac mass at $\omega = \omega'$. The Bayes risk to be minimized is:

$$E\{R(\omega, d(F))\}, \tag{6}$$

where d is a decision function based on our observations. Considering that F is constant in our model and combining the above equations we get:

$$\omega_{OPT} = \arg\min_{\omega' \in \Omega} \int_{\omega \in \Omega} R(\omega, \omega') P(\omega \mid F) d\omega = \arg\max_{\omega \in \Omega} P(\omega \mid F) = \arg\max_{\omega \in \Omega} P(F \mid \omega) P(\omega). \tag{7}$$

Now, utilize the Hammersley-Clifford theorem to substitute $P(\omega)$ probabilities with energy potentials and write the above equation in the following form:

$$\omega_{OPT} = \arg\max_{\omega \in \Omega} \prod_{s \in \boldsymbol{S}} P(f_s \mid \omega_s) \prod_{C \in \boldsymbol{C}} \exp\left(-E_C(\omega)\right). \tag{8}$$

2.3 The Energy Function

Taking the logarithm of Eq. (8) and assuming that $P(f_s \mid \omega_s)$ is Gaussian we can formulate an energy function:

$$\boldsymbol{E}(\omega, F) = E_1(\omega, F) + E_2(\omega), \tag{9}$$

where

$$E_1(\omega, F) = \sum_{s \in \boldsymbol{S}} \ln(\sqrt{2\pi}\sigma_{\omega_s}) + \frac{(f_s - \mu_{\omega_s})^2}{2\sigma^2{}_{\omega_s}}, \tag{10}$$

$$E_2(\omega) = \sum_{C \in \boldsymbol{C}} E_C(\omega), \tag{11}$$

$$E_C(\omega) = E_{\{s,r\}}(\omega_s, \omega_r) = \begin{cases} -\beta & if \quad \omega_s = \omega_r \\ \beta & if \quad \omega_s \neq \omega_r \end{cases}, \tag{12}$$

(This last one is a penalty function to encourage homogeneity of labels ω_s and ω_r in cliques).

That is to find an optimum solution for the Bayesian estimation problem (Eq. (8)) we can run an energy optimization algorithm during which we are to get the smallest energy $\boldsymbol{E}$ (Eq. (9)) over the lattice **S**.

A basic assumption in image segmentation is that the observed input image can be well characterized by a finite number of classes. Within a Gaussian model each class can be represented by its mean value μ and by its standard deviation σ. That is each possible label should have its appropriate mean μ_λ and deviation σ_λ. The first energy term (E_1) is responsible to keep the labeling close to observations, while the second energy term (E_2) is to achieve fairly homogenous regions, which is an important requirement in image segmentation. β is a positive model parameter controlling the homogeneity of regions of the segmented image. Typically, small β retains little image segments, while larger β causes the formation of larger regions. For more details of more general image models the reader should refer to [50, 56].

2.4 Optimization by Modified Metropolis Dynamics (MMD)

During the relaxation process, new random labels (also called states) are generated in each iteration and compared to the previous labels. Labels are compared on the basis of the already defined energy function (Eq. (9)), however, the decision criteria can vary according to the optimization algorithm.

Basically, the MMD algorithm [50] is a modified version of Metropolis Dynamics [60] and it turns the original algorithm into a pseudo-stochastic relaxation process. In our parallel computation framework the MMD algorithm proved to be a very effective solution: while it is a fast algorithm with satisfactory optimization abilities [50] its parallel implementation does not require high complexity.

The difference between the original Metropolis method and the MMD is the choice of a threshold value ξ used in the dynamics to accept or reject a new candidate state. In the original method ξ is chosen randomly at each iteration. In the MMD algorithm ξ is a constant threshold, now denoted by α, chosen at the beginning of the algorithm. This means that a jump to a new state η is allowed if this change does not increase the

energy "excessively". The threshold α controls the possible increase of energy that is allowed for a transition to a new state. The algorithm can be run parallel to all sites [4] and can be described as follows:

1. Pick randomly an initial configuration ω^0, with k=0 and T=T_0.
2. Using a uniform distribution, pick a global state $\eta \in \Omega \setminus \{\omega^k\}$. Compute the local energy $\mathbf{E}_s(\eta_s)$ with $\eta = [\omega_{s_1}^k, \omega_{s_2}^k, ..., \eta_s, ..., \omega_{s_N}^k]$.
3. Compute $\Delta\mathbf{E}_s = \mathbf{E}_s(\eta_s) - \mathbf{E}_s(\omega^k)$. The new label η_s at site s is accepted according to the following rule:

$$\omega_s^{k+1} = \begin{cases} \eta & if \quad \Delta E_s \le 0 \\ \eta & if \quad \Delta E_s > 0 \quad and \quad \ln(\alpha) \le \left(-\frac{\Delta E_s}{T} \right) \\ \omega_s^k & otherwise \end{cases} \tag{13}$$

   α is a constant threshold ($\alpha \in]0,1[$) chosen at the beginning of the algorithm.
4. Decrease temperature T. If $\sum_s \Delta\mathbf{E}_s$ is greater than a given threshold (or the number of changed sites is above a predefined limit), then go to Step 2, otherwise stop.

There is no explicit formula for the threshold α. In practice, in case of a noisy image α is chosen nearly equal to zero, otherwise α is chosen nearly equal to one. If the temperature is less than a certain threshold $(\Delta\mathbf{E}_{min}/(-ln\ \alpha))$ then only the jumps to states of lower energy are allowed. While this algorithm converges to a local minimum its performance is close to the Metropolis algorithm according to tests in [50]. The typical rate to decrease temperature is given by $T_{k+1} = 0.95 \cdot T_k$.

Since α is fixed, there is no need for updating its logarithmic value in every step, moreover, more sites can be updated successfully in one step. This latter property is not obvious, but according to [4] we know that the algorithm will converge as long as less than 100% of the pixels are updated at the same iteration step. It is true in our case except for events that occur with almost zero probability, and it enables fast parallel implementations without the restriction on the number of simultaneously changed sites. These features make the MMD method a simple algorithm that could be easily implemented in parallel VLSI architectures.

3 THE CNN-MRF SEGMENATION MODELS

3.1 Parallel Solution on the CNN

MRF image segmentation methods are based on a large number of local (pixel-level) calculations of potential functions. If we use a Complex Instruction Set Computer (CISC) then all functions at arbitrary complexity level can be carried out. However, if we want to use massively parallel architecture solutions, with thousand of processing elements, we should keep low the necessary complexity for local computations of the computation model. In this special case (in analog implementations at limited accuracy) it is also important to take into consideration the precision of the hardware architecture as detailed in Section 3.9 [96].

One example for parallel implementation is the Connection Machine (CM) [36,50,51]. The CM is a Single Instruction Multiple Data (SIMD) architecture, where each image pixel is assigned to one *virtual processor* at high accuracy and complexity. It is a partly parallel solution since several *virtual processors* are mapped to one *physical processor*.

If we use fully parallel machines of smart but reduced complexity cells, such as CNN, where global computations are more difficult to be carried out, we should redefine the segmentation model [96].

Basically, functions used in the segmentation can be divided into two groups:

1. There are global processes, such as:
   - Image grabbing from camera or file (G1).
   - Checking the stopping conditions (G2).
   - Image saving or image transfer (G3).
   - Image statistics computation (histograms, estimation of labeling parameters) (G4).
   - Controlling Simulated Annealing (G5).
2. There are local processes operating in a limited neighborhood:
   - Comparing labels of the neighborhood (L1).
   - Calculation of energy functions (L2).
   - Evaluating the decision function (L3).

An imaging cellular system, such as some CNN VLSI chips [21], can grab the image using on-board photo sensors. Such a cellular sensor chip can operate at video rate, and we do not need to deal with stopping conditions, since:

- The segmentation process should be convergent in time. Overtime is not a problem.
- The process should be faster than the standard video frame-rate (for an analog CNN chip the whole MRF process is approximately *0.1 msec* as simulations with physical VLSI parameters [21] show).
- We stop the iteration at a pre-defined but satisfactory number of steps.

The computation of image statistics and parameter estimation (such as determining μ_λ and σ_λ.) is a sequential algorithm. However, it does not mean any limitations for the parallel model, since these calculations are to be made only once during the whole algorithm and in some cases can be done during image transfer. These statistical parameters and the parameters of simulated annealing should be transmitted to every cell simultaneously, meaning that the system needs parallel data loading and controlling, which are basic functions of the CNN VLSI chips. However, as we show later, parameter estimation for labeling may be based on pixel-level computations using a parallel unsupervised estimation method.

Local calculations (L1-L3) can be made in parallel processes operating in the close neighborhood of a pixel. On the other hand, the cellular parallel processing architecture is not capable of all kinds of local computations. For example, if we used the Gibbs Sampler [30] for combinatorial optimization we would have to compute exponential functions to evaluate the decision functions. While it was available in the CM, it is not in the CNN framework. Fortunately the MMD algorithm avoids this problem.

According to the latest chip designs [21,73] only the following operations are accessible in our framework: addition, multiplication [39], on-board photo sensors, local and global memories, and logical functions. Besides these it was also suggested to use simple stored functions (jigsaw, comparison) [96], which can be easily implemented at the current technological level.

As for the generation of new random candidates in the MMD algorithm, random number generation can be a pseudo random process. A serial process can compute the random numbers of an initial random configuration, and then new random global states can be generated by spatially constant offsets. Resulted values can be transformed into the valid data range with the help of a jigsaw-like function.

Division is only done in pre-processing when the inverse of standard deviations of classes is calculated (see Eq. (10)). Using the MMD algorithm [50] *7-8* memories by cell are needed. The inverse of variance and mean of a possible class should be fed into the cells by parallel lines. If these values are stored in local memories during the whole segmentation algorithm, then a cell needs additional memories as much as twice the number of the possible labels. Later it is shown that only *2* additional memories are necessary if we use a simplified pixel-level parameter method.

For a detailed explanation of the implementation of the MRF model on the CNN-UM the reader is referred to [96].

3.2 A Monogrid CNN-MRF Model Based on Local Statistics

The previous implementation of the MRF segmentation model on 2D processor arrays requires a large amount of local memories for storing class statistics and for the computation of the energy functions. However, the number of these pixel-based memories is limited due to technological reasons. To reduce the need for local memories a constant deviation model (the standard deviation of all classes was substituted by one value) was introduced in [96].

In this section we discuss a different model that is based on local statistics and was also first proposed in [96]. It does not require the estimation and store of global class characteristics; instead, a local Bayesian model is used.

It is supposed that a rough estimation of the gray level of classes can be made by finding the histogram peaks of the smoothed input image. Thus μ_{ω_s} means the estimated gray level of class ω at pixel s. It is also assumed that the different classes can be very roughly approximated locally by the expected value and variance defined:

$$\mu_s = \frac{f_s + s_s}{2}, \tag{14}$$

and

$$\sigma_s = \frac{|f_s - s_s|}{2}, \tag{15}$$

where s_s is the average value around pixel s obtained by a smoothing operator. This means that we expect the segmented image to be somewhere halfway between the observed value and its smoothed version. Although it seems to be a strong simplification, it fits the general segmentation model where the different classes are approximated through a smoothing operator. Since this kind of local approximation fits the CNN architecture very well, let us call this model the CNN-MRF model in this work.

Now, the energy term defined in Eq. (10) is rewritten:

$$E_1(\omega, F) = \sum_{s \in \mathbf{S}} \ln(\sqrt{2\pi}\sigma_s) + \frac{\left(\mu_s - \mu_{\omega_s}\right)^2}{2\sigma_s^2}. \tag{16}$$

Since variances are constant at each position, the calculation of $\Delta\mathbf{E}_s = \mathbf{E}_s(\eta) - \mathbf{E}_s(\omega^k)$ is greatly simplified compared to the original MRF model. The first part of Eq. (16) can be simply neglected, while the numerator of the second part can be written with the help of 3 addictions/subtractions and 1 multiplication:

$$\left(\mu_{\omega_s^{k+1}} - \mu_s\right)^2 - \left(\mu_{\omega_s^k} - \mu_s\right)^2 = \left(\mu_{\omega_s^{k+1}} - \mu_{\omega_s^k}\right)\left(\mu_{\omega_s^{k+1}} + \mu_{\omega_s^k} - 2\mu_s\right). \tag{17}$$

E_2 is calculated using a simple equality detector for each neighbor of s for the current state ω_s^k and for the new candidate ω_s^{k+1}.

However, in some cases we might need to increase neighborhood connectivity to achieve good segmentation results. Fig. 2 illustrates the third order neighborhood system used in the CNN-MRF implementation.

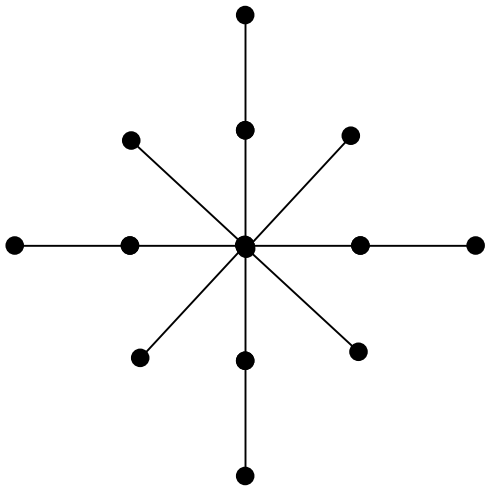


Figure 2.
A central site and its 12 neighbors in the third order neighborhood-system.

According to the MMD algorithm temperature control can also be realized by simple functions. After setting an initial temperature the current temperature should be decreased in each iteration step. Since T is uniform over the whole image field it can be represented and updated in a global memory, then the new value can be downloaded to local memories. Logical operators, appropriate for the parallel array model, can then execute the MMD decision rule given by Eq. (13).

Fig. 3 shows the flow chart of the initialization (dotted lines) and one cycle of the segmentation process [96]. Local memories are denoted by M#, 8 of which is required in the CNN-MRF architecture. Every step of the algorithm can be done parallel in the CNN-UM except for the initial classification and the calculation of some global

parameters. Since the division operator is not available in the CNN VLSI environment, the calculation of $\frac{1}{\sigma_s^2}$ is also a serial process and part of the initialization. However, some restricted division can be implemented in VLSI using a nonlinear function but at limited accuracy. Due to the nature of the σ calculus no high precision is needed for the division.

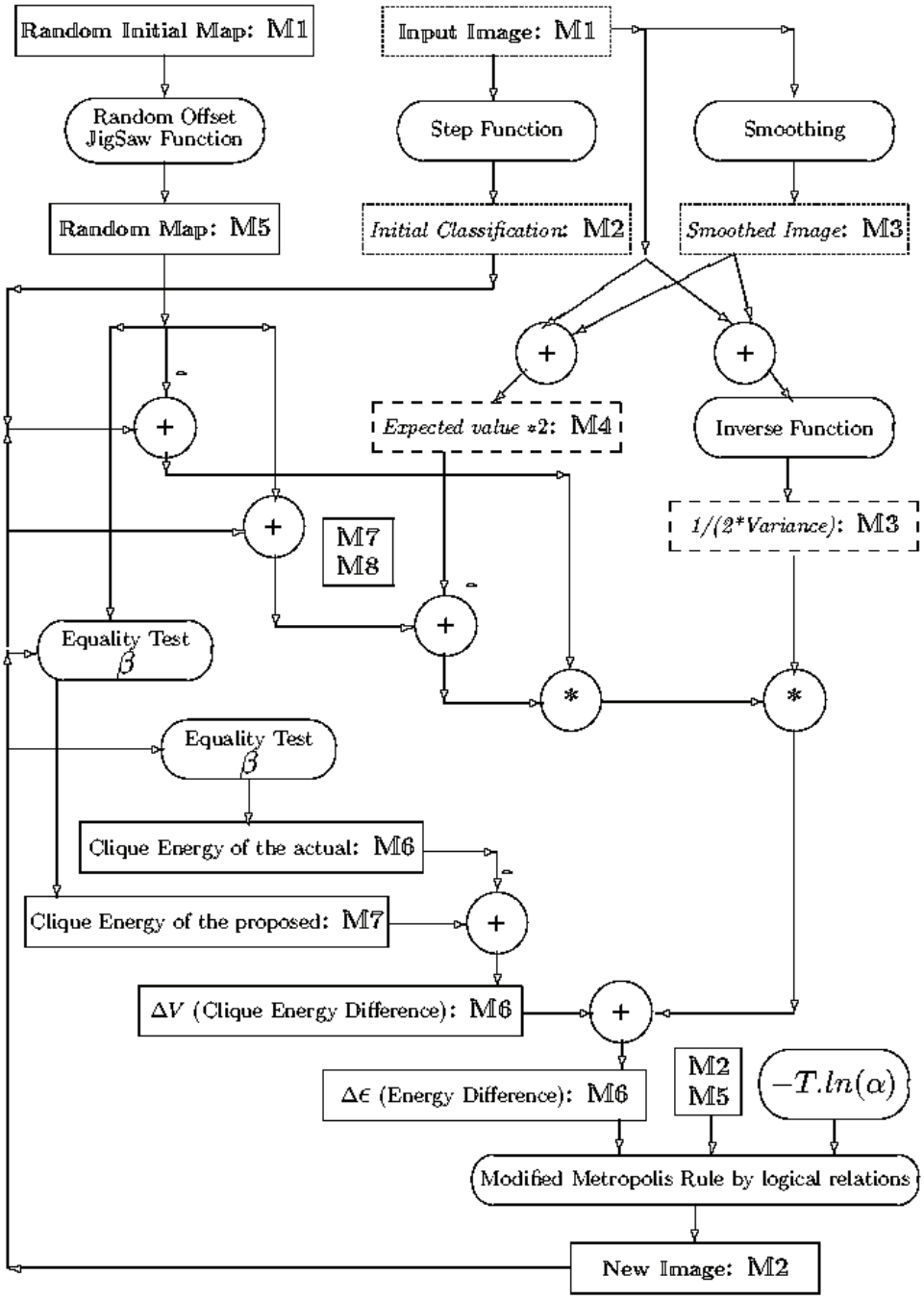


Figure 3.
Architecture of the CNN-MRF model with 8 local memories and simple functions appropriate for the CNN-UM [96].

3.3 Experiments with the Monogrid CNN-MRF model

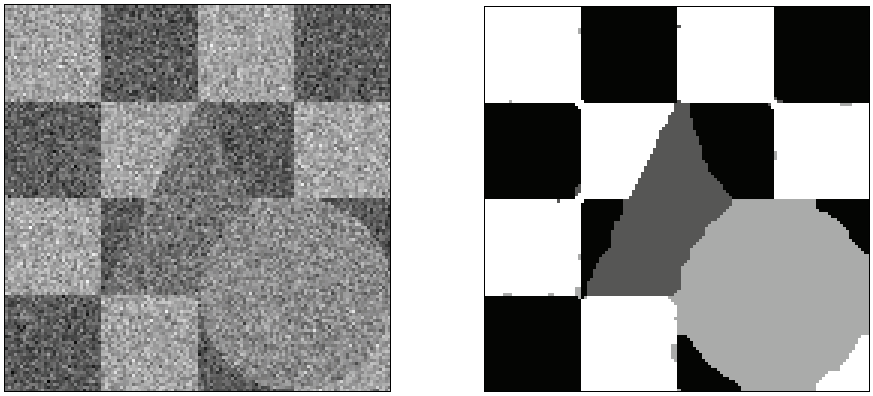
The experiments have been carried out on a software simulator system using a fixed-point hardware accelerator board installed in a PC [76]. The simulator has limited accuracy, so values had to be normalized to avoid over- or under-flow problems. The accumulated rounding error and the nonlinear saturation of the CNN simulator results in a loss of precision. Since still satisfactory results could be achieved in this environment, it predicts that an image processing system implemented in VLSI circuits should be robust [90] either.

The CNN-MRF model was tested both on artificial and natural images. In the first case the artificially generated pictures consisted of well-defined regions of given number of classes. These original images were loaded with heavy noise and the segmentation algorithm was expected to reconstruct the same homogenous regions as the original image had. In this case comparing the noise-free and the segmented image pixel by pixel we could measure the segmentation error. In case of natural born images the evaluation of the segmentation results is much more difficult. Generally, the comparison is based on hand-made segmentation, or results are evaluated by other subjective means. Only a few cases allow reliable comparisons to ground truth data.

Fig. 4 shows the segmentation results using a noisy input test image (SNR=5dB). Parameters in the segmentation process are: $T_{k+1}=0.95 \cdot T_k$, $\alpha=0.3$, and $\beta=10.0$. The misclassification error is *1.5%*. Using the original MMD algorithm on a CM [36] the error is 1.3 %, and it is 1.0% with the Metropolis algorithm [50,48].

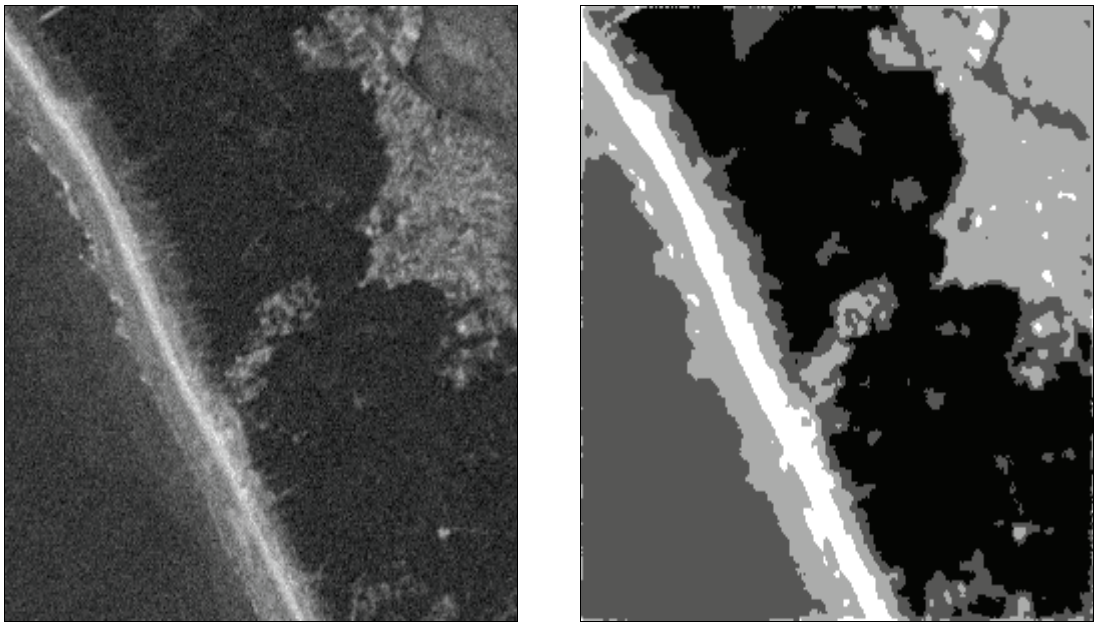
We have checked the segmentation error with respect to the number of iteration steps for different α values. We have found that the process is convergent and settles in about 100 iteration steps. In Fig. 5, the unsupervised segmentation of a SPOT satellite image is shown. Here 4 output classes were defined. Parameters are $T_{k+1} = 0.95 \cdot T_k$, $\alpha=0.3$ and $\beta=0.5$.

It is important to mention that these segmentation results, and also other results generated with a stochastic iterative process are not always absolutely stable and exactly reproducible. Depending on the speed of convergence, initial temperature, etc. there can be a few fractions of percentage differences in the outcome of the same experiments.



Input noisy image (SNR=5dB) *CNN-MRF Segmented Result*

Figure 4.
Monogrid segmentation of an artificial test image.



Input SPOT Image *CNN-MRF Segmentation*

Figure 5.
Satellite Image Segmentation, 4 classes.

3.4 Nonlinear Diffusion for Detail Conservation

In [93] it has been shown that the segmentation error of the CNN-MRF algorithm significantly depends on the type of diffusion used to calculate the expected value and the variance in each pixel position. Nonlinear (or anisotropic) diffusion methods are

much more satisfactory to generate s_s, the average value around pixel s used in Eq. (14) and Eq. (15), than linear smoothing operators.

While linear smoothing is defined by equation:

$$\frac{dI}{dt} = div(gradI), \tag{18}$$

edge driven diffusion defined by Perona, Malik and Catté [12,67] is given by:

$$\frac{dI}{dt} = div(g \circ (\|grad(G * I)\|) gradI), \tag{19}$$

where G is a Gaussian pre-smoothing filter for noise suppression, and

$$g = \exp\left(-\left\| \frac{id}{K} \right\|^n \right), \tag{20}$$

K>0, n>0, typically n=2. K can control the edge conservation property of the filter.

A simpler equation for nonlinear isotropic diffusion was proposed in [79]:

$$\frac{dI}{dt} = g \circ (\|gradI\|) div(gradI). \tag{21}$$

Since in case of both nonlinear solutions g is given by an exponential form the direct implementation of these nonlinear models would require a physically hardly realizable function at each cell. That is why the exponential function is replaced by a linear term detailed in a following section.

3.5 Multigrid Models

Nowadays multiresolution, multiscale, hierarchical approaches are widely applied in the field of image processing. Markov Random Fields are one typical area where the advantages of such techniques seem to be tremendous. A review on this widespreading area can be found in [32]. It is worthy to note that there is a certain confusion in the terminology regarding techniques processing images on more than one representation level. The words like multigrid, multiresolution, multiscale, hierarchical are to be considered generally synonyms, while in some papers they are addressed specially to some well-defined techniques. In this dissertation I usually call the models simply monogrid or multiscale/multigrid depending on whether we apply multiresolution processing or not.

Here are some reasons why multigrid models are usually preferable:

- In a general case many phenomena (e.g. fractal-like signals) have intrinsic multiscale properties so it may have a natural meaning to apply similar operators on different scales.
- In our case we face an optimization problem with possible local minima. Multiscale optimization can avoid being trapped in local minima. It can also result in faster convergence and become less sensitive to initial configurations.

Generally speaking, the common feature of all multiscale models is the representation of images on several levels with decreasing resolution, while there can be significant differences in the definition of cliques and energy functions in the different approaches (see [32] for details).

The VLSI implementation of the CNN-MRF model with a third order neighborhood-system can be technologically costly, on the other hand, as experienced, our simple cell-array system with first order neighborhood connectivity and with unsupervised pixel-level parameter estimation does not give satisfactory results. What it is expected from multigrid implementations is to reduce the necessary neighborhood connectivity of the monogrid CNN-MRF model at comparable segmentation results and acceptable ratio of the number of operations per iteration.

Multiscale representation can be introduced into a parallel 2D cell-processing framework in two different ways:

1. At lower scale representations rectangular groups of pixels are restricted to have the same value. The size of the image array is constant; these restrictions are responsible to achieve lower resolutions. In this case there is no need for moving or reorganizing the pixel values when changing resolution. However, if we do not want to exceed first order connectivity, only two level image pyramids are feasible this way.
2. The size of the image arrays is changing with resolution. When increasing or decreasing the resolution, pixels should be read out from the 2D array, reorganized and downloaded in a serial process.

These two solutions are equivalent in segmentation, however, they need different computational complexity and memory requirements when implemented in the parallel 2D processor array.

3.6 The CNN-MRF Multiscale Model

Now, I introduce a multiscale structure for the CNN-MRF model [18]. While the proposed multiscale model is similar to [35], I certainly had to take into consideration the capabilities of the CNN-UM platform where it is supposed to be implemented. In this model we have a top-down strategy from coarser representation of the image to finer scales (in experiments 2x2 sites were used to build up a coarser block, i.e. the re-scale ratio n equals 2). The optimization starts on the coarsest level and the obtained equilibrium state serves as the initial state for the further optimization on the finer layer below. During the optimization process there are no interactions between the scales except for the initial data.

An important feature of this model is the calculation of clique potentials. At a given scale the second part of the energy term (E_2) is calculated through the cliques of the finest scale. While at all levels the cliques of the finest representation are considered for computation, there is a restriction that blocks of n^{2i} sites must have the same value. Here n is the re-scale ratio, i is the actual level of representation in the image pyramid.

Cliques, always defined on the finest scale, corresponding to a block at a certain level, can be partitioned into two sets. The first set contains cliques that belong to sites all inside the block, whereas the other set of cliques consists of the ones that connect sites of two neighboring blocks. The energy contribution of the first set is given by $p\beta$ in Eq. (22) while Eq. (24) gives the energy of the other set. Clique potentials at a given scale are the sum of these energies measured on the finest resolution.

The following forms represent all energy components at level i similarly to Eq. (9) with the modifications corresponding to the description given above:

$$E_1^i(\omega^i, F) = \sum_{s^i \in S^i} E_1^i(\omega_{s^i}, F) = \sum_{s \in b_{s^i}^i} \left(\ln(\sqrt{2\pi}\sigma_s) + \frac{(\mu_s - \mu_{\omega_s})^2}{2\sigma_s^2} \right) - p\beta \tag{22}$$

and

$$E_2^i(\omega^i) = \sum_{C^i \in \mathbf{C^i}} E^i(\omega^i) \tag{23}$$

where

$$E^i(\omega^i) = \sum_{(r,s) \in D_{C^i}} E(\omega_r, \omega_s) = \begin{cases} -q\beta & if \quad \omega_r = \omega_s \\ +q\beta & if \quad \omega_r \neq \omega_s \end{cases}. \tag{24}$$

Explanations to the notations of the above equations:

- ω_{s^i} means the labeling of one block at scale *i*.
- $s \in b^i_{s^i}$ means sites of the finest scale, which build up a block at scale *i*.
- C^i is one clique, while $\boldsymbol{C}^i$ is the set of all cliques at scale *i*.
- D_{C^i} is the set of those sites which build up clique C^i.
- The number of cliques included in one block at scale *i* is $p=2n^i(n^i-1)$, while the number of cliques between two neighboring blocks is $q = n^i$.

Note that these equations apply only for a first order neighborhood-system since our purpose is to reduce the necessary neighborhood connectivity of the monogrid CNN-MRF model.

Fig. 6 represents the two kinds of cliques mentioned, and Fig. 7 illustrates the main steps in the optimization that are:

1. Energy optimization of a layer.
2. Initialization of the next layer from a coarser layer above.

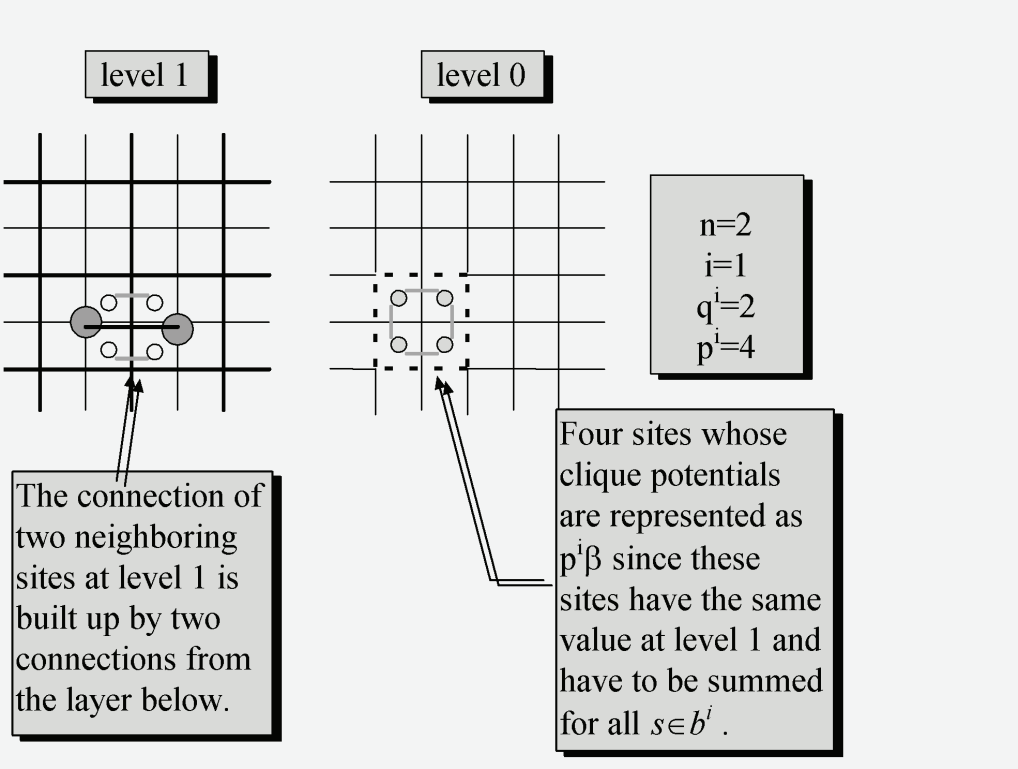


Figure 6.
Multiscale cliques.

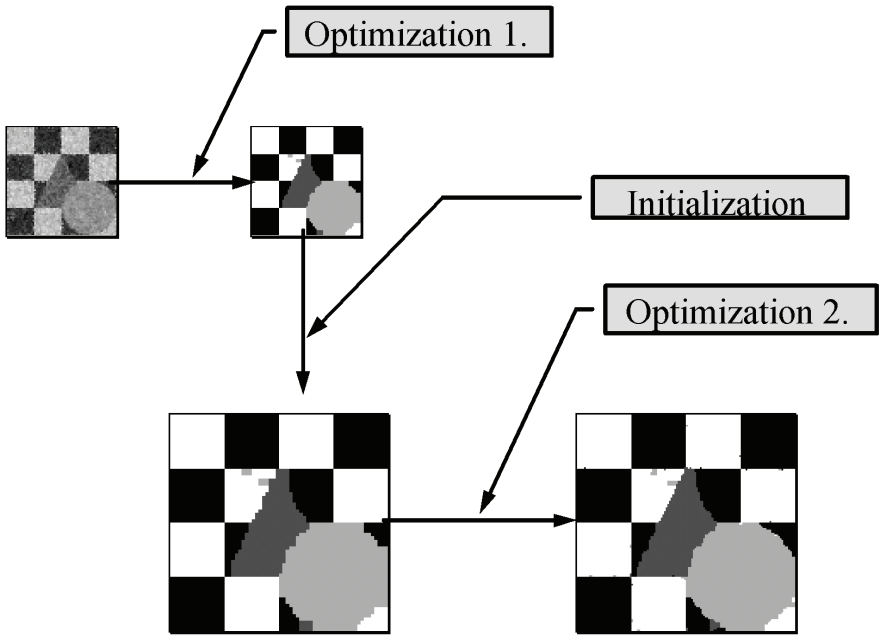


Figure 7.
Main steps of multiscale segmentation.

3.7 Experiments with the Multigrid CNN-MRF Model

Besides the model described in the previous section I investigated other multigrid models, e.g. a causal hierarchical model similar to [10]. Here I do not deal with other multigrid algorithms, since the limited set of operations available in our parallel framework did not lead to satisfactory segmentation results with these models.

We have found that two implementations of the CNN-MRF multiscale model could work properly in the CNN-UM or in other similar structures. Both of these multiscale implementations need more local memories per cell and functionality than the monogrid version but use only first order neighborhood connectivity.

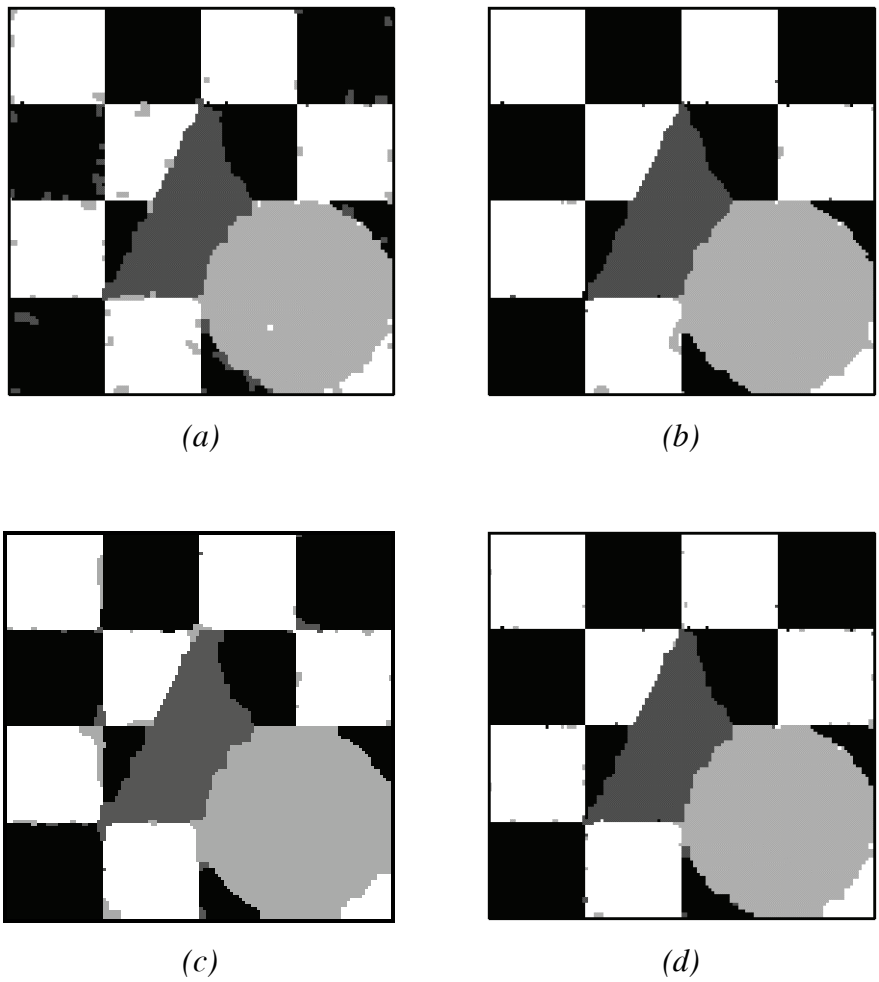
There are two possible algorithmic implementations of the proposed multiscale model:

1. The first realizes the model just as given by Eq. (22) and Eq. (24). We start the algorithm at the coarsest scale where the image size is naturally smaller than the input image. Since we work on a chip of fixed size, other parts of the array are not used except for the finest level. Local observations (Eq. (14) and

Eq. (15)) are smoothed then re-scaled and stored for all scales independently. When the segmented image is stable, then all labels are read out and mapped to a higher resolution by duplicating sites.

2. The second implementation realizes directly the idea that is behind the formulae. In this case we have only the finest scale represented in the cell array and instead of building a coarser (smaller) grid we restrict the values of sites so as to be the same in every block defined on a higher scale. It means that there is no need for the reconfiguration of sites when changing resolution. Initial observations (expected value and variance) are downsampled for lower resolutions and doubletons (Eq. (12)) are computed for all sites of the finest resolution. The crucial point of this implementation is the summation of energy terms over blocks of $n^i x n^i$ pixels at scale i. Since we should not exceed first order neighborhood connectivity only blocks of size *2x2* are appropriate, larger areas are not adequate for collecting and summing up clique potentials. Thus only two level pyramids are supported by this construction. The generation of random labels is clear: since they are generated by offset of the first iteration, spatial restrictions apply only in the first random step.

Since both techniques are based on the same theory, very similar segmentation results are expected, however, they have different complexity and different computation time. In our experiments both models were optimized with the MMD technique with various ad hoc parameters.

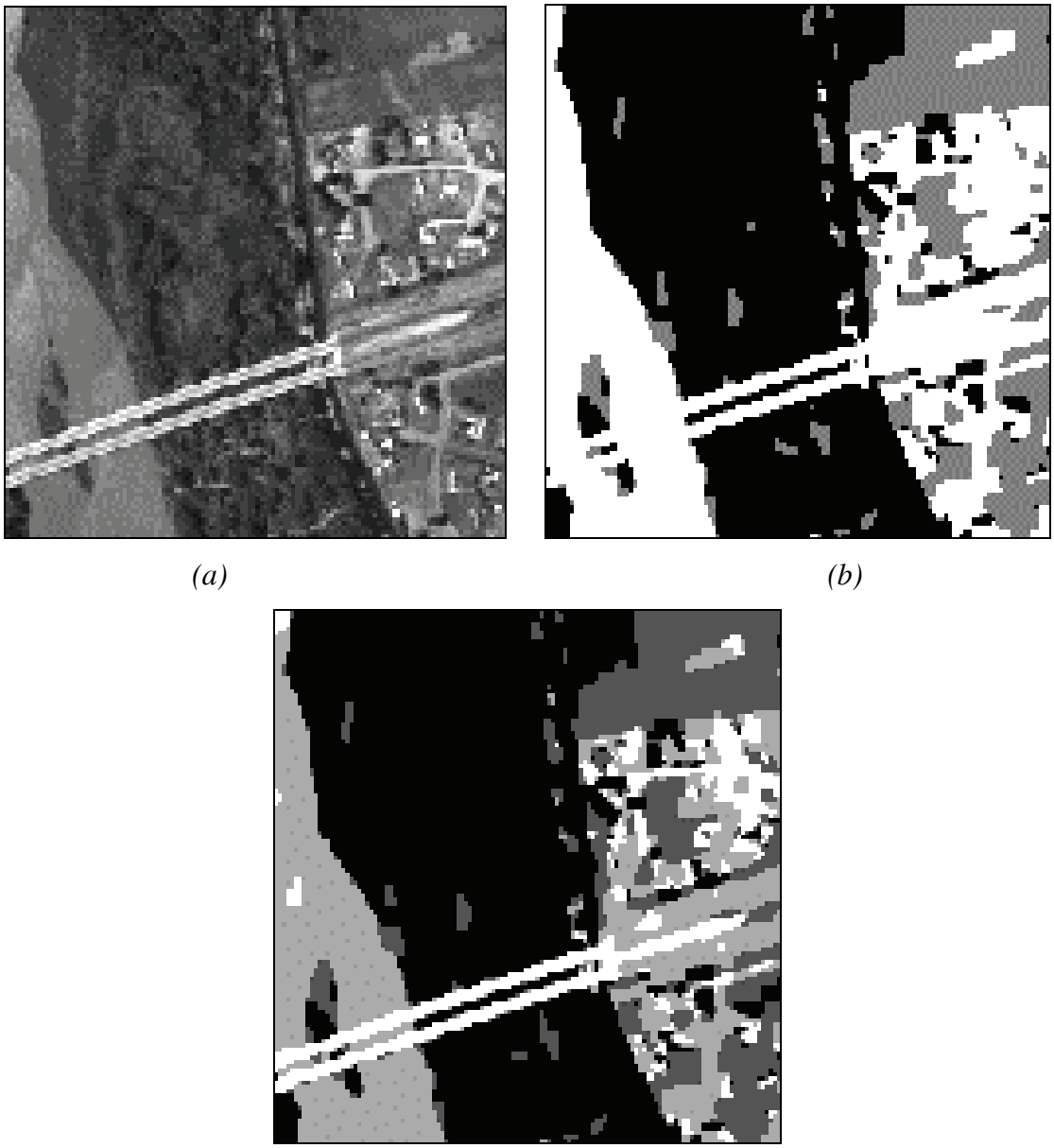


(a) *(b)*

(c) *(d)*

Figure 8.

*Segmentation results applying different smoothing methods in the pre-processing and different CNN-MRF. (a) Monogrid model, first order neighborhood, 150 iteration steps, nonuniform smoothing in the pre-processing, misclassification error: 3.7%. (b) Monogrid model, third order neighborhood, 150 iteration steps, nonuniform smoothing in the pre-processing, misclassification error: 1.6%. (c) Multiscale (2 scales) representation, first order neighborhood, 2*80 iterations, uniform smoothing in the pre-processing, misclassification error: 3.5%, (d) Multiscale (2 scales) representation, first order neighborhood, 2*80 iteration steps, nonuniform smoothing in the pre-processing, misclassification error: 1.7%.*



(a) *(b)*

(c)

Figure 9.
Multiscale MRF segmentation
(a) Air view of a scene with river, bridge, forest, green area and town (from left to right) at Rio Grande, New Mexico. Image is segmented into (b) 3 classes, (c) 4 classes.

As Fig. 8 illustrates, there is a significant increase in segmentation quality compared to the first order monogrid model, while best results of the first order multiscale model are quite close to the results of the third order monogrid model.

In Fig. 9, we can see the segmentation of an air born image. The input image (*a*) is a part of an image from Airborne Multisensor Pod System Data Access Catalogue

(http://info.amps.gov:2080/). Segmentation was done with a 2 level multiscale method in *2*80* iteration steps.

3.8 Characteristics of the Implemented Models: Comparing Monogrid and Multigrid CNN-MRF Models

Table 1 summarizes the most important properties of the monogrid and the two implementations of the multiscale model. Multiscale models need more memory per cell but need smaller neighborhood connectivity than the monogrid model to achieve comparable results. To decide which one to use in a given parallel environment depends on the available technological potential.

	Monogrid	**Multigrid 1**	**Multigrid 2**
Memories/Cell	8	15	10
Operations/Iteration	8	24^{+} / $8^{\#}$	13
Array Size (input is of size $N \cdot N$)	$N \cdot N$	$(N/2) \cdot (N/2)^{+}$; $N \cdot N^{\#}$	$N \cdot N$

Table 1
Some characteristics of the models in case of two scales. **#** *stands for the finest scale,* **+** *for the coarse scale.*

3.9 The Robustness of the CNN-MRF Model on Imprecise Analog Circuits

In [90] it has been shown that the analog structure of the CNN is highly robust against parameter noise, image noise, and the imperfect estimation of parameters in different algorithms including feedback effects, such as the tasks of image deblurring and texture segmentation.

Now the question is how robust the proposed segmentation model is to noise originating from imperfect physical realization. To clarify this question different noise models have been generated to simulate possible analog circuit noise. Table 2 lists variations of the different noise levels (all have zero expected value). Absolute noise was used in additions and relative noise in case of multiplications.

Noise Model	Absolute Noise (Variation)	Relative Noise (Variation)
Noise0	0.0	0.0
Noise1	0.0017	0.0029
Noise2	0.0029	0.0058
Noise3	0.0046	0.0087
Noise4	0.0058	0.012
Noise5	0.0087	0.014
Noise6	0.012	0.017
Noise7	0.014	0.02
Noise8	0.02	0.026
Noise9	0.023	0.029
Noise10	0.035	0.04
Noise11	0.046	0.52

Table 2
Noise models of zero expected value and different variance.

There are some key points where the original model should be modified to fit a noisy representation and computation model built of simple operations:

1. The exponential function *g*, given in Eq. (20) necessary for the nonlinear diffusion, should be replaced by a linear approximation and this approximation should be set to the noise level present during computations. First *g* should be scaled from the range [0,1] to [0,10] otherwise it would be too much affected by absolute noise. Then the proposed linear function is given by:

$$g(x) = \begin{cases} f(x) & \text{if } x \geq 0 \text{ and } f(x) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where

$$f(x) = \frac{-(10 - Noise)x}{\sqrt{-K^2 \ln\left(\frac{Noise}{10}\right)}} + 10, \tag{26}$$

where *Noise* is a parameter related to the maximal amount or the variation of noise. Fig. 10 illustrates the original exponential function and its linear approximation.

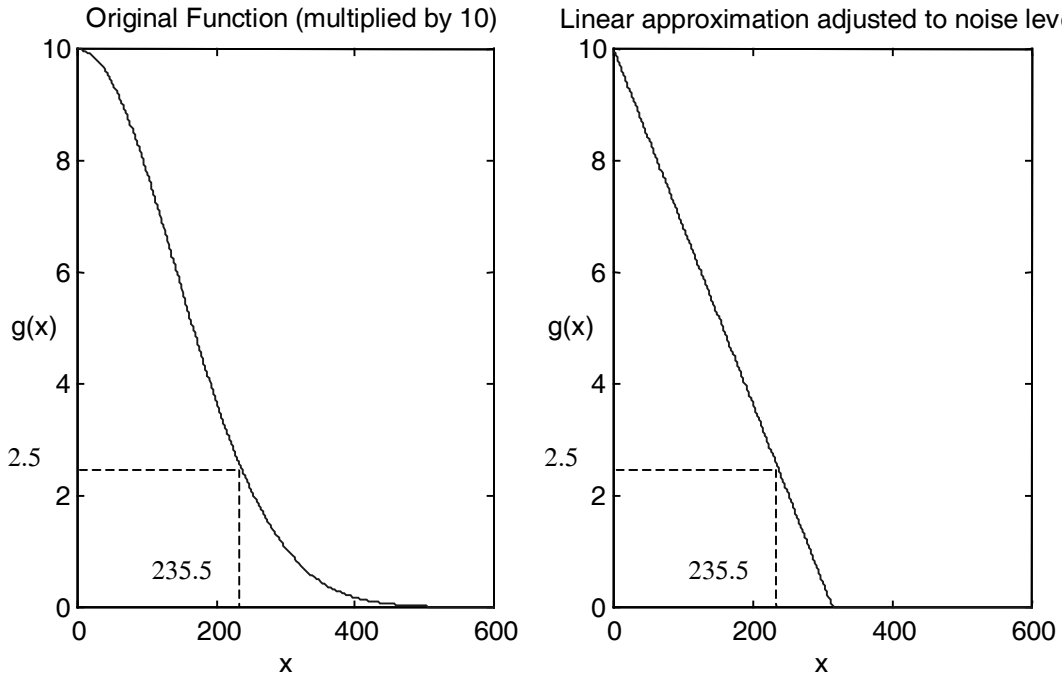


Figure 10.
Linear approximation of function ***g*** *of exponential form, Noise=2.5, K=200.*

2. Eq. (12) should also be modified, since the noisy representation of variables makes it impossible to compare neighboring pixels without introducing a threshold at comparisons. Then Eq. (12) becomes:

$$E_C(\omega) = E_{\{s,r\}}(\omega_s, \omega_r) = \begin{cases} -\beta & if \quad |\omega_s - \omega_r| < Noise \\ +\beta & if \quad |\omega_s - \omega_r| > Noise \end{cases} \tag{27}$$

(If the absolute function is not easily feasible in the CNN implementation, a quadratic form can replace it.)

There are other important parts of the segmentation algorithm where noise can greatly effect the outcome of the segmentation:

3. The calculation of expected value given in Eq. (14).
4. The evaluation of energy potentials in the MMD decision process defined in Eq. (13).
5. Random numbers stored in local analog memories can also be affected by noise. If these perturbations accumulate over hundreds of iteration steps, the convergence of the segmentation process can be lost. Therefore, within a given number of iterations (e.g., in our examples, in approximately every 40 iterations) the image should be classified to the original label values, similarly to the method of initial classification.

6. The contribution of β to the energy given in Eq. (12) is also affected by analog circuit noise. This perturbation contributes to the uncertainty of the energy (Eq. (9)), which at last, has a similar effect than a stochastic modification in the temperature parameter of the MMD decision process.

Fig. 11 illustrates the segmentation error on a test image when the exponential function was approximated with a piecewise linear form. The efficiency was investigated at different noise levels and at different diffusion steps of the form given by Eq. (19). As the graphs show segmentation error increases at higher noise levels, nevertheless this loss of performance is not monotonic. It is also observable that the number of diffusion steps can modify the outcome of the algorithm, but not significantly.

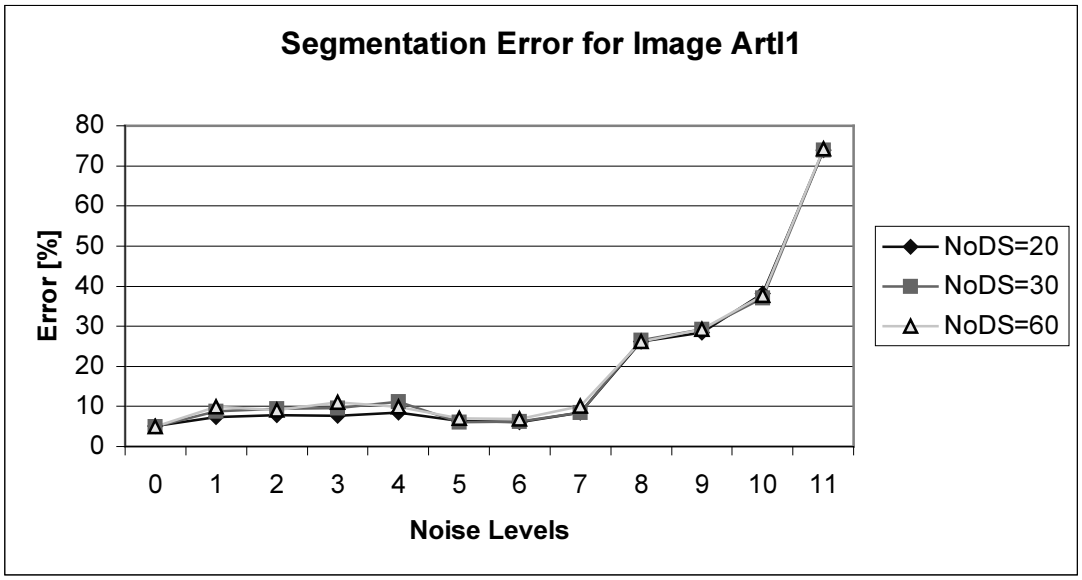


Figure 11. Segmentation error vs. computational noise at different number of diffusion steps (NoDS) when g has exponential form. Input test image is in Fig. 15.

We also investigated the effects of noise in case of the proposed linear approximation of function *g* (Eq. (25)). Fig. 12 shows the results of the same segmentation algorithm as Fig. 11 but with the modified diffusion characteristics. (For images see Fig. 15 and Fig. 17.) It is interesting to see that the segmentation error even decreases at moderate noise levels. The form of nonlinear diffusion can extensively modify the performance of the CNN-MRF models, and the proposed approximation of *g* (Eq. (25)) is much more robust at the different noise levels, suitable to replace the original function in the noisy environment.

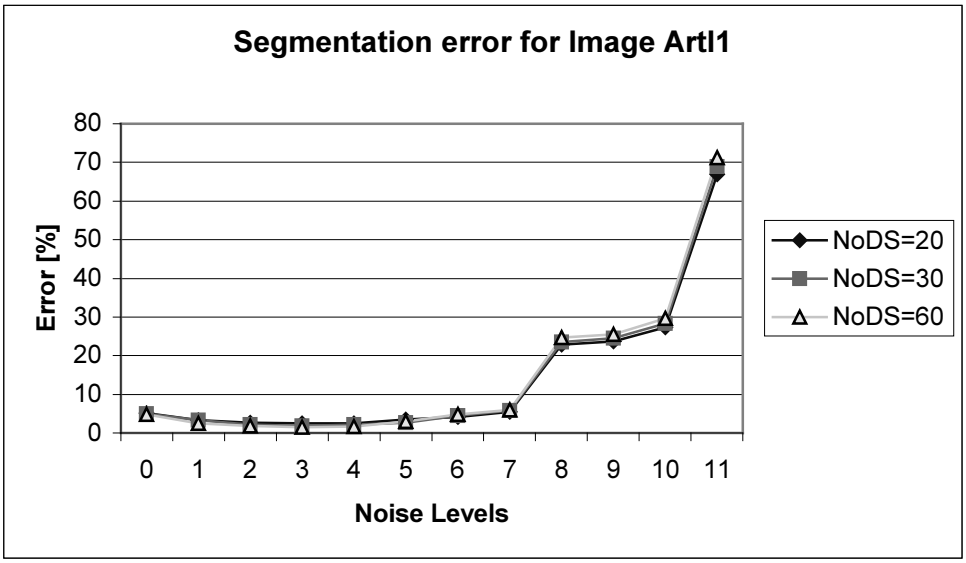


Figure 12. Segmentation error vs. computational noise for different number of diffusion steps (NoDS) when g has linear form. For images see Fig. 15 and Fig. 17.

Now the question is what causes the increase of performance of noisy models. The reason for this can be found in the application of the MMD optimization method. MMD is a pseudo-stochastic relaxation process for energy minimization where energy increase is also allowed above a certain temperature given by $\Delta\mathbf{E}_{min}/(-ln\ \alpha)$. Below this threshold the algorithm becomes deterministic, only better configurations are accepted with lower energy terms. However, in Simulated Annealing [52], the most widely used stochastic optimization algorithm, a random number ξ is used in the decision function instead of the fixed α, that is $\xi \le \left(-\frac{\Delta \boldsymbol{E}_s}{T}\right)$ is applied instead of $\ln(\alpha) \le \left(-\frac{\Delta \boldsymbol{E}_s}{T}\right)$ in Eq. (13) and stochastic relaxation methods are proved to be more efficient in global optimization than deterministic ones.

In our model analog circuit noise can have effects on the CNN-MRF segmentation algorithm in points 1-6 given above. In all cases they have influence on the computation of local energy, that is the uncertainty of the MMD decision is increased. In case of noise the MMD algorithm is no more a pseudo-stochastic process consisting of a pseudo-stochastic and a deterministic phase, but becomes a truly stochastic relaxation. It is justified if we plot the numbers of sites that are altered in each iteration. As can be seen on Fig. 13 and Fig. 14 the number of altered pixels is decreasing very fast in case of the noise free model (in the second example the difference is less remarkable but still significant in the first 60 iterations).

Besides testing the algorithm on images loaded with Gaussian noise, one may ask how the procedure reacts if the input image is translated/rotated and the segmentation parameters are unchanged. Although in case of 1 order connectivity the smoothness term does not supports homogeneity in the diagonal direction, no significant fall in segmentation quality were observed in experiments (see Fig. 17 for example).

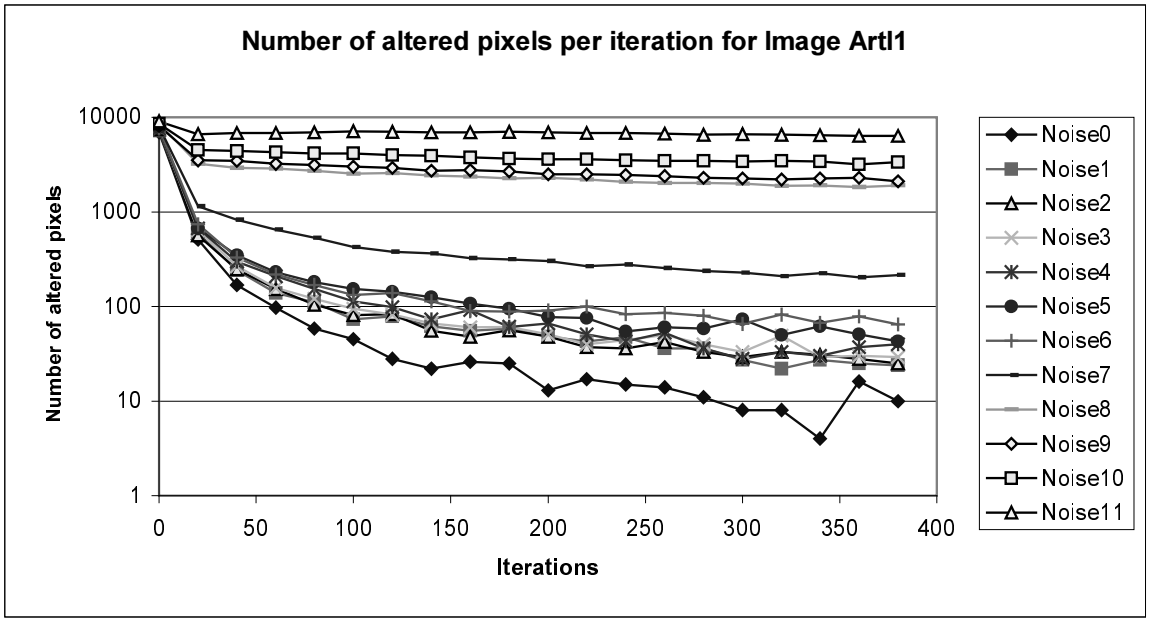


Figure 13. Number of altered sites per iteration for image ArtI1.

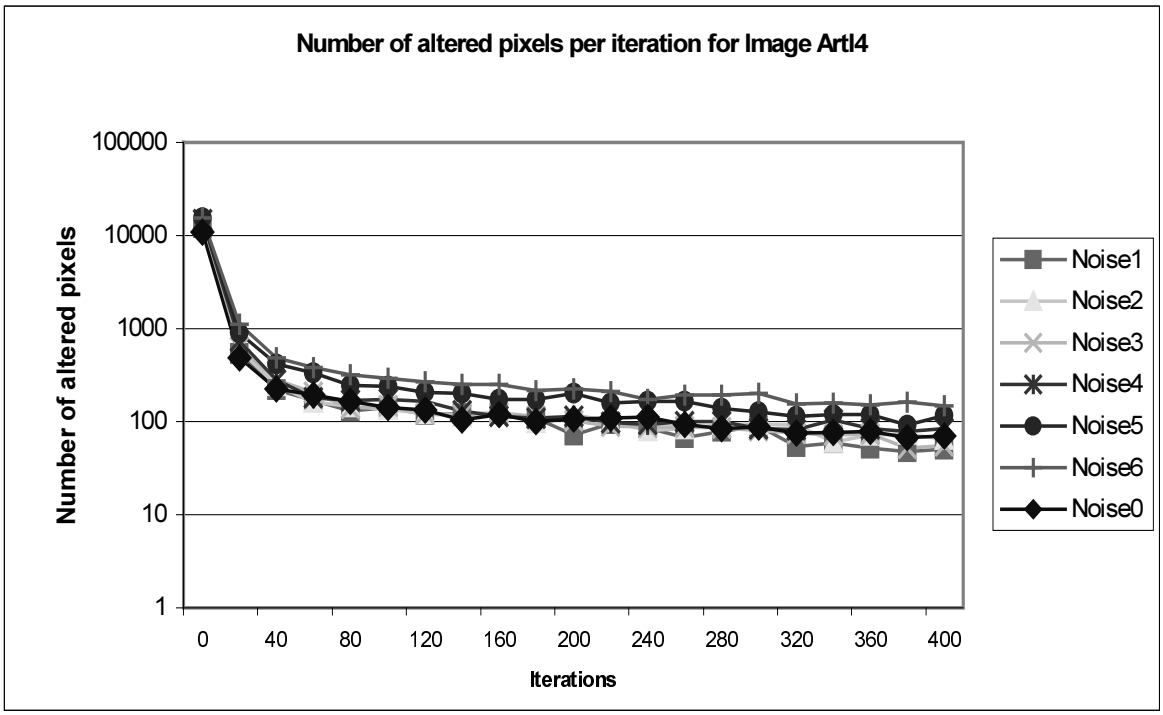
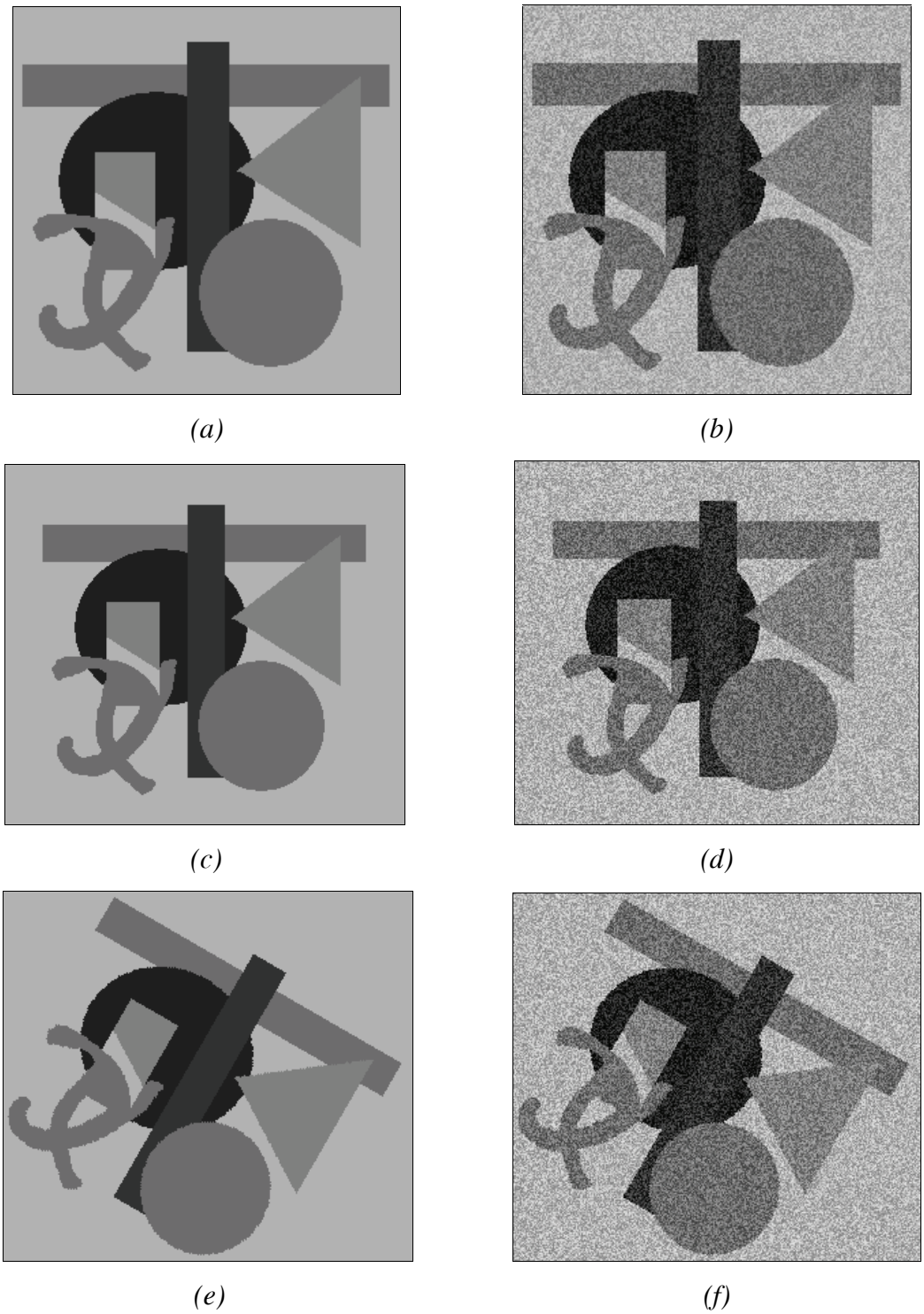


Figure 14. Number of altered sites per iteration for image ArtI4.



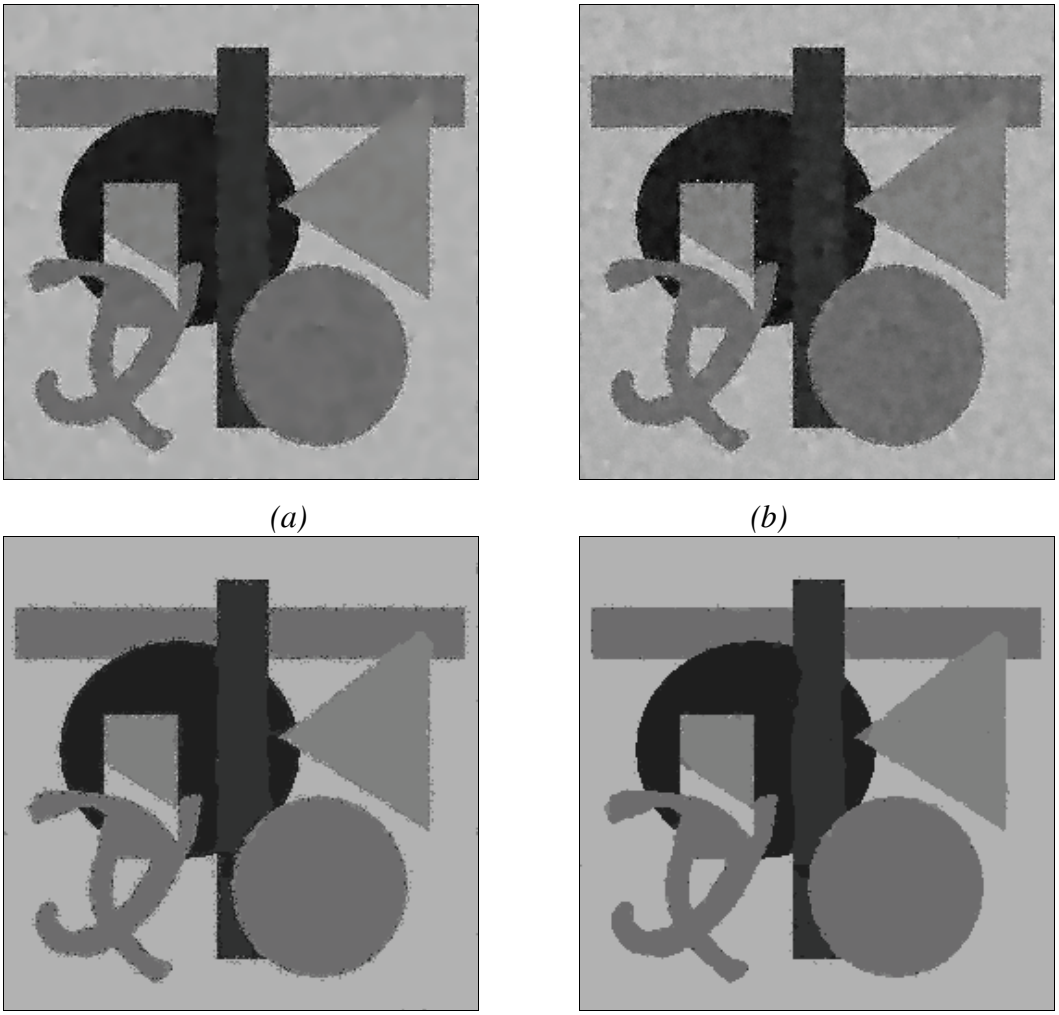
(a) *(b)*

(c) *(d)*

(e) *(f)*

Figure 15. Test images: (a)"ArtI1" (b) "a" with noise PSNR[1]=21.36dB (c) "ArtI2" (d) "c" with noise PSNR=18.84dB, (e)"c" rotated "ArtI3" (f) "e" with noise PSNR=18.9dB respectively.

[1] $PSNR = 10\log_{10}\frac{255^2}{MSE}$ where $MSE = \frac{1}{NM}\sum_{j=0}^{M-1}\sum_{i=0}^{N-1}(x_{i,j} - \bar{x}_{i,j})^2$ in case of images x and $\bar{x}$ of size *NxM*.



(a) *(b)*

(c) *(d)*

Figure 16. Segmentation of "ArtI1" (a) Noise-free nonlinear diffusion (Noise0), (b) nonlinear diffusion in model Noise3, (c) segmentation or "AritI1" model Noise0, error= 4.7%, (d) segmentation in model Noise4, error=1.7%, NoDS=30 and Beta=18 for both cases.

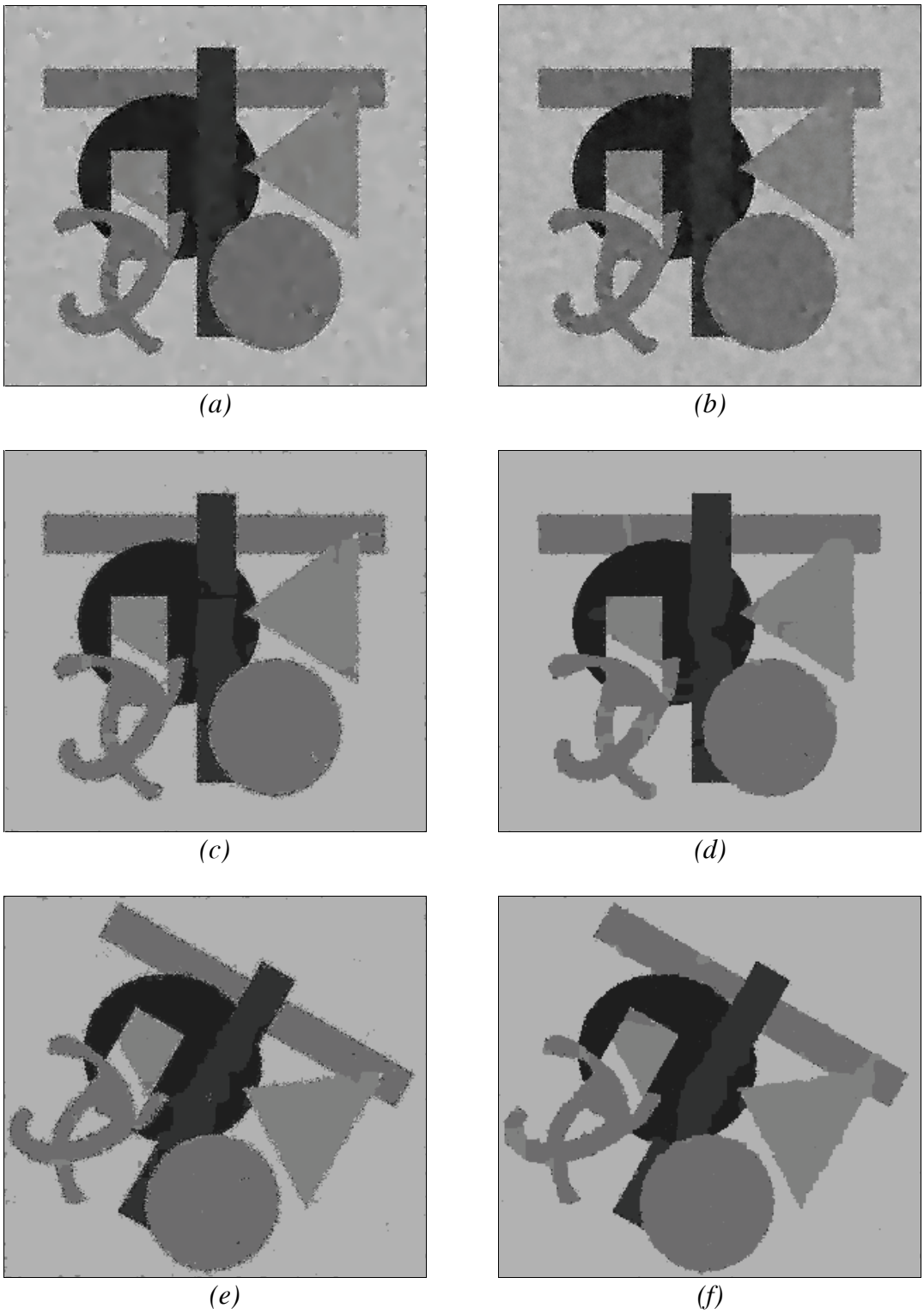


Figure 17. Segmentation of "ArtI2" and "ArtI3"
(a) Noise-free nonlinear diffusion (Noise0), (b) nonlinear diffusion in model Noise4, (c) segmentation result in Noise0, error= 5.74%, (d) segmentation result in Noise4, error=3.29%, (e) segmentation result in Noise0, error= 5.67% (f) segmentation result in Noise4, error=3.21%. NoDS=60 and Beta=15 for both cases.

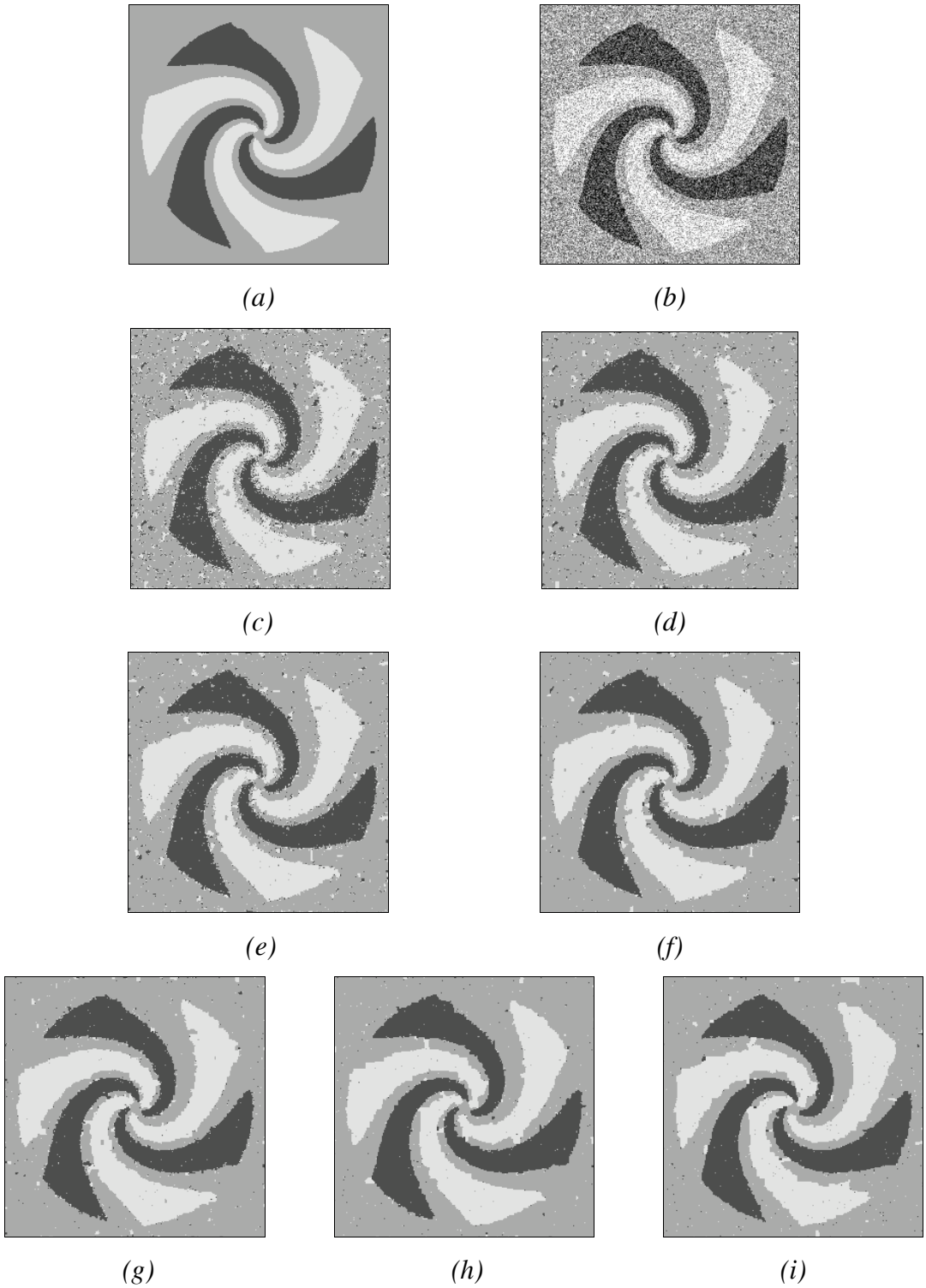


Figure 18.
Segmentation of "ArtI4" (a) Noise-free image ArtI4, (b) noisy input image PSNR=14.83dB (c) segmented image in model Noise0, segmentation error=12.96% (d) segmented image in model Noise1, segmentation error=8.78% (e) segmented image in model Noise2, segmentation error=6.31%, (f) segmented image in model Noise3, segmentation error=4.78%, (g) segmented image in model Noise4, segmentation error=4.07%, (h) segmented image in model Noise5, segmentation error=3.65%, (i) segmented image in model Noise5, segmentation error=4.35% NoDS=30 and Beta=68 in all cases.

4 CONCLUSIONS

In this chapter different MRF-based image segmentation models, implemented on parallel cellular arrays by simple functions realizable in VLSI, were discussed. Using an analog solution, like the CNN-UM architecture, the original MRF models had to be modified to meet the requirements of analog VLSI implementations. These modifications do not decrease the performance of the original methods significantly.

In the proposed models segmentation is based on an unsupervised labeling method where local statistics are estimated for image pixels and energy functions to be minimized are responsible to remain close to observations while achieving a homogenous image.

Multiscale implementations have been investigated to reduce the necessary neighborhood connectivity of the CNN-MRF model and to get similar segmentation performance like models with higher connectivity achieve. In general, multiscale models need more memory per cell but need smaller neighborhood connectivity than the monogrid model to achieve comparable results.

Robustness of the analogue architectures is an important question. With slight modifications to the monogrid CNN-MRF model it is shown that moderate noise can even increase the segmentation abilities, a few percent higher segmentation accuracy was achieved. This result comes from the behavior of the MMD relaxation method in the noisy environment. Its original pseudo-stochastic feature turns to be a real stochastic decision function due to computational noise.

CHAPTER II

Image Compression with Noisy 2D Processor Arrays

1 Introduction

The possible role of 2D processor arrays like CNN has already been showed for many image processing purposes. Image coding is one of the most important areas of image processing, while there are not many papers dealing with image compression and coding in the CNN environment [100,101].

What would be the advantage of 2D arrays in image compression against other existing hardware implementations? One would think that compression algorithms implemented on analog circuits have less accuracy, and the speed of existing digital codecs is already satisfactory to consumer needs. However, as Fig. 1 shows, there is always an exponentially increasing demand for new multimedia applications. In effect, the main advantage would be that the same hardware, one analog 2D array chip and some accompanying hardware elements, could solve several different tasks at high speed, such as capturing the image with its built-in sensor, filtering, analyzing and compressing. The larger number of tasks is implemented in the same hardware the cheaper can the architecture be on a large-scale market.

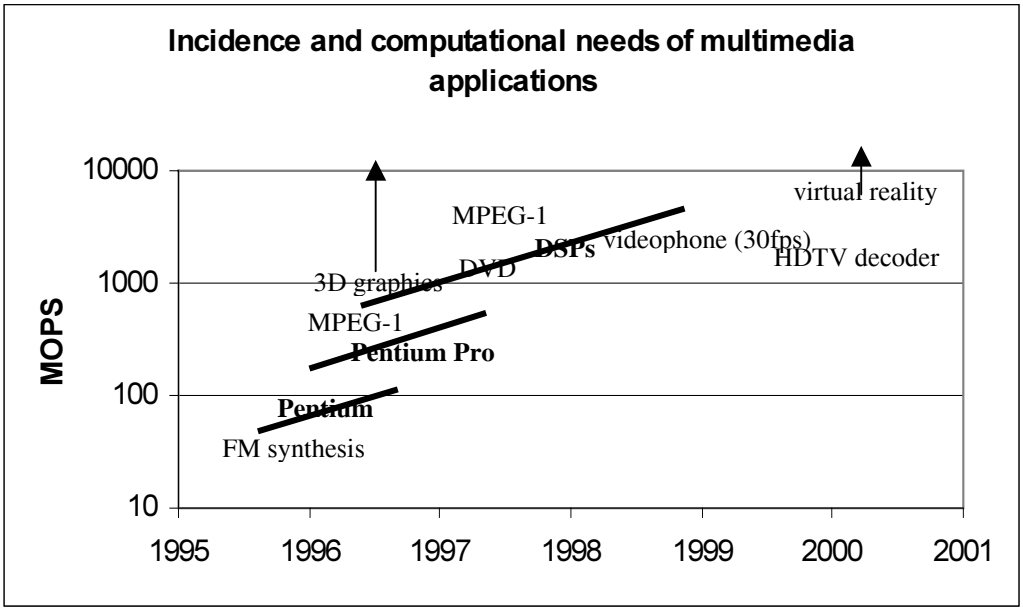


Figure 1.
Incidence and demands of some major multimedia applications [70].

In this chapter it is shown that there is an image coding model that is capable of coding grayscale images with the help of the CNN and some accompanying hardware elements (or generally speaking with 2D processor arrays) at an affordable performance. Simulations show that the chip set built of analog hardware of 8-bit precision gives satisfactory coding results at high speed also in the so-called Dynamic Coding environment. The described model is based on the orthogonal decomposition of image blocks and the simultaneous inverse operation for rate-distortion control.

First the coding model is given, then the necessary hardware components are listed and the architecture of the orthogonal analyzer is explained. In the proceeding section the processing cycle is outlined followed by a short analysis of computational complexity.

Since the analog feature of the chip-set may play a crucial role in the quality of the encoding, some simulation results and the effect of imprecise calculations are given in Section 2. In the last section of this chapter an algorithm for dynamic image coding is described to achieve optimal rate-distortion characteristics.

1.1 A Parallel Image Coding Model in the CNN Environment

The well-known idea behind orthogonal decomposition coding and compression is to represent (store, transmit) the image by its transform coefficients submitted to thresholding, quantization and entropy coding. In [94] it has been shown how the calculation of transform coefficients can be implemented in a system containing analog CNN chip and some accessories. Besides Discrete Cosine Transformation (DCT) we also use Hadamard Transformation (HT), which usually has worse compression results than that of DCT, but HT can be implemented much more easily in VLSI and is more robust against computational noise (if implemented in analog hardware). To get a more optimal code there is a need for the real-time computation of error-rate and bit-rate for all image blocks. As it was already mentioned the calculation of error-rate can be quite complex since in many cases conventional error measures are not satisfactory compared to the human visual perception. That is why the use of Human Visual System (HVS) like filters is proposed, fortunately some also feasible in the CNN environment [16].

As it will be shown through the next sections the proposed CNN-based image coding system has the following advantages:

- It processes pixels of an image block in parallel.
- Encoding and decoding are made simultaneously.

- Error-rate and bit-rate are computed coefficient-by-coefficient.
- The generation of coefficients for an image block can be stopped if required image quality is reached.

Since CNN can perform parallel pixel-level operations at high speed, we do not use the Fast Fourier Transformation (FFT) for the computation of the DCT (or HT) coefficients. Instead we obtain the coefficients by parallel multiplication and summation of image blocks with the basis functions. The complexity of the proposed method is $O(n)$ contrary to the $O(n\log n)$ complexity of FFT

.

1.2 The Basic CNN Chip-set

In [75] a chip-set capable of many image processing tasks has been defined, where the analog input sensors, digital signal processors, and control units are coupled to the CNN chip and analog memories. Main accessories of this set are as follows:

- CNN-UM chip (32∗32 or 64∗64).
- A-RAM (analog buffer memory).
- V-RAM (video memory).
- Digital microprocessors for control purposes.
- DSPs or special compression chips for data processing and entropy coding.
- Digital bus.
- Analog bus.

1.3 The Architecture of the Orthogonal Analyzer and Decoder

To accomplish the parallel encoding/decoding model the previous architecture should be extended with some additional functions and the necessary hardware equipment. In the new model the CNN chip itself represents an image block. Thus it should be capable of loading data parallel to columns and rows and it should be able to support parallel summation of the cell values for the calculation of coefficients.

In the coding architecture the CNN chip is to be surrounded with digital circuits for controlling the operation and for the evaluation of data (like in [92]). To be more specific, the chip and its accessories are specified in the following (see Fig. 2) according to the available tools of [75].

The proposed CNN chip contains:

1. Quick image loading line (active only once) (*ILL*).
2. Global data line for multiplication (*ML*).
3. Lines for loading data parallel to columns (*DC*). The same value is to be transmitted to a column (binary value in case of the HT).
4. Lines for loading data parallel to rows (*DR*). The same value is to be transmitted to a raw (binary value in case of the HT).
5. 5 local analog memories (LAMs) per cell (*M1-M5*).
6. Circuit for multiplication per cell (*mp*).
7. Circuit for addition per cell (*ad*).
8. Circuits for the absolute subtraction operation (*as*).
9. Circuits for the exclusive OR operation (*xor*) (only in case of HT). It is a so-called Local Logic Unit (LLU).
10. Circuit for the parallel summation of cell-outputs (*POS*).
11. Circuits for running a simple *3x3* CNN filter (template) (*filt*) for adopting the Human Visual System (HVS) expectation.
12. Fast D/A converter for the input image.

Further accessories required:

13. 2 analog memories for storing the row and column components of the basis functions of the DCT. In case of a CNN chip of size *32x32* both analog memories contain 32 lines each 32 wide. 2D basis functions are generated from the values of these two memories.
14. 2 digital memories (both with 32 entries of 32 bit wide) for storing the row and column components of the basis functions of the HT. 2D basis functions are generated from the values of these two memories.
15. CNN chip controller.
16. Fast A/D converter for the coefficients (*adcPOS*).
17. Fast code table for digital coefficients (*ctPOS*).
18. Entropy codec (*arPOS*).
19. Fast D/A converter for the generation of analog *POS*-code and transmission unit to *ML (anPOS)*. This part is necessary for decoding purposes.
20. CPU for the optimization of the Lagrangian cost function (*LCF*) considering error-rate and bit-rate [72].

Remarks:

1. Notations in the brackets are given for the easier matching of the chip components and functions of the processing cycle described in the next section.
2. The Lagrangian cost function mentioned in the above list is used to obtain an optimal code for the whole image. It is a function of the error-rate (ER) and the bit-rate (BR), which depend on the coding parameters (e.g. cut-off frequency[1], transformation method, etc.).
3. Digital code series (coming from ctPOS) contains redundant information. For this reason an entropy coder compresses the data into the final code that is transmitted to the receiver.
4. The HVS filter *(filt)* is an optional feature. It is analyzed in details in [16].

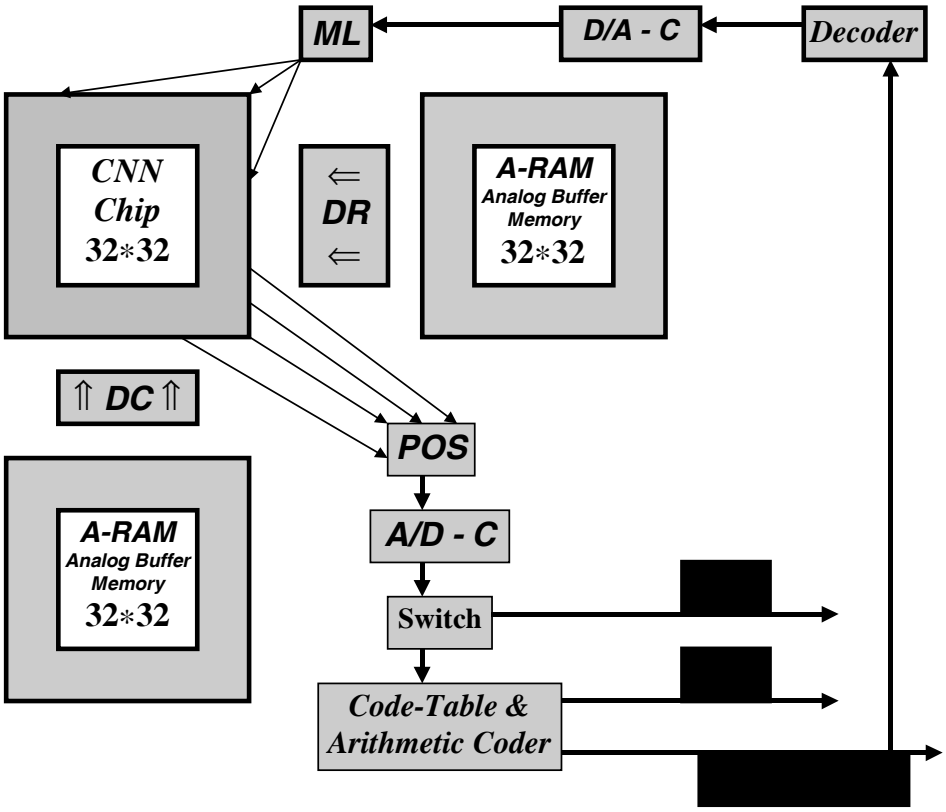


Figure 2.
The architecture for parallel transform encoding and decoding.

[1] Cut-off frequency: The frequency above which all coefficients are zeroed. It is given in discrete frequency or in percentage of the full frequency bandwidth and is often denoted with CF.

1.4 The Processing Cycle

Using the above notations we give the series of basic steps of the proposed CNN encoder/decoder. Before listing the individual operations the algorithm is summarized as follows:

Regardless of generating DCT or HT, the transformation function is stored by the transformation matrix in analog memory. Basis functions can be generated in the CNN chip by the multiplication (DCT) or the XOR logical function (HT) of the lines of the transformation matrix. Then these basis functions are multiplied with the input image block pixel by pixel and the results are summed up. The obtained transformation coefficient is digitized and can already be used for further coding, transmission or for the reconstruction of the input image by simply multiplying it with the basis function still in memory. This enables the system to continuously monitor the achieved error-rate and to stop the encoding process if a pre-defined image quality criterion is already reached. Fig. 3 illustrates the generation of coefficients and image reconstruction.

The individual steps of the algorithm according to previous notations are given:

1. Image block is loaded into the memory (LAM - *M1*) through the image loading line (*ILL).*
2. Decoded image of the previous step is erased from *M5.*
3. Pre-processing, image analysis, motion detection [54,72,87,89] are optional processes for the image in *M1.*

Making basis functions and calculation of the coefficients:

4. Loading transform matrix elements into the *DR.*
5. Loading transform matrix elements into the *DC.*
6. Calculating and storing the 2D basis function for every cell in LAM *M2* by
   - multiplication (*mp)* for the DCT
   - logical function (*xor)* for the HT
7. Multiplication *(mp)* of the image (*M1)* with the basis function (*M2)* into LAM *M3.*
8. The coefficient is calculated by adding up (*POS)* the elements of *M3.*

Coding the coefficient (CNN output):

9. Converting analog value to digital *(adcPOS).*
10. Possible look up table for efficient coding *(ctPOS).*
11. Entropy coding (*arPOS).*

12. Calculation of bit-rate.

Parallel decoding:

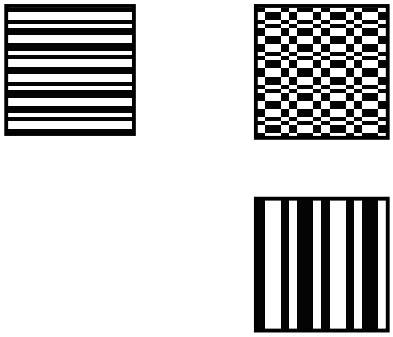
13. Analog coefficient (*anPOS)* is loaded to the global multiplicator line (*ML).*
14. Multiplication (*mp*) of coefficient (*ML*) with basis function (*M2*) into LAM *M4* (weighted basis function).
15. Adding up *(ad)* of weighted basis function *(M4)* into *M5* (continuously upgraded decoded image).
16. Extraction *(as)* of *M1* and *M5* into LAM *M4* (difference image).
17. *POS* sums analog error of *M4.*
18. Error-rate is obtained by A/D conversion of *POS (ER = adcPOS).*
19. The calculation of the Lagrangian cost function (*LCF).*

Evaluation:

20. Evaluation of cost function (*LCF)*
    - if required quality is reached then go to step 21
    - if not, go to step 4 in the cycle for the next coefficient
21. Coding is terminated and the next image block is to be processed.

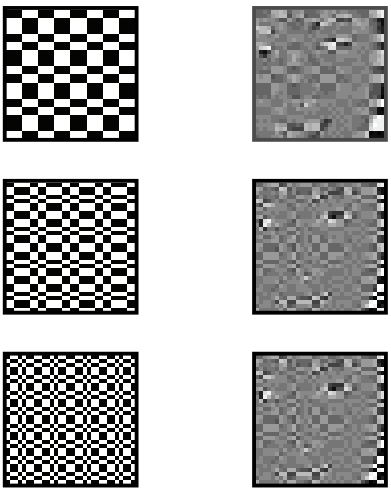
While usually more compact codes can be generated with the help of DCT than with HT, in this coding model the HT is easier to be implemented in VLSI since only binary data are driven through the *DC* and *DR* lines. Usually *DR* and *DC* are mounted on a CNN chip as memory lines, only a general switch (common line) is needed to control these lines to be a data not an address for a given cycle. Only 5 local memories are used in our encoding process. Although in Figure 2 only the transmitter is shown, it is easy to see that a very similar system can be used for receiving and decoding data in parallel as well. Data transfer and the control of the receiver and transmitter functions should be synchronized.

Generation of a 2D Hadamard basis function from the transformation matrix rows and columns by logical operations. Now, these operands are represented as 2D matrices but they can be stored as 1D rows and columns in analogue memories.



Input image block.

Making the product of the basis functions with the input image block. The summation of the elements of the product matrix gives the coefficient.



The subsequent decoded image series using the summation of the weighted basis functions.

85% 41% 15% 8%

On-the-fly measurement of error as the difference of the input and the reconstructed image block.

Figure 3.
The main processing steps of the orthogonal codec.

1.5 Computation Times

In the above algorithm for an image of size $N*N$ the number of parallel processing steps for the orthogonal transformation is $5*N*N$. In case of a conventional digital computer running Fast DCT [69] the number of serial arithmetic steps is about $8*N^2 * log_2N$. In case of a special FDCT processor this number is approximately $2*N^2 * log_2N$. If $N=8 \rightarrow 64$, the number of steps of the FDCT is in the same range as that of the CNN but the processing speed of CNN is still much better because of the following reasons:

- In our solution local memory processes are used (LAMs and local logic memories (LLMs)) instead of RAM and memory block transfers. For the parallel analog arithmetic/logic processes the time constant is approximately 10nsec.
- Inverse transformation is executed at the same time with the 2D basis functions retained in memory.
- In some cases it is not necessary to calculate all transformation coefficients. Since the encoding and the computation of the coding error are simultaneous processes the algorithm can stop the encoding process after the critical error-rate is achieved.
- Pre-processing methods may run in the same chip ensuring the local one-chip execution of the whole encoding process.

Table 1 and Table 2 shows the timing data. In our calculations we considered the following time constants (given for a recent CNN chip [21,57]):

- parallel loading of a line of analog values (LAM): 8nsec
- parallel read-out of a line of logic values (LLM): 10nsec
- serial readout of an analog pixel (LAM): 60nsec
- time constant for analog templates: 100nsec
- time constant for arithmetic steps: 10nsec
- LAM holding time: 1msec.

Timing results of Table 1 and Table 2 tell us that the encoding process can be executed at satisfactory speed with the help of only one CNN-UM chip. Considering the standard video-rate, a lot of pre-processing operations [16,54,75,89,90,92] and other different encoding mechanisms (e.g. DVC [72]) can be incorporated into this system. Using more CNN chips and sophisticated segmentation methods [54,87,89] the system can still operate at high speed. Depending on technological facilities it is also possible that large

chips (128*128 or larger) can measure the coefficients of several image blocks at the same time.

No. Step	***G**lobal **C**ycle **S**ingle*	*Function*	*Execution Time*	*Event/ Cycle Time*
1.	G,S	Image loading	32*8= 256 nsec	256nsec
2.	G,S	Clearing M5	1 nsec	1nsec
Coding process with orthogonal decomposition, calculation of bit-rate (BR)				
4-5.	G,C	Loading Horizontal and Vertical basis functions	16 nsec	
6.	C	Calculating the 2D basis functions	10nsec	
7.	C	Multiplication	10nsec	
8.	GC	Read-out of the sum of the cell-outputs: Coefficient	10nsec	47nsec
		(In case of row by row read-out)	(320nsec)	
Decoding from the coded coefficients using the same basis functions parallel with the coding, calculation of error-rate (ER)				
13.	GC	Loading single multiplicator	10nsec	
14.	C	Multiplying the basis func. by the coefficient	10nsec	
15.	C	Adding to the decoded image	10nsec	
16.	C	Difference between the input and the decoded image	10nsec	
17.	C	HVS-sensitive filtering of error image (e.g. halftoning)	150nsec	
18.	GC	Read-out of the sum of the cell-outputs: error-rate	10nsec	200nsec
		(without HVS filtering)		(50nsec)
Total time/cycle				***247nsec***
without HVS filtering				*(97nsec)*

Table 1
*Computation times for orthogonal decomposition in a CNN chip of size 32*32.*

Image size	Complexity	Time
32*32	Total processing time for 16*16 coefficients: 256*247+257nsec	63μsec
32*32	without HVS filter:256*97+257nsec	25μsec
32*32	Total processing time for 32*32 coefficients: 1024*247+257nsec	253μsec
	without HVS filter:1024*97+257nsec	100μsec
32*32	Total processing time for 24*24 coefficients, when decoding is evaluated rarely: 576*87+257nsec	***50μsec***
256*256	Processing an image using 1 chip with overlapping areas, ***100 sub-images***	
256*256	*Minimum*	***2.5msec***
256*256	***Optimum***	***5msec***
256*256	*Maximum*	***25msec***

Table 2
Total processing time for the decomposition/encoding process using a CNN-UM chip, without pre-processing.

2 Coding in a Noisy Computation Model

In this section I demonstrate that the presence of noise and structural inaccuracies of VLSI CNN chips influence the efficiency of our computational model, so it is important to take into consideration these parameters in the design of the algorithm.
In the proposed model we should consider that 1D components of 2D basis functions are converted through D/A converters into the analog buffer-memory, then the 2D functions are generated from the 1D components. In the decoding process the basis functions, weighted by the coefficients, are added up to get the decoded image. The large number of multiplication and addition of basis functions results in the accumulation of noise.
Now, in our experiments we encode all coefficients in 8 bits. In the following we simulated the CNN encoder architecture with two noise effects:

1. Absolute noise: the magnitude of additive noise is related to the maximal dynamic range of signals.
2. Relative noise: noise is related to the amplitude of the actual signal value.

These are approximations for the simulated circuits, and while they might be not exact models of VLSI circuits, they are only supposed to give the characteristics of the analog behavior of the encoding/decoding process.

In the simulations I used standard images (“Lena 256”, “Lena 512”, “Crowd”, “Peppers”, see Fig. 4)) and for the entropy coding of coefficients an easily available well-known entropy encoder GZIP V1.2.4, 1997, which is based on the LZ77 algorithm [33], was used.

Figure 4.
Test images “Lena 512” ”Crowd” and “Peppers”.

2.1 The Effect of A/D Conversion and the Accumulation of Coefficient Inaccuracy

Since all operations are to run on analog circuits, the generation of DCT basis images is also loaded with noise. Since 1D basis functions are stored in digital memories first they should be converted to analog values then 2D matrices are generated by the outer product of the 1D functions. The question is how the A/D conversion process effects precision. It is also interesting whether errors originating from this A/D conversion accumulate as more and more coefficients are added up in the image reconstruction process. As Fig. 5 shows there is only a slight effect of A/D data conversion, the error measured in the decoded images is approximately 0.1 dB at 7bits.

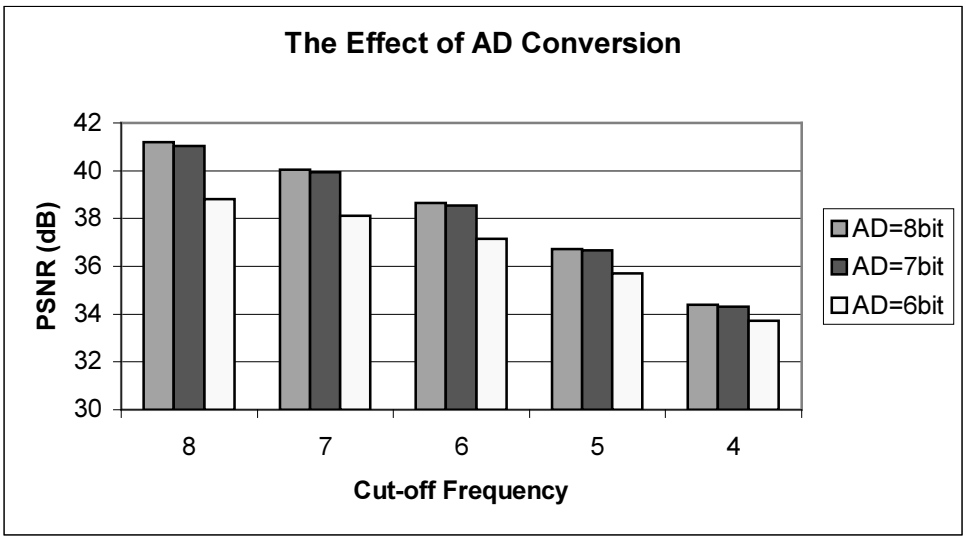


Figure 5.
*PSNR between original and decoded image (blocks are of size 8*8 of "Lena 512") at different cut-off frequency and at different A/D converter accuracy (AD).*

In case of HT we should not deal with this kind of error, since the basis functions are binary. Their analog values can be saturated and even in the calculation of coefficients they are not being multiplied with image pixels, they are just used as binary masks for image pixel read-out and summation in the POS (parallel output summation) unit.

Now, let us examine the effect of inaccurate coefficient computation. Fig. 6 illustrates the DCT encoding/decoding error of "Lena 512" at different noise parameters and different cut-off frequencies. Above the given discrete frequency all coefficients were neglected. Noise models tested are given in the following table (Table 3).

Noise Model	Absolute Noise (Variation)	Relative Noise (Variation)
Noise0	0.0	0.0
Noise1	0.0058	0.012
Noise2	0.0029	0.0058
Noise3	0.0012	0.0029

Table 3
Noise models of zero expected value and different variance.

Absolute noise was added to every addition while every multiplication was loaded with relative noise.

In Fig. 6 it can be seen that according to the different noise levels the compression behaves differently at the different cut-off frequencies. At low noise (Noise0 and Noise3) the loss of higher frequency components (cut-off frequency decreases) increases coding error (PSNR decreases) while at higher noise levels (Noise1, Noise2) the loss of high frequency decreases coding error (PSNR increases). It is important to note that the decrease of performance is much more significant than in case of imprecise AD conversion.

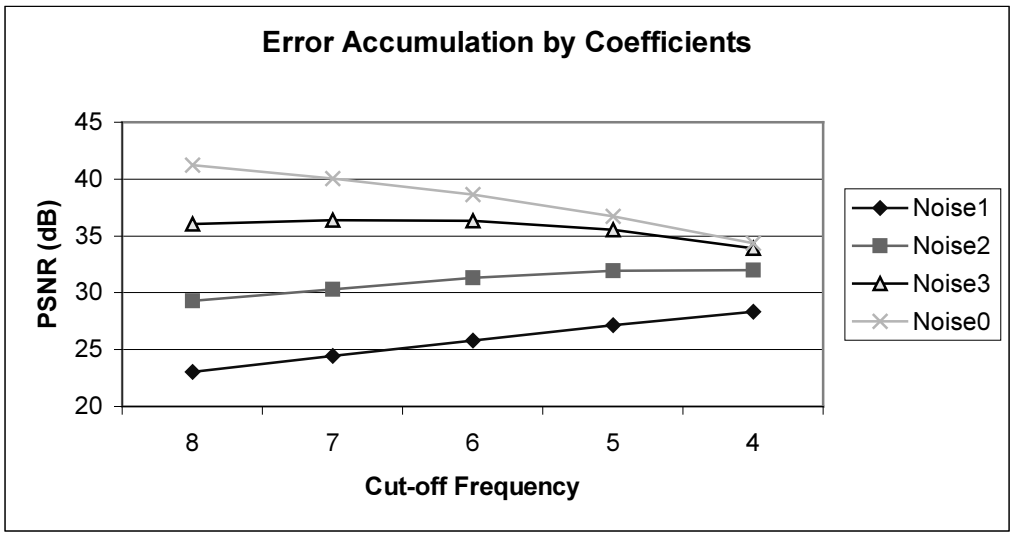


Figure 6.
*PSNR between original and DCT decoded images (blocks are of size 8*8, of "Lena 512") at different cut-off frequency and at different Noise parameters.*

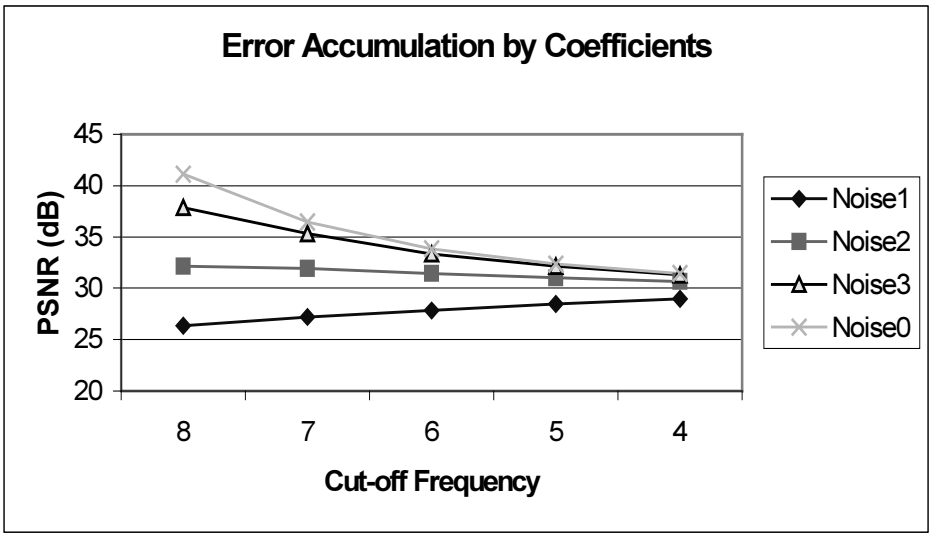


Figure 7.
*PSNR between original and HT decoded images (blocks are of size 8*8, of "Lena 512") at different cut-off frequency and at different Noise parameters.*

Fig. 7 illustrates the same experiments, but for the Hadamard Transformation. As it is expected in case of low noise (or noise free computation) the HT is less effective (curves of Noise0 and Noise3), however HT is also less sensitive for noise, i.e. graphs have much less slope in case of Noise1 and Noise2.

In the above examples all of the coefficients below the cut-off frequency were taken into account in the reconstruction/decoding process irrespectively of their value. However, in our case, and generally in all bit allocation problems, a large amount of redundancy can be reduced with the proper coding (e.g. discarding) of coefficients close to zero. Naturally we should consider the noise level to find an optimal threshold for coefficient zeroing.

2.2 Thresholding Coefficients

If we automatically take all of the coefficients into consideration in the decoding process then we accumulate a high amount of noise, since many of the coefficients are zero or below the noise level. For this reason now we take into account only those coefficients that are above the noise level. Only the reasonable values are transmitted through the system significantly decreasing the code length and coding error.
In the following I simulated the CNN architecture with the second noise model (Noise2) given above, that is:

- The accuracy of D/A and A/D converters is 8 bit.
- The variation for absolute noise is 0.0029 and for relative noise is 0.0058.
- Cut-off limit for thresholding coefficients is given by ε.

Noise parameters of the above model can be considered as achievable VLSI parameters of analog chips of good quality [21].
The coding algorithm is the following:

1. The input test images are divided into disjoint image blocks.
2. Each block is transformed with the appropriate encoder, analog computations are simulated with noise model Noise2.
3. Coefficients below cut-off frequency are read out in a zigzag order, similarly to that of implemented in the JPEG standard [105].
4. Values below noise are thresholded and substituted by zero.
5. Zero coefficients are run-length encoded.
6. All coefficients are written to a data stream and encoded with GZIP.

Here we should note that considering the architecture of recent CNN chips and our coding model the optimum block size (depending also on noise parameters) may differ from the size of the chip itself. In case of image blocks of size 16∗16 a CNN chip of size 32*32 is divided into 4 equal parts. It means insignificant modifications in the specification of the model, like changing the summation output (*POS*) to give the 4 coefficients of the 4 different areas.
The next diagrams (Fig. 8 - Fig. 11) show the reconstruction error (given in PSNR) and the compression ratio (in bit/pixel) versus threshold (ε), achieved with the help of the mentioned GZIP entropy encoder run on the 8bit DCT coefficients.

As we have seen in a previous example (Fig. 6) if we increase the threshold, the value of PSNR is not decreasing unambiguously, at a certain threshold level it even increases while the compression ratio increases monotonously.

However, it is not easy to find the right threshold value where PSNR reaches its top, moreover it does not seem to be an easy task to find an optimum coding if the set of possible solutions includes different cut-off frequencies, thresholds, and block sizes.

Yet, the things that are important for us are the slopes of the diagrams. If we compare the diagrams of the full-length coefficient results and the truncated results, we observe that the full-length data varies much more according to the threshold values. They have higher dynamics through the threshold parameter, still their peak PSNR and compression values are not as much better (if they are better at all). So it seems that the truncated codes are much more tolerant for the different thresholds. An important question might be what cut-off frequency to choose at a given noise parameter and what is the proper block size. This way we might simplify the search for a more efficient code, however, in some cases we would not reach the optimal result.

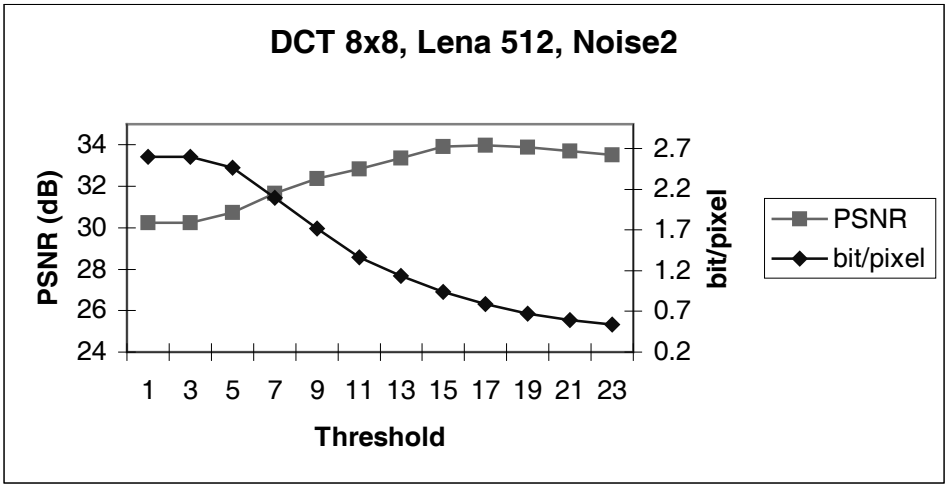


Figure 8.
PSNR and Compression Ratio for "Lena 512", DCT.

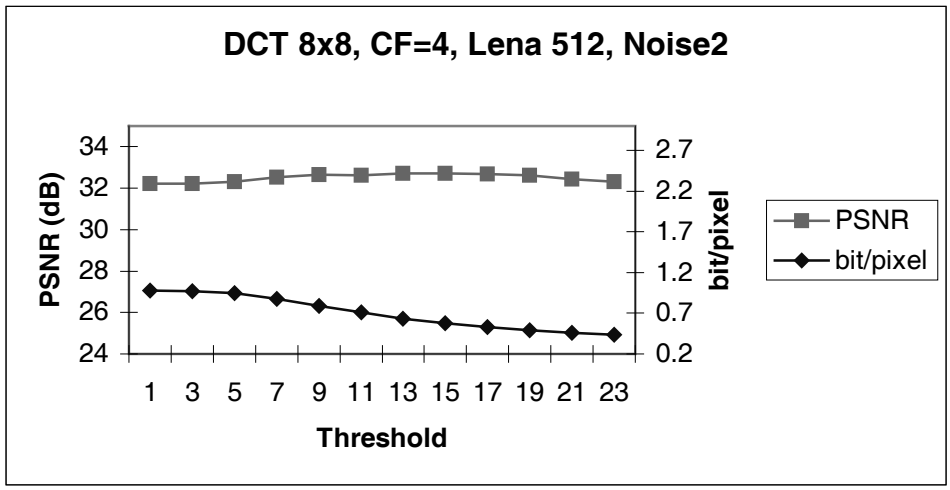


Figure 9. PSNR and Compression Ratio for "Lena 512", DCT, cut-off requency = 4.

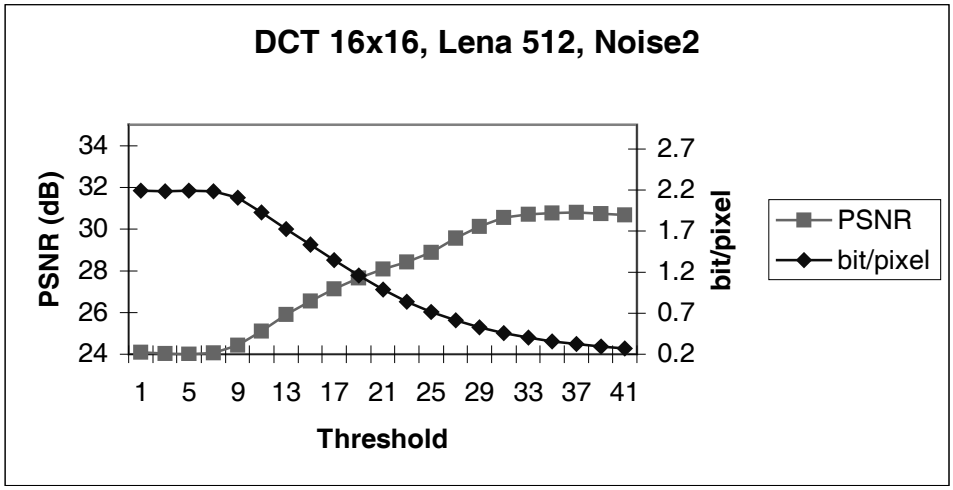


Figure 10. PSNR and Compression Ratio for "Lena 512", block size 16x16, DCT.

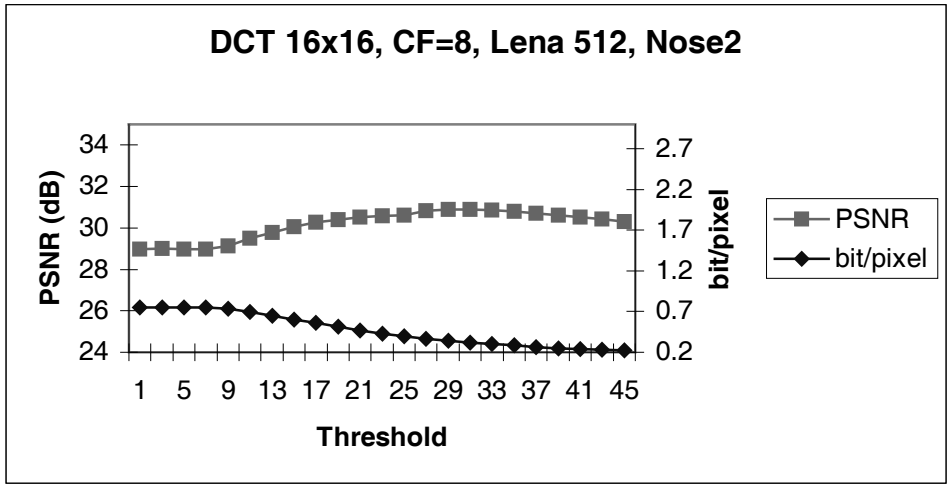


Figure 11. PSNR and Compression Ratio for "Lena 512", block size 16x16, DCT, cut-off frequency=8.

We should also investigate the effect of noise and thresholding in case of the Hadamard Transformation. As it was already mentioned this transformation is less affected by

noise due to its binary feature, that is why increasing the threshold level does not lead to so powerful PSNR increase. See Fig. 12 and Fig. 13 for some data examples.

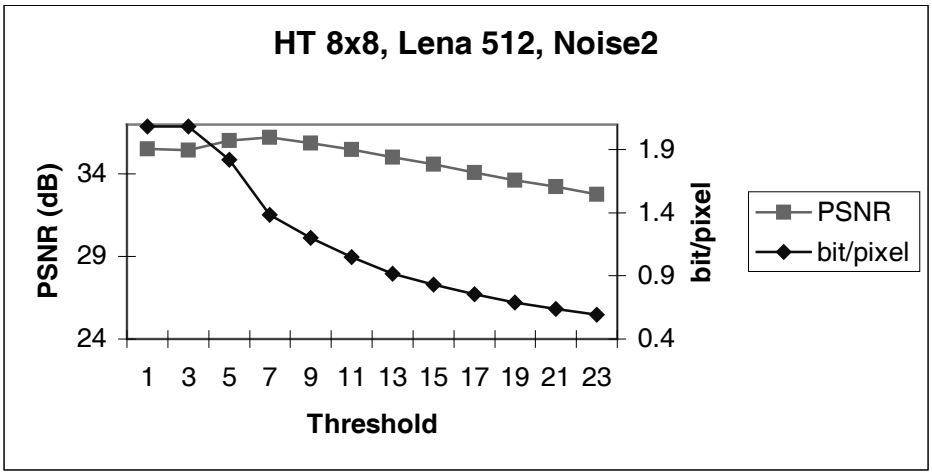


Figure 12.
PSNR and Compression Ratio for "Lena 512", block size 8x8,HT.

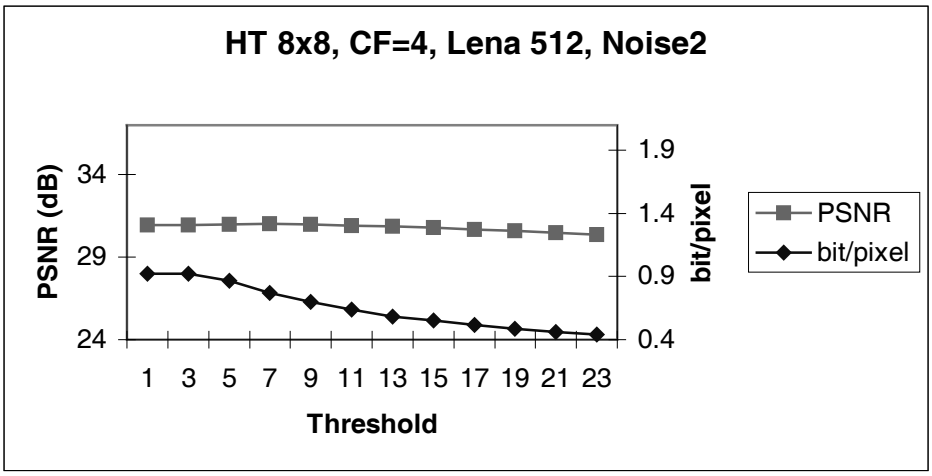


Figure 13.
PSNR and Compression Ratio for "Lena 512", block size 8x8,HT, cut-off frequency is 4.

2.3 Parameters for the Compression of an Image Block

As we have seen there are a lot of parameters to be optimized in case of the analog coding architecture. They are the following:

1. The size of the basic image blocks *(BS)*. It can be smaller than the size of the CNN chip.
2. Cut-off frequency (CF) of the DCT or HT coefficients *(above the value of CF all coefficients are substituted with zero).*
3. Threshold level for coefficients (ε).
4. Type of orthogonal transformation: DCT or HT. In case noiseless coding DCT usually have higher energy compression features. On the other hand, as we have seen, HT is more robust against error.

Now, in this work I do not deal with the optimization of the quantization levels. Our aim is to show the effect of analog noise and to investigate if the analog CNN-based encoder/decoder is capable of efficient coding of still images. However, the proposed method could include various quantization levels, now 8 bits represent all coefficients.

3 Optimizing the Compression in a Dynamic Coding Environment

There are a lot of approaches to achieve high compression ratio by choosing an optimal bit allocation strategy [15]. Generally speaking, fixed allocation algorithms do not perform very well compared to adaptive procedures. At the same time, the problem of adaptive codecs is the overhead introduced by the transmission of additional information necessary to describe the parameters of the adaptive bit allocation, such as bit allocation tables, block classification, and normalization factors. In the following sections we will propose an optimization based on the variable size of blocks, variable thresholding, cut-off frequency and the type of transformation.

3.1 Dynamic Image Coding with Lagrange Optimization

Dynamic Video Coding (DVC) [72,22] is a new method for effective image compression. It is based on the assumption that by coding the different image parts the most appropriate way - according to the local image structure - we may get a more efficient and compact code of a video sequence or a single image frame. To achieve this, different coding methods are executed for each small basic image block simultaneously then the optimum representation is chosen from all possible candidates as a final code representation of the image.

Many of the optimization methods are established on the framework of Shannon's rate-distortion theory [83]. Image transmission problems can usually be formalized in one of the following two ways:

1. $$C_{Optimal} = \min_{C_i}\{E(C_i) \mid C_i \in C \text{ and } R(C_i) \le BR_{Max}\} \quad (1)$$

2. $$C_{Optimal} = \min_{C_i}\{R(C_i) \mid C_i \in C \text{ and } E(C_i) \le ER_{Max}\} \quad (2)$$

where C_i is one of all possible encodings ($C_i \in \mathbf{C}$), $E(C_i)$ is the error-rate, $R(C_i)$ is the bit-rate belonging to a given code C_i. ER_{Max} is the maximum allowed coding error, while BR_{Max} is the admissible bit-rate, and one of them is a constraint in a possible image coding problem. We want to find the solution for the constrained problems of Eq. (1) or Eq. (2) and an optimal solution should be given for the whole image frame or even for several frames of a video sequence. To avoid extreme computational loads it is

advised to restrict the model to intra frame coding where no redundancy between frames are reduced, only the image redundancy in frames are processed.

The algorithmic solution for the above problems is given by converting the constraint problems to unconstrained problems as proposed in [25].

Equation (1) can be transformed into the minimization of the following cost function with different values of λ:

$$Cost(\lambda, C_i) = E(C_i) + \lambda R(C_i), \tag{3}$$

where λ is called the Lagrange multiplier. By sweeping λ over a positive interval and minimizing the cost function for each λ we get different values for *E* and *R* denoted by E_λ^* and R_λ^*. As λ increases we get smaller values for R_λ^* and larger values for E_λ^*, in particular the previous value is monotone decreasing while the later is monotone increasing. If we reach a predefined value with R_λ^* then the C_i that minimized $Cost(\lambda, C_i)$ is the solution for Eq. (1). Eq. (2) can be similarly solved by transforming the problem to equation:

$$Cost(\lambda, C_i) = R(C_i) + \lambda E(C_i). \tag{4}$$

These algorithms do not guarantee solutions for the constraint problems but if there is a solution they are able to find it at a certain precision.

Now, what follows is to give the set of admissible solutions (*C*) in the framework of our analog coding architecture.

3.2 Optimal Image Coding with Lagrange Optimization in Quadtree Representation

So the task is to define an admissible set of solutions within the proposed architecture, to give the constraint problem, to convert this problem to an unconstrained problem, then to define the encoding model, the image representation, and the bit allocation.

As it comes from a previous section the set of admissible solutions consists of the two type of transformation DCT and HT, different block sizes and several cut-off frequency settings. Assuming that we have a given noise model, the thresholds for coefficients are fixed. Generally, HT has smaller threshold than DCT and larger blocks also owe larger thresholding than blocks of smaller size.

The proposed method can be adequate for solving both Eq. (1) and Eq. (2), however if BR_{Max} is given as a constraint (Eq. (1)) then the image quality can be highly varying

according to the content, which can be very annoying in many situations. Therefore, in this work we deal with Eq. (2), that is we are looking for the most compact coding of the image above a given quality criterion.

Our following assumption states that the minimization of the function $Cost(\lambda, C_i) = R(C_i) + \lambda E(C_i)$ for the whole image can be solved by cutting the image into different blocks. Namely, our energy function can be decomposed:

$$Cost(\lambda, C_i) = R(C_i) + \lambda E(C_i) = \sum_{n=1}^{N} \left(R(C_{i,n}) + \lambda E(C_{i,n}) \right) \tag{5}$$

where *n* stands for indexing the different image partitions (they can even be overlapping image blocks). Naturally, it is not necessarily true in many situations, since the assumed equation does not take into consideration the redundancy in the code of groups of separate image partitions. For example if $R(C_i)$ is measured by the product of C_i's entropy and the number of code words in C_i, then the sum of $R(C_{i.n1})$ and $R(C_{i.n2})$ is larger or equal than $R(C_{i.n1,n2})$, measured on the larger block made be the merge of the two separate partitions. However, in the examples we will see that the sum of the entropy-based measure of several image blocks highly correlates with the code length necessary for the encoding of the whole image.

As to specify the elements of the cost function by entropy:

$$R(C_i) = -\left|C_i\right| \cdot \sum_{k=0}^{255} p_k \log_2 p_k \tag{6}$$

where C_i is a transform coding (DCT or HT), $\left|C_i\right|$ is the number of code words, and p_k is the probability of 8-bit coded coefficients.

$$E(C_i) = \sum_{k=1}^{MM} \left(b_{k,C_i} - \hat{b}_{k,C_i} \right)^2 \tag{7}$$

where b_{kCi} are the pixel values belonging to the block of size *MxM* coded by C_i, and $\hat{b}$-s are the original image values.

To generate different block partitionings of the image frame we use a quadtree representation, where there is a maximum and minimum block resolution, typically ranging from 32x32 to 8x8 pixels. This way an image block can be represented in 16 possible configurations, namely (see Fig. 14):

1. 1 parent (1 configuration)
2. 1 parent and 1 child (4 configurations)

3. 1 parent and 2 children (6 configurations)
4. 1 parent and 3 children (4 configurations)
5. 4 children (1 configuration).

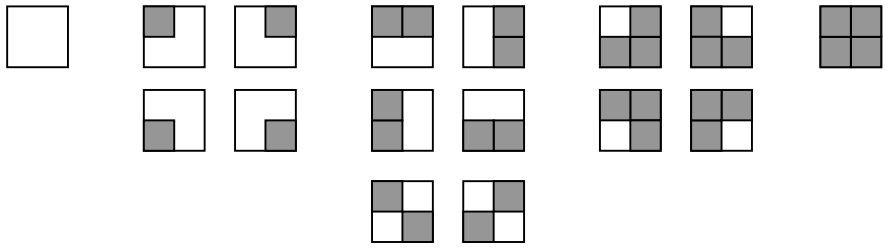


Figure 14. The different possible configurations of parent and child nodes. Children are dark gray and parents are white.

Each node of the quadtree can have codes for all admissible solutions. The algorithm for finding the most optimal code and generating the compressed bit series is the following:

Code Generation:

1. The quadtree for representing the input image is built.
2. Each node of the tree is transformed with DCT and HT. Considering two different values for cut-off frequency (defined e.g. at 30% and 70% of the maximum frequency) we get four codes for each node.
3. Coefficients are thresholded then quantized to 8 bit resolution.
4. $R(C_i)$ and $E(C_i)$ is computed for each solution.

Optimization:

5. Initial value for λ is selected.
6. Each node of the tree is visited once and the most efficient code of the four for the current node is selected by computing $Cost(\lambda, C_{i,j}) = R(C_{i,j}) + \lambda E(C_{i,j})$.
7. Starting a recursion at the parents of the leaves the best solution is selected from the 16 configuration listed above. At the end of the recursion we have the most optimal code for a given λ.
8. If $E(C_i)$, computed for the whole image, is below a required critical value, we can stop the optimization and go to step 10.

9. Increase λ and step to 6.

Generation of final code:

10. Bytes from the selected optimal codes of image nodes are read out in a zigzag order.
11. Zeros are run-length encoded.
12. Coefficients and the type of optimal codes at each quadtree node are written to a data stream and encoded with GZIP. If we have 4 kinds of possible codes for each node, it means 2bit information/node to encode.

Here we should note that according to Parseval's theorem the encoding error could also be measured in the transformed domain. However, in our case the decoding algorithm is also based on inaccurate analog circuits. That is why it is necessary to compute the coding error by the inverse transformation on the proposed architecture.

3.3 Analysis of Code Efficiency

Here I examine the basic properties of the proposed algorithm through some experimental data. In all measurements two DCT and two HT methods were available for the image blocks at different cut-off frequency and threshold settings.

First the usage frequency of the two transformations (DCT and HT) in case of noisy (model Noise2) and noiseless (model Noise0) simulations are compared.

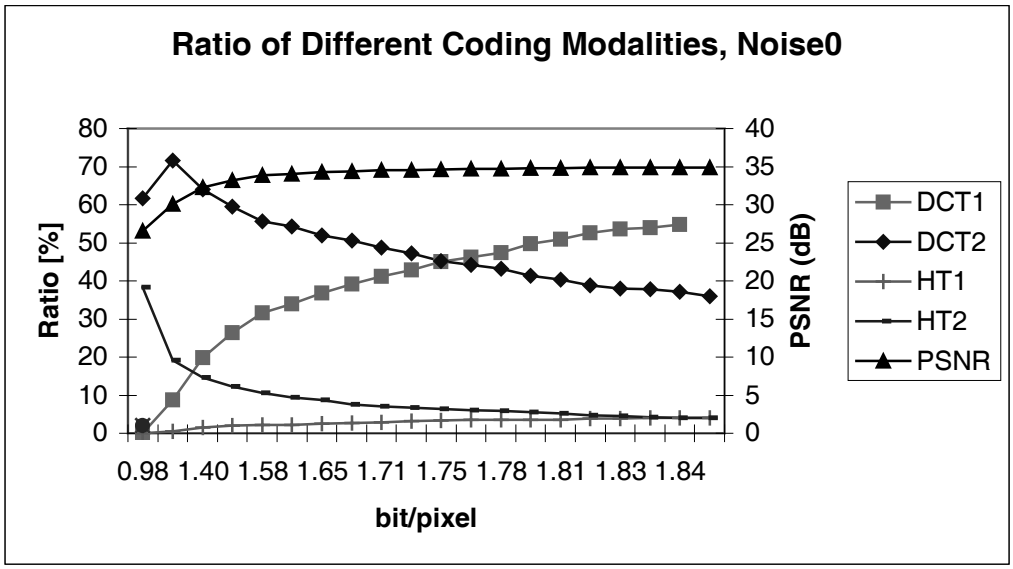


Figure 15. The ratio of the different coding modalities and the measured PSNR values for different compression ratios for "Lena 256", block size is 8x8, no circuit noise is present.

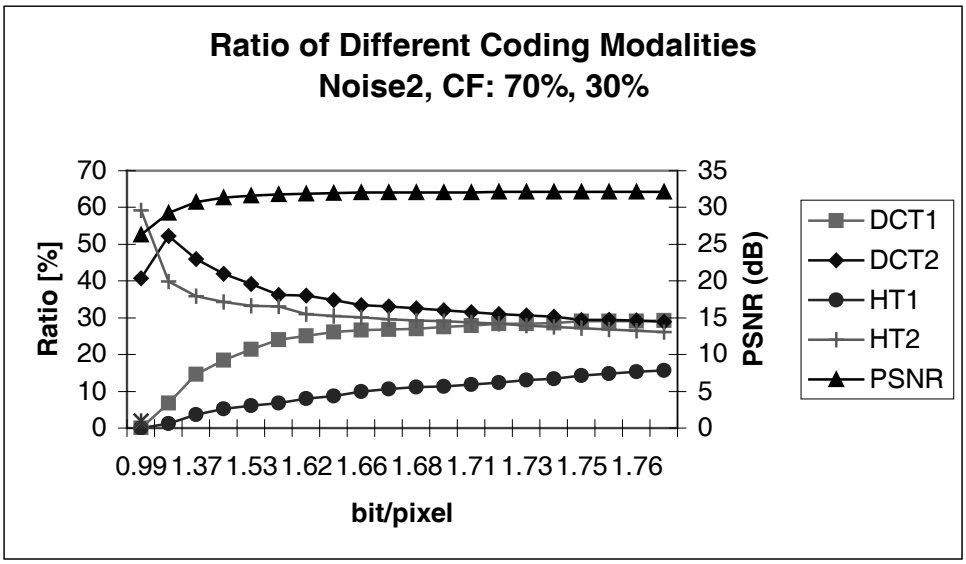


Figure 16. The ratio of the different coding modalities and the measured PSNR values for different compression ratios for "Lena 256", block size is 8x8, Noise2, cut-off frequencies were set at 30 and 70 percent of 8 (maximum frequency were 2 and 5 in both directions).

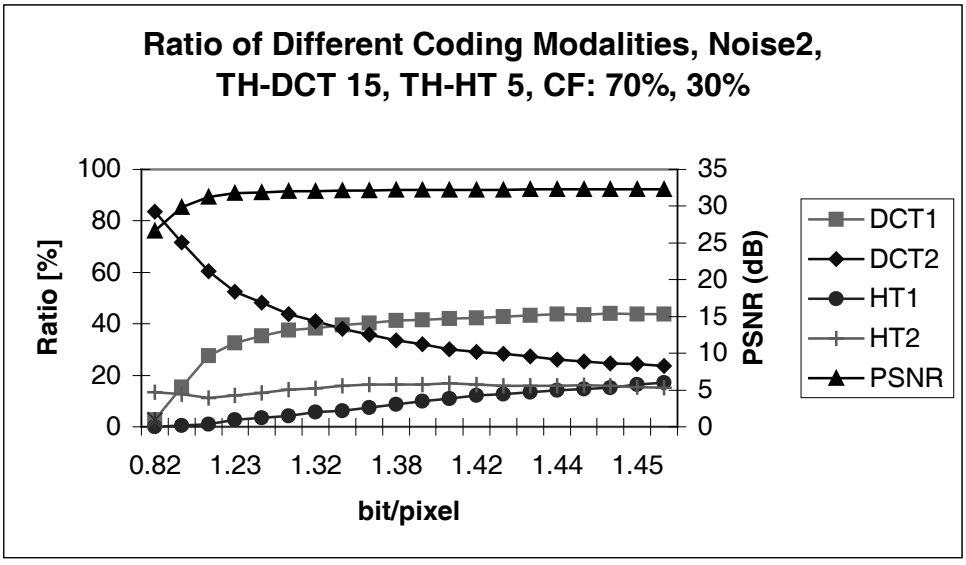
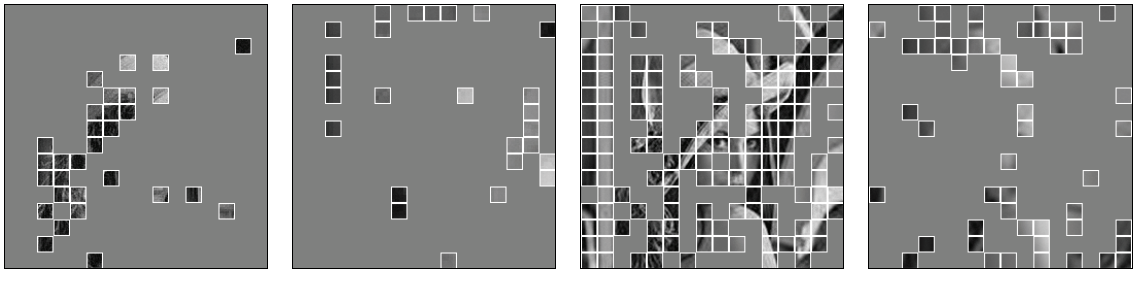


Figure 17. The ratio of the different coding modalities and the measured PSNR values for different compression ratios for "Lena 256", block size is 8x8, Noise2, threshold for DCT and HT is 15 and 5 respectively. Cut-off frequencies were set at 30 and 70 percent of 8 (maximum frequency were 2 and 5 in both directions).



(a) HT1 11% *(b) HT2 11%* *(c) DCT1 57%* *(d) DCT2 20%*

(e)

Figure 18.
The result of the optimization for blocks of size 16x16, "Lena 256". Decoded image at 0.6 bit/pixel and 28.7 dB, CF: 70%, 30%, noise model: Noise2. The blocks of the different transformation methods are also drawn.

As Fig. 15-17 show, in case of noiseless simulations the ration of HT transformed blocks in the optimal code are usually smaller than in the case of noisy circuits. It may justify the robustness of HT against noise as discussed before.

It is important to consider the overhead information originating from the code necessary to encode the structure of the optimal quadtree. The larger range is allowed for the selection of the optimal block size the larger the overhead might be. Typically if we want to achieve higher quality, smaller block partitions are required, while at lower quality encoding the number of small blocks is smaller. Unfortunately, the Lagrange optimization does not take into account the amount of data for the encoding of the structure of the optimal quadtree. The following graphs (see Fig. 19 - Fig. 21 and the corresponding table Table 4) illustrate the ratio of the overall entropy based code-length measure ($R^*(C_i)$) and the real size of the compressed data, both given in bytes. It can be seen that while the effective length is always much above the measure applied in the optimization, there is a high correlation between the two data, the effective code length

increases monotonically as the entropy grows. The difference originates from the overhead of the partitioning information, the overhead of the GZIP compression and the sub-optimal entropy coding. The first reason can be justified by the fact, that in Fig. 21 (the variable size encoder) the difference between the two data functions is much more significant, that is more code is allocated for coding the structure of the optimal quadtree.

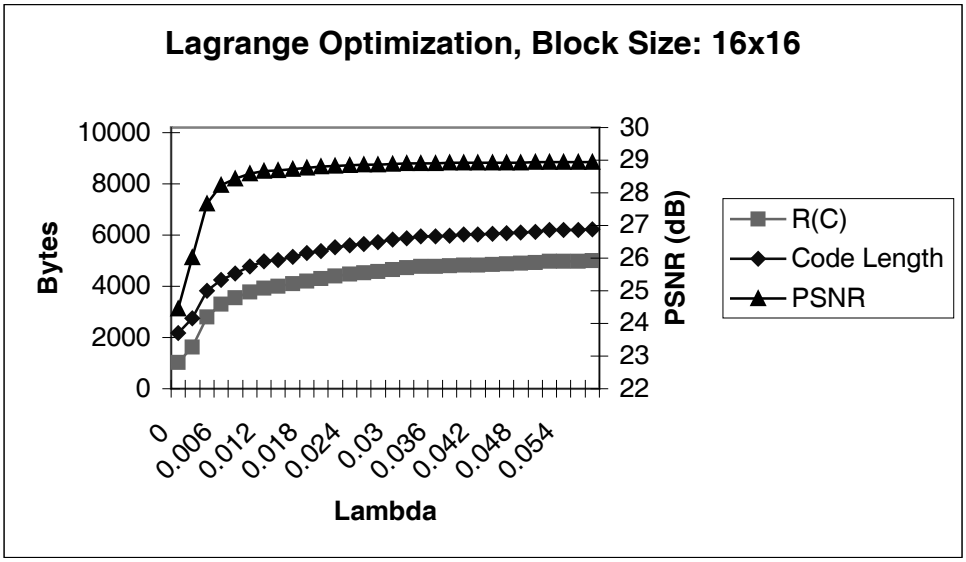


Figure 19.
Lagrange optimization of "Lena 256" in a noisy model at fixed block size (16x16). PSNR is given on the right side while code length and estimated code length is given on the left side in bytes.

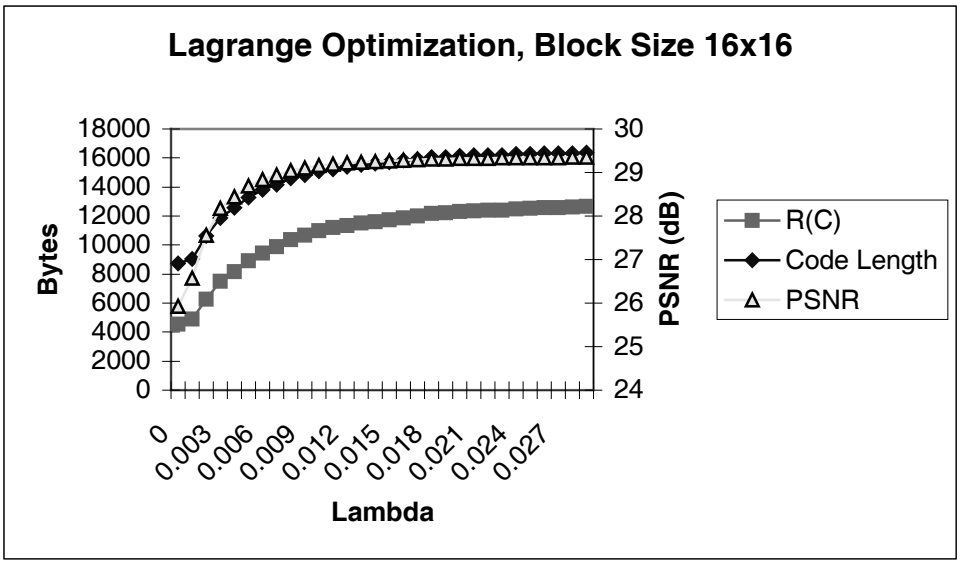


Figure 20.
Lagrange optimization of "Crowd" in a noisy model at fixed block size (16x16). PSNR is given on the right side while code length and estimated code length is given on the left side in bytes.

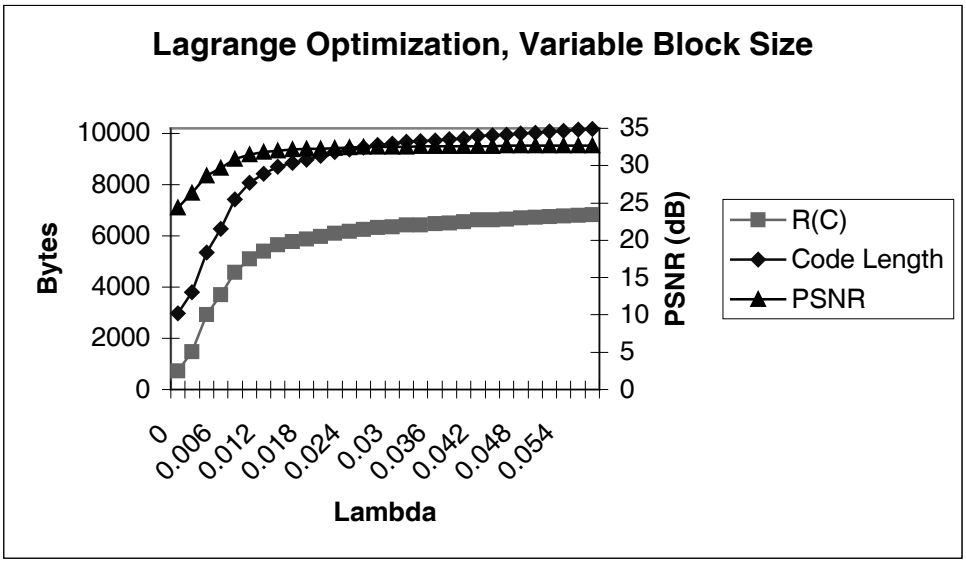


Figure 21.
Lagrange optimization of "Lena 256" in a noisy model at variable block size (8x8 and 16x16). PSNR is given on the right side while code length and estimated code length is given on the left side in bytes.

The problem that still remains is how to select the range of possible block sizes. Generally, we can state that if we want to reach high quality it is worth allowing the use of small blocks, while if the necessary image quality is low, it is not worth using small blocks in the optimization process.

Let us examine the optimal codes with the help of Table 4 when PSNR is cc. 26dB. When variable sizes are allowed we expect shorter optimal codes than in the fixed size case, since the set of possible codes (C) is much larger. It is justified by the values of $R^*(C_i)$ in the second line of the table below (at λ=0.002). However, the final code lengths are better for the fixed size algorithm. Fig. 22 and Fig. 23 also illustrate this effect, where fixed size algorithms outperform their competitors usually in the high compression regions.

The consequence of these observations is that to get the most optimal code we should run the Lagrange optimization more than once with different block range settings, then we can avoid sub-optimal code due to the effect of overhead information.

	Fixed block size (16x16)			Variable block size (8x8, 16x16)		
λ	PSNR	R*(C)	Length(C)	PSNR	R*(C)	Length(C)
0	24.452	1032.3	2169	24.328	737.23	2972
0.002	**26.021**	**1613.2**	**2740**	**26.338**	**1477.5**	**3809**
0.004	27.676	2792.8	3818	28.663	2917.1	5353
0.006	28.232	3309.2	4255	29.707	3693	6273
0.008	28.435	3558.3	4502	30.845	4585	7433
0.01	28.581	3781.5	4780	31.479	5110.5	8084
0.012	28.664	3937.3	4984	31.804	5406.2	8437
0.014	28.688	3989.9	5032	32.039	5649.5	8690
0.016	28.731	4096	5160	32.142	5770.4	8851
0.018	28.771	4210.7	5295	32.228	5876.8	8981
0.02	28.796	4288.8	5386	32.297	5973.4	9115
0.022	28.829	4402.7	5521	32.38	6099.3	9275
0.024	28.846	4468.3	5601	32.431	6182.8	9372
0.026	28.86	4524.8	5655	32.47	6252.1	9451
0.028	28.872	4577.4	5721	32.502	6312.8	9544
0.03	28.888	4656.7	5815	32.524	6355.8	9597
0.032	28.899	4714	5884	32.553	6417	9670
0.034	28.91	4772	5959	32.559	6433.7	9693
0.036	28.91	4772	5959	32.576	6471.1	9725
0.038	28.914	4797.1	5983	32.591	6509.1	9763
0.04	28.919	4828	6017	32.604	6544.5	9796
0.042	28.919	4830.7	6019	32.629	6612.7	9890
0.044	28.922	4852.3	6042	32.636	6633	9914
0.046	28.926	4877.8	6073	32.646	6659.8	9945
0.048	28.93	4904.4	6105	32.66	6708.5	10006
0.05	28.933	4930.4	6137	32.667	6729.4	10024
0.052	28.94	4985.9	6194	32.673	6752.2	10064
0.054	28.94	4985.9	6194	32.68	6773.7	10094
0.056	28.94	4985.9	6194	32.689	6804.9	10141
0.058	28.942	5005.2	6216	32.696	6829.2	10174

Table 4

Lagrange optimization of "Lena 256" in a noisy model at fixed (16x16) and variable block size (8x8 and 16x16). See Fig. 19 and Fig. 21 for graphical representations.

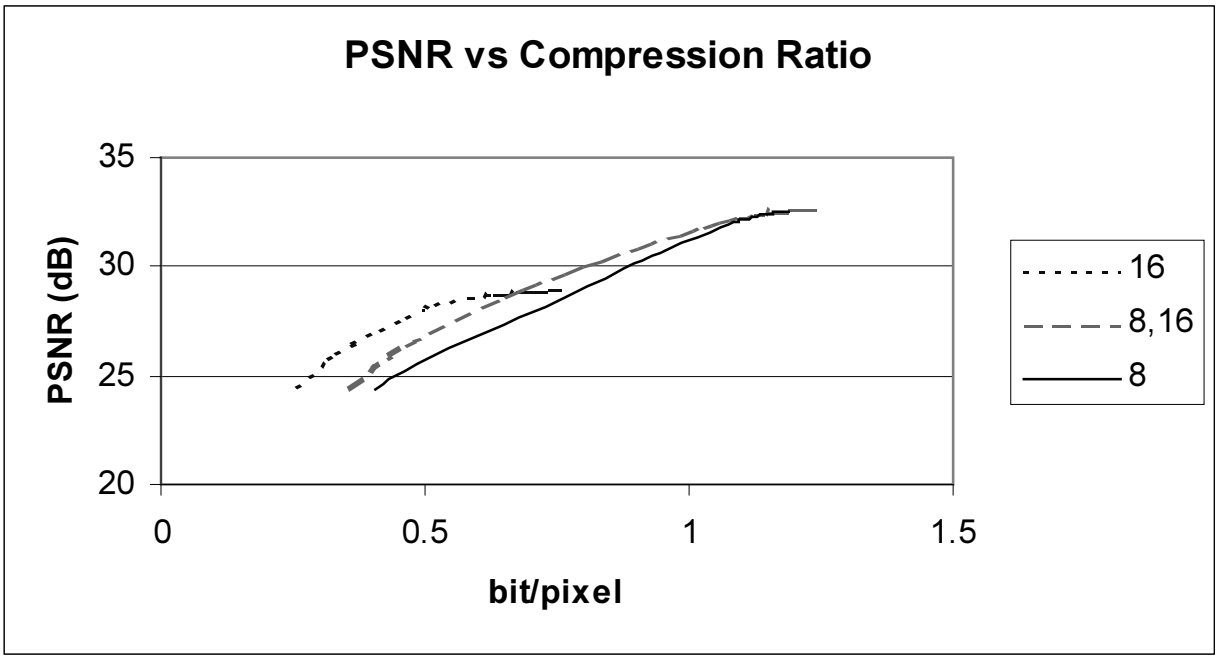


Figure 22.
PSNR (dB) vs. Compression Ratio (bit/pixel) at different possible block sizes for "Lena 256" in a noisy coding model (Noise2)

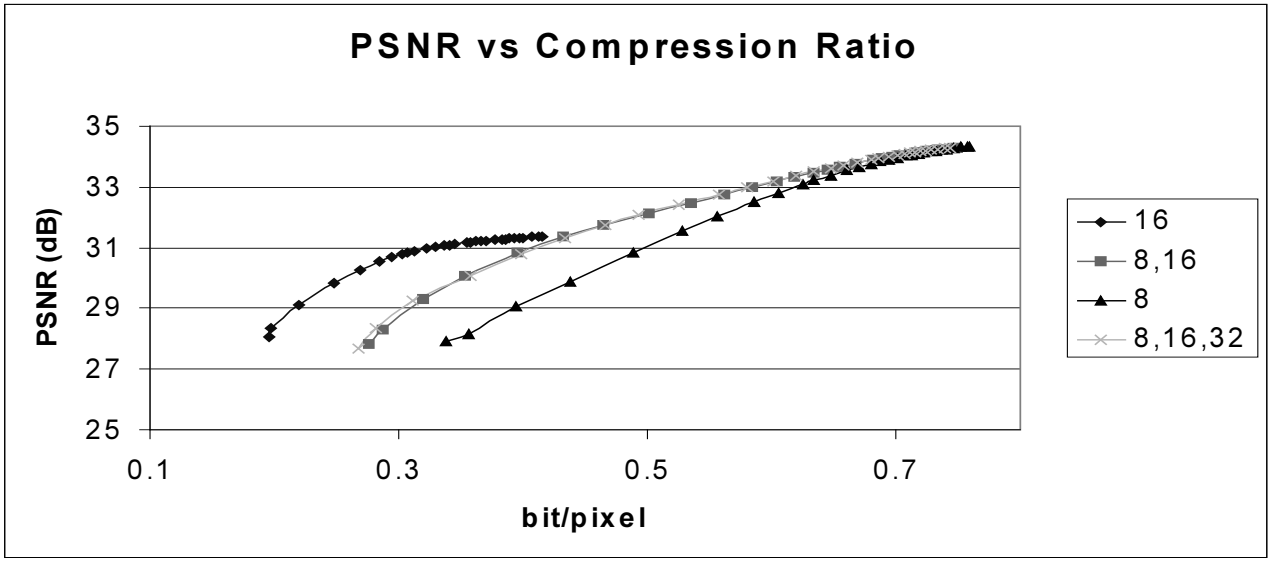


Figure 23.
PSNR (dB) vs. Compression Ratio (bit/pixel) at different possible block sizes for "Lena 512" in a noisy coding model (Noise2)

4 Conclusions

In this chapter I have examined some VLSI constraints [13,21,77] based on the necessary accessories [75, 94] to build a parallel analog system for image coding by orthogonal transformations. In [94] DCT and HT were given as possible methods for energy compression. Results show that DCT is usually better in terms of performance, but HT is more robust against circuit noise and it is easier to be implemented in VLSI (instead of analog lines only logical data lines are needed for transferring the components of the basis functions).

On the other hand the dissertation has shown that analog noise can accumulate with the increasing number of coefficients used in coding. While the proposed compression models are loaded with circuit noise, the estimated time required to carry out the compression algorithms in analog VLSI is noticeably small. Compression efficiency achieved by the discussed encoding/decoding methods is comparable with standard and highly optimized algorithms, however, in some degree remains under best competitive techniques. For examples see images Fig. 25-27. Fig. 28 and Fig. 29 serve visual comparisons, while Fig. 24 compares compression performance with some other highly efficient methods.

There can be several reasons for this:

1. The proposed architecture is based on analog circuits with finite (~8bit) precision and computational noise.
2. The implemented encoding algorithm is less efficient in bit allocation than other methods. For example, in our examples 8 bits are used to encode all coefficients and the entropy encoder is not particularly configured for the coding of different coefficients as in the case of JPEG [66,105].
3. Usually, frequency threshold techniques are less effective than other bit allocation methods [41], also wavelet or fractal based algorithms are more efficient both in standard error measures or subjective human vision-based measures.

It has also been shown in this work that the overhead from the encoding of the quadtree may distort the results of the Lagrange optimization method. The cost of the overhead can not be included in the Lagrange method, so the optimization should be run for the different block range settings independently.

Besides all, its is important to mention that the proposed optimization algorithm is able to involve other different allocation techniques, such as the optimization of the coefficient quantization matrices not discussed in this work. By implementing it and with the use of optimized entropy codecs better efficiency is expected.

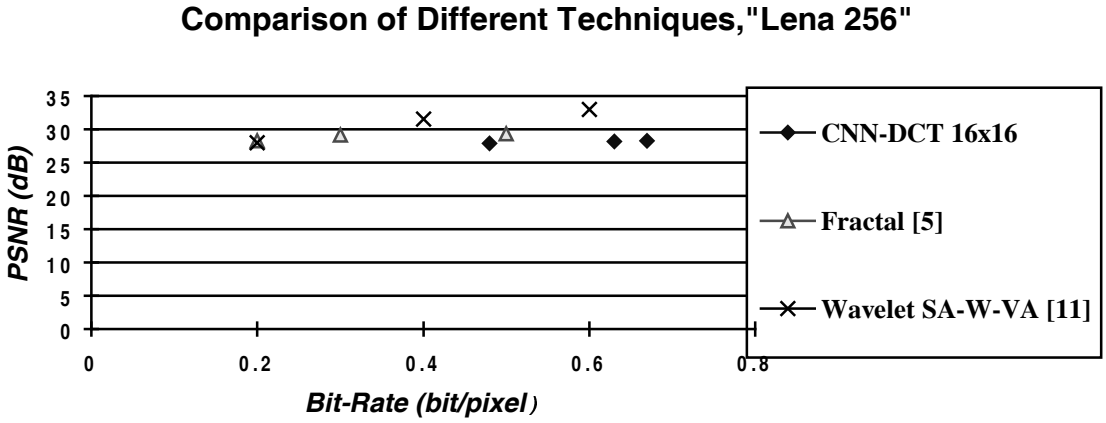
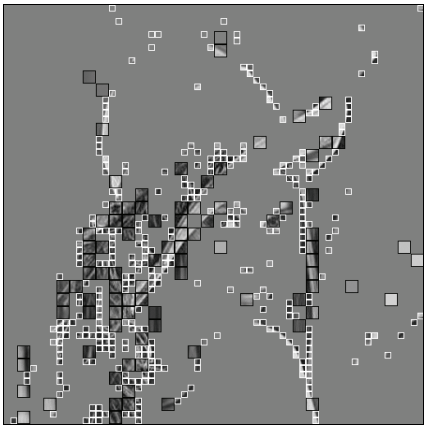
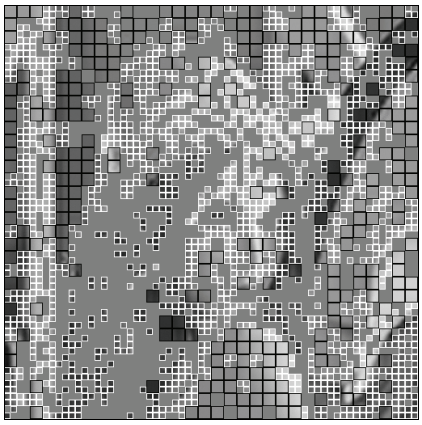


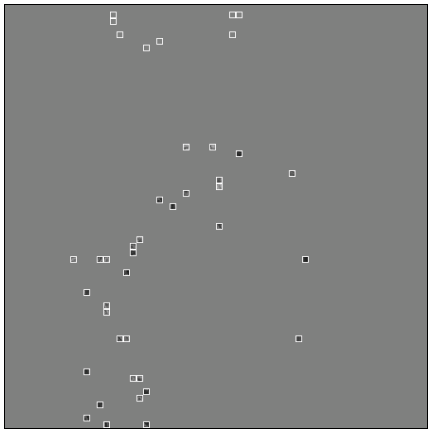
Figure 24.
Comparison of the analog CNN DCT and other optimized compression methods on "Lena 256" [27,86].



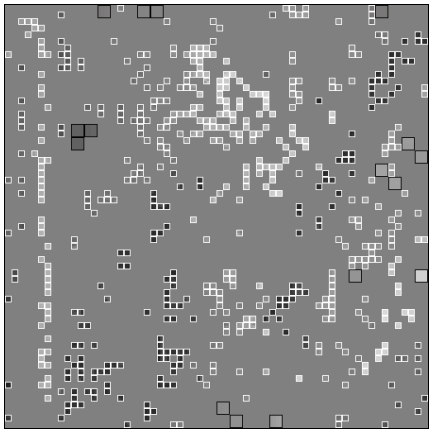
(a) DCT1, CF=90%



(b) DCT2, CF=30%



(c) HT1, CF=70%



(d) HT2, CF=30%

(e)

Figure 25.
Decoded "Lena 512" at 0.49bit/pixel and 32.1dB in the noisy model.

Figure 26. "Peppers", coded/decoded in the simulated analog noisy CNN model, at 0.246bit/pixel and 29.42dB.

Figure 27. "Crowd", coded/decoded in the simulated analog noisy CNN model, at 0.384bit/pixel and 28.44 dB.

Figure 28. "Lena 512", coded/decoded with baseline JPEG at 0.269bit/pixel and 31.9dB.

Figure 29. "Lena 512", coded/decoded in the simulated analog noisy CNN model, optimized at 0.268bit/pixel and 30.7dB.

CHAPTER III

Spatio-temporal Segmentation of Video Sequences with 2D Processor Arrays

1 Introduction

Generally, the class of spatio-temporal image analysis tasks requires both low-level and high-level optimization procedures with a huge amount of computing power [98]. In this chapter, a fully parallel methodology to investigate some motion analysis and motion segmentation problems with low-level algorithms based on limited local neighborhood connectivity is described [17,95]. Our efforts are aimed at finding solutions to these problems that need almost low-level, simple functions that can be implemented on special parallel VLSI architectures at superior speed, such as chip-sets built around the CNN-UM. The output of these low-level operations can be forwarded to high-level processors responsible for controlling the whole operation and for final interpretation. Since most of the computations would be done on a parallel processor array, significant speed-up could be achieved compared to other processor architectures as shown in later sections.

1.1 General Tasks in Motion Segmentation and Tracking

Motion models may consist of estimations about the components of the 3D scene and the motion of the camera itself. The latter is called ego-motion and if present without known motion parameters, it increases computation demands significantly. Motion observed on the image plane is formed by a projection system. It is called 2D motion, apparent motion or also optical flow. Generally perspective projections are used, but is some cases special optics generate unusual flow fields [19].

Tasks in motion segmentation and tracking can be classified into the following main groups:

- Estimation of motion information.
- Segmentation of motion fields.
- Spatio-temporal segmentation and tracking of video sequences.

Although there is, from a certain aspect, a sequential order of these tasks, an implicit, iterative estimation may result in better performance.

Spatio-temporal segmentation can be considered as the most important information source to detect or to track different moving objects in a video sequence. One important application is the analysis of image sequences for encoding objects in the MPEG4 environment. In spatio-temporal segmentation we use both spatial and motion information and by combining the two it may answer questions like whether two image areas belong to the same object or not. This is often impossible to answer correctly if only spatial or only temporal information is employed. On the other hand, many times it is difficult to make these decisions any way, since the two information sources can not only complement but can also conflict with each other. A more perfect solution would be the use of 3D geometric models of objects but that is beyond this work.

Estimation of optical flow fields can be crucial for the performance of the whole problem of spatio-temporal segmentation, since the estimated motion field is a basic input to all higher level algorithms. During estimation it is necessary to assume some constraints such as the local homogeneity of optical flow vectors. In most of the models smooth motion fields are assumed, so estimation is usually followed by a segmentation procedure or estimation and segmentation are run simultaneously.

Tracking motion information means that not only the current velocity of video objects is analyzed but also other characteristics such as the trajectory of moving objects or the motion in past frames.

To carry out dense optical flow estimation, segmentation and tracking on a single processor computer is hardly achievable on today's computing platforms (although feature based, non-dense optical flow methods are much closer to real time [58,85]). It is important to note, that as the image complexity increases (e.g. more moving objects appear), the required computational time also increases tremendously. This effect is avoidable if massively parallel architectures are used.

1.2 Basic Features of the Proposed CNN-based Model

In this chapter it is shown that besides the existing sophisticated algorithms many of the basic motion analysis tasks can be implemented in the special computation environment of VLSI feasible 2D arrays, which have a restricted number and type of operations.

The analog hardware platform of the proposed model has the following advantages:

- Low energy consumption.
- Integrated visual sensor.
- Real-time operation with locally parallel computations.
- High pixel densities are possible.

On the other hand, analog implementations are less precise and less versatile than the digital counterparts. In [82] 1D motion sensing problems are discussed with an emphasis on the effects of different SNR values on measurable speed range. Example applications such as heading direction estimation and time-to-contact measurements are given.

In the following model it is assumed that no ego-motion (the motion of the camera) is present and we concentrate on the segmentation of optical fields in the spatial and temporal domain. Since we mainly apply low-level processing rather than object-level motion analysis we neglect trajectory information but utilize the history of motion on the pixel level. This introduces some tracking abilities.

In our approach we utilize the dense spatial distribution of the cells to estimate the dense optical vector field. While most of the compression techniques (H.261, H.263, MPEG-1, MPEG-2 [43,44,46]) use only one 2D vector per block, the new MPEG-4 standard offers a region-based model for more flexible and precise video processing. In the proposed algorithm, starting from an initial estimation of the dense field we are approaching a region-based representation with the help of segmentation using spatial and temporal information.

Since the discussed methods are to be implemented on a discrete 2D field, motion vectors may have limited accuracy as well; especially we do not deal with sub-pixel or fractional velocity components. This approximation is reasonable if we consider that it is not our aim to get a precise motion field, rather to find, eliminate and describe the different video objects.

To sum up, the proposed architecture would be a useful intelligent sensor for stand-alone applications or front-ends in machine vision systems.

1.3 Elementary Functions of the Cell Array

In our model the elementary functions can be executed in parallel on all image pixels and can be realized by a set of VLSI functions in analog circuits like [21]. An important limitation of physical realization is the radius of local connectivity. We use only first (a pixel is connected to 4 of its neighbors) or second order (pixel is connected to 8 of its neighbors) neighborhood relations, since higher order connectivity would make it difficult to design the circuits for hardware manufacturers. In some cases larger neighborhood relations can be replaced by multi-scale representation and processing (see the first chapter for multi-scale processing in a pixel-level MRF environment).

The necessary cell functions and components are as follows:

- Comparison of neighboring pixels.
- Convolution operators. Convolution is a basic function already realized in many CNN chips. Since it is often used in many algorithms its execution speed is crucial for real time algorithms.
- Arithmetic and logic functions, relations. These functions are also core elements and since they are executed on the whole pixel array simultaneously, their implementation also calls for fast parallel processors.
- Cell memories. Results of arithmetic and logic functions are stored locally in cell memories. These local memories can be analog or logical - storing usually 8-bit precision or only binary values. Naturally, the number of memory per pixel has an upper limit depending on hardware technology. We should keep the number of necessary local memories per pixel as low as possible.
- Nonlinear functions: absolute value, gradient, etc.

Fig. 1 shows some sample-frames used in our experiments.

(a)

(b)

(c)

Fig. 1.
Samples from the video sequences used in our tests. (a) "Table Tennis", (b) "Mother and Daughter", (c) "Hamburg Taxi".

2 Estimation and Segmentation of the Optical Flow

2.1 Estimation of the Motion Displacement Field

Our approach of motion segmentation is based on estimating and segmenting the dense 2D optical vectors rather than using an image motion model of 6 or 8 parameters like in [42]. This is not a limitation, since camera ego-motion can be determined from motion vectors if needed. Instead of using a 2D or 3D parametric model, projections of vector flow fields can be used for ego-motion estimation as described in [26]. On the other hand, our model can be described with a small number of parameters reducing computation demands.

In most motion models there are several assumptions about the dependence of image data on the real motion of the scene. A very often used and important hypothesis is that image intensity remains unchanged along motion trajectories. However, this assumption can be violated by spatio-temporally varying illumination, non-opaque surface reflectance, occlusion, and noise. In our model we do not deal with accelerating motion and our dense model also disregards the description of non-rigid objects.

Basically, there are three main approaches to estimate dense optical vectors without involving a parametric model of more than two parameters:

1. In correlation or block matching techniques small patches of the image are compared with nearby patches of the consecutive frames.

2. Gradient-based algorithms compute optical vectors from spatial and temporal derivatives of image intensity.
3. Spatio-temporal filtering methods estimate optical flow in the frequency domain.

In the next sections we describe two models for the estimation and segmentation of motion information. Both solutions have approximately the same complexity and can be used with stochastic relaxation steps. The first is a correlation technique, which does not combine motion field estimation with optimization into one step. The second is basically a gradient-based approach that jointly estimates and classifies optical vectors in an iterative process.

2.2 Motion Estimation by a Parallel Correlation Technique

If motion field estimation itself is reliable then it is not always necessary to combine the estimation and segmentation into one process. Its main advantage and disadvantage originates from the same fact: we do not reevaluate motion information during segmentation. Obviously, this is computationally more effective but no sophisticated algorithm ensures the confidence of results. Since in many cases this method, followed by a segmentation algorithm, can still achieve good motion segmentation, results can be satisfactory for many motion-based applications. Its use is particularly advantageous when there is relatively fast motion, the sequence is temporarily undersampled or there are serious memory storage limits.

In this approach the most time consuming task is the computation of the displaced frame difference, or the so-called sum of squared differences (SSD):

$$SSD_{t,t+1}(x,y) = \sum_{(x_i,y_i)\in N_{(x,y)}} \left(I_t(x_i + V_{x,t}, y_i + V_{y,t}) - I_{t+1}(x_i, y_i)\right)^2 \quad (1)$$

That is the SSD at point x,y is calculated by comparing a translated pixel and its spatial neighborhood with the following frame. The vector (V_x,V_y) that gives the smallest error is considered as the estimated velocity vector. Since the quadratic form is very sensitive to outliers, it is often substituted with the more robust absolute error criterion [88] or by correlation techniques. To speed this search up, to find the most appropriate vector with the least SSD value, there are two basic approaches:

1. The number of points investigated can be reduced to some critical feature points. It is convenient because there may be homogeneous regions where the estimation is not reliable.
2. Sophisticated search methods can be applied to replace full search with special traversals.

Instead of these techniques we propose an algorithm of five steps to implement a full search. Since each step can be easily implemented in cell array architectures, the solution is not computationally prohibitive:

1. S*piral* movement of the *whole* current frame. The current frame is shifted with one pixel and the direction of motion is given in a spiral order. By moving the whole frame the search is performed for each pixel simultaneously. See Fig. 2 for illustration.
2. Subtraction from the next frame to get the difference image.
3. Multiplication to get the square.
4. Smoothing in a local neighborhood with heat diffusion or convolution. This step gives the spatial support $x_i, y_i \in N_{(x,y)}$.
5. If the resulting SSD value is smaller than the previously stored minimum value, store it as a new reference and also store the recent spiral-offset (the new motion vector candidate).

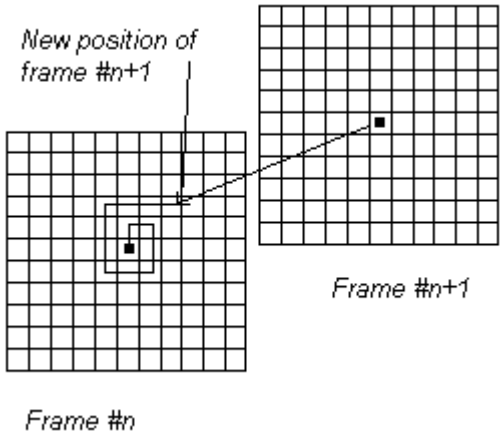


Fig. 2.
The spiral movement of an image frame over the preceding frame. The current position is shown after the series of steps: up, right, down, down, left, left, up, up, up, right, right, right.

This way the calculation is performed not only at one location but also at all pixels' neighborhood in one computation cycle. To run the search for all image pixels in a 5 by 5 window, 24 steps of one-pixel shifts (in the spiral traversal) of the image frame are necessary. See Fig. 4 showing motion fields for the "Table Tennis" sequence obtained with this correlation technique.

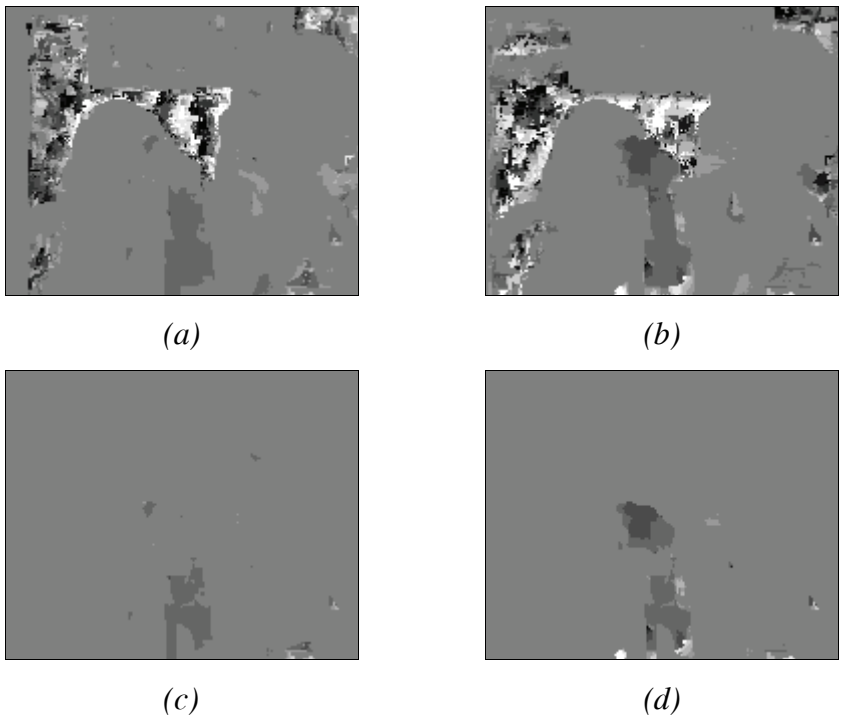


Fig. 3.
Motion field estimation of the frame #76-77 of the "Mother and Daughter" sequence.
x component, (b) y component of velocity vectors,
(c) x component and (d) y component filtered with statistical change detection.

A possible solution for noise filtering is to apply statistical change detection [1], where differences (changes) of succeeding image frames are smoothed and thresholded. This threshold can be based on a general noise model or on the specific noise parameter of the camera. Where no change is detected by statistical change detection between two subsequent frames, the estimated displacement can be neglected. Fig. 3 illustrates motion of the "Mother and Daughter" sequence in the *x* (3*a*) and *y* (3*b*) directions. The corresponding masked motion fields (3*c* and 3*d*) were obtained with statistical change detection.

Optimization of the motion field is not possible during motion field estimation in the correlation approach, since the iterative reevaluation of the SSD does not match the

spiral-translation model. Instead, segmentation is carried out after the estimation process as illustrated in Fig. 4 and Fig. 6.

2.3 Segmentation with an MRF-based method

In the first chapter a CNN-based MRF model was described with very simple functions to solve a general segmentation problem based on local observations of image intensity. Generally, the algorithm can be used for the segmentation of any scalar valued 2D field e.g. the magnitude of motion vectors as illustrated by Fig. 4. As for the energy optimization, the use of the Modified Metropolis Dynamics (MMD) [50] algorithm is proposed, which is a pseudo-stochastic relaxation process.
Since motion information is not scalar, this method's efficiency is very limited in our case. On the other hand the optimization of optical vector fields would involve too large number of possible classes in case of general image flows, accordingly, the direct transition of this optimization model to motion vector segmentation is limited to some special cases.

Instead, now we restrict the possible states, of 2D velocity vector candidates, at each image location to be one of its neighbors. Naturally, this restriction increases the probability that the algorithm will be trapped in a local minimum, that is why a good initial state is required. The definition of sites, neighbors and cliques does not change, however, the random variables at each site (pixel) are vectors from a common state space: $X = \{X_s \mid s \in \mathbf{S}\}$ so that $\forall s \in \mathbf{S} : X_s = (X_x, X_y) \in \Lambda$, where $\Lambda = \{1,.....,L\}$ are the labels for the different velocity vectors and let $\Omega = \{\omega = (\omega_{s_1}, \omega_{s_2}, \omega_{s_3}, ..., \omega_{s_N}) : \omega_{s_i} \in \Lambda,\ 1 \le i \le N\}$ be the set of all possible configurations.

The energy function to be minimized is the following:

$$\boldsymbol{E}(\omega, F) = E_1(\omega, F) + E_2(\omega), \tag{2}$$

where

$$E_1(\omega, F) = \sum_{s \in \boldsymbol{S}} \left(\mu_s - \mu_{\omega_s}\right)^2, \tag{3}$$

$$E_2(\omega) = \sum_{C \in \boldsymbol{C}} E_C(\omega), \tag{4}$$

and

$$E_C(\omega) = E_{\{s,r\}}(\omega_s, \omega_r) = \begin{cases} -\beta & if \quad \omega_s = \omega_r \\ \beta & if \quad \omega_s \neq \omega_r \end{cases}. \tag{5}$$

Local observation is generated similarly to the original CNN-MRF model:

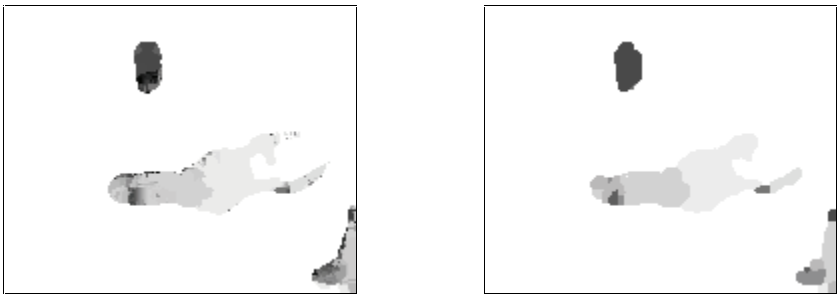
$$\mu_s = \frac{f_s + s_s}{2}, \tag{6}$$

where f_s is the initially estimated motion field and s_s is its smoothed version. In our experiments we used a second order neighborhood system for generating the new vector candidates but cliques were defined only on a first order neighborhood system. This way we achieved relatively fast convergence (compared to a truly stochastic optimization method where possible vector candidates are not limited to be chosen from the neighborhood).

Naturally, such an optimization of a vector field requires 2 cell array layers for the state representation of V_x and V_y respectively.

Fig. 5 (a) shows an artificial test image where the two blobs are moving to the North East and North West direction respectively. These input images were loaded with different Gaussian noise (Fig. 5 (b), (c)). Motion-segmentation results of the noisy sequences are illustrated by Fig. 5 (f) and (g).

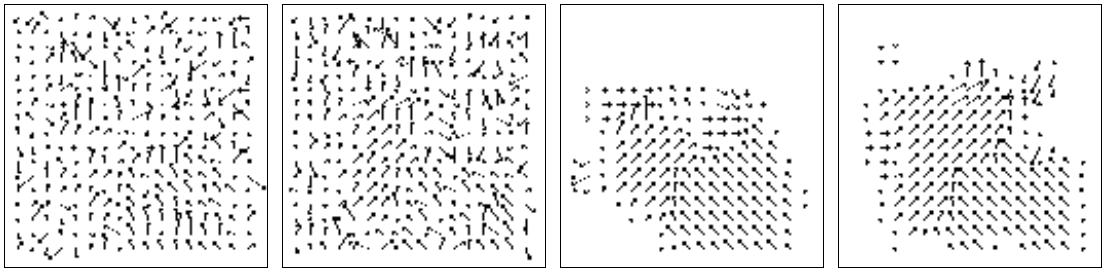
Fig. 6 shows the 2D optical vector field for the sequence “Hamburg Taxi”. Both the initial estimation, obtained with the correlation method, and the segmented vector field is illustrated.



(a) *(b)*

Fig. 4.
(a) Magnitude of optical vectors, (b) segmented with the CNN-MRF model to 9 classes. The optimization ran for 50 iterations. First the motion field was estimated with the correlation technique, then it was segmented with the MRF-based method.

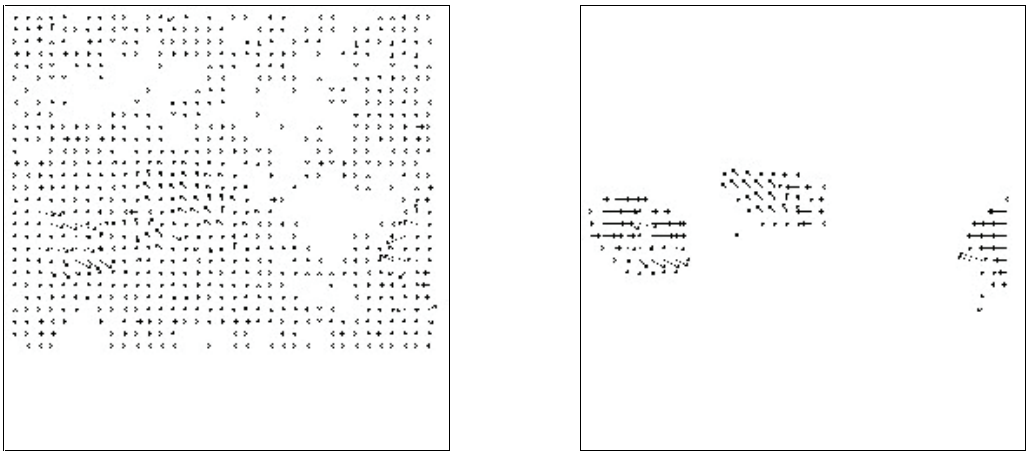
(a) *(b)* *(c)*



(d) *(e)* *(f)* *(g)*

Fig. 5.

(a) Two grayscale rectangles moving NE and NW at speed $\sqrt{2}$, (b) original image loaded with Gaussian noise, RMSE≈20, (c) RMSE ≈ 15, (d) initial motion field of (b) obtained by the correlation method, (e) initial motion field of (c), (f)-(g) motion segmentation of (b) and (c).



(a) *(b)*

(c)

Fig. 6.
(a) Initial vector field after a rough correlation estimation, (b) segmented vector field with the CNN-MRF method, (c) result projected onto frame #2 of the sequence "Hamburg Taxi".

2.4 A Gradient-Based Method: Simultaneous Estimation and segmentation of the Optical Flow by Energy Minimization

When estimated and segmented motion information is not satisfactory due to heavy noise or other effects, then we can run the motion estimation and segmentation in one optimization process. In [40] and [65] this technique is used with the correlation approach. Since motion information should be reconsidered many times during the segmentation process the above algorithms can be extremely time consuming.

On the contrary, we have chosen the gradient approach for optimization, because the continuous reevaluation of the SSD in the correlation-based model does not fit our parallel framework. In other words, the spiral movement of frames made it possible to calculate the SSD in all pixel locations simultaneously but an optimization involving the motion model would require the repetition of the spiral image translation process. This problem is resolvable with using the gradient-based estimation approach, where the evaluation of a possible motion vector candidate can be achieved for all pixels simultaneously in the parallel framework.

2.4.1 Constraints for the Optical Flow from the Intensity Conservation

One way of deriving the gradient constraint is based on image intensity conservation, stating that the image intensity is constant along the motion trajectories (see [38] for details):

$$\frac{d}{dt} I(x(t), y(t), t) = 0 . \tag{7}$$

Applying the chain rule:

$$\frac{d}{dt} I(x(t), y(t), t) = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}\frac{dt}{dt} = I_x V_x + I_y V_y + I_t . \tag{8}$$

Since we have two unknowns and one constraint equation, it is common to combine constraints from several neighboring locations:

$$E(V_x, V_y) = \sum_{x_i, y_i \in N_{(x,y)}} W_{x-x_i, y-y_i} \left(I_{x_i} V_x + I_{y_i} V_y + I_{t_i} \right)^2 \tag{9}$$

I, with corresponding lower indices, stands for spatial (I_x, I_y) and temporal (I_t) derivatives of the image. This equation assumes that motion vectors are distributed smoothly (it is naturally not always true). *W* is a decaying window (e.g. Gaussian form) and $E(V_x,V_y)$ can be considered as a measure of how much the set of constraints are satisfied by a motion vector candidate.

Giving an analytical expression for the velocity vectors in the least square estimate sense we can write:

$$\begin{aligned} \frac{\partial E(V_x, V_y)}{\partial V_x} &= \sum_{x_i, y_i \in N_{(x,y)}} W \left(I_{x_i}^2 V_{x_i} + I_{x_i} I_{y_i} V_{y_i} + I_{x_i} I_{t_i} \right) = 0 \\ \frac{\partial E(V_x, V_y)}{\partial V_y} &= \sum_{x_i, y_i \in N_{(x,y)}} W \left(I_{x_i} I_{y_i} V_{x_i} + I_{y_i}^2 V_{y_i} + I_{y_i} I_{t_i} \right) = 0 \end{aligned} \tag{10}$$

Leaving small indexes the above equations can be written in a matrix form:

$$M(x, y) \cdot V(x, y) + b(x, y) = 0 \tag{11}$$

where *V* is the optical flow field of components V_x and V_y and *M* and *b* are:

$$M = \begin{bmatrix} \sum W I_x^2 & \sum W I_x I_y \\ \sum W I_x I_y & \sum W I_y^2 \end{bmatrix} \quad b = \begin{pmatrix} \sum W I_x I_t \\ \sum W I_y I_t \end{pmatrix}. \tag{12}$$

Assuming that *M* is invertible the solution is given:

$$\hat{V} = -M^{-1} b \tag{13}$$

This description uses a constant model for V in a small neighborhood, giving further constraints to solve the problem in the least square sense. As to find the most appropriate motion vectors, we must obviously choose the vector that approximates the constraint equation with the least error.

It is easy to see that the elements of M and b can be directly computed in a parallel way, well suited to our cell array framework and this calculation is required only once for each video frame. The evaluation of a motion vector candidate in each pixel position at time t, carried out by some multiplications and additions with the elements of M and b, foretells that the evaluation can be a part of a motion optimization/segmentation algorithm. This way, motion vector estimation and segmentation can be combined into one model and can be processed by the iterative change of V_x and V_y and evaluating an energy function defined below.

2.4.2 Segmentation by Energy Minimization

To achieve our final goal of spatio-temporal video segmentation it is not necessary to achieve a very accurate optical flow, but rather to get a smooth segmentation of the moving areas. Next we propose a segmentation model that is based on the energy minimization of a formulae including the smoothness of the optical flow and the gradient-based motion estimation model.

The optical flow segmentation model is similar to the previous MRF based image segmentation. Now, the energy term of this model to be minimized is the following:

$$E(\omega, M) = \sum_{s \in \mathbf{S}} \left\| M_s \cdot V_{\omega_S} + b_s \right\|^2 + \sum_{C \in \boldsymbol{C}} E_C(\omega) \tag{14}$$

where

$$E_C(\omega_C) = E_{\{s,r\}}(\omega_s, \omega_r) = \begin{cases} -\beta & if \quad \omega_s = \omega_r \\ +\beta & if \quad \omega_s \neq \omega_r \end{cases}. \tag{15}$$

In the equations ω is the appropriate label field and $V_{\omega s}$ is the corresponding velocity candidate. As we want to optimize a vector field, there might be too much possible vector candidates (in other words in general motion fields the number of possible classes could be too large) so it would make it impossible to achieve fast convergence. Due to this reason we made two restrictions on the label field (similarly to the segmentation method in the previous section):

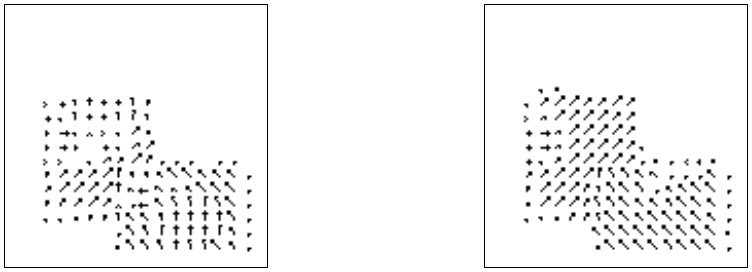
1. For the initial vector field we use the vectors obtained by an initial estimation. During the optimization no new values are introduced.

2. In the optimization process a label can be changed only to the value of one of its neighbors.

We found other authors applying the same constraints with success under similar conditions [11,40].

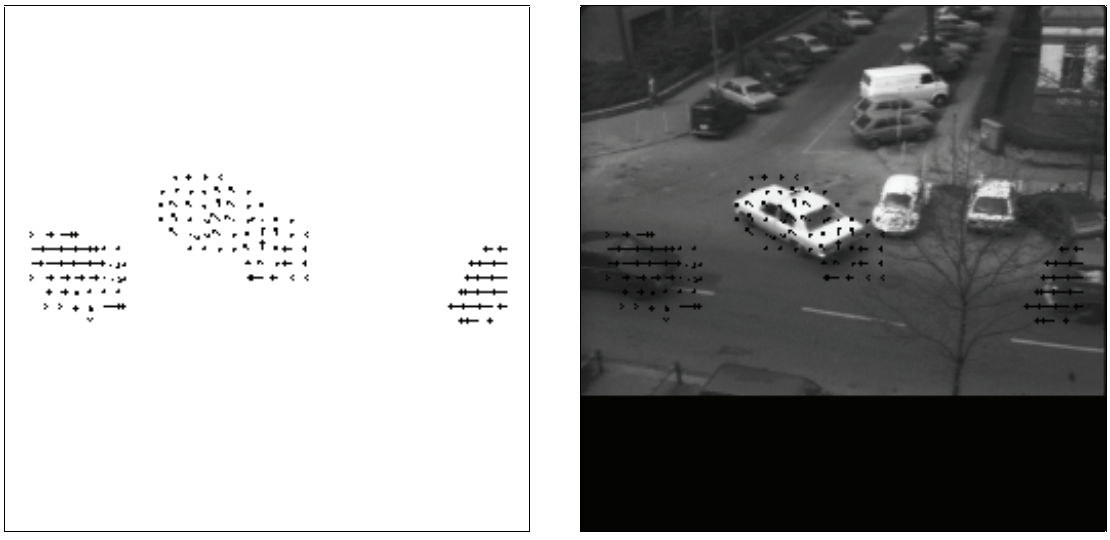
With these restrictions fast convergence can be reached within some hundred iterations using the MMD optimization method. Similarly to the previous motion segmentation model, this algorithm also requires 2 cell array layers for the state representation and optimization of V_x and V_y respectively.

Fig. 7 (a) and (b) shows the results of segmentation on the input images Fig. 5 (b) and (c) respectively. Fig. 8 shows motion segmentation results from the sequence "Hamburg Taxi".



(a) *(b)*

Fig. 7.
Segmented vector fields with the gradient-based optimization method of input images Fig. 5 (b) and (c).



(a) *(b)*

Fig. 8.
(a) Segmented vector field with the gradient-based optimization method, (b) result projected onto frame #2 of the sequence "Hamburg Taxi".

2.4.3 Related Motion Segmentation Models Based on Energy Minimization

Here we describe shortly other motion segmentation models, that we found similar to the proposed method. The most similar segmentation algorithms found in the literature are based on the model of Horn and Schunk [38], where the estimated velocity field is obtained by the minimization of the form:

$$\Sigma\left(\left(I_x V_x + I_y V_y + I_t\right)^2 + \lambda\left(\left\|\nabla V_x\right\|^2 + \left\|\nabla V_y\right\|^2\right)\right) \tag{16}$$

where the magnitude of λ determines the influence of the smoothness term. Iterative algorithms are used to minimize Eq. (16), e.g.:

$$\begin{aligned} V_x^{k+1} &= \hat{V}_x^k - \frac{I_x\left(I_x\hat{V}_x^k + I_y\hat{V}_y^k + I_t\right)}{\alpha^2 + I_x^2 + I_y^2} \\ V_y^{k+1} &= \hat{V}_y^k - \frac{I_y\left(I_x\hat{V}_x^k + I_y\hat{V}_y^k + I_t\right)}{\alpha^2 + I_x^2 + I_y^2} \end{aligned} \tag{17}$$

where k stands for the iteration, and $\hat{V}_x^k$ and $\hat{V}_y^k$ are for spatial averages of V_x^k and V_y^k. Both terms of Eq. (16) have the same meaning as the members of Eq. (14), however, our model simplifies computations in several aspects. In our solution the first term contains smoothing effects itself, and there is no need to reevaluate $\hat{V}_x^k$ and $\hat{V}_y^k$ during the iterations, M and b are invariant during our algorithm. While the second terms are very similar, our smoothness constraint is justified to the Gibbs/Markov model, similarly to the optimization procedure, where our Markovian model contains a pseudo-stochastic optimization - instead of the iterative calculation of Eq. (17) - to solve the minimization problem.

Other Markovian segmentation models are described in [11,29] but with different estimation models, both referenced papers deal with global affine motion models not well-suited to our pixel-based 2D array model.

3 Edge Optimization for Spatio-Temporal Segmentation and Tracking

3.1 Subroutines of the Proposed Model

Since the proposed model for the whole spatio-temporal segmentation algorithm includes a large number of image manipulation steps, it might be useful to list them separately, then to simply reference them in the next sections. These sub-tasks, such as finding edges, filtering noise, estimating motion parameters can be considered as subroutines of the parallel solution.

3.1.1 Nonlinear Diffusion

Nonlinear or anisotropic diffusion [2,12,67,79] is an effective tool in image enhancement, capable of smoothing images in an adaptive way. While linear smoothing removes not only noise but the edge content also, nonlinear and anisotropic smoothing can preserve edge information, often crucial in the solution of image analysis problems.

Two equations for nonlinear filtering were already given in Section 3.4 in Chapter 1. In a later section it was also discussed, how the exponential characteristics of edge sensitivity could be substituted by a linear approximation.

Fig. 9 shows a sample image as the result of this diffusion. Edges are preserved while noise is removed. Since other nonlinear filters, such as rank order filters, can also be implemented in our parallel framework [71], we will use them for oversegmentation purposes.

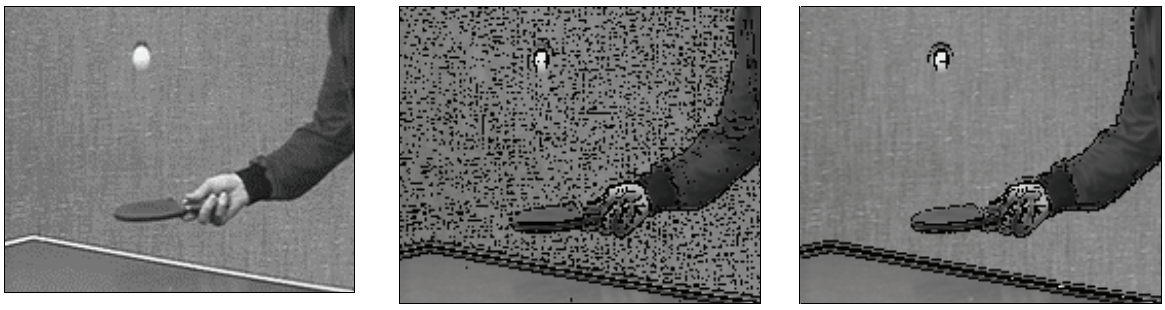


Table Tennis Sequence, Frame #6 | Edge Map Projected onto Original Image | Edge Map of *Nonlinear Diffusion* Filtered Image Projected onto Original Frame

Fig. 9.
The effect of nonlinear diffusion for the enhancement of main edges.

3.1.2 Pixel Level Tracking: Motion History

As we will see, accumulated motion information of the recent frames is an important part of our spatio-temporal segmentation model. In many cases the currently measured motion field itself cannot describe the motion very well. With the help of motion history information, temporal uncertainty of estimation can be reduced and long-range information can be accumulated. We track the motion of each point and register if it has stopped or was in motion within a given period of time:

$$I_{MH}(x,y,t+1)=\begin{cases} I_{MH}(x-V_x(x,y,t+1),y-V_y(x,y,t+1),t)+1 & \text{if} \quad V(x,y,t+1)\neq 0 \quad \text{and} \quad I_{MH}(x,y,t)<M \\ I_{MH}(x-V_x(x,y,t+1),y-V_y(x,y,t+1),t)-1 & \text{if} \quad V(x,y,t+1)=0 \quad \text{and} \quad I_{MH}(x,y,t)>-M \end{cases} \tag{18}$$

where V is the corresponding motion field (with components V_x and V_y) and the magnitude of M determines the memory-length (or temporal support) of the process. This way we get a motion history field denoted I_{MH}, where areas with greater value mean regions that have been moving further in the last M frames. Greater M means that the algorithm has longer memory and thus motion transparency is weaker. Fig. 10 shows two succeeding motion maps and the corresponding motion history maps of the sequence “Mother and Daughter”.

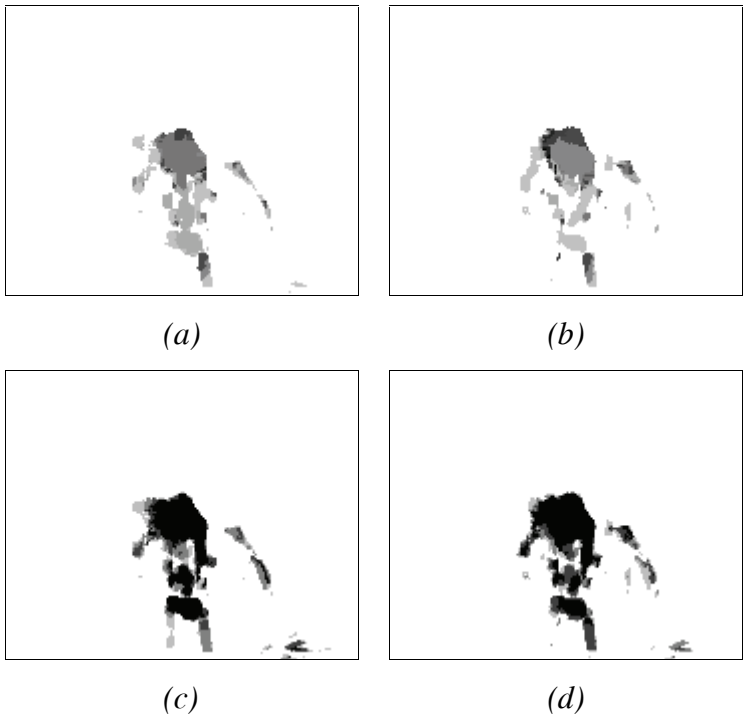


Fig. 10.
Motion and motion history of the sequence “Mother and Daughter”.
Speed #81 (a) speed #82 (b) motion history #81 (c) motion history #82 (d).

3.1.3 Morphology Operators on Cell Arrays

The implementation of morphology operators on parallel machines is very reasonable [68], since these functions are simple and operate in the close neighborhood of a pixel. But to hug to the CNN-UM computation model, we refer to [103] where the implementations of many grayscale and binary morphological operators on the CNN-UM are described. For our purpose we need only binary operators especially for the thinning of edge maps obtained during the optimization process. Small patch removal is also useful in motion analysis e.g. to mask out moving regions below a certain size.

3.1.4 Disocclusion Removal in Parallel

Vector fields, obtained by any motion field estimation technique, generally suffer from errors of disocclusion [7]. It is a systematic error, since we know that estimating motion from image projections is an ill posed problem. The removal of disocclusion effects needs higher interpretation of motion, at least (observable) background areas should be recognized. If we assume that background regions are separated from objects in the front, then we can give an approximation to the solution of the problem of disocclusion with parallel low-level steps.

Measuring the optical vector for every pixel, the false stripes (disoccluded background areas) can be removed from the estimated motion field in a consecutive series of steps. The disadvantage of our approximation that we are able to handle only a finite number of directions and magnitude of velocity vectors. The number of iterations of the disocclusion removal algorithm depends on the maximum velocity and the number of directions present in the segmented motion field.

We should repeat the following steps below for all featuring directions ϕ. ϕ can take a value from the 8 basic orientation: E, W, S, N, NE, NW, SW, SE and can be represented numerically in the cell memories of every pixel. V_ϕ equals the maximum speed in direction ϕ. The initial state of the algorithm is the segmented motion field.

0. Choose ϕ as one possible direction present in the motion field and initialize $V\phi$.
1. Set every pixel to “background” if there is a "background" neighbor on the opposite direction (E–W, S-N, etc.) and has different value than *"P"* (for the definition of *“P”* see the next step).

2. Decrease the speed of those pixels that lay in the direction of the pixels that are set to “background” in the previous step (#1). If the speed of them reduces to zero, mark them *"P"* (as “processed”, a new value different from “background” and any other speed.
3. Decrease $V\phi$. If $V\phi$ is greater then zero then go to step #1 otherwise go to step 0.

While most steps of the algorithm is a deterministic parallel labeling process based on local operations and decisions, some serial functions are still needed. Such as those for histogram analysis of the initial segmented motion field to determine the possible set of directions of motion and the maximum speed in all directions.

3.2 The Basis of the Spatio-Temporal Segmentation Algorithm

While many probabilistic approaches use labeling algorithms for spatio-temporal segmentation [11,31,49], in our model we employ fast contour-based segmentation. In this contour-based optimization method there is no need for registering regions or to deal with graph-based representations that is not possible in the framework of cell array processors.

Our algorithm is mainly based on three inputs: the oversegmented image (based on grayscale information), the estimated and segmented optical flow and the motion history information. We found that in many cases the joint utilization of intensity values and the current motion information (motion estimated between two consecutive frames) was not enough to satisfactorily define the objects’ contours. On the other hand, the probability that two neighboring image blobs belong to the same object is the higher the more of the following requirements are satisfied:

- The two blobs have similar color (or grayscale intensity value).

  (In case of textured areas, texture filters [91] can be applied to label these regions with colors.)
- The two blobs have similar velocity.
- The two blobs had similar activities in the recent past.

In our spatio-temporal segmentation process we apply a contour controlled split & merge algorithm to find coherent image areas based on these three features of neighboring regions.

(To reduce the dimensionality of the problem it is possible to replace motion vectors with scalars by a clustering method. In some of our experiments we simply dropped one component, the segmentation algorithm seemed to be quite robust and gave satisfactory results when we considered only the magnitude of velocity vectors.)

3.3 The Segmentation Process

Now, instead of constructing an explicit energy model like in Sections 2.3 and 2.4, we introduce an implicit optimization algorithm where contours are responsible to get an optimal spatio-temporal segmentation of video sequences.

Four edge maps are generated during the algorithm: edges separating areas of different intensity values ($\boldsymbol{E_{in}}$), edges separating different motion fields ($\boldsymbol{E_{mx}}$,$\boldsymbol{E_{my}}$) and edges separating fields of different motion history values ($\boldsymbol{E_{mh}}$). Edge-fragments of these maps are different subsets of the spatio-temporal binary edge map $\boldsymbol{E_{segm}}$, which is a subset of the edge map of the initially oversegmented image ($\boldsymbol{E_{os}}$).

The edge maps ($\boldsymbol{E_{in}}$, $\boldsymbol{E_{mx}}$, $\boldsymbol{E_{my}}$, $\boldsymbol{E_{mh}}$) are weighted and then added to form a unified edge map ($\boldsymbol{E_u}$) that is thresholded and used to modify the actual $\boldsymbol{E_{segm}}$. Then the intensity, motion and motion history fields are updated by diffusion inside the contours of the new $\boldsymbol{E_{segm}}$. If the difference between the new state of the three feature fields and their previous state is too large, some edges may be restored. Then at the next iteration the different edge maps are measured again and a new unified map is formed, etc.

The optimization is based on the following implicit model:

When the four edge maps are added to form a new unified edge map, the applied threshold criterion is analogous to evaluating a *Dam-potential* between the neighboring segments S_i and S_j:

$$D(S_i, S_j) = \sum_{k=1}^{4} w_k \left| L_k(S_i) - L_k(S_j) \right| \tag{19}$$

where L_1 = intensity, $L_{2,3}$ = motion, L_4 = motion history (see Section 3.1.2), while w_k is a weighting coefficient. If $D(S_i, S_j)$ is above a threshold, then the edge is kept, otherwise deleted at that location.

The reconstruction of edges is a necessary part of the algorithm, because the merging of similar neighboring regions *in one step* can result in the merging of distant areas that have very different values (see Fig. 12). Hence we use the following expressions

to measure the effects of edge removal. First, we define the new average feature values over a segment:

$$L_k(S_M) = \sum_{S_i \subseteq S_M} \frac{A_i L_k(S_i)}{A_M} \tag{20}$$

is the k^{th} feature value of the unified region S_M obtained by merging regions S_i, corresponding segment-areas are denoted by A_M and A_i. The change due to the formation of a new region S_M is expressed for each S_i ($S_i \subseteq S_M$) by the difference of the old and the new levels:

$$Q(S_M, S_i) = \sum_{k=2}^{4} \left| L_k(S_M) - L_k(S_i) \right| \quad . \tag{21}$$

If $Q(S_M, S_i)$ is above a predefined value, then the previously eliminated but stored edge-fragments around S_i are reconstructed again. Notice, that no intensity is considered in the edge reconstruction process. It means that regions with different intensity can be merged more easily than with different motion information.

Alternatively, instead of measuring the change for each S_i separately, we can measure the accumulated error over S_M as a volumetric average:

$$Q(S_M) = \frac{1}{A_M} \sum_{k=2}^{4} \sum_{\forall S_i \in S_M} \left| L_k(S_i) A_i - L_k(S_M) A_i \right| . \tag{22}$$

In this case the reconstruction of edges applies for the whole area of S_M, causing the restoration of all previous edges over S_M. This second solution results in larger areas while the other is rather to maintain smaller segments with large contrast. Note that in Eq. (21) and (22) averaging over an area is carried out diffusion inside the edge-defined borders.

The individual steps of the proposed algorithm are the following (see Fig. 11 for illustration while Fig. 16 shows some examples):

1. ***Segment the input image***, based on intensity observations, possibly to a large number of segments of characteristic closed regions. The obtained segmented image is called oversegmentation, and it gives the finest partitioning that could be achieved in the whole spatio-temporal segmentation process. Good oversegmentation can be generated with the help of anisotropic diffusion or median filtering of the input frame. Both can be implemented in the parallel framework [71,79].

2. ***Produce the edge map of the oversegmented intensity field (E_{os}) by an edge-detector*** [103]. E_{os} is a binary map showing the more-or-less closed segment-borders of the oversegmented image parts. In the segmentation process the state variable is the actual edge map, the binary E_{segm}.
   - Starting condition: $E_{segm} = E_{os}$.
3. ***Diffuse intensity, motion and motion history fields inside the regions defined by E_{segm} with the help of external edge controlled diffusion. Then make the grayscale edge maps of these fields, namely E_{in}, E_{mx}, E_{my}, and E_{mh} respectively.*** *These non-binary maps contain the edge-strength values between the different diffused areas in the same points where the oversegmented binary edge-segments are in E_{segm}.*

External edge controlled diffusion is similar to anisotropic diffusion (see Section 3.1.1), however, the edge control should act from the external E_{segm} edge map. We found that in some cases the anisotropic diffusion may not smooth the feature fields inside strong contours uniformly – in spite of the large number of iteration steps of the numerical approximation of the formula given in the first chapter of this work. Naturally, we do not expect precise averaging like in a region-based segmentation method with conventional numerical solutions. Instead, after a given number of steps we stop the diffusion process. Then, as a supplement, a new series of operations begin when we change every pixel's value to its greatest neighbor, except if a pixel has at least two neighbors with corresponding edge points represented in the external edge map E_{segm}. This last condition ensures that we get homogenous areas inside contours and it prevents averaging between regions separated by the "external" edges. This last series of steps is also responsible to get continuous contours rather than leaking edge lines and curves.

Note that while diffusion takes a long computation time on a conventional digital computer (or any SISD architecture), this time is proportional with the diffusion radius on the parallel array. For comparison, a diffusion process on the CNN chip on the whole image of radius r should take similar time than reading out r different values from a memory!

4. ***Weight and add together the maps E_{in}, E_{mx}, E_{my}, and E_{mh} to form a unified map E_u.*** With the increase of the weight of one component we can control how much the segmentation process should lean on that given type of information. In our

experiments we applied approximately the same weights (w) for the two motion type maps and a significantly smaller weight for the intensity map, e.g.:

$$w(\boldsymbol{E}_{in}) : w(\boldsymbol{E}_{mx}) : w(\boldsymbol{E}_{my}) : w(\boldsymbol{E}_{mh}) = 0.2 : 1.2 : 1.2 : 1.2.$$

5. ***Threshold the superimposed edge-map*** $\boldsymbol{E}_u$ and reduce the edges in $\boldsymbol{E}_{segm}$:

$$\boldsymbol{E}_{segm} := \boldsymbol{E}_{segm} \setminus \boldsymbol{E}_u^{(thresholded)}.$$

Edges of $\boldsymbol{E}_{segm}$ below a threshold in $\boldsymbol{E}_u$ are neglected: closed contours can leak or whole edges can disappear this way.

6. ***Approximate the average motion and motion history feature fields by external edge controlled diffusion inside the contours of the modified*** $\boldsymbol{E}_{segm}$. This diffusion is just similar to step 3.
7. ***Correct*** $\boldsymbol{E}_{segm}$ ***with reconstruction (***$\boldsymbol{E}_{rec}$***).*** Naturally, the optimal control of the merging of different areas with different intensity, motion and motion history must be a reversible method [62]. Although, our cell array framework does not enable us to process a graph-based optimization or higher-level understanding, we can still make a feedback to rebuild some lost edges. In every cycle, the change between the current motion fields and the previously segmented motion fields is measured. Over those areas, where the difference of the old and the new features (given by Eq. (21) or (22)) is greater than a predefined value, a mask is generated ***(***$\boldsymbol{E}_{rec}$***)***. Then with the help of this mask we can reconstruct edges from the stored edge map of the previous iteration cycle: $\boldsymbol{E}_{segm} := \boldsymbol{E}_{segm} \cup \boldsymbol{E}_{rec}$. Fig. 12 illustrates a typical situation (applying Eq. (21)) when a cascade of edges are removed because neighboring areas were similar to each other, but the regions at the two margins had significant differences. One may think that the contour-based segmentation is very sensitive for leaks on edges but this feedback can interact and correct the segmentation process.
8. Cycle controlling
   - ***Decrease edge weights.*** In our experiments we decreased edge weights by 0-20%. If this relaxation-factor is small, then edge destruction is slow; otherwise the different regions merge into each other faster.
   - ***Go to step 3.***

According to our test results, approximately 10-15 iterations were sufficient to get stable edge contours. Morphology operators may then be used to get thin lines as a

final result. The segmented motion history of the last iteration cycle can also be used as the input for the calculation of motion history of succeeding frames.

The resulted map ($\boldsymbol{E_{segm}}$) contains the contours of the spatio-temporal objects. This map can be forwarded to a vector based DSP processor for further analysis, such as MPEG-4 video transmission or motion analysis applications.

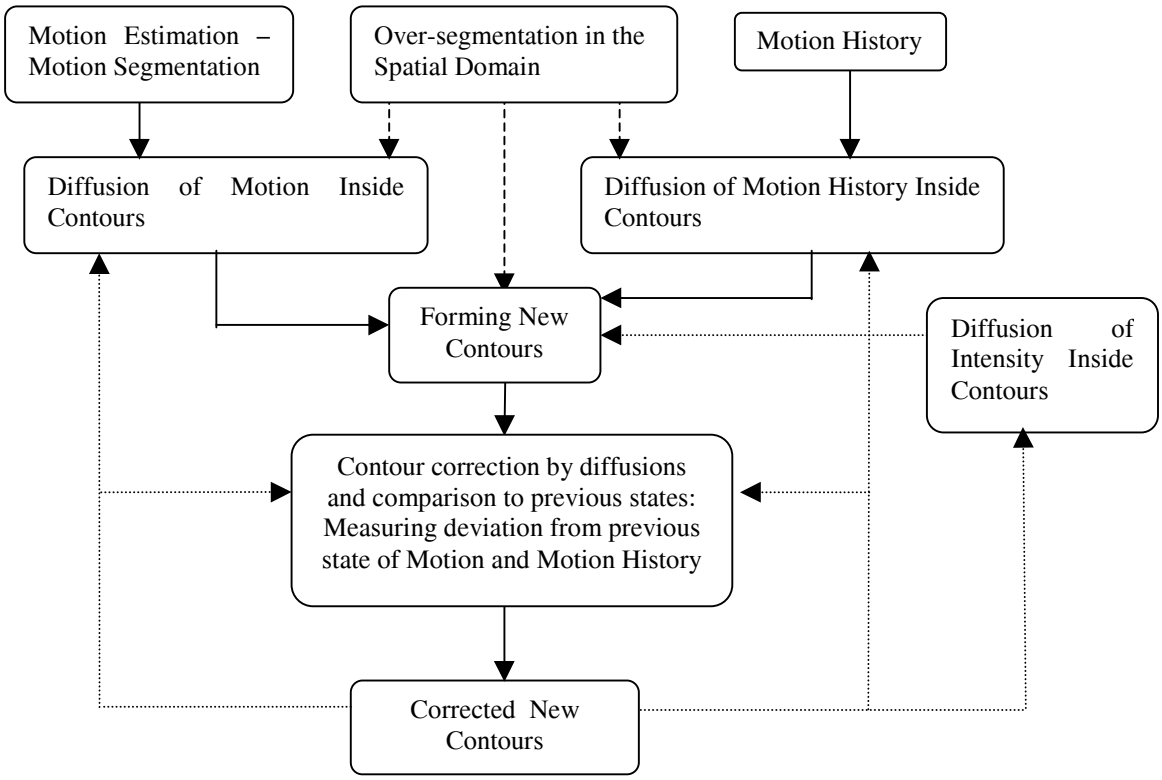


Fig. 11.
Iterative edge-based spatio-temporal segmentation.
Dotted lines symbolize feedback of the new contour to the following iterations and to the error computation. The broken lines mean data transfer needed only for the first iteration.

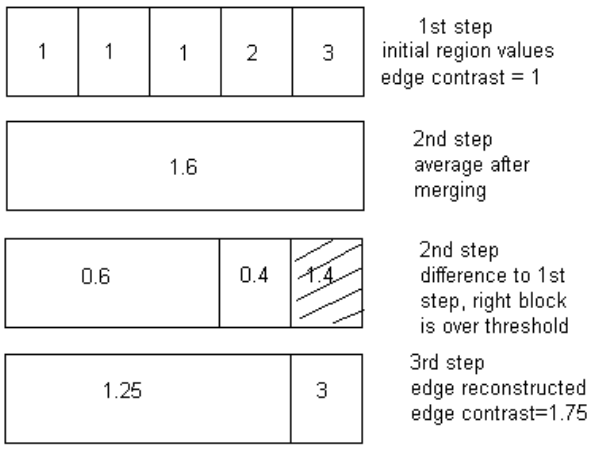


Fig. 12.
Edge reconstruction in the edge-based optimization model. In the first step all five regions are merged but then at the next step the one on the right is separated. The difference between its value and the average of the five blocks was over a threshold of 1.0.

4 Experimental Results

The following examples are the results of a parallel simulation (with the help of a parallel VLSI programming language (Analogic Macro Code, [53]). All steps, except the controlling and global parameter-setting operations, are defined with simple analog low-level operations.

Fig. 16 and Fig. 14 contains some results of the proposed algorithm. We also show some images of segmentation of motion/motion history at different steps of the iteration cycle (Fig. 16 *(d),(e),(f),*(*g),(h),(i)*). In the first example (the sequence "Table Tennis") the algorithm marked those parts that had different motions. The upper part of the arm was handled separately from the hand since it had different color and either motion, or motion history was far from being uniform within the object's borders. Fig. 16 *(j)* and *(k)* shows the average speed and motion history within the detected region's borders after the 10^{th} iteration. Edge maps are also demonstrated at two different iteration levels.

Fig. 14 illustrates how edges are being modified through an iteration cycle of 10 steps. The optimization for this image is stable after 10 steps. Small moving areas were removed because no spatial content was present at those blobs, however the effect of shadow was only partly removed. Fig. 15 is another example how incomplete motion field is reconstructed by spatio-temporal segmentation.

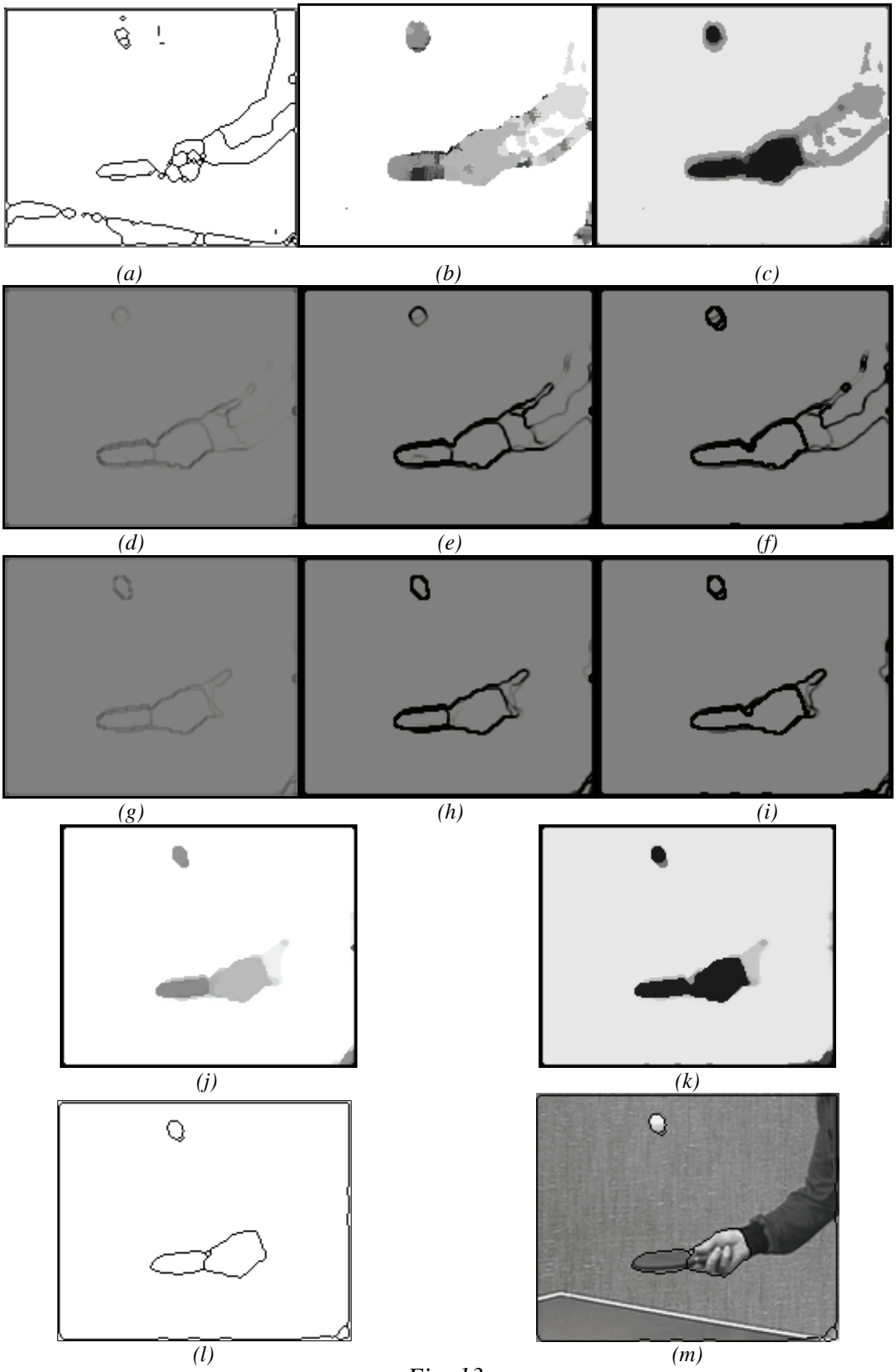
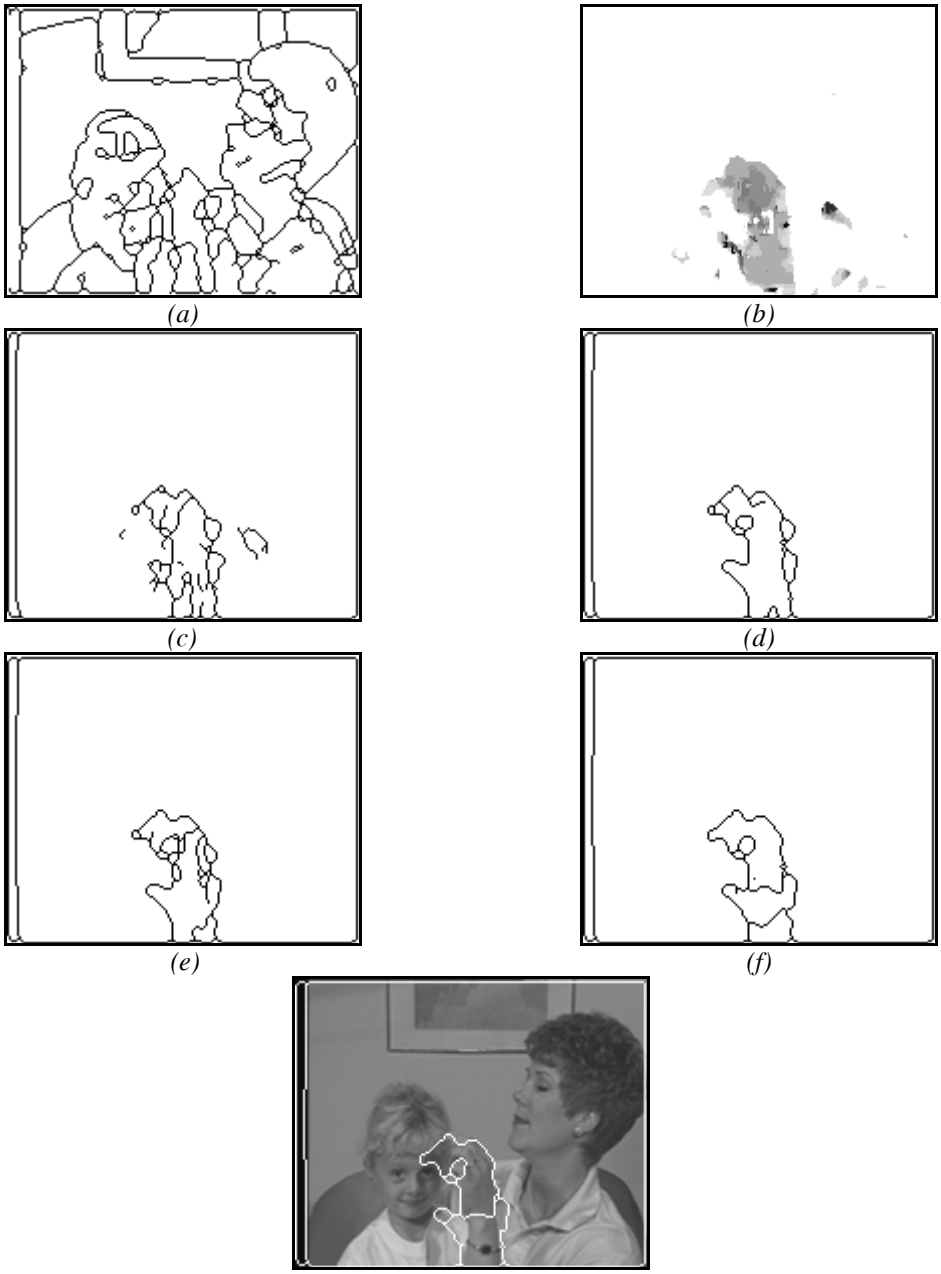


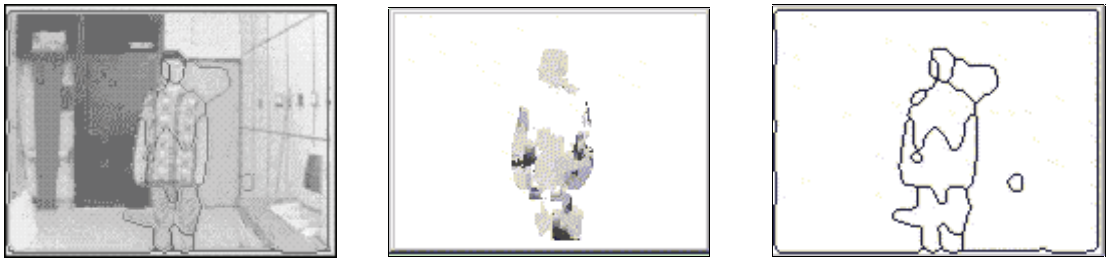
Fig. 13.

Spatio-temporal segmentation of the sequence "Table Tennis". (a) Oversegmentation obtained by nonlinear diffusion, median filtering and edge detection, (b) magnitude of velocity vectors, (c) motion history, (d) edge map of color, (e) edge map of velocity and (f) edge map of motion history after the 2nd iteration; (g) edge map of color, (h) edge map of velocity, (i) edge map of motion history after the 8th iteration; , (j) edge map of velocity, (k) edge map of motion history after the 10th iteration; (l) final contours after the 10th iteration, (m) final contours projected onto the input image.



(a) *(b)* *(c)* *(d)* *(e)* *(f)* *(g)*

Fig. 14.
Edge optimization for the spatio-temporal segmentation of "Mother and Daughter". (a) Oversegmented input frame, (b) motion of the current frame, edges of the (c) 1st, (d) 3rd, (e) 5th, (f) 9th, iterations, (g) final edge map (10th iteration) projected onto the input image.



Input image with motion-border *Speed-map* *Contour of moving object*

Fig. 15. A moving person with detected object borders. In spite of the scanty motion field, the upper part of the body is still detected as a moving area (Some artifacts are also present, e.g. the shadows on the background.)

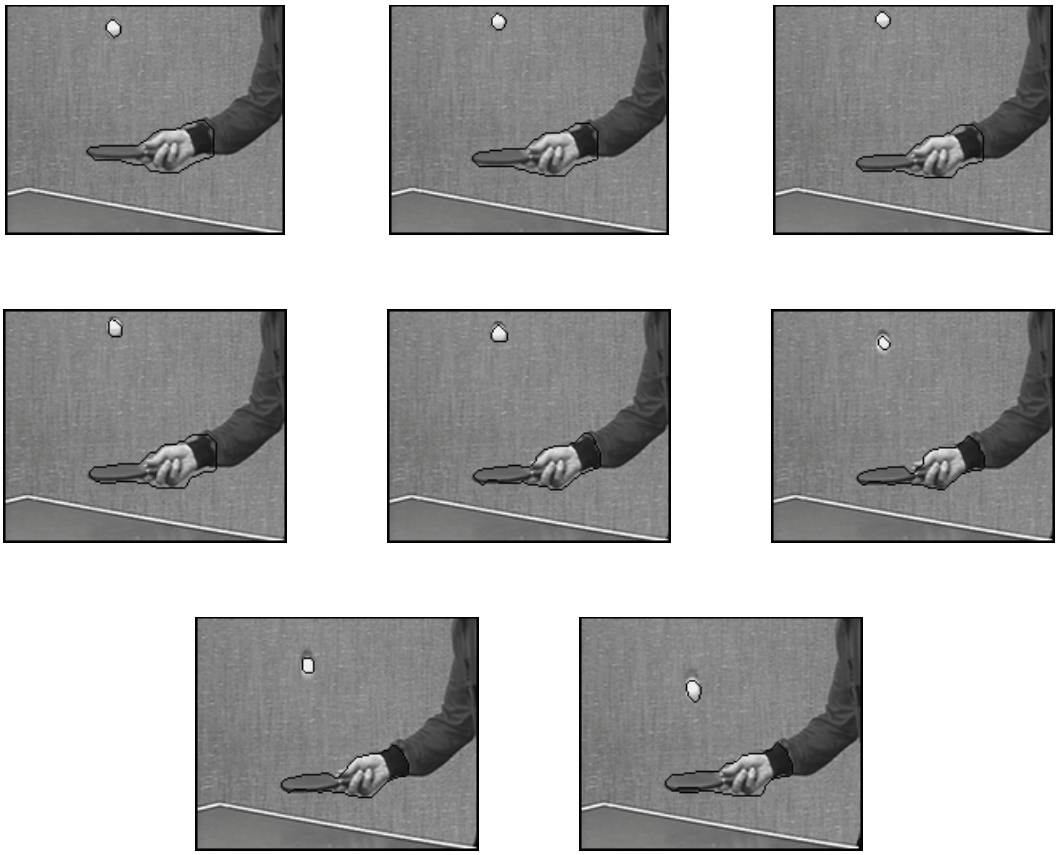


Fig. 16.
Spatio-temporal segmentation of "Table Tennis" frames #16-23.

5 VLSI Chip Speed and Complexity Estimation

Since spatio-temporal segmentation is a high-complexity task, even our fully parallel solutions require fast hardware implementations. We give computation complexity and running time estimations for the proposed algorithms on the CNN-UM parallel computation platform. With the help of our test programs, most of which ran on a software simulator [53], we estimated the number of different steps and the executions-times on the CNN-UM. For VLSI chip speed of basic operations we used data based on empirical tests [57]. Table 1 contains some physical parameters of a CNN chip [21] compared to other computation platforms.

Technology	*CNN* τ=200ns $\lambda = 0.5\mu m$	*Pentium®II. @ 400 MHz & MMX™* $\lambda = 0.25\mu m$	*TMS320C80 @40 MHz* $\lambda = 0.35\mu m$	*Matrox Genesis with C80 & NOA ASIC @ 50 MHz* $\lambda = 0.35\mu m$
Image Save/Load	90 *μsec*	40 *μsec*	80 *μsec*	80 *μsec*
Arithmetic Operation	500 *nsec*	38 *μsec*	156 *μsec*	47 *μsec*
Logical Operation	100 *nsec*	30 *μsec*	125 *μsec*	40 *μsec*
Conversion from analog values to binary values/Memory Transfer	200 *nsec*			
Convolution, 3x3	2*μsec*	125 *μsec*	383 *μsec*	28 *μsec*
Feedback Convolution (Dynamic IIR Spatial Filter)	5*μsec*			

Table 1

Comparison of execution times on different image processor platforms. Image size is 64x64. τ is the time constant of the analog chip [57].

		Number of instructions per iterations					Time
	Number of iterations	Parallel Data-Transfer in Chip	Serial Data-Transfer	Arithmetic Operations	Logical Operations	Convolution Template	*msec*
MDF	120	11	-	12	-	2	1.4
DR	100	7	-	11	6	1	1.1
MRF	100	7	-	16	6	5	2.4
NLDIF	30	4	-	10	-	8	0.7
MH	120	5	-	3	-	1	0.6
STS	15	22	2	13	2	3	5.6
Σ							**11.8**

Table 2.

Execution-time estimations for the different algorithms. The table gives typical data for processing a 64x64 image. In the columns there are the necessary numbers of steps per iteration and estimated time given in msecs. MDF: estimating motion displacement field with the correlation technique. DR: disocclusion removal. MRF: Markov Random Field based segmentation. NLDIF: nonlinear diffusion. MH: estimating motion history. STS: spatio-temporal segmentation.

Table 2 gives estimations about how fast our algorithms would run on a fully parallel architecture (like the CNN-UM). The full operation time would satisfy real-time requirements.

The following conditions are supposed in our comparison:

1. Test image size: 64x64 (currently available CNN chip is of size 64x64).
2. Grayscale image (8bit/pixel).
3. DSP is used for parameter setting and controlling the segmentation cycles.

On the other hand, there are other physical parameters than speed, such as area, power consumption, pin-number etc. Table 3 shows the most important technical data for common image processing platforms. This table shows that with a CNN chip we can achieve extremely fast computation at low power consumption on a small area. And what is about the technology? While the CNN-UM technology parameters are restrained: 0.5 μm VLSI technology with very low clock-speed, the computing power of the CNN-UM is still superior regarding the other image-processor architectures.

Processor	*Technology [μm]*	*Chip area [mm2]*	*Pin count*	*Worst case power consumption [W]*	*Clock speed [MHz]*
Pentium®II @ 400 MHz & MMX™	*0.25*	*230*	*242*	*25.0*	*400*
TMS320C80 40 MHz	*0.35*	*~240*	*305*	*8.3*	*40*
TMS320C62x 250 MHz	*0.25*	*n.a.*	*352*	*1.9*	*250*
CNN τ *=200 nsec* cP4000	*0.5*	*90*	*120*	*1.2*	*Digital: 10 Analog: 1*

Table 3.
Comparing some physical properties of different image processing platforms.

6 Conclusion

In this chapter fully-parallel motion segmentation and spatio-temporal segmentation schemes based on local computations and optimization were outlined.

I introduced two basic approaches for the segmentation of the optical flow, for both main techniques (gradient-based method, correlation-based method) I described the algorithms that have relatively small complexity. These algorithms can be used as inputs for higher level analysis. This analysis is carried out in the spatio-temporal segmentation process accomplished also with low-level operations.

The spatio-temporal approach consists of two main modules:

1. Algorithms for image and motion segmentation of spatial and temporal information by optimization.
2. Contour-based split-and-merge spatio-temporal segmentation to utilize the information obtained in the first module. It is showed that with the usage of motion history, intensity and the current motion it is possible to segment video sequences in the spatio-temporal domain.

Both parts can be realized with the same set of simple operations; the need for high-level control is not considerable. Basic local instructions are convolution operators, simple arithmetic steps, logical relations and the simplest nonlinear functions (sigmoid and gradient in a neighborhood). As it was found in the current and previous tests [17,96,95,], these algorithms are fast and give stable results in a reasonable number of steps.

The aims were to design optimal algorithms for fast implementations on parallel processor arrays. As time complexity estimations in Section 5 show the proposed approaches can result in real-time operation if implemented in VLSI. The parameters of the latest CNN chip have been applied to estimate the possible implementation and time complexity of the proposed system.

Theses

Thesis 1

1.1 New multiscale segmentation models have been introduced into the MRF-based segmentation model in the CNN environment. Two theoretically equivalent realizations are made with different memory and instruction requirements.
The range of neighborhood connectivity is an important limitation in CNN models. The aim of the proposed technique is to tailor the requirements of the CNN-MRF model to these neighborhood constraints. Its is shown that the multiscale model is capable of similar segmentation performance with reduced neighborhood connectivity. (Chapter I, Sections 3.5-3.8)

1.2 The noise sensitivity of the CNN-MRF model has been investigated, the robustness of the method is shown experimentally. It is also shown by simulations that a certain amount of noise, originating from VLSI implementation, does not decrease the segmentation precision, even segmentation accuracy increases at low noise.
The application of the optimization method, called Modified Metropolis Dynamics (MMD), is advantageous due to its low complexity and fast convergence [50,96], however it gives local optimum and it is only a pseudo-stochastic process. According to my simulations, low noise added to the MMD decision algorithm modifies its deterministic behavior and results in a real stochastic process. (Chapter I, Section 3.9)

Thesis 2

2.1 I have investigated the role of parameters in DCT coding, implemented on noisy analog parallel hardware (CNN-UM architecture). Experimentally I have showed that the increase of the number of DCT coefficients can increase coding error.

While in the case of conventional digital implementations of transform coding the increase of the number (or quantization levels) of transform coefficients increases coding quality and decreases compression rate, contrary in the proposed model of [94] the increase of the number transform coefficients can cause the increase of coding error. (Chapter, II Section 2)

2.2 I have introduced a dynamic coding method for the use of HT and DCT coding on analog 2D processor arrays.
With the help of the features of the model of [94] I have showed how to optimize the use of the two transformation methods in case of noisy implementations. At the same time it is shown that the dynamically changing block size causes an unwanted overhead and reduces compression ratio, which can be avoided by repeating the Lagrange multiplication method with different block size settings. (Chapter II, Section 3)

Thesis 3

3.1 I gave an MRF-based method for the segmentation of estimated optical motion field. The proposed algorithm does not require the definition of classes and fits well the structure of 2D processor arrays.
The method is similar to the CNN-MRF model of [96] but it is applicable for the segmentation of 2D vector fields. The new candidates for possible states are selected from the neighborhood of each pixel. (Chapter III, Section 2.3)

3.2 I have introduced a parallel gradient-based optical vector field segmentation method for 2D processor arrays.
The segmentation algorithm is based on the minimization of an energy function composed of two components: one component is formed from the gradient-based motion equation while the other component forces homogeneity. (Chapter III, Section 2.4)

3.3 I have showed that the pixel-level tracking, together with the segmented intensity information and segmented motion information, enables spatio-temporal segmentation of video sequences in the parallel 2D processor array environment.

Tracking implemented in the spatio-temporal segmentation algorithm consists of pixel-level steps and it generates a field describing the motion of some recent frames called motion history. Edge fields defined on motion history, motion field and the intensity information are the bases of the contour-based segmentation algorithm. The advantage of this method is that it contains simple operations executable on parallel 2D arrays. (Chapter III, Section 3)

Acknowledgements

I would like to thank the help of my supervisor Tamás Szirányi who supported my studies and research in the last years. Also a lot I got from my friends and colleges, both technically and personally, at the Department of Image Processing and Neurocomputing, UofV and at other laboratories like the Analogical and Neural Computing Laboratory, MTA SZTAKI.

Last I would like to thank the patience and support of my family, my parents and Judit and I hope that it was worthy to standby in the last couple of years.

References

[1] T. Aach, A. Kaup, R. Mester, "Statistical model-based change detection in moving video", *Signal Processing*, Vol.31, pp.165-180, **1993.**

[2] L. Alvarez, F. Guichard, P. L. Lions, J. M. Morel, "Axioms and Fundamental Equations of Image Processing", *Arch. Rational Mech. Anal.*, V.123, pp.199-257, **1993**.

[3] R. Azencott, "Markov Fields and image analysis", Proceedings of the *AFCET*, Antibes, **1987.**

[4] R. Azencott, "Parallel Simulated Annealing, Parallelization techniques", *Wiley,* **1992.**

[5] P. Baldi, W. Heiligenberg, "How sensory maps could enhance resolution through ordered arrangements of broadly tuned receivers", *Biol. Cybernetics*, Vol.59, pp.313-318, **1988**.

[6] J. L. Barron, D. J. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, Vol. 12(1), pp.43-77, **1994**.

[7] S. S. Beauchemin, J. L. Barron, "On the Fourier Properties of Discontinuous Motion", Submitted to *Journal of Mathematical Imaging and Vision*, **1999.**

[8] J. Besag, "On the statistical analysis of dirty pictures", *Journal of the Royal Statistical Society, B-68*, pp. 259-302, **1986.**

[9] A. Blake, "Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.11, pp.2-12, **1989.**

[10] C. A. Bouman, M. Shapiro, "Multiscale Random Field Model for Bayesian Image Segmentation", *IEEE Transactions on Image Processing*, Vol.3, pp.162-177, **1994.**

[11] P. Bouthemy and E. Francois, "Motion Segmentation and Qualitative Dynamic Scene Analysis from an Image Sequence," *International Journal of Computer Vision,* Vol.10:2, pp.157-182, **1993.**

[12] F. Catté, T. Coll, P. L. Lions, and J. M. Morel, "Image selective smoothing and edge detection by nonlinear diffusion", *SIAM J. Numerical Anal*., Vol.29, pp.182-193, **1992.**

[13] L. O. Chua, L. Yang, "Cellular Neural Networks: Theory", *IEEE Transactions on Circuits and Systems,* Vol.35, pp.1257-1272, **1988.**

[14] L. O. Chua, L. Yang, "Cellular Neural Networks: Applications", *IEEE Transactions on Circuits and Systems,* Vol.35, pp.1273-1290, **1988.**

[15] Roger J. Clarke, "Digital Compression of Still Images and Video", Academic Press, **1995.**

[16] K. Crounse, T. Roska, L.O. Chua, "Image halftoning with Cellular Neural Networks", *IEEE Transactions on Circuits and Systems* Vol.40, pp.267-283, **1993**.

[17] L. Czúni, T. Szirányi, "Motion Segmentation and Tracking with Edge Relaxation and Optimization using Fully Parallel Methods in the Cellular Nonlinear Network Architecture", *Real-Time Imaging*, accepted, **2000.**

[18] L. Czúni, T. Szirányi, J. Zerubia, "Multigrid MRF Based Picture Segmentation with Cellular Neural Networks," *CAIP'97*, Kiel, Proceedings in *Lecture Notes in Computer Science*, Vol.1296, pp.345-352, **1997.**

[19] L. Czúni, B. Vágvölgyi, T. Szirányi, T. Greguss, "A Compact Panoramic Visual Sensor for Intelligent Applications", *Proceedings of the 4th Asian Conference on Computer Vision (ACCV2000)*, Taiwan, IEEE, pp.258-263, **2000.**

[20] L. S. Davis, A. Rosenfeld, "Cooperating Processes for Low-level Vision: A Survey", *Artificial Intelligence*, 17, pp.245-263, **1981.**

[21] R. Domínguez-Castro, S. Espejo, A. Rodríguez-Vázquez, A. Carmona, P. Földesy, Á. Zarándy, P. Szolgay, T. Szirányi, T. Roska, "A 0.8μm CMOS Two-Dimensional Programmable Mixed-Signal Focal-Plane Array Processor with On-Chip Binary Imaging and Instructions Storage", *IEEE Journal of Solid-State Circuits*, Vol.32, No.7, pp.1013-1026, **1997.**

[22] T. Ebrahimi et al., "Dynamic coding of visual information," technical description ISO/IEC JTC1/SC2/WG11/M0320, *MPEG-4,* Swiss Federal Institute of Technology, October, **1995.**

[23] S. Espejo, R. Carmona, R. Dominguez-Castro and A. Rodriguez-Vazquez, "A CNN Universal Chip in CMOS Technology", *International Jorunal of Circuit Theory and Applications*, Vol.24, pp.93-110, **1996**.

[24] R. Etienne-Cummings, S. A. Fernando, J. Van der Spiegel, and P. Mueller, "Real-time 2D analog motion detector VLSI circuit" *Proceedings of IEEE International Joint Conference on Neural Networks,* New York, Vol.4, pp.426-431, **1992.**

[25] H. Everett III., "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources", *Operation Research*, Vol.11, pp.399-417, **1963.**

[26] S. Fejes and L. S. Davis, "What can projections of flow fields tell us about the visual motion," *Proceedings of the ICCV*, Bombay, India, **1998.**

[27] Y. Fisher (ed.), "Fractal Image Compression*"*, *Springer Verlag* , **1994.**

[28] D. J. Fleet, K. Langley, "Recursive Filters for Optical Flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.17, pp.61-67, **1995.**

[29] E. Francois, J-F. Vial, and B. Chupeau, "Coding Algorithm with Region-Based Motion Compensation", *IEEE Transactions on Circuits and Systems for Video Technology,* Vol.7, No.1, February **1997.**

[30] S. Geman, D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*," IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.6, pp.721-741, **1984.**

[31] M. Gelgon, P. Bouthemy, "A Region-Level Graph Labeling Approach to Motion-Based Segmentation", Technical Report, *INRIA*, France, **1996.**

[32] C. Graffigne, F. Heitz, P. Pérez, F. Prteux, M. Sigelle, J. Zerubia, "Hierarchical Markov random field models applied to image analysis: a review", *SPIE Conference, San Diego*, July 10-11, **1995.**

[33] GZIP Manual: http://www.gnu.org/manual/gzip-1.2.4/html_mono/gzip.html

[34] R. Haralick, "Image segmentation survey", Fundamentals in Computer Vision, *Cambridge University Press*, **1983.**

[35] F. Heitz, P. Perez, P. Bouthemy, "Multiscale minimization of global energy functions in some visual recovery problems", *CVGIP: Image Understanding*, Vol. 59, No.1, pp.125-134, **1994.**

[36] W. D. Hillis, "The Connection Machine", *MIT Press*, **1985**.

[37] M. W. Hirsch, S. Smale, "Differential Equations, Dynamical Systems, and Linear Algebra", *Academic Press*, San Diego, pp.180-203, **1974**.

[38] B. K. P. Horn and B. G. Schunk, "Determining optical flow", *AI* 17, pp.185-204, **1981.**

[39] T. Ikenaga, T. Ogura, "Discrete time Cellular Neural Networks using highly parallel 2D cellular automata CAM", *Proceedings of NOLTA'96,* Japan, pp.221, **1996.**

[40] K. Illgner and F. Müller, “Image segmentation using motion estimation” In V. Cappellini, editor, Time-Varying Image Processing and Moving Object Recognition, Vol. 4, *Elsevier Science* B.V., Amsterdam, pp.238-243, **1997.**

[41] K. Illgner and M. Braess, “On optimized selection of DCT-coefficients in H.261-like videocodecs”, *Proceedings of the IEEE International Workshop on Intelligent Signal Processing and Communication Systems ISPACS’93*, pp.339-344, Tohoku University, Sendai, Japan, October, **1993.**

[42] M. Irani, P. Anandan, “A Unified Approach to Moving Object Detection in 2D and 3D Scenes”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.20, No.6, pp.577-589, **1998.**

[43] ISO/IEC JTC1 IS 11172-2 (MPEG-1), Information technology – coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbits/s, **1993.**

[44] ISO/IEC JTC1 IS 13818-2 (MPEG-2), Information technology – generic coding of moving pictures and associated audio information, **1996.**

[45] ISO/IEC JTC1/SC29/WG11 N3536, Overview of the MPEG-4 Standard, Editor: Rob Koenen, http://www.cselt.it/mpeg/standards.htm, **2000.**

[46] ITU-T Recommendation H.263, Video coding of narrow telecommunication channels at <64 kbit/s, **1995.**

[47] Z. Kató, “Bayesian color image segmentation using reversible jump Markov chain Monte Carlo”, CWI Report, **1999.**

[48] Z. Kató, “Modélisations Markoviennes Multirésulution en Vision Par Ordinateur”, Thése, *L’Université de Nice-Sophia Antipolis*, pp.52-55, **1994.**

[49] Z. Kató, T.C. Pong, J.C.M. Lee, “Motion Compensated Color Video Classification Using Markov Random Fields”, *Proceedings of ACCV,* Vol.I, pp.738-745, **1998.**

[50] Z. Kató, J. Zerubia, M. Berthod, “Satellite image classification using a modified Metropolis dynamics”, *Proceedings of ICASSP*, San Francisco, **1992.**

[51] Z. Kató, Zerubia, M. Berthod, “A Hierarchical Markov Random Field Model and Multitemperature Annealing for Parallel Classification, *Graphical Models and Image Processing*, Vol.58, No. 1, pp.18-37, **1996.**

[52] S. Kirkpatrick, C. Gellatt, M. Vecchi, “Optimization by simulated annealing”, *Science* 220, pp. 671-690, **1983.**

[53] T. Kozek, Á Zarándy, S. Zöld, T. Roska, P. Szolgay, "Analogic Macro Code (AMC) - Extended Assembly Language for CNN Computers," Technical Report, MTA SZTAKI, Budapest, **1998.**

[54] T. Kozek, C. W. Wu, Á. Zarándy, H. Chen, T. Roska, M. Kunt, L.O. Chua, "New results and measurements related to dynamic image coding using CNN universal machine chips", *IEEE Transactions on Circuits and Systems-VT*, Vol.7, pp.606-614, **1997.**

[55] P. A. Laplante, A. D. Stoyenko Editors, "Real-Time Imaging, Theory, Techniques, and Applications", *IEEE Press*, **1996.**

[56] S. Z. Li, Markov Random Field Modeling in Computer Vision, Springer-Verlag, **1995.**

[57] G. Linan, S. Espejo, R. Dominguez-Castro, E. Roca, A. Rodríguez-Vázquez, "A Mixed Signal 64x64 CNN Universal Machine Chip", *MicroNeuro'99, IEEE*, Granada, Spain, pp.61-68, **1999.**

[58] H. Liu, Tsai-H. Hong, M. Herman, T. Camus, R. Chellappa, "Accuracy vs. Efficiency Trade-offs in Optical Flow Algorithms", Computer Vision and Image Understanding, Vol.72, No.3, pp.271-286, **1998.**

[59] R. Mester and U. Franke, "Spectral entropy-activity classification in adaptive transform coding", *IEEE J. Selected Areas Commun.* 10, pp.913-917, **1992.**

[60] N. Metropolis, A.W. Rosenbluth, M. N. Rosenbluth, A.H. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, Vol.21, No.6, pp.1087-1092, **1953.**

[61] J. P. Miller, T. Roska, T. Szirányi, K. Crounse, L. O. Chua, L. Nemes, "Deblurring of Images by Cellular Neural Networks with applications to Microscopy", Procedings of *3rd IEEE Workshop on CNN and their Applications*, Rome, December, pp.237-242, **1994.**

[62] F. Moscheni, S. Bhattacharjee, M. Kunt, "Spatiotemporal Segmentation Based on Region Merging," *IEEE Transaction on Patter Analysis and Machine Intelligence*, Vol.20, No.9, pp.897-915, **1998.**

[63] H. H. Nguyen and P. Cohen, "Gibbs Random Fields, Fuzzy Clustering, and the Unsupervised Segmentation of Textured Images", *CVGIP: Graphical Models and Image Processing*, Vol.55, pp.1-19, **1993**.

[64] N. Pal and S. Pal, "A review on Image Segmentation techniques", *Pattern Recognition*, Vol.26, No.9, pp.1277-1294, **1993.**

[65] J. N. Pan, Y. Q. Shi, and C. Q. Shu, "Correlation-Feedback Technique in Optical Flow Determination," *IEEE Transactions on Image Processing*, Vol.7, July, pp.1061-1067, **1998.**

[66] W. B. Pennebaker, J. L. Mitchell, "JPEG Still Image Data Compression Standard", *Van Nostrand Reinhold*, **1993.**

[67] P. Perona, T. Shiota, J. Malik, "Anisotropic Diffusion, Geometry - Driven Diff. In Computer Vision", *Kluwer Academic Publishers* , pp.73-92, **1992.**

[68] I. Pitas, editor, "Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks", *Wiley Professional Computing*, **1993**.

[69] A. D. Poularikas, editor, "The Transformations and Applications Handbook", *CRC and IEEE Press*, **1996.**

[70] Steve Purcell, "The Impact of Mpact 2", *IEEE Signal Processing Magazine*, March, **1998.**

[71] Cs. Rekeczky, T. Roska, and A. Ushida, "CNN Based Difference-controlled Adaptive Nonlinear Image Filters", *International Journal of Circuit Theory and Applications*, Vol.26, pp.375-423, **1998.**

[72] E. Reusens, T. Ebrahimi, M. Kunt, "Dynamic approach to visual data compression", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.7, pp.197-211, **1997**.

[73] Á. Rodriguez-Vázquez, S. Espejo, R. Dominguez-Castro, and G. Linan, "The 64x64 Analog Input CNN Universal Machine Chip and its ARAM", *Proceedings of the International Symposium on Nonlinear Theory and Applications,* (*NOLTA'98*), pp.667-670, Le Régent, Switzerland, 2-88074-391-5, **1998.**

[74] Y. Rosanov, "Markov Random Fields", Springer Verlag, **1982.**

[75] T. Roska, "CNN Chip set Architectures and the Visual Mouse", Proceedings of CNNA '96 (Seville), IEEE, pp.487-492, **1996.**

[76] T. Roska, G. Bártfay, P. Szolgay, T. Szirányi, A. Radványi, T. Kozek, Zs. Ugray and Á. Zarándy, "A digital multiprocessor hardware accelerator board for Cellular Neural Networks: CNN-HAC*", International Journal of Circuit Theory and Application*, Vol.20, pp.589-599, **1992.**

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

[78] T. Roska, L. Kék, L. Nemes, Á. Zarándy, M. Brendel and P. Szolgay (editors), "CNN Software Library (Templates and Algorithms), Version 7.2", *DNS-1-1998*, (CADET-15), Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, **1998**.

[79] T. Roska, T. Szirányi, "Classes of Analogic CNN Algorithms and Their Practical Use in Complex Processing", *Proceedings of the IEEE Non-linear Signal and Image Processing*, June, pp.767-770, **1995**.

[80] T. Roska, P. Szolgay, T. Kozek, Á. Zarándy, Cs. Rekeczky, L. Nemes, L. Kék, K.László, I.Szatmári, M.Csapodi, "CADETWIN", Budapest, MTA SZTAKI, **1997.**

[81] T. Roska, G. Bártfai, P. Szolgay, T. Szirányi, A. Radványi, T. Kozek, Zs. Ugray and Á. Zarándy, "A Digital Multiprocessor Hardware Accelerator Board for Cellular Neural Networks: CNN-HAC", *International Journal of Circuit Theory and Applications*, Vol.20, pp.589-599, **1992.**

[82] R. Sarpeshkar, J. Kramer, G. Indiveri, and C. Koch, Analog VLSI Architectures for Motion Processing: From Fundamental Limits to System Applications, Proceedings of the IEEE, Special Issue on Parallel Architecture for Image Processing, Vol.84, pp.969-987, July, **1996.**

[83] Claude E. Shannon, Warren Weaver. A Kommunikáció matematikai elmélete, OMIKK, Budapest, **1986.**

[84] B. E. Shi, T. Roska and L. O. Chua, "Estimating Optical Flow with Cellular Neural Networks," *International Journal of Circuit Theory and Applications*, Vol.26, No.4, July, pp.343-364, **1998.**

[85] J. Shi, C. Tomasi, "Good Features to Track", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Seattle, pp.593-600, June, **1994.**

[86] E. A. B. da Silva, "SA-W-VQ: wavelet based vector quantization", *IEEE Transactions on Image Processing*, Vol.6, No.2, pp.299-310, **1996.**

[87] K. Slot, L. O. Chua, T. Roska, "Very low-bitrate video coding using Cellular Neural network Universal Machine", *UCB/ERL, Memo* M97/46**, 1997.**

[88] C. Stiller, and J. Konrad, "Estimating Motion in Image Sequences" IEEE Signal Processing Magazine, Vol.16, No.4, pp.70-90, **1999.**

[89] A. Stoffels, T. Roska, L.O. Chua, "Object oriented image analysis for very-low-bitrate video-coding systems using the CNN Universal Machine", *International Journal of Circuit Theory and Applications*, Vol.25, pp.235-258, **1997.**

[90] T. Szirányi, “Robustness of Cellular Neural Networks in image deblurring and texture segmentation”, *International Journal of Circuit Theory and Applications*, Vol.24, pp.381-396, May **1996.**

[91] T. Szirányi, M. Csapodi, “Texture Classification and Segmentation by Cellular Neural Network using Genetic Learning”, *Computer Vision and Image Understanding,* Vol.71, No.3, pp.255-270, **1998.**

[92] T. Szirányi, J. Csicsvári, ”High speed character recognition using a dual Cellular Neural Network architecture”, *IEEE Transactions on Circuits and Systems*, V.40, No.3(II.), pp.223-231, **1993.**

[93] T. Szirányi, L. Czúni**, “**Picture Segmentation with Introducing an Anisotropic Preliminary Step to an MRF Model with Cellular Neural Networks”, *Proceedings of the 13th ICPR*, IEEE, Vienna, pp.366-370, **1996.**

[94] T. Szirányi, L. Czúni, “Image Compression by Orthogonal Decomposition Using Cellular Neural Network Chips”, *International Journal of Circuit Theory and Applications*, Vol.27, No.1, pp.117-134, **1999.**

[95] T. Szirányi, K. László, L. Czúni, F. Ziliani, “Object oriented motion-segmentation for video-compression in the CNN-UM”, *Journal of VLSI Signal Processing*, V.23, No.2-3, pp.479-496, **1999.**

[96] T. Szirányi, J. Zerubia, “Markov Random Field Image Segmentation using Cellular Neural Network,” *IEEE Transactions on Circuits and Systems I.*, Vol.44, January, pp.86-89, **1997.**

[97] T. Szirányi, J. Zerubia, L. Czúni, D. Geldreich, Z. Kato,”Image Segmentation Using Markov Random Field Model in Fully Parallel Cellular Network Architectures,” *Real-Time Imaging*, accepted, **2000.**

[98] L. Torres, M. Kunt (Editors), “Video Coding: The Second Generation Approach”, *Kluwer Academic Publishers*, ISBN: 0 7923 9680 4, **1996**.

[99] T. Toyoda, Y. Nitta, E. Funatsu, Y. Miyake, W. Freeman, J. Ohta, and K. Kyuma, “Artificial retina chips as image input interfaces for multimedia systems,” *Proceedings of the Optoelectronics and Communications Conference,* OECC'96, Chiba, Japan, July, **1996.**

[100] P. L. Venetianer, T. Roska, “Image Compression by CNN”, *Proceedings of the IEEE ICNN-96*, Washington, DC, 3, pp.1510-1515, **1996.**

[101] P. L. Venetianer, F. Werblin, T. Roska, L.O. Chua, “Analogic CNN algorithms for some image compression and restoration tasks”, *IEEE Transactions on*

Circuits and Systems I: Fundamental Theory and Applications, (CAS-I), Vol.42, pp.278-284, **1995.**

[102] F. Werblin, A. Jacobs, J. Teeters, "The computational eye", *IEEE Spectrum,* pp.30-37, **1996.**

[103] Á. Zarándy, A. Stoffels, T. Roska and L.O. Chua, "Implementation of Binary and Gray-Scale Mathematical Morphology on the CNN Universal Machine"*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* (CAS-I), Vol.45, No.2, pp.163-168, **1998.**

[104] J. Zerubia, R. Chellappa, "Mean field approximation using Compound Gauss-Markov Random Field for edge detection and image estimation", *IEEE Transactions on Neural Networks,* Vol.8, pp.703-709, **1993.**

[105] Independent JPEG Group's CJPEG, version 6a, 7-Feb-96, Independent *JPEG Group's DJPEG*