JYU DISSERTATIONS 340

Billy Braithwaite

Neurocomputing and Probabilistic Propagation in Computer Vision

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY

JYU DISSERTATIONS 340

Billy Braithwaite

Neurocomputing and Probabilistic Propagation in Computer Vision

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi joulukuun 16. päivänä 2020 kello 11.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
on December 16, 2020 at 11 o'clock am.

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2020

Editors
Timo Männikkö
Faculty of Information Technology, University of Jyväskylä
Ville Korkiakangas
Open Science Centre, University of Jyväskylä

Permanent link to this publication: http://urn.fi/URN:ISBN:978-951-39-8467-0

ISBN 978-951-39-8467-0 (PDF)
URN:ISBN:978-951-39-8467-0
ISSN 2489-9003

ABSTRACT

Braithwaite, Billy
Neurocomputing and probabilistic propagation in computer vision
Jyväskylä: University of Jyväskylä, 2020, 132 p.
(JYU Dissertations
ISSN 2489-9003; 340)
ISBN 978-951-39-8467-0 (PDF)

One of the earliest (also well-studied) research areas in *artificial intelligence* is the study of *visual perception*, and the study of neurons of the brain using connectivist models or *neurocomputing*. Where cognitive and mathematical psychology, and neuroscience studied how the brain and perception works in their own paradigms, artificial intelligence provided tools from theoretical and applied computer science to study the aforementioned areas using digital computers.

This study focuses on examining two sides of neurocomputing, namely *probabilistic graphical models* and *artificial neural networks* in solving early perception, or *early vision* and inference tasks. More specifically, the study examines *probabilistic propagation* such as denoising tasks under similarity measures and parallelization schemes. And finally, combining probabilistic graphical models and artificial neural networks into a pipeline model for solving inference tasks from a set of imaging measurements.

Keywords: Algorithms, Artificial intelligence, Inverse problems, scientific computing.

TIIVISTELMÄ (ABSTRACT IN FINNISH)

Braithwaite, Billy
Neurolaskenta ja todennäköisyyspropagaatio konenäössä
Jyväskylä: University of Jyväskylä, 2020, 132 s.
(JYU Dissertations
ISSN 2489-9003; 340)
ISBN 978-951-39-8467-0 (PDF)

Yksi *tekoälyn* ensimmäisistä, ja vanhimmista, tutkimusalueista ovat *näköaisti* ja aivojen neuronien mallintaminen *neurolaskenta* menetelmillä. Kognitiivisessa ja matemaattisessa psykologiassa, ja neurotieteessä tutkittiin näköaistia sekä neuronien toimintaa tutkimusalueiden kehittämillä malleilla, tekoäly tarjoaa työkaluja teoreettisesta ja sovelletusta tietojenkäsittelytieteestä edellämainnittujen alueiden tutkimiseen hyödyntämällä tietokoneita.

Tässä työssä käsitellään neurolaskennan kahta osapuolta: *todennänöisyysverkkoja* ja *neuroverkkoja*. Työn tavoitteena on kehittää menetelmiä *todennäköisyyspropagaatioon konenäkösovelluksiin* kuten kohinanpoistoon ja päättelytehtäviin, soveltamalla samankaltaisuusmittoja ja rinnakkaislaskentaa. Työssä käsitellään myös, kuinka yhdistelemällä todennäköisyysverkkoja ja neuroverkkoja voidaan ratkaista päättelytehtäviä kuvaantamissovelluksissa.

Avainsanat: Tekoäly, Neurolaskenta, Inversio-ongelma, tieteellinen laskenta

Author

Billy Braithwaite
Faculty of Information Technology
University of Jyväskylä
Finland

Supervisors

Professor Pekka Neittaanmäki
Faculty of Information Technology
University of Jyväskylä
Finland

Docent Ilkka Pölönen
Faculty of Information Technology
University of Jyväskylä
Finland

Reviewers

Associate professor Tuomas Eerola
LUT University
Finland

Associate professor Janne Kujala
University of Turku
Finland

Opponent

Tuomo Kauranne
Docent, University of Eastern Finland
President, CEO, Arbonaut Ltd.
Finland

ACKNOWLEDGEMENTS

Like biological evolution, this work progressed in small, random, and discrete jumps. To start with, I wish to thank my previous peers who had initiated me into the world of computational research: Kimmo Fredriksson, Tomi Kinnunen, Ville Hautamäki, Pasi Fränti, Pekka Toivanen and Zaid Al-Ars. However the biggest (academic) thanks goes to my supervisors Pekka Neittaanmäki and Ilkka Pölönen, for their anecdotes, tips, giving me the space and freedom in pursuing my "scholarly interests". And more importantly, thank you for pointing me to additional resources (when needed), giving me the opportunity to further develop myself through various miscellaneous projects, and providing funding for visiting essential workshops related my work. I appreciate my peer-reviewers Tuomas Eerola and Janne Kujala for taking the time to point out the short-comings and memory-lapses of my research work. Thank you also to Valery Zheludev and Luke Decker for discussions on seismic signal processing, and Kalevi Mursula for various discussions on other technicalities regarding my work.

I would like also thank my colleagues in the spectral imaging lab, who kept me company and giving interesting discussions on various, esoteric topics during my visits to Jyväskylä: Matti Eskellinen, Leevi Annala, Sampsa Kiiskinen, Kimmo Riihinaho, Anna-Mari Hakola, Anna-Leena Erkkilä and Samuli Rahkonen. Additional thank yous: Milena Veneva for helping me with various LaTex issues; Juha Kekäläinen for being my personal linux admin; Alessandro Foi, Sergei Repin, Blair Rajamaki, Radu Mariescu-Istodor, Matti Mononen and Waleed Akhtar for additional comments of my work.

A special thanks goes to Jaana Nevalainen. Thank you for housing me during my "few" visits to Helsinki. And another special thanks goes to Ville Soikkeli for also housing me at Casa de Villeneuve during my visits to Jyväskylä.

Überthanks goes to Heli, Mom, Dad, Brother, my band Shadow's Mortuary, my Canadian side of the family, and many other friends who put up with my weirdness, foul-moods, harshness and over all being human, all too human. I regret nothing.

This research work was funded by The Faculty of Information Technology JYU (doctoral programme funding), Jenny & Antti Wihuri Foundation, and Jane and Aatos Erkko Foundation (grant 170015),

LIST OF ACRONYMS

A.I	Artificial intelligence
P.G.M	Probabilistic Graphical Models
S.S.A	Seismic Signal Analysis
H.P.C	High-Performance Computing
B.P	Belief Propagation
C.M.G	Cascadic Multigrid
r.v	Random variable
P.D.F	Probabilistic Density Function
M.G	Multigrid
n.i	Network-isomorph
D.A.G	Directed Acyclical Graph
D.E	Differential Equations
M.R.F	Markov Random Fields
A.N.N	Artificial Neural Networks
C.N.N	Convolutional Networks
N.M.O	Normal Moveout
S.N.R	Signal-to-Noise Ratio
M.A.P	Maximum a Posteriori
P.S.N.R	Peak Signal-to-Noise Ratio
S.S.I	Structural Similarity Index
G.B.N	Gaussian Bayesian Network
K.M	Kubelka-Munk
C.P.D	Conditional Probability Distribution
M.L.E	Maximum Likelihood Estimation
K.D.E	Kernel Density Estimation
M.M	Mathematical Morphology
P.D.E	Partial Differential Equations
L.P	Linear Programming
B.L.P	Basic Linear Programming

LIST OF FIGURES

LIST OF TABLES

CONTENTS

0 INTRODUCTION

"Science and pseudo-science are the same thing."

Paul Feyerabend – Against method

High-dimensional structured or (raw) unstructured data poses great challenges in terms how they are processed and utilized. As technology advances, not only does our ability to collect more data, also the size of the data is increasing. For example in digital media, the contents are not only bulky but also rich in content. This creates challenges, for example, in content retrieval which (at scale) should be simple, fast and relevant. Another challenge arises with the increase in information is our ability to detect the true, underlying signal from the data.

One of the key challenges in high-dimensional image analysis and processing is how to do efficient processing and analysis operations. For time-critical applications, such as surveillance and (bio)medical applications, it is important to have methods which processes images fast while preserving an acceptable level of quality. Currently, there is a trend to approach image analysis and processing using *machine intelligence*. That is, to use methods which combines *statistical learning theory* (Vapnik (1998)) and *mathematical optimization* (Luenberger (1973)).

0.1 Research motivation

The field of *artificial intelligence* (A.I) can be characterized as a subfield of computer science[1], which aims at studying basic cognitive problem solving skills, or *intelligence*, of living organisms using machines. One of A.Is earliest field of study is the study of human perception in terms of the *neuronal mechanisms* of the eye (Anderson et al. (1988); Pitts and McCulloch (1947)) using computer programs. Today, this field is known as *computer vision*[2]. William James (1984) was the first

[1] Or computational neuroscience. Computer science is the educational background of this study's author so that's why the bias.

[2] Through out the study, we will refer to computer vision as **vision** for short.

to foreshadow some of the main ideas of neural networks, and Alan Turing was perhaps the first to consider building machines to emulate neuronal activities of the brain (Teuscher and Sanchez (2001)). However, it was the publications of McCulloch and Pitts (1943) and Rosenblatt (1958) which sparked the interest[3] in *neurocomputing*, which is a technological discipline on how to process information autonomously and adaptively given the information environment.

In essence, the goal of vision is to study *spatial, structural* aspects of image formation, transformations and recognition in order to explain the visual system of organisms (Johnson-Laird (1988)), by applying methods which *transforms* the image. The methods derived from vision research are applied in analyzing complex objects in digital images, where two difficult questions are confronted: the problem of object *description* and the problem of *quantifying* the description (Serra (1983)). For example, natural images can contain a combination of complex objects[4] which may or may not give a neutral descriptions. In order to answer these difficult questions on descriptions, images are transformed using *digital image processing* techniques (Jähne (2005)), which are derived from *digital signal processing* (Oppenheim (1978)). From these methods, we are able to study the spatial and structural properties occurring in images and then give descriptions of the occurring objects.

In this study, we examine ways to process and analyze spatial, structural aspects of images by applying *Probabilistic Graphical Models* (P.G.Ms). The use of P.G.Ms in vision were popularized by the brothers Geman and Geman (1984), where they experimented with how to define a priori solutions to vision tasks using stochastic modelling (Brémaud (2013)). There is also the case of researching suitable *transformation* steps of the inputs before feeding them to a neurocomputing model[5] (Anderson et al. (1988)). Structurally, P.G.Ms are a unified framework combining *network*[6] *theory* (Erdös (1959)) and *probability theory* (Kolmogorov (1983)), which offers interpretations from both *statistical mechanics* (Chalmond (2003)), A.I (Pearl (2014)), neurocomputing (Jordan et al. (2001); Hertz (2018)) and *statistics* (Jordan (2004); Lauritzen (1996)).

The interest in P.G.Ms in vision research lies in their generality: instead of applying ad-hoc approaches, P.G.Ms offers probabilistic approaches from both *Bayesian* and *Markov* formalisms for various tasks in dealing with images, such as denoising, edge detection and texture discrimination (Li (2009); Winkler (2012)). From this generality, there is a heavy price to pay, namely in *computation*. To process and analyze images with P.G.Ms, one requires to solve a potentially large collection of possible discrete states (Winkler (2012)). (With discrete states it is meant that of pixel values or objects in an image.) Certain types of P.G.Ms are

---

3 Minus the two "A.I. winters" and a possible third one incoming.

4 Forests, trees, mountains in scenery images; or in histology images (Figure 28, Section 3.3), the arrangements of cell nuclei and the texture of tissues.

5 This was based on biological and/or physiological arguments (Anderson et al. (1988)). This is known as *feature extraction*, which is a common procedure in pattern analysis (Fukunaga (2013)).

6 In many literature, *graphs* are commonly used instead of networks. In this work, the term network is used for not to be confused with *graph of a function*.

computationally intractable[7], while for P.G.Ms with binary states there exists methods which are *computationally tractable*.

The representation of a P.G.M depends on the application at hand. A Markov formalism is the prominent representation when dealing with vision tasks: the image is represented as an *undirected network*, where each node in the network represents some type of *statistical dependency* between nodes. To derive descriptive quantities of objects from and image, the description task is posed as an *energy minimization* problem: given a P.G.M as a *hypergraph* $\mathcal{H} = (\mathcal{V}, \mathcal{C}, \mathcal{L})$, where $\mathcal{V}$ is a set of vertices, $\mathcal{C}$ are the cliques in $\mathcal{H}$, and $\mathcal{L}$ is a set of labels. The optimization problem in a Markov formalism is formulated as a *combinatorial optimization*, where the aim is to find a *global solution* to the following energy function[8] E[9] : $\mathbb{Q}$[10] $\leftarrow \mathcal{L}^{|\mathcal{V}|}$, which is defined as

$$E(\mathbf{X}) = \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{c \in \mathcal{C}} f_c(x_c), \ f_i, f_{\geq} 0, \tag{1}$$

where f_i denotes the *unary potential energy* of the ith variable and f_c denotes the *interactions* between unary potentials in a set defined by $\mathcal{C}$. (This optimization problem will be elaborated in Section 2.1. Or the impatient reader may refer to Appendix 1.8.) The clique size $|\mathcal{C}|$ determines the *level* of interaction between nodes in $\mathcal{C}$, and has direct consequences on how easily we can solve the optimization problem. When $|\mathcal{C}| = 2$, we obtain a special case of first-order Markov network. And for $|\mathcal{L}| = 2$ we obtain a binary labelling case. For solving the special case, we can use for example Graph-Cuts, which are known to be computationally tractable (Boykov and Zabih (1999); Hadlock (1975)). For $|\mathcal{C}| > 2$, finding a global optimum becomes computationally *NP-Hard* (Boykov and Zabih (1999); Freedman and Drineas (2005)). Appendix 3.3 illustrates the severity of optimization problem.

To use a Bayesian representation of a P.G.M, a *directed network* is used, where the dependencies between nodes are explicitly assigned, given some background information of the application at hand. In this case, nodes represents *propositions* (Pearl (2014)), where we can do probabilistic queries. The nodes can have either discrete or continues states. A P.G.M with a Bayesian representation allows one to construct a different computational representation for Markov networks (such as Belief propagation (Pearl (2014); Felzenszwalb and Huttenlocher (2006))). Another way to use the Bayesian formalism is to use the network as an *expert system* (Jones and Graham (1988); Sotos (1990)), where instead of using *symbolic logic* to represent propositions we use *conditional probabilistic* propositions.

---

7 Or not solvable in polynomial "time" (Appendix 3.1)

8 The term *energy function* comes from statistical mechanics, which the P.G.M literature has adopted.

9 The normal notation for function mappings is $f : A \rightarrow B$, however here a "programming friendly" notation $f : A \leftarrow B$ is adopted. Our adopted notation reads: a function f is of (output) type A with B as its (input) source (Bird and De Moor (1996)).

10 Instead of using the domain of real numbers $\mathbb{R}$, we use the domain of rational numbers $\mathbb{Q}$. The reason being the inability of computers to model infinite precision numbers.

0.2 Aim of research work

This research work has two objectives:

1. To study how similarity measures used in *seismic signal analysis* (S.S.A) affect the energy minimization problem (1) on networks with arbitrary structures.
2. To describe image objects quantitatively using Bayes network in image analysis.

The first objective poses an *algorithmic* question:

Question 0.1. *The key algorithmic questions are*
– *How to approximate the energy minimization problem (1) without resorting to high-order representations in the energy term (that is, with* $|\mathcal{C}| > 2$*).*
– *For time-critical applications, the energy minimization problem (1) may be computationally slow. If one does not re-formulate the optimization problem, approaches from* high-performance computing *(H.P.C) should be used (Section 1.7). If one has a suitable energy function form, what speed-ups can we get from the method with H.P.C methods?*

The above algorithmic questions will be experimented on image denoising, an elementary vision problem. The approach taken here is to analyze and experiment how similarity measures affect the energy minimization problem using *probabilistic propagation* or *belief propagation* (B.P) using a *cascadic multigrid* (C.M.G) data structure (chapter 2). Particularly, we apply similarity measures from S.S.A (Yilmaz (2001)), which aims to locate signals from noisy seismic measurements, to augment the basic optimization methods for solving the energy minimization problem. To speed-up the B.P computation, we experiment with basic concepts from H.P.C on the C.M.G structure to evaluate computational bottle-necks and approaches.

The second objective poses an *application oriented* question. application oriented question, we experiment with the case of modelling physical parameters of skin layers, which are derived from a simulated light propagation model. The goal is to describe and probabilistically evaluate parameter dependencies in the form of probabilistic queries.

Question 0.2. *The key application questions are:*

– *How to effectively model the physical parameters which are obtained by using spectral imaging?*
– *What types of dependencies can we find from the obtained spectra using a Bayesian network model?*

0.2.1 Main results of research work

The main results of this research work are the following.

1. Additional insights on the effect of unary potential optimization f_i, specifically in B.P inference using C.M.G structure.
2. A proposal to evaluate probabilistically the modification effects of unary potential functions f_i.
3. H.P.C approaches to do computational speed-ups on the B.P inference approach with C.M.G structure. Existing H.P.C speed-ups are known in M.G literature, however not yet reported in the context of B.P inference.
4. Combining the dual nature of neurocomputing to probabilistically evaluate variable (or parameter) dependencies of imaging measurements.

While this research work focuses on P.G.M approaches in vision, it is important to note that the efficiency and advantage of a method has several dependencies[11]: the (concrete) task at hand, data structures and image models are some examples of such dependencies. Appendix 1 gives a short description of different image models and methods applied to image processing and analysis.

The Chapters are organized as follows. Chapter 1 gives a introduction to the necessary theoretical and technical aspects of this research work. Chapter 2 describes the B.P inference model and its S.S.A and H.P.C extensions. Chapter 3 covers the experimental section of this study, and finally conclusions and discussions are covered in Chapter 4.

[11] Also stated by the *No Free Lunch* theorem (Wolpert and Macready (1997)).

1 THEORETICAL & TECHNICAL BACKGROUND

> "I would like to understand things better, but I don't want to understand them perfectly."
>
> *Douglas R. Hofstadter – Metamagical Themas: Questing for the Essence of Mind and Pattern*

This chapter covers the necessary theoretical and technical backgrounds needed for chapters 3 and 4. The reader may skip Sections 1.1 and 1.2 if he or she has previous knowledge on the basics of vision research.

1.1 Processing visual information

A good practice when talking about vision is to give a crash course on theories of the human visual system, from which perception research in A.I is motivated.

1.1.1 Theories for perception

In psychological research, there are many theories on how organisms, particularly humans, visual perception is formulated[1], for example the *Gestalt* school of perception theory (Ehrenfels (1890)), or the *mathematical psychology* approach of visual perception (Leeuwenberg and Buffart (1978)). However, from a computational standpoint, David Marr's theory of perception (Marr (1982, 1980)) is more appropriate given the current computational capabilities of machines[2]. Note here, that there is a distinction between those which are *biologically plausible* mechanisms of perception and those mechanism which are more plausible

[1] More historical theories of perceptions are due to Newton, Hemholtz and Wertheimer, to name a few.

[2] To be fair and to point out, some say that Marr's vision model has failed due to inability to include natural constraints and therefore is not robust. See Warren (2012) for a related discussion. Also there is the limitation of the *encoding* paradigm used in current A.I and cognitive sciences in solving various intelligence problems (as intelligence is defined in this research) (Bickhard and Terveen (1996); Müller and Bostrom (2016))

for perceiving "artificially" generated images, such as *magnetic resonance imaging*, *positron emission tomography*, or *spectral* modalities.

Marr's computational model of vision. Marr theorized, that the stimulus, or input, to the visual system is given by the information in the optic array of the eyes. Processing the stimulus requires several, complicated stages by the visual system in order to detect the incoming information. After detecting the information received from the stimulus, the information is organized in such a way, that *feature representations* of the surroundings of the visual system will occur.

To explain the model of vision, Marr proposed three levels of explanations in terms of computation:

Computational theory : *what* kind of computational procedures are necessary for computing intensity changes in the visual system into output representations.

Algorithmic : *how* to define i) the input-output *relation*, ii) the *transformation* of the input to the required output, and iii) the *encoding* of the input.

Hardware : *what* is the *physical realization* of the algorithmic process.

From a computational view, the vision model processes the input in a sequence of processes with various sensory modules, where the visual information is first extracted from one representation, then organize the representations, and then making the organized representations explicit to be further used by other modules to compute other representations. This sequence is done until the "final" representation is reached.

A complementary theory of perception is given by Gibson (2002), stating that perception is based on the "correspondence" between some *invariant components* and the multiplicity of the visual system's stimulus. However, this theory will not be pursued any further in this study.

1.1.2 A computational approach to vision

For most living organisms, *the eye* functions as an "apparatus" to navigate in a surrounding, consisting of light sensitive elements (retina), converting the incoming light quanta into internal symbolic code. As the eye scans over the environment, the *representation* becomes explicit where the eye is in relation to its environment.

To make sense of how this representation if formed, Marr proposes a sequence of computational processes to reach the final output representation: the *primal sketch*, the $2D\frac{1}{2}$ *sketch*, and the $3D$ *object model*.

The primal sketch. This early stage vision process seeks for a representation which describes light intensity changes all over the retina. Different changes in intensities facilitates in identifying, for example, edge points, edge segments or small blobs. Detecting intensity changes do not determine what precisely causes the intensity change, so additional information is needed to solve this ambiguity problem. The additional information is given by the *grouping rule*: similar

structures are grouped together using Gestalt principles of *proximity, continuity, closure, etc*[3].

The primal sketch stage has no notion of "things". Despite the grouping rule, it is misguided to say that intensity changes are *inferred* by correlated object structures. If this were the case, then computing the early stage representation would mean that the computed correlated structures are *built-in* features, ensuring said correlation. Additionally, the *same* correlation is *inferred* by the visual system. The notion of "things" is computed in the second sequential stage.

$2D\frac{1}{2}$ **sketch.** This sketch provides information on surface layouts, implicitly solving an "image segmentation" problem, that is, making discontinuities between surfaces and objects more explicit. Various computational models combines the results from the primal sketch, and are combined into a $2D\frac{1}{2}$ sketch. This step acts as a short-term memory for other computational steps of *stereopsis*, that is depth perception.

$3D$ **object model.** A $3D$ object model is then constructed from the shape and relative distance representations of the $2D\frac{1}{2}$ sketch. This object model constructs a representation of object which then can be identified and recognize as *shapes of particular objects*. The construction is done in a formal or *grammar* system, where rules for constructing computed representations from the $2D\frac{1}{2}$ sketch. The identification of object is assumed to be done using top-down procedures and information.

1.2 Principles of image processing

Image processing is a field of converting an image, which is captured using a suitable acquisition system (for example a camera), into a new image (Jähne (2005); Horn (1977)). After an image has been captured, suitable processing techniques are implemented for further usage of the image. For example, correcting image quality by reducing noise, adjusting brightness and/or contrast, and restoring geometrical distortions. Vision research depends heavily on the methods and theories developed in image processing, because where image processing offers elementary processing tasks, vision offers methods how to combine these basic elementary processing tasks to perform mechanized perception with machines. For example, object recognition from a variety of image scenes, do image interpretation and help self-driving cars to navigate through its environment.

[3] For example, combining edge points which object boundaries are assumed to be continuous.

1.2.1 Image representation

How information is represented in an image can be done in different ways. However, the most important way of representing information is using spatial or wave number representation. These two representations are equivalent and complete, meaning that we can convert a spatial representation into a wave number representation, and vice versa. For example, *Fourier transform* can be used to convert spatial representation into a wave representation. We will limit our discussion to spatial representation and omit the discussion of wave number representation because it is not relevant in this study. See Jähne (2005) for discussions on wave number representations of digital images.

Spatial representation is a way of representing image information using a spatial distribution of *pixels*, which are spatial elements of an image. This spatial distribution constitutes the *irradiance*[4] at a plane. Mathematically, the spatial distribution can be described as a continuous function of two spatial variables x_0, x_1:

$$u(x_0, x_1) = u(\vec{x}). \tag{2}$$

Computers cannot intrinsically handle continuous representations of images. A "natural" way to represent images in a computer is to use a rectangular array of digital numbers (pixels), which is a convenient way of manipulating and notating an array of images and its elements, using matrix notation. Each element in an image represents a rectangular region with an associated value, giving the average irradiance of the corresponding spatial location. Image arrays can be represented in 2D or 3D spaces, depending on the application.

To analyze the properties of an image, *neighbourhood relations* is an important property. This is because, by analyzing neighborhood relations, we can define connected regions. When dealing with 2D rectangular images, there are two elementary ways to analyze neighborhood relations: a *4-neighbourhood*, where pixels have a joint edge; or *8-neighbourhood*, where additional to a joint edge, the pixel also has a joint corner neighbour (see Figure 1). For 3D rectangular images, neighborhood relations become more complex: additionally to joint edges (*18-neighbors*) and joint corners (*26-neighbors*), we also need to define joint faces (*6-neighbors*).

Discrete geometry

Because of the discrete nature of digital images, basic elementary geometrical properties (for example, distance, slope of a line) and coordinate transforms (for example, translation, rotation and scaling) need to be redefined to accommodate object definitions and geometric parameter estimations. We will use a *grid vector* definition for representing a position of a pixel in a digital image.

A grid vector in a 2D or 3D spatio-temporal image can be defined as

[4] The radiant flux received by a surface per unit area.

$$\vec{r}_{m,n} = [n\Delta x, m\Delta y]^T \tag{3a}$$

$$\vec{r}_{m,n,k} = [n\Delta x, m\Delta y, k\Delta z]^T, m, n, k \in \mathbb{Z}_+, \tag{3b}$$

where $\mathbb{Z}_+$ is an integer lattice and $\Delta x, y > 0$ is a positive step in the grid. In order to measure distances in digital images, we redefine the distance metric into a discrete setting using the grid vector definition. For example, the *Minkowsky distance* metric in a discrete setting becomes

$$d_l(\vec{r}, \vec{r}') = \| \vec{r} - \vec{r}' \|_l = \left[(n - n')^l \Delta x^l + (m - m')^l \Delta y^l \right]^{1/l}, l \geq 1. \tag{4}$$

Setting $l = 1, l = 2, l = \infty$, we get the well known *City block*, *Euclidean* and *Chessboard* distances respectively. For (most) practical applications in digital images, the Euclidian distance is the most relevant, because the Euclidian distance preserves the isotropy of the continuous space.

Scaling digital images can be done, for example, with integer multiples of the chosen scaling factor. That is, taking every ith pixel on every jth line. This discrete scaling operation acts as a subsampling approach on an image grid.

In general, there is no clear and correct representation, even for the simplest geometric objects, in digital images. For example, lines can be represented only for values with multiples of 45°. In any other directions, the lines will appear as jagged, staircase-like sequences of pixels.

Quantization

The process of mapping the measured irradiance at the image plane into a rectangular image is called *quantization*. Usually quantization is limited to a number of Q discrete gray values. Greater the quantization number, finer the image will become. If the quantization level is too low, this will produce, for example, false edges and it will be difficult to recognize spatial variations between objects. However, quantization will always produce errors when mapping a continuous image into a discrete form.

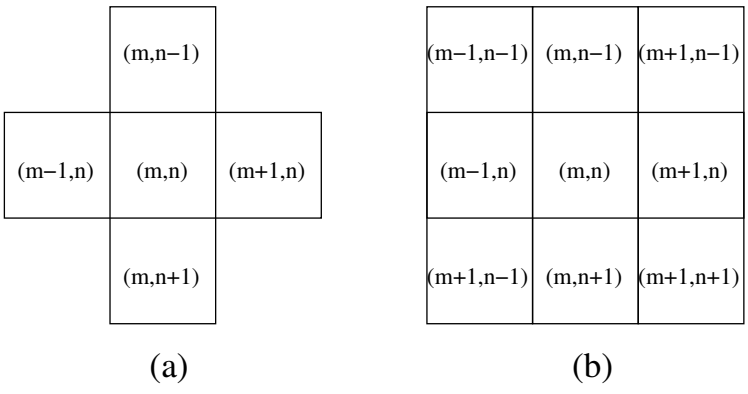


FIGURE 1 (a) $4-$ neighbourhood system.; (b) $8-$ neighbourhood system. (Jähne (2005)).

The level of quantization is dependent on the application domain. Usually the image is quantized into 256 gray values, making each pixel occupying 8bits (or one byte). This quantization approach is well suited for processing images with modern computer architectures. For some computer vision applications, a two level quantization (a binary image) could be sufficient if dealing with homogeneously illuminated objects. Applications in, for example, imaging spectroscopy or medical diagnosis with X-ray images, the standard 8-bit quantization could be too coarse, because of the faint changes in intensity values of objects.

Digitization

When a suitable level of quantization has been chosen, the discrete set of pixel points are sampled. This process is called *digitization*, which may occur when an imaging sensor collects photons into an electric signal or afterwards, as in video signal processing.

There are several crucial questions regarding the validity of sampled points when a continuous image is digitized into a discrete digital image (no loss of information), and what are the limitations and methods on doing the sampling. For example, digitization of fine structures may suffer from distortions due to different periodicity and direction of sampled grid points (*Moiré effect*) or sinusoidal oscillation (*aliasing*) when dealing with 1D signals. From these observed phenomenons, we can formulate conditions under we can do correct sampling and construct a complete representation of a continuous image in discrete form. This formulation is known in image (and signal) processing as the *sampling theorem*.

Image formation.

Let $u(\vec{x})$ be an infinite, continuous image and $g(\vec{x})$ be a resulting image from mapping $u(\vec{x})$ into a discrete space. (We consider now only 2D representations but the principle applies to 3D representations too.) In the process of digitization, the image $g(\vec{x})$ is sampled at certain points from the grid vector $\vec{r}_{m,n}$.

Instead of collecting the illumination intensity exactly from the grid points, the intensity values are collected in a certain area around the grid points. For an idealized *charge-coupled device* camera, which collects the photons with a matrix of photodiodes without light-sensitive strips in between, the signal at the grid points $\vec{r}_{m,n}$ is the integral over the area of the individual photodiodes:

$$g(\vec{r}_{m,n}) = \int_{(m-1/2)\Delta x_1}^{(m+1/2)\Delta x_1} \int_{(n-1/2)\Delta x_2}^{(n+1/2)\Delta x_2} u(\vec{x}) d(x_1, x_2), \tag{5}$$

assuming that the photodiodes are equally sensitive over the whole area.

Operation (5) includes *convolving* with a rectangular box function and sampling the grid points. By first applying continuous convolution and then performing sampling, the image formation can be generalized and perform the sampling step separately. In general, the process of image formation results in blurring the fine details, causing the gray intensity values to be *band-limited*.

Sampling

When an image is sampled, the grid points are the locations where the information is stored during image formation. The *sampling theorem* tells us how we can avoid distortions in the signal: by restricting the image spectrum around the area that extends around the grid point up to the lines parting between all other grid points. By doing so, we can separate the spectral amplitudes that originate from the original spectrum at the central grid point or one of its copies. In practice, this means that sampling restricts the number of wave numbers and thus the spatial resolution of the image, because we can only work with finite image matrices.

1.2.2 Random variables

After an image is captured, we want to analyze the objects it contains (positions, shapes, etc.). This can be done by measuring, for example, the position of the object or its reflectance properties. However, the obtained measurements are useful if we can also measure its uncertainty. There are two important statistical classes used in digital image processing. *Statistical error* describes how scattered the measurement is when the *same* measurement is repeated over and over again. From this, we obtain a distribution, from which the width of the distribution tells us the statistical error of the measurement, whilst the centroid of the distribution gives us the mean value of the measurements. The second class is the called *systematic error*, which tells us how much the true value deviates from the mean value. Systematic errors are much harder to control compared to statistical errors. The reason is that they are often caused by the lack of knowledge and understanding of the measuring setup and procedure. If the parameters are unknown or uncontrolled during the measurement, this could easily lead to systematic errors.

To treat image data statistically, we treat the data as a *random variable* (r.v). In the simplest case, digital image processing operations operate on a single point, that is using *point operations* (These will be covered more in the next Section below.) A more elaborate operations are done a group of points in order to produce new quantities, for example, a time series of random variables or a spatial array of random variables.

As previously mentioned, when imaging objects with an imaging device, the imaging device measures quantity called *irradiance* at a certain point in the image plane. This observed process is statistical in nature, so that each irradiance measurement will give a different value. Instead of measuring each single value from the object, a *probabilistic density function* (P.D.F) is used to measure and quantify the observed irradiance. Each measurable quantity in this stochastic process is a r.v, and is discretized with respect to the chosen quantization level.

In general, we do not know the probability distribution before hand, and we cannot represent individual r.vs with individual P.D.Fs. To study the random properties of r.vs, a *joint probability density function* is required. For cases where r.vs are independent, we can study the resulting independent P.D.Fs by estimating their *marginal P.D.Fs* (see Appendix 2.1 for formal definitions). Devel-

oping statistics of an image considering *both* spatial and temporal relationships between points, *random fields* are used. In vision literature, random fields are known as P.G.Ms and they will be presented in Section 1.4.

1.2.3 Operating on pixels

Recognition of objects in images is done by analyzing spatial relationships of pixel values in a (small) *neighborhood*. By examining pixel values inside a neighborhood, we can determine does a particular set of values belong to an object or to an edge. However, when processing images with these neighborhood operations, information is generally lost because the neighborhood operations "create a new image" inside the neighborhood (Jähne (2005)). The shape of the neighborhood can be of any form, but a common starting point is to choose a $(2k + 1) \times (2k + 1)$, $k > 0$ neighborhood size and operate on the center pixel of the chosen neighborhood. Even-sized neighborhoods are discouraged because they do not have a center pixel, causing a shift in pixel distances between the original pixels and the pixels in the neighborhood. If these results would be added to the original pixel values, considerable errors in pixel values will occur.

In contrast, *point operations* are used for analyzing single pixels in an image, and are useful as a first step procedure, in order to correct heterogeneous and nonlinear responses from the imaging device. Some examples of point operations are contrast enhancement and computing average values. Complementary operation to point operations is *geometric operations*, where a pixel is located into a new position. Point operations can be divided into two types of operations:

1. *Homogeneous point operations*, where a point operation is independent with respect to pixel location. This operation is non-invertible, because two different pixel values may be mapped onto one value (for example, threshold operations).
2. *Heterogeneous point operations*, where a point operation also depends on the location of the pixel. These operations are computationally more time consuming and are used, for example, doing calibration operations. Image averaging is an example of the simplest heterogenous point operation.

1.2.4 Multiscale data structures

Neighborhood operations are the main starting point in image analysis. By definition, neighborhood operations operate on a local level or *local scale*, allowing to extract pixel features from a small set of distances. To go beyond local scale and to obtain feature information from greater distances, we need to extract these features from a *larger scale*. A naïve approach would be to use a larger neighborhood, but this results in an increase in computational cost[5].

The challenge posed by different image resolutions, consider the task of detecting lines. For high resolution images, local distances between pixels will be

[5] For example, using a neighborhood size $\mathcal{N}^W$, $W \in \mathbb{Z}_+$, the number of operations needed in $\mathcal{N}$ is proportional to $\mathcal{N}^W$.

dominated by the noisy background of the image. Detecting lines, in this case, will be a challenge because of the contrast between objects and background is inaccurate and erroneous. That is, we are dealing with *scale mismatch*. Larger scales should be used to capture the varying pixel values properly. In the low resolution case, lines would be blurred such that the contrast between lines will decrease proportionally to the used image resolution. With these observations in mind, it would be reasonable to consider detecting objects using suitable scale representations.

To this end, *multiscale data structures* (or *multiscale representations*) are designed to capture object features from different scales. An efficient data structure for processing images in multiple scales is the *multigrid*[6] (M.G) representation. The idea is to represent the image, such that, the fine scale (level) in the M.G structure is the full resolution of the image, while the lower resolution of the image is represented in the coarsest level. With a M.G representation, speed ups can be obtained for many image processing methods.

In Section 2.1 will demonstrate how a *cascadic M.G* (Appendix 3.4) representations can be used for solving optimization problems with B.P.

1.3 Probabilistic graphical models

P.G.Ms are useful probabilistic models, which enable the capturing of dependencies of r.vs in a graphical way, where the nodes of the network are treated as propositions of variables (Pearl (2014)). Given a certain set of initial independence relationships and certain axioms (discussed below), new independencies can be inferred using non-numeric, logical manipulations. From these certain axioms we can identify different structural properties that can be captured using a graphical representation. There are two main graphical representations: undirected representations, known as *Markov networks* (Section 1.3.2), and directed representations, known as *Bayesian networks* (Section 1.3.3). We will first discuss axioms used to identify structural properties of P.G.Ms, and then discuss more about the two network representations, which can be used for probabilistic inference. This Section follows Pearl (2014) unless otherwise stated.

1.3.1 Set of axioms for graphical representation

Motivation. In order to do probabilistic reasoning, basic textbooks on probability theory gives the impression that one must *literally* construct a joint distribution function $P(x_0, \dots, x_{n-1})$ over all propositions and their combinations. The problem of this approach is that even for moderate size n, we would have to store an arbitrary $P(\dots)$ into a table with 2^n entries. This leads to storage requirements which are unreasonable and uneconomical. Additionally, computing this table would be cumbersome. (The same pitfalls occurs when computing conditional

[6] Multigrids are also efficient numerical solvers for partial differential equations.

probabilities, which means dividing two marginal distributions.)

How then does one solve the computation of the joint distribution function $P(\dots)$ over all propositions and its combinations? This can be done by how the notion of *independence* is defined in P.G.Ms. Numerically speaking, the notion of independence uses the equality $P(x, y) = P(x)P(y)$ given propositions x and y. That is, we should test are the joint distributions X and Y independent, from which propositions x and y are drawn. Things become even more complicated when we are introduced additional propositions from additional joint distribution(s). Explicitly encoding the dependencies would be unreasonable because the number of evaluated combinations will explode extremely fast, especially when encountering new evidence or data. *Conditional independence* could provide informational relevance qualitatively by intuitively capturing how dependencies between propositions should change when presented with new evidence. This however, still requires checking equalities numerically.

To avoid numerically verifying equality between propositions, a structure in the form of a *dependency network* helps facilitating simple and local operations, which can be applied to propositions. Networks are a common metaphor for conceptual dependencies. The type of information relevance and probabilistic dependencies is determined by the network topology[7].

The axioms. Let $\mathbb{U}$ be a finite universe of discrete random variables, where each r.v may take values from a finite domain $\mathbb{W}$. We will denote a r.v in boldfaced, uppercase ($\mathbf{X}$) when we mean a partition[8] of variables in $\mathbb{U}$, and a boldfaced, lowercase ($\mathbf{x}$) with a *configuration* of variables taking values from a partition. For any given $\mathbf{X} \in \mathbb{U}$, we will denote $\mathbb{W}_{\mathbf{X}}$ as the domain of values $\mathbf{X}$ may take.

In order to express dependency relationships between partitions and their configuration, we need the following definition on conditionally independency.

Definition 1.1. *Given a finite and discrete universe of* $\mathbb{U}$ *and a joint probability function* $P(\cdot)$ *over the variables of* $\mathbb{U}$*. Let* $\mathbf{X}$,$\mathbf{Y}$ *and* $\mathbf{Z}$ *be partitions of* $\mathbb{U}$*.* $\mathbf{X}$ *and* $\mathbf{Y}$ *are said to be* conditionally independent given $\mathbf{Z}$*, if*

$$P(\mathbf{x} \mid \mathbf{y}, \mathbf{z}) = P(\mathbf{x} \mid \mathbf{z}), \tag{6}$$

when $P(\mathbf{y}, \mathbf{z}) > 0$*.*

The above independence relation between $\mathbf{X}$ and $\mathbf{Y}$, given $\mathbf{Z}$ will be denoted as $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$.

Now without referencing to numerical forms, are there conditions that should constrain the relation $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ for some distribution P? Let us first state the following axiom.

Axiom 1.1. *Given disjoint partitions* $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subset \mathbb{U}$ *and* $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ *for a distribution* P*. Then* $I(\cdot)$ *must satisfy the following independent conditions:*

---

7 When discussing networks, topology and structure are used interchangeably.

8 Here partitions are referred as *subsets*.

- *Symmetry:* $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \iff I(\mathbf{Y}, \mathbf{Z}, \mathbf{X})$
- *Decomposition:* $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W}) \implies I(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \wedge I(\mathbf{X}, \mathbf{Z}, \mathbf{W})$
- *Weak union:* $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W}) \implies I(\mathbf{X}, \mathbf{Z} \cup \mathbf{W}, \mathbf{Y})$
- *Contraction:* $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \wedge I(\mathbf{X}, \mathbf{Z} \cup \mathbf{Y}, \mathbf{W}) \implies I(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$.

If $P > 0$, then we have the additional condition:

- *Intersection:* $I(\mathbf{X}, \mathbf{Z} \cup \mathbf{W}, \mathbf{Y}) \wedge I(\mathbf{X}, \mathbf{Z} \cup \mathbf{Y}, \mathbf{W}) \implies I(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$.

The above axiom states the following: by *symmetry*, the knowledge of **Z** at any state, if the knowledge of **Y** does not give anything *new* about **X**, then knowing **X** does not give us anything new about **Y**. *Decomposition* asserts that if combining two propositions $\mathbf{Y} \cup \mathbf{W}$ are irrelevant to proposition **X**, then the propositions **Y** and **W** are irrelevant **separately** to **X**. *Weak union* tells us that learning an irrelevant proposition **W** will not make an irrelevant proposition **Y** to become relevant to proposition **X**. *Contraction* states that if proposition **W** is irrelevant to **X** after learning about irrelevant proposition **Y**, then **W** is irrelevant before learning about proposition **Y**. Taking the weak union and contraction axioms together, they mean that irrelevant propositions should not alter the relevance of other propositions. The *intersection* condition tells us, that if **Y** affects **X** when **W** is constant, or **W** affects **X** when **Y** is constant, then neither propositions **W** or **Y** (or their combination) has an affect on **X**.

When we want probabilistic formulations of dependencies between propositions, Axiom 1.1 will be enough. However for qualitative formulations of dependencies between propositions we need the following extra properties from the following conjecture from Pearl and Paz (1985).

Conjecture 1.1. *The symmetry – contraction properties from Axiom 1.1 are said to be* complete *if $I(\cdot)$ is interpreted as a conditional independence relation. That is, for a distribution P*

$$\exists P \text{ such that} P(\mathbf{x}|\mathbf{y}, \mathbf{z}) = P(\mathbf{x}|\mathbf{z}) \iff I(\mathbf{X}, \mathbf{Z}, \mathbf{Y}).$$

If the intersection property is also satisfied in Axiom 1.1, then $\exists P \geq 0$ satisfying the above relation.

By doing the above axiomatizaion of probabilistic dependencies offers the following benefits.

1. Powerful theorems can be derived and conjectured, which may or may not be obvious when using numerical representation of probabilities, for example, the chaining rule for $I(\cdot)$ (Lauritzen (2002)) (direct consequence of the properties of symmetry – contraction in Axiom 1.1).
2. We can derive new independencies from the initial set of propositions as a rule for qualitative inference. For example, initially given a set of qualitative independence propositions $\mathcal{A}$, we can test wether a new set of independence propositions $\mathcal{A}'$ are a "consequence" of the initial set of propositions. In principle, this type of query may be computationally undecidable[9]

[9] See Languages (Appendix 3.1)

(Beeri (1980)), because in order to check this consequence, we would have to check from an infinite number of possible distributions. However, if we use proper axioms, we can derive $\mathcal{A}'$ straight from $\mathcal{A}$ without resorting to search from an infinitely large space. This is to say, that we can decide if $\mathcal{A}'$ follows from $\mathcal{A}$ by finding a complete set of axioms for conditional independence.

3. The axioms provide a convenient way to comparing features of several dependency formalisms.

How to represent then the dependencies or independencies of propositions in a graphical way? Nodes in the network can be linked either in an undirected or directed manner. To capture the interactions between propositions in a network, a natural way to illustrate interactions would be to add an *edge* between interacting propositions. The lack of interaction between propositions would be the absence of an edge. This way of capturing interactions between propositions deprives the expressive power given by the graphical representation.

The reasoning being, that illustrating the interaction of propositions in such a rigid requirement, all connected propositions in the network are treated as *equal*, robbing any special meaning to the structure of each connected proposition. In order to properly exploit the expressive power of presenting proposition interactions in a graphical way, a *semantic* distinction should be made between *direct* (propositions directly interacting through an edge) and *indirect* (proposition interaction through a *mediator* proposition). By this semantic distinction of the network topology, the edges between propositions reflects a *conditional* interaction. In this way, the interactions may become stronger, weaker, or non-existent.

Before reviewing different types of graphical representation of propositions, we will touch on dependency models and dependency maps. A *dependency model* determines the truth value of a partition $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$. Using equation (6) to verify the validity of $I(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$, then any joint probability distribution P is a dependency model. That is, equation (6) tests does a partition of propositions $\mathbf{Z}$ *intervenes* in a relation between propositions $\mathbf{X}$ and $\mathbf{Y}$. Our goal is then to characterize the connection between propositions in a network which is used to encode these interventions.

To relate a graphical representation of a dependency model $\mathcal{M}$, what we are looking for is a correspondence between elements of $\mathbb{U}$ of $\mathcal{M}$, and a set of propositions in a network. This correspondence should reflect some properties of $\mathcal{M}$ from the topology of the used network. To discuss more about this correspondence, let $G \stackrel{\text{def}}{=} (\mathbf{V}, \mathbf{E})$ be an undirected[10] network, where $\mathbf{V}$ is a set of nodes in the network and $\mathbf{E}$ is the set of edges connecting the nodes in G.

Definition 1.2. *If there is an injective correspondence between the elements* $\mathbb{U}$ *and the set of nodes* $\mathbf{V}$ *in* G*, we say* G *is a* dependency map *(D-map) of a dependency model* $\mathcal{M}$*. That is, we have*

$$I(\mathbf{X}, \mathbf{Z}, \mathbf{Y})_{\mathcal{M}} \implies < \mathbf{X}|\mathbf{Z}|\mathbf{Y} >_G . \tag{7}$$

[10] Unless stated otherwise, when talking about G, we mean an undirected network.

G is an independency map *(I-map) of a dependency model $\mathcal{M}$ if*

$$I(\mathbf{X},\mathbf{Z},\mathbf{Y})_{\mathcal{M}} \impliedby < \mathbf{X}|\mathbf{Z}|\mathbf{Y} >_G . \tag{8}$$

$< \mathbf{X}|\mathbf{Z}|\mathbf{Y} >_G$ is read as, a partition of propositions $\mathbf{Z}$ intercepts all paths between propositions $\mathbf{X}$ and $\mathbf{Y}$. The correspondence given by Definition 1.2 provides a graphical representation of $\mathbf{X}$ not affected by $\mathbf{Y}$ (directly), with $\mathbf{Z}$ being in between them.

Equation (7) says that all the nodes which are connected in G are dependent in $\mathcal{M}$[11]. Equation (8) then says that nodes to be separated correspond to independent propositions. However, this does not guarantee that all those nodes which are shown to be connected are dependent.

The weakness of undirected representation of propositions is that the ability to represent informational dependencies is limited. An example of this would be that a model $\mathcal{M}$ could make unrelated propositions in G become relevant to each other when we obtain new information. This is to say (in technical terms) that G would have a graphical representation with both a D-map and an I-map, which is a contradiction. In order to overcome this problem, we need classes of $\mathcal{M}$ that are able to characterize a family of graphical representations which are isomorphic to node separation in G. This characterization is given by the following definition.

Definition 1.3. *Let $G = (\mathbf{V},\mathbf{E})$, $\mathcal{M}$ a dependency model of G, $\mathbf{X},\mathbf{Y},\mathbf{Z} \subset \mathbb{U}$ disjoint partitions. If for every $\mathbf{X},\mathbf{Y},\mathbf{Z}$ we have*

$$I(\mathbf{X},\mathbf{Z},\mathbf{Y})_{\mathcal{M}} \iff < \mathbf{X}|\mathbf{Z}|\mathbf{Y} >_G, \tag{9}$$

then $\mathcal{M}$ is said to be a network-isomorph *(n.i) of G.*

A necessary and sufficient condition for n.i is given by Pearl and Paz (1985):

Axiom 1.2. *In order a dependency model $\mathcal{M}$ to be a n.i of G, a necessary and sufficient condition is that the relation $I(\mathbf{X},\mathbf{Z},\mathbf{Y})_{\mathcal{M}}$ satisfies the following independent axioms*

- *Symmetry:* $I(\mathbf{X},\mathbf{Z},\mathbf{Y})_{\mathcal{M}} \iff I(\mathbf{Y},\mathbf{Z},\mathbf{X})_{\mathcal{M}}$.
- *Decomposition:* $I(\mathbf{X},\mathbf{Z},\mathbf{Y})_{\mathcal{M}} \cup \mathbf{W}) \implies I(\mathbf{X},\mathbf{Z},\mathbf{Y})_{\mathcal{M}} \wedge I(\mathbf{X},\mathbf{Z},\mathbf{W})_{\mathcal{M}}$.
- *Intersection:* $I(\mathbf{X},\mathbf{Z}\cup\mathbf{W},\mathbf{Y}) \wedge I(\mathbf{X},\mathbf{Z}\cup\mathbf{Y},\mathbf{W}) \implies I(\mathbf{X},\mathbf{Z},\mathbf{Y}\cup\mathbf{W})$.
- *Strong union:* $I(\mathbf{X},\mathbf{Z},\mathbf{Y})_{\mathcal{M}} \implies I(\mathbf{X},\mathbf{Z}\cup\mathbf{W},\mathbf{Y})$
- *Transitivity:* $I(\mathbf{X},\mathbf{Z},\mathbf{Y})_{\mathcal{M}} \implies I(\mathbf{X},\mathbf{Z},\gamma)_{\mathcal{M}} \vee I(\gamma,\mathbf{Z},\mathbf{Y})_{\mathcal{M}},\ \gamma \in \mathbb{U}$.

With Definition 1.3 and Axiom 1.2, we can test is a model $\mathcal{M}$ a valid graphical representation. For convenience for using probabilistic models, I-maps (equation (8)) are considered when using P.G.Ms (Pearl (2014)) because probabilistic models may fail both Strong union and Transitivity axioms in Axiom 1.2[12].

[11] It is possible for a pair of dependent propositions to be displayed as a pair of separate nodes.

[12] For example, using an unfair coin would violate the Transitivity axioms.

1.3.2 Markov networks

From the dependency axioms stated in Section 1.3.1, given a joint probability distribution P and a network G, how do we test is G an I-map of P? *Markov networks* provides tools for this question. The solution to this question is to find a unique and *minimum* set of edges in G for strictly positive P. This is done by searching how each proposition in a network influences each other, given a dependency model. That is, we are searching for a set of propositions that form a *Markov boundary*, which in turn induces a strictly positive distribution of *neighborhood systems*:

Definition 1.4. *Given an element $u \in \mathbb{U}$, a* Markov blanket *is a partition $\mathbf{S}$ of $\mathbb{U}$, such that*

$$I(u, \mathbf{S}, \mathbb{U} \setminus \mathbf{S} \setminus \{u\}), u \notin \mathbf{S}. \tag{10}$$

If a Markov blanket is minimal of an element u, that is, none of the proper partitions of a Markov blanket satisfies equation (10), then it is called a Markov boundary $\mathbf{B}_\partial$.

If every element $u \in \mathbb{U}$ and the I-map of a dependency model satisfies Axiom 1.1., then each u has a unique $\mathbf{B}_\partial$ and the I-map is minimal (Pearl and Paz (1985)). The elements of $\mathbf{B}_\partial$ are adjacent to a minimal I-map. This property allows two identical interpretations of *direct neighborhoods*.

1. The neighborhood shields the element u from the influence of other elements.
2. The neighborhood *binds* the elements in the neighborhood, such that the connection between the elements cannot be weakened by other elements in the system.

The Definition 1.4 helps making a Markov network into a probability distribution with following definitions:

Definition 1.5. *Given a probability distribution $P > 0$, the Markov blankets creates* neighborhood systems *of the elements of P. That is, given a collection of Markov blankets $\mathbf{B}^*_\partial \stackrel{def}{=} \{\mathbf{B}_\partial(u) \ : \ u \in \mathbb{U}\} \in \mathbb{U}$, for a pair of elements $(u, w) \in \mathbb{U}$, we have*

1. $u \notin \mathbf{B}_\partial(u)$, *and*
2. $u \in \mathbf{B}_\partial(w) \equiv w \in \mathbf{B}_\partial(u)$.

Definition 1.6. *A network which is I-minimal, has a strictly positive distribution, and can be constructed by connecting all elements u to all elements of its Markov boundary $\mathbf{B}_\partial(u)$, is called a* Markov network.

To test does a Markov network have an I-map, it suffices to check for the Markov boundary condition.

Markov networks captures the essential qualities of Axiom 1.1 regarding conditional independence, and is essential for constructing undirected networks. The properties of symmetry, decomposition and intersection allows to construct a minimal Markov network, which then offers an inference mechanism for deducing propositions. Figure 2 shows a simple Markov network with four propositions.

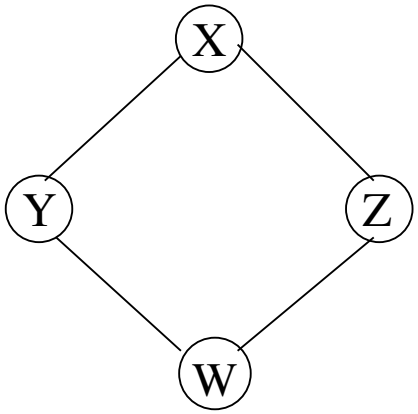


FIGURE 2 A simple *Markov network* with four propositions (Pearl (2014)).

1.3.3 Bayesian networks

Markov networks are capable of modelling the conditional independence between propositions, but fail to represent induced and non-transitive dependencies between propositions. *Bayesian networks* allows to make *explicit* and directional dependencies between propositions.

Bayesian networks tries to answer similar questions what Markov networks tries to address, by adding a *directed acyclical network* (D.A.G)[13] constraint into the question:

1. Given a joint probability distribution P, how can we construct an edge-minimal D.A.G, which is an I-map of P?
2. Given P and D.A.G D, how can we test that D is a (minimal) I-map of P?
3. Given a D.A.G D, can we construct a P, such that D is a perfect map of P?

Similar to Markov networks, the notion of conditional independence (Axiom 1.1), is used to construct I-maps. However, because of the directional dependencies between propositions, the intersection axiom in Axiom 1.1 is no longer a strict requirement to derive local dependencies in a network.

D.A.Gs allows us to inspect which set of propositions are considered to be independent. Directional edges are used to illustrate direct causal influences between propositions, and the strength of these edges are expressed by a forward conditional probability.

Bayesian networks have a more complicated separability criteria compared to Markov networks. As a reminder, if the removal of a proposition $\mathbf{Z}$ makes the propositions $\mathbf{X}$ and $\mathbf{Y}$ no longer connected to each other, then $\mathbf{X}$ and $\mathbf{Y}$ are independent of each other given $\mathbf{Z}$. In Bayes networks, the separation criteria is called a *d-separation* criteria:

Definition 1.7. *Given partitions* $\mathbf{X}, \mathbf{Y}$ *and* $\mathbf{Z}$ *which are disjoint in a D.A.G* D*, then* $\mathbf{Z}$ *d*-separates $\mathbf{X}$ *from* $\mathbf{Y}$*. Conditions for d-separation are:*

1. *Every proposition has a descendent in* $\mathbf{Z}$*.*

[13] For this acronym, the "traditional" version is used.

2. *Every other proposition is outside of* $\mathbf{Z}$.

If the d-separation condition holds for a path, then that path is called *active*. The path of arrows represents predicted events and are considered blocked activated by evidental information. Figure 3 shows a simple Bayesian network with three proposition.

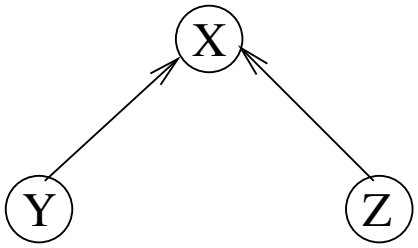


FIGURE 3 A simple *Bayesian network* with three propositions (Pearl (2014)).

Definition 1.8. *A D.A.G D is an I-map of a dependency model $\mathcal{M}$, if every d-separation condition in D correspond to a valid conditional independence relationship in $\mathcal{M}$. That is, if for disjoint propositions $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ we have*

$$< \mathbf{X}, \mathbf{Z}, \mathbf{Y} >_D \implies I(\mathbf{X}, \mathbf{Z}, \mathbf{Y})_{\mathcal{M}}, \tag{11}$$

where $< \mathbf{X}, \mathbf{Z}, \mathbf{Y} >_D$ is the d-separation of given propositions.

Definition 1.9. *Let P be a joint probability distribution of $\mathbb{U}$. The the D.A.G $D = (\mathbb{U}, \mathbf{E})$ is a* Bayesian network *of P, if and only if D is a minimal I-map of P.*

We can construct a Bayesian network in the following way:

Definition 1.10. *Let $\mathcal{M}$ be a dependency model defined over elements of $\mathbb{U}$. The* boundary strata *of $\mathcal{M}$ is a relative ordering of d partitions Markov boundaries $\mathbf{B}_{\partial_i}$, $i = 0, \ldots, d-1 \in \mathbb{U}$. Each Markov boundary is a minimal set, satisfying $\mathbf{B}_{\partial_i} \subseteq \mathbb{U}_i$, and $I(X_i, \mathbf{B}_{\partial_i}, \mathbb{U}_i \setminus \mathbf{B}_{\partial_i})$, $X_i \in \mathbb{U}, \forall i = 0, \ldots, d-1$. The D.A.G D is then created by assigning each Markov boundary (as a parent) to each X_i, and is called a* boundary *D.A.G of $\mathcal{M}$ relative to the ordering d.*

The key for constructing a Bayesian network is that the resulting D.A.G should satisfy Conjecture 1.1. That is, if a D.A.G D is a boundary D.A.G, then D is a minimal I-map of a dependency model $\mathcal{M}$ (Verma and Pearl (1990)).

Definition 1.11. *Let P be a joint probability distribution with d ordering of its variables. A D.A.G is constructed by assigning any minimal set Π_{X_i} as parents with X_i as descendants, with the criteria*

$$P(x_i \,|\Pi_{X_i}) = P(x_i \mid x_i, \ldots, x_{i-1}),\ \Pi_{X_i} \subseteq \{X_0, \ldots, X_{i-1}\}. \tag{12}$$

The D.A.G satisfying the equation above is a Bayesian network of P. If $P > 0$, then the parents in the D.A.G are unique.

Constructing a Bayesian network by boundary strata, with ordering d, ensures that the ordering is consistent with the direction of the edges assigned in the network. That is, any new ordering of Π_{X_i} will satisfy equation (12), as long as the new set of predecessors of X_i do not contain any old descendants of X_i.

In principle, any given joint distribution with ordering d over $\mathbb{U}$ can constructed into a Bayesian network. In practice however, a numerical representation of the joint distribution is rarely available, but an intuitive understanding of the (major) constraints in the problem domain. The important feature of network representation is to express *directly* qualitative relationships of direct influence between propositions. The network augments these direct influences by deriving *indirect* influences between propositions and preserves them between.

Quantifying links is easier in Bayesian networks, compared to Markov networks. In Bayesian networks, specifying the strength of links between propositions, one needs only assess the conditional probability between parent partitions and its descendants by some function. For the model builder, the conditional probabilities quantifies many conceptual relationships that can be obtained by direct measurement[14].

It is worthy to note, that the topology of Bayesian networks are sensitive to the ordering numbering d of the propositions. Ordering d_1 could result into a tree-like structure while another ordering d_2 could result into a complete network. This is because the standard ordering imposes an indirect induces identical network topologies. which is caused by the direction of causation given by the model builder.

1.4 Probabilistic graphical models in vision

Before elaborating the usage of P.G.Ms in vision tasks, a brief motivation of P.G.Ms is given from statistical mechanics from where most theoretical and technical ideas are derived from.

Statistical mechanics is an approach of studying the behaviour and properties of macroscopic bodies, which are composed of a very large number of individual particles using statistical laws (Landau and Lifshitz (1969)). When describing, for example, motion of a mechanical system, one constructs and integrates equations of motion of the system. If the concerned system obeys the laws of classical mechanics, one must construct and solve a set of differential equations (D.Es) which are equal to the number of degrees of freedom that are present in the given system. Solving such a large number of D.Es becomes tedious and impractical. An additional problem is that even if we could solve these large number of D.Es by integration, substituting a general solution to the initial conditions becomes impossible due to the size of the problem. For this reason, statistical and probabilistic approaches should be used.

When there is a large number of degrees of freedom, solving the mechanical

[14] Also psychologically convenient.

system with statistical means that we can find new types of regularities. These regularities can be found by treating the macroscopic body as a closed system and study its properties and behaviour with respect to the *phase space* of its *subsystems*. (That is, study the coordinate points and momenta of the subsystems, which determines the total energy of the system.) These subsystems are also treated as macroscopic bodies but these are not closed anymore, because of their complex interaction with each other, and their varying states over time. By solving the mechanical problem of these subsystems we can obtain a solution to the whole macroscopic body. The fundamental feature of using a statistical approach to solving the mechanical problem of the macroscopic body, is that given long enough time, the complex interactions between subsystems will visit all possible states.

Here the macroscopic body is analogous to a digital image, and the different subsystems of the macroscopic body are analogous to for example the interaction between foreground and background (or objects of the digital image), and the phase space of each subsystem are analogous to pixel (values) or objects states.

1.4.1 P.G.Ms applied in vision tasks

For processing and analyzing complex and "natural" images, stochastic image modellings offers powerful tools for various low-level and high-level vision tasks (Chan and Shen (2005)). The visual perception of humans is able to separate and identify similar objects with varying sizes and shapes from different background by computing various statistical properties. By quickly computing these statistical properties, the human visual perception system can utilize robust features for visual detection. By approaching image modelling from a stochastic viewpoint, each image sample is viewed as a statistical ensemble of different ques or features from the image. In other words, these statistical ensemble encode the probability distribution from the perceived images which enables the detection of smoothly varying features (for an ensemble interpretation of probability, see Appendix 2.3).

With this stochastic image modelling in mind, P.G.Ms are popular and effective methods for modelling the aforementioned (statistical) feature distributions, and spatial relationships between neighboring pixels in digital images. P.G.Ms have been used for stereo matching (Geiger et al. (2010); Sun et al. (2003)), image segmentation (Won and Derin (1992); Boykov and Funka-Lea (2006); Zhang and Ji (2009)), image classification (Jordan (2004); Torralba et al. (2004)), image denoising (Geman and Geman (1984); Xie et al. (2012); Malfait and Roose (1997)) and object-matching (Murphy et al. (2003); Caetano et al. (2006)), to name a few.

1.4.2 A note on differences between directed and undirected networks in vision

The structure of a network represents a particular, unique factorization of the joint distribution induced by the edges and nodes. That is, the patterns of conditional independence are expressed differently in directed networks compared to

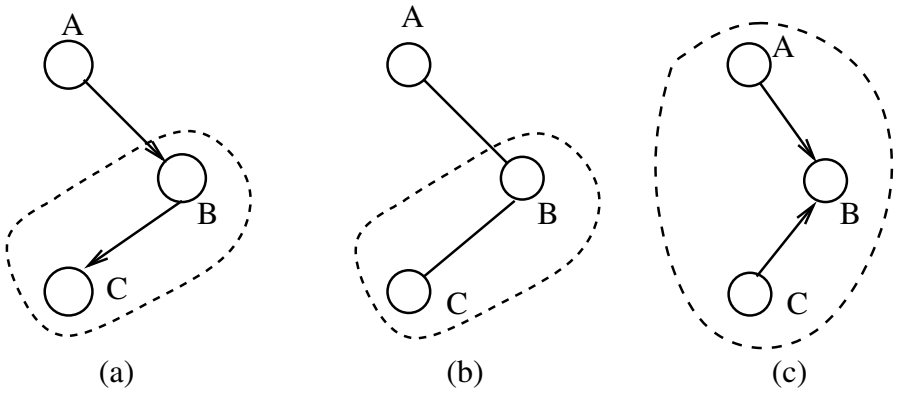


FIGURE 4 Difference between patterns of conditional independence in directed and undirected networks (Prince (2012)).

undirected ones (Figure 4).

In Figure 4 (a), node C is the Markov blanket of node B (dashed area). That is, B is shielded from the influence of node A, making B independent from A. In notation: $C \perp A|B$. The same independence relation is present in Figure 4 (b). However in Figure 4 c) A and C are independent, but we cannot write $A \perp C$ unless we condition both A and C on B. That is, we cannot have an undirected network structure with three nodes with the type of independence pattern in Figure 4 (c).

The most common P.G.M types used in vision are the following:

- *Hidden Markov Models*: Measurements are observed in a chain-like fashion, where a discrete state of a chain is determined only by its previous state.
- *Markov tree*: Measurements and their states are in a tree-like structure.
- *Markov Random Fields* (M.R.Fs): Measurements and their states are formed into a undirected network structure, where each node correspond to a pixel in the digital image.
- *Kalman filter*: Same structure as the hidden markov models but the states are continuous.

1.4.3 Markov Random Fields

Pixels in a digital image can be seen to form a lattice[15] of nodes. To evaluate the relations of the pixels and their states, each state of a pixel has a direct, probabilistic connection to its neighboring pixels. Each pixel, in itself, gives ambiguous information about its associated state in the digital image. However, it is known

[15] That is, an abstract structure which is a partially ordered set: given a homogeneous binary relation $\leq$, a set P and $x, y, z \in P$, then

- *reflexivity*: $x \leq x$,
- *antisymmetry*: if $x \leq y$ and $y \leq x$, then if $x = y$,
- *transitivity*: if $x \leq y$ and $y \leq z$, then $x \leq z$,

and where every two given nodes have a unique supremum or infimum.

that a certain spatial configuration of states are more common than others. To exploit these spatial configurations, we now describe the M.R.F model, which will be used to do inference on the states of pixels.

Neighborhoods and Local specification. Next we state the M.R.F model in terms of *neighborhoods* (or Markov Blankets) and *configurations*[16]. We will first restate the Markov blanket property into a more conventional form used in vision literature.

Definition 1.12. *A* neighborhood *system of* $\mathbb{U}$ *is a family of* $N \stackrel{def}{=} \{\mathcal{N}_u\}_{u\in\mathbb{U}}$ *partitions of* $\mathbb{U}$*, such that*

- $u \notin \mathcal{N}_u,\ \forall u \in \mathbb{U}$,
- $v \in \mathcal{N}_u \implies u \in \mathcal{N}_v,\ \forall u, v \in \mathbb{U}$.

The tuple $(\mathbb{U}, N)$ *defines a network. A boundary of a set* $A \subset \mathbb{U}$ *is defined by the partition* $\partial A \stackrel{def}{=} (\cup_{u\in A}\mathcal{N} \setminus A)$.

Here $\mathbb{U}$ are the nodes of the network and N defines the edges connecting the nodes. $v \in \mathcal{N}_u$ is called a *neighborhood site*, connecting sites u and v with an edge. Next we give a definition of a M.R.F.

Definition 1.13. *An undirected network is called a* Markov Random Field *with respect to its neighborhood system* $\mathcal{N}$*, if for all neighborhood sites in* $\mathbb{U}$*, r.vs* $\mathbf{X}(u)$ *and* $\mathbf{X}(\mathbb{U} \setminus \mathcal{N}_u \cup \{u\})$ *are independent given* $\mathbf{X}(\mathcal{N}_u)$*. That is,*

$$I(\mathbf{X}(u), \mathcal{N}_u, \mathbf{X}(\mathbb{U} \setminus \mathcal{N}_u \cup \{u\})),\ \forall u \in \mathbb{U}. \tag{13}$$

Or equivalently in traditional, probabilistic notation:

$$\mathbb{P}(\mathbf{X}(u) = \mathbf{x}(u) \mid \mathbf{X}(\mathbb{U} \setminus u) = \mathbf{x}(\mathbf{U} \setminus u)) = \mathbb{P}(\mathbf{X}(u) = \mathbf{x}(u) \mid \mathbf{X}(\mathcal{N}_u) = \mathbf{x}(\mathcal{N}_u)), \tag{14}$$

for all $u \in \mathbb{U},\ \mathbf{x} \in \mathbb{W}^{\mathbb{U}}$.

Definition 1.13 states, that the distribution of $\mathbb{W}$ is directly influenced only by the neighboring sites. The probability distribution is characterized by the *local specification* of the neighboring sites:

Definition 1.14. *Let* $\pi^u : [0,1] \leftarrow \mathbb{W}^{\mathbb{U}}$ *be the* local characteristic *of site u in a M.R.F. Define* π^u *as*

$$\pi^u(\mathbf{x}) = \mathbb{P}(\mathbf{X}(u) = \mathbf{x}(u) \mid \mathbf{X}(\mathcal{N}_u) = \mathbf{x}(\mathcal{N}_u)). \tag{15}$$

Now $\{\pi^u\}_{u\in\mathbb{U}}$ *defines a* local specification *of the M.R.F.*

[16] Recall these definitions from Section 1.2.1.

Clique potentials and Gibbs Distribution. The notion of clique potentials comes from physics, introduced by Gibbs (1902). The probability distribution of the state space $\mathbb{W}^{\mathbb{U}}$ is given by

$$\pi_T(\mathbf{x}) = \frac{1}{Z_T}\exp\{-\frac{1}{T}\mathcal{E}(\mathbf{x})\},\ \pi_T(\mathbf{x}) \in [0,1],\ \mathcal{E}(\mathbf{x}) \in [-\infty,\infty] \tag{16}$$

where Z is the *Zustandssumme*[17], $T > 0$ is the temperature, $\mathcal{E}(\mathbf{x})$ is the energy of the configuration $\mathbf{x}$, with $\mathcal{E}(\mathbf{x}) < \infty \iff \pi_T(\mathbf{x}) > 0$. These clique potential energies describe the local interactions of configurations. Any singleton $\{u\}$ is a clique on its own. A partition $\mathbf{C} \subset \mathbb{U}$ having more than one element is a clique in the network $(\mathbb{U}, N)$ if and only if two distinct sites of $\mathbf{C}$ are neighbors. $\mathbf{C}$ is called a *maximal* clique if for any site u makes $\mathbf{C} \cup \{u\}$ **not** a clique.

Sampling in undirected networks in vision. The difficulty of sampling the M.R.F comes from the fact, that there is no way know which of the variables are parents to another variables. This makes it difficult to make decisions to do ordered sampling on the variables of the network. Another difficult poses the factorization of the cliques, which are not probability distributions. One way to sample the M.R.F is to use *Markov Chain Monte Carlo* method (Carlo (2004)), which generates a series of samples from the distribution where the sampling depends only on the previous sample in the chain. This can be done, for example, using *Gibb's sampling* (Brémaud (2013)).

1.5 Artificial Neural Networks

Recently there has been an explosion of published research[18] on the subject of *artificial neural networks* (A.N.Ns). Only the fundamental aspects of A.N.Ns will be covered here. The treatment of *Convolutional Networks* will be more explicit, since it more relevant to this study (Section 3.4). Readers familiar with A.N.Ns may skip this Section. We mainly follow Hecht-Nielsen (1990) on the fundamental aspects of A.N.Ns, and Goodfellow et al. (2016) on C.N.Ns in this Section unless otherwise stated.

1.5.1 Fundamentals

The general structure of a A.N.N is of a set of *neurons* (processing elements) and *connections* (an instantaneous unidirectional signal-conduction path) in a form of a directed network[19]. The structure itself can be considered as a parallel distributed information processing structure. Each neuron has

- any number of *incoming* or *outgoing* connections[20].

[17] The *partition function*, for you German illiterates.

[18] Also commercial use.

[19] Not to be confused with Bayesian Networks from Section 1.3.3

[20] The output signal must be of identical sign because of multiple (possible) outputs.

– *local memory*[21] to store values.
– a *transfer function* which uses and alters local memory, and input signals producing output signals. Transfer functions operate either *episodically*: an input is "active", causing the transfer function operating on the current input signals and local memory, producing and updated output signal (may also alter values in local memory); or *continuously*: neurons are *always* active and information is passed in the network by *scheduling*.

Further more, the A.N.Ns have connection coming outside the network from the real world, and finally the connection out from the network. A classical example of an A.N.N is the *perceptron* model by Frank Rosenblatt (1958) (Figure 5).

In neurocomputing, by an A.N.N *architecture* it is meant as a *mathematical description*. Likewise a computer program[22] has nothing to do how it is run in a computer, here an A.N.N architecture has no *implementational* meaning (how it is implemented in a computer or software).

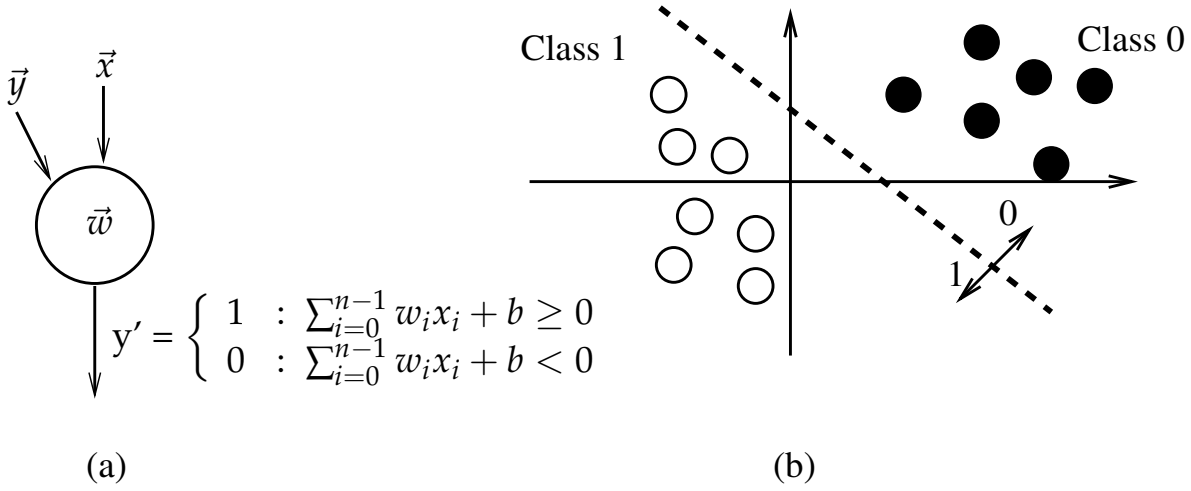


FIGURE 5 (a) The perceptron model. The model is trained with inputs $\vec{x}$ and "correct" output $\vec{y}$. (b) Pattern classification done by the perceptron model: searching a separating hyperplane, according to learned weights $\vec{w}$, between the data points (Hecht-Nielsen (1990)).

The perceptron model illustrates a single-layered A.N.N architecture, which aims at searching for a *linear* separating hyperplane between data points. The first A.I winter was caused by the analysis of Minsky and Papert (1969), on how the perceptron model cannot solve the *XOR-problem*. Their point was, that the perceptron model cannot cope with nonlinear problems. In order the perceptron model to cope with searching for nonlinear hyperplanes, the solution is to add *hidden neural layers* between the input and output neurons (Figure 6). These hidden layers are disjoint partitions of neurons, still possessing transfer functions.

[21] May have.

[22] As an **actual computer program** there is of course implementation differences to specific computer architectures. Usually computer programs are referred as **algorithms**. Because the notion of an algorithm has no rigorous definition in current literature, algorithms are referred here as computer programs or routines.

This is basically a *universal* definition for all A.N.Ns. Adding additional hidden layers to the network adds additional nonlinearities into the architecture[23].

Conceptually, each neuron is an "isolated island". In order for the neurons to adapt into its local environment, *statistical learning theory* (Vapnik (1998)) comes into play, so that the transfer functions can be used in various A.N.N architectures. Most A.N.Ns learning is done by modifying the *weights* of the neurons, where the weights determines the set of all possible connection configurations in the network. There are three distinct learning approaches in modern statistical learning theory:

- *Supervised learning*: A parametric sampling approach, where the A.N.N model is shown a tuple of an input-output relation $(\vec{X}, \vec{Y})$, from which the underlying probability distribution $P(\mathbf{X}, \mathbf{Y})$ is modeled.
- *Unsupervised learning*: A nonparametric sampling approach, where the probability distribution of $P(\mathbf{X})$ is modeled w.r.t. some, for example, proximity criteria.
- *Reinforcement learning* (Mine and Osaki (1970)): A *(Semi-)Markovian Decision Process*, which aims at maximizing some *utility function*.

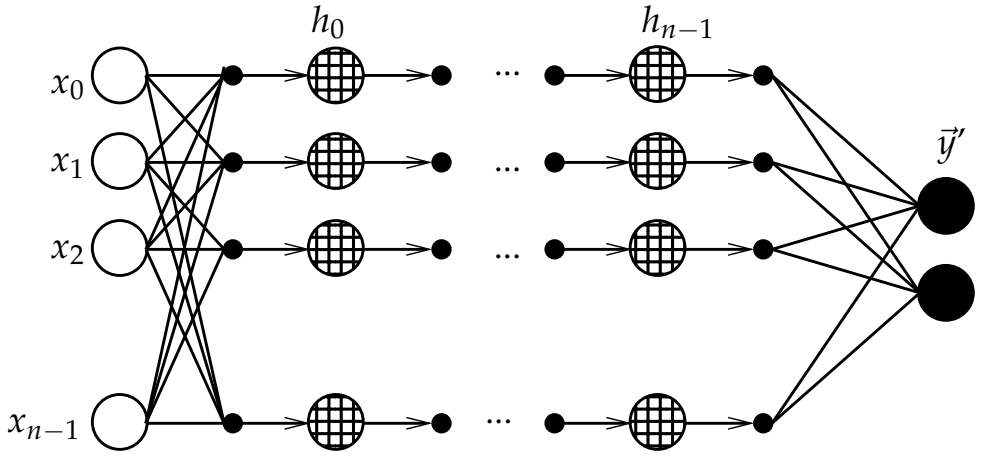


FIGURE 6 A prototypical A.N.N architecture. Each input neuron is connected to a hidden layer. The final hidden layer is then connected to the output neurons (Hecht-Nielsen (1990)).

1.5.2 Convolutional Networks

Convolutional networks (C.N.N) is a network architecture specialized in signal processing tasks[24]. A C.N.N can be specified by defining at least *one* (hidden) layer which utilized the *convolution* operation: given a time t dependent output $x(t)$, $x, t \in$

[23] With the perceptron the hyperplane is a straight line. Adding one hidden layer the separating hyperplane becomes a *polygon*. Adding a second hidden layer the hyperplane becomes a *polyhedra*, etc.

[24] Also other tasks which are suited for grid-like computational models, such as time-series.

$\mathbb{R}$, the convolution operation computes a weighted average w with a given displacement h. Or formally,

$$(x * w)(t) \stackrel{\text{def}}{=} \int x(h)w(t-h)\, dh,\ w > 0, \tag{17}$$

where weighted function, or *kernel*, w must be a valid P.D.F. The only requirement for the operation 17 is that the integral exists, for the operation can we used for other purposes than computing weighted averages. What $(x * w)(t)$ produces is a *feature map*. For a discrete convolution, the operation (17) is defined as

$$(x * w)(t) \stackrel{\text{def}}{=} \sum_{h=-\infty}^{\infty} x(h)w(t-h),\ x, t \in \mathbb{Z}. \tag{18}$$

The appeal of using convolution layers are the following.

- *sparse connectivity*: The layer can store the detected features into a smaller number of parameters. This results in smaller memory requirements[25] and improves statistical efficiency.
- *parameter sharing*: Instead of learning *separate* parameters for each point location, the convolution operation *shares* the learned parameters by the kernel w. This also reduces memory requirements.
- *equivariant representation*: The parameter sharing has the property of *equivariance* to translation: if the input changes by some amount, the output changes by the same amount[26].

An additional feature of the C.N.N architecture is the *pooling function*, which is performed after the convolution and transfer function operations in the network. The pooling function replaces certain locations of the output in the network with a summerized statistics of these certain locations. This pooling function helps making feature representation *approximately* invariant to small translations in the data.

1.5.3 Connection to P.G.Ms

We will illustrate the connection between P.G.Ms and A.N.Ns using an analogy from statistical mechanics, and an example using a modified version of the *Ising model* which is used to understand the phase transitions in ferromagnetic materials. Given a network $(\mathbb{U}, N)$, where $\mathbb{U} \stackrel{\text{def}}{=} \mathbb{Z}_m^2, m > 0$ and N is a family of $\mathcal{N}$ neighborhood systems of $\mathbb{W} \stackrel{\text{def}}{=} \{0, 1\}$ sites. In statistical mechanics terms, we are solving the phase spaces (states) of the different subsystems defined by the configurations of $\mathbf{x} \in \mathbb{W}^{\mathbb{U}}$.

---

[25] That is, given m inputs and n outputs, a "standard" hidden layer requires $\mathcal{O}(mn)$ matrix multiplication operations, whilst limiting the inputs into k requires $\mathcal{O}(kn)$ operations. See Definition 3.5, Appendix 3.2 for the definition of $\mathcal{O}(\cdot)$.

[26] $f(g(x)) \equiv g(f(x))$ for you techincally minded readers.

Now the site $u \in \mathbb{U}$ is being interpreted as a **neuron** and is *excited* when $\mathbf{x}(u) = 1$ and *inhibited* when $\mathbf{x}(u) = 0$. If we have $v \in \mathcal{N}_u$, then u is connected to v and has the weight w_{uv}. Now the energy function becomes

$$\mathcal{E}(\mathbf{x}) = \sum_{u \in \mathbb{U}} \sum_{v \in \mathcal{N}_u} w_{vu}\mathbf{x}(v)\mathbf{x}(u) - \sum_{u \in \mathbb{U}} h_u \mathbf{x}(u), \tag{19}$$

where h_u is the *activation threshold* of the *transfer function* of neuron u.

1.6 Similarity measures from seismic signal analysis

Here we adopt similarity measures which are utilized in *seismic signal analysis* (S.S.A), where the goal is to analyze the reflection of wave propagation in some medium (Yilmaz (2001)). There are three principle applications for applying S.S.A:

1. *Engineering seismology*, where tools from S.S.A are applied to near-surface studies, such as delineation of near-surface geology for engineering studies.
2. *Exploration seismology*, for exploring and developing hydrocarbon for oil and gas fields.
3. *Earthquake seismology*, where the crustal structure of the earth is investigated.

Seismic data contains reflections of wave propagation in a medium, recorded and stored as a temporal image. These temporal images are processed either considering each image channel separately or as a multichannel image as a whole. The tools for processing and analyzing seismic data almost always consists of these three techniques: *deconvolution*, *common-midpoint stacking* and *migration*.

1.6.1 Coherence measure

The relevant parts and theories of S.S.A for our purposes is to use the concepts known as *coherency measures*. What seismic data actually captures is the *indirect measurement* of the *velocity* of the seismic wave travelling in the earth. With the addition of *sonic logs*, which are *direct measurements* of the seismic waves travelling in the earth, different types of velocities can be derived. Coherency measures are applied to do signal corrections from noisy seismic data measurements and is one of the main tools for analyzing temporal images obtained from seismograms (Quincy and Tomich (1985); Bahorich and Farmer (1995)).

To estimate seismic velocities, the seismic data is required to be measured from nonzero offsets, provided by a measuring device. This is to say, that the seismic data is captured by a recording device and multiple wave reflections. The difference in traveltime at a given offset and zero offset is called a *normal moveout* (N.M.O) type velocity.

The object of using coherency measures in S.S.A is to find similar data among multidimensional signals. Thus coherency provides a similarity measure and is

used for enhancing, extracting or estimating shapes of common signals. Even if the data is weak or noisy the coherency measure provides a measure of similarity between similar data signals. How the adopted coherency measure depends upon the application in question.

Let's give an elementary S.S.A example. For analyzing seismic data, assume there is k signal traces (or channels) which are assumed to be independent. From each k trace, $s+1$ samples are obtained and a coherent signal is assumed to cross the k traces with linear N.M.O trajectory. Let this trajectory be denoted as m_j. The coherency of a signal is evaluated pairwise across the channels using the following window function

$$R_{j,l}(0) = \sum_{i=0}^{s} t^{j}_{i+m_j} t^{l}_{i+m_k}, \tag{20}$$

which measures the zeroth-lag correlations between channels j and l. Here t^{j}_{i} is the amplitude of the data sample i at trace j. Using the previous window function, the unnormalized crosscorrelations between pairwise channels is given by the sum

$$\sum_{l>j} \sum_{j=1}^{k-1} \frac{1}{s+1} R_{j,l}(0). \tag{21}$$

The Simpson's dissimilarity measure (Simpson Jr (1967)) is used to measure the crosscorrelation between pairwise channels. This dissimilarity measure is based on the energy ratio's output/input where the channels within the window $R_{j,l}(0)$ are linearly combined. The dissimilarity measure is expressed as

$$\lambda' = \frac{\frac{k-1}{2} \sum_{j=1}^{k} R_{j,j}(0) - \sum_{l>j} \sum_{j=1}^{k-1} R_{j,l}(0)}{\frac{k-1}{2} \sum_{j=1}^{k} R_{j,j}(0) + \sum_{l>j} \sum_{j=1}^{k-1} R_{j,l}(0)}, \tag{22}$$

where $\lambda' \in [0, \frac{k}{k-2}]$. Within the window $R_{j,l}(0)$, λ' will give an estimate for the Signal-to-Noise Ratio (S.N.R), defined by the samples of k traces. As $R_{j,l}(0)$ traverses the k traces, it may overlap in both space and time which gives better estimation of the data samples. This better estimation comes with an increase in data storage requirements. The window $R_{j,l}(0)$ may also be rotated, giving estimations within the given range of moveouts. For seismograms, this gives a S.N.R estimate of coherent line-ups, the moveout line-ups and their space-time location within the seismogram (Quincy and Tomich (1985)).

The information provided by λ' is used to assign weights to a 2D enhancement mask, where there exists a bijection between the location of the weights in the enhancement mask and the sample of the regions scanned by $R_{j,l}(0)$. The weights designate the maximum S.N.R at the particular weight location passed by the window estimators at the corresponding data sample. For extracting the weights, the user inputs two threshold parameters $\alpha, \beta \in [0,1], \alpha > \beta$. All weights w which satisfy $w > \alpha$ are replaced by α, and all weights which satisfy $w < \beta$ are set to zero. After the weights are extracted, the weights are multiplied with the corresponding data sample in the image.

1.7 High-performance computing

High-performance computing (H.P.C) deals with program *implementations* and *hardware* aspects for a given computing task (Hager and Wellein (2010)). That is, when a desired method is chosen for solving a computing task, efficient implementations of the method is desired on the given hardware. This can be done either efficient use of the underlying hardware architecture (for example, field-programmable gate array hardware, cache-based architectures, ...) or formulating the chosen method into parallel program, such that the method can be implemented using multicore architectures or graphical processing units.

Choosing an appropriate computing model helps abstracting and encapsulating the main features of a computing architecture. This also helps clarifying the program description and design for a given problem and evaluating program performance. A naïve approach to evaluating performance of a parallel program is to assume having access to an unlimited number of processors with instantaneous data transfer capabilities. With this naïve assumption, the theoretical or maximum expected benefits of the program can be evaluated, and also helps in search for a more feasible method for a given computational task.

From a computational stand point, method feasibility for a given computational task depends on context and is not particularly transferable between different tasks and situations (Greenlaw et al. (1995)). However, it is desirable that for large computation problems that the selected method does not have an exponential (or factorial) growth rate in terms of computing time and space requirements. This is not only desirable for sequential computing, but also for parallel computing. In practical terms, a parallel method is feasible if we can solutions to a n-sized problem in polynomial time $n^{\mathcal{O}(1)}$, using $n^{\mathcal{O}(1)}$ processors. This is not a universally accepted definition for parallel method feasibility. The use of this definition is justified, when considering that we are attempting to trade the number of processors for speed. The goal for parallel computing comes then to develop methods that use a reasonable number of processors[27] and are computationally tractable.

1.7.1 Limiting factors in H.P.C

When discussing limiting factors in H.P.C, it comes down to two factors: *technological* and *computational* [28]. Concerning technological limitations, *Ahmdal's law* (Amdahl (1967)) is one of the widely used propositions when evaluating parallel performance of a program, which governs the amount of parallelism a method may obtain.

Proposition 1.1. *Let $f_p \in (0,1)$ be the fraction of number of arithmetic operations which are inherently sequential, where p is the number of available processors. The speed up S_p*

[27] What is a "reasonable" number of processors is up to debate.

[28] These two limitations often are intertwined. That is, even though a method is formulated into a parallel program, technological limitations (may) still apply.

of a parallel program is bounded by

$$S_p < \frac{1}{f_p}. \tag{23}$$

Ahmdal's law assumes a fixed sized computation, which then gives a speed up and efficiency fraction that is actually achievable by the parallel program. Even for a full parallel program, there is a loss of efficiency caused by the communication time cost, memory references and parallel management between processors. With these overhead costs, Ahmdal's law usually implies $S_p < p$. For $S_p > p$ cases, the available data for the computation is too large for the local memory of one processor. For such cases, then distributing the data across p processors becomes possible and desirable.

However, it is worthy to note that Ahmdal's law mainly addresses so called *Single instruction, multiple data* systems (Hager and Wellein (2010)). That is, running a single user-mode on a parallel computer and does not address the effects of other computational processes handled by the system, nor heterogeneous systems are addressed.

2 PROBABILISTIC PROPAGATION

"Algorithm *(noun)* is a word used by programmers they do not want explain what they did."

Qwertee Tee print

In this chapter, the Belief Propagation (B.P) method will be reviewed. First in Section 2.1 the B.P scheme by Pearl (Pearl (2014); Jordan et al. (2001)) will be described. That is a probabilistic propagation for doing inference in networks, with a particular emphasis on M.A.P estimation (to be explained later). In Section 2.1.2 we describe a hierarchical propagation approach by Felzenszwalb and Huttenlocher (2006), on the emphasis in image denoising. In Section 2.2, we present a modification of the B.P procedure by applying the coherence measure to solve the M.A.P estimation. Accelerating the B.P method is done by utilizing elementary H.P.C approaches, which is then presented in Section 2.3.

2.1 Propagation in networks

B.P methods are for solving probabilistic inferences using local message-passing schemes on a set of unobserved variables given a set of observed variables. The probability distribution is induced by the network's structure and connection of the variable nodes in the network. Network structures which are singly connected, that is there exists only one path between any nodes[1], there exists tractable message-passing schemes for computing the posterior probability (or *beliefs*) of the set of unobserved node variables given observed ones (Pearl (2014)). We will focus on network structures where multiple paths are present, that is, multiple *loops*. For various message-passing schemes for singly connected networks, see Smyth et al. (1997) for a review.

[1] For example, tree structures (Aho and Hopcroft (1974)).

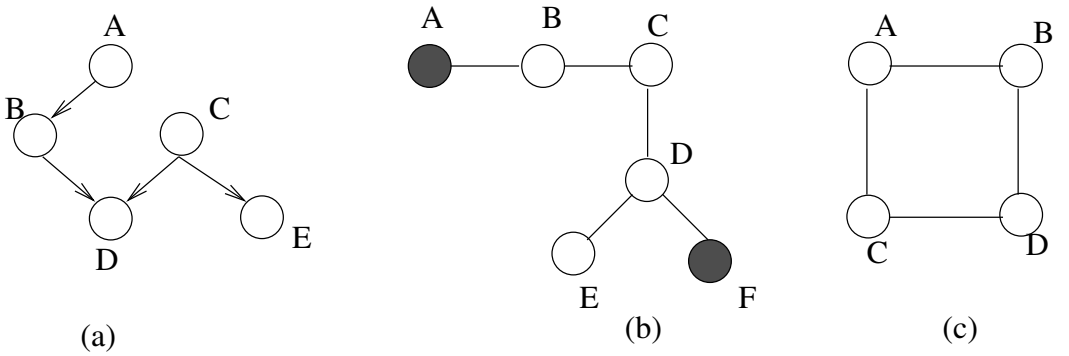


FIGURE 7 Example of three different P.G.Ms (Jordan et al. (2001)). The structure and connection of the nodes represents the constraints (structure) and the qualitative aspects of the joint distribution induced by the node connections. a) A singly connected Bayesian Network. b) A singly connected Markov Network, where nodes A and F are observed. c) A Markov Network with a loop.

Belief propagation formulation. As just mentioned, the structure of the network implies a product form of the node variables of the induced probability distribution. Using the chain rule of probability, the complete probability distribution over the node variables can be expressed as a product of conditional and prior probabilities. For example, for the Bayesian Network in Figure 7 a), we can express the complete probability distribution in the following product form:

$$\mathbb{P}(ABCDEF) = \mathbb{P}(A)\mathbb{P}(B|A)\mathbb{P}(C)\mathbb{P}(D|B,C)\mathbb{P}(E|C). \tag{24}$$

In the case of Markov Networks, the Hammersley-Clifford theorem (Besag (1974)) guarantees, that the probability distribution can be factorized into a product of functions in terms of the maximal cliques in the network. For example, the Markov Network with a loop in Figure 7 c), we can write the probability distribution into the following product form:

$$\mathbb{P}(ABCD) = \frac{1}{Z}\mathbb{P}(AB)\mathbb{P}(AC)\mathbb{P}(CD)\mathbb{P}(DB). \tag{25}$$

There are three inferences tasks what probabilistic propagation can be used as a solver:

- *Marginalization*: Computing the marginal probabilities of a variable given an observed variable(s).
- *Maximum a Posteriori (M.A.P)*: Finding the most probable assignments to unobserved variables given the observed variables.
- *Maximum Marginal*: Finding value assignments to the unobserved variables that maximizes the marginal probability assignments.

In this work only the M.A.P estimation problem is considered. See Appendix 1.8 the M.A.P formulation for vision in P.G.Ms.

Here we give a message-passing scheme for pairwise Markov networks, which we will be experimenting in image denoising. In the case of where there

are many paths between a set of nodes, computing correct beliefs is no longer possible, because the messages will be passed around the network indefinitely and the messages will not converge to a stable answer. Empirically, B.Ps are shown to have good performance on approximate inference when multiple loops are present in the network (Murphy et al. (2013); Jordan et al. (2001)). However, theoretically there is no definite answer yet why these empirical results hold. Ihler and Willsky (2005) presents some theoretical bounds on the performance of B.Ps on arbitrary network structures.

2.1.1 Solving M.A.P estimation with message-passing in the presence of loops

In this work, we solve the M.A.P estimation on a grid, which consists of points in the image. This makes each point on a grid *independently, identically distributed*, which acts as a relaxation technique to solve the M.A.P estimation. We use a *variational* formulation of the problem (Appendix 1.5), in order to take advantage of the M.R.F theory presented in Section 1.4.3.

The M.A.P estimation on a grid of points translates to finding the most probable *pixel label* assignment. The most probable assignment is done by solving the following energy function:

$$\arg\min_{\mathbf{X}} E(\mathbf{X}) = \sum_{i \in V^{\mathcal{L}}} f_i(x_i) + \sum_{c \in \mathcal{C}} f_c(x_c), \tag{26}$$

where $\mathcal{V}$ is a set of points on the image grid $\mathcal{G}$, $\mathcal{C}$ is a set of cliques, and $\mathcal{L}$ the label set. In statistical physics terms, the goal is to minimized the total energy of the grid points by finding the most probable pixel label assignments. Since we are restricting our model where $|\mathcal{C}| = 2$, the M.A.P problem (26) is rewritten as

$$\arg\min_{\mathbf{X}} E(\mathbf{X}) = \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{j \in \mathcal{N}_i} f_{ij}(x_i, x_j), \tag{27}$$

where $\mathcal{N}_j$ is the $4-$neighborhood of the variable i. Here, $f_i(x_i)$ denotes the *unary potential* of x_i, and $f_{ij}(x_i, x_j)$ denotes the clique energy potential.

Energy minimization via max-product B.P translates the clique potentials as messages $m_{ji}(x_j)$. Here the message is to the jth variable from variable i, giving information about the state the variable j should be. The message is encoded as a vector of dimension $|\mathcal{L}|$ with vector components proportional to the likelihood, or the belief, of the state that x_i thinks x_j should be.

Minimizing the energy function (27) with B.P is done using a *max-product* approach, where the posterior probability distribution for each pixel label is computed iteratively in an approximate fashion. A *sum-product* approach could also be used but we use the max-product formulation, because it is computationally equivalent to minimizing the sum of log probabilities, which corresponds directly to the M.A.P estimation (27). Computing the messages and passing the messages in a network is defined by the selected neighborhood structure $\mathcal{N}$. Using the

max-product approach, denote $m^t_{i\to j}$ as the message passed from i to j at iteration t. Then the messages are computed as

$$m^t_{i\to j}(f_j(x_j)) = \min_{f_i(x_i)} \{f_i(x_i) + f_{ij}(x_i, x_j) + \sum_{s\in\mathcal{N}_i\setminus j} m^{t-1}_{s\to j}(x_i))\}. \tag{28}$$

After computing T iterations, for each node we have

$$b_j(f_j(x_j)) = f_j(x_j) + \sum_{i\in\mathcal{N}_j} m^T_{i\to j}(f_j(x_j)), \tag{29}$$

where the value of $b_j(f_j(x_j))$ is the most probable labelling for the node x_j.

2.1.2 Speeding up max-product on arbitrary network structures

The challenges with B.P methods with multiple loops is that messages are observed, or *counted*, multiple times, and the difficulty of converging to a correct posterior estimate[2]. Counting messages multiple times causes high computational costs for doing inferences. The basic implementation of message passing via max-product on a network would require $\mathcal{O}(|\mathcal{V}||\mathcal{L}|^2T)$ operations to converge to a correct M.A.P estimate (that is, if a correct estimate is to be found), where T is the number of iterations . Furthermore, each message $\Theta^3(|\mathcal{L}|^2)$ operations is required for computing each message vector and there are $|\mathcal{V}|$ messages to be computed per iteration. To minimize the computational cost of the max-product scheme, Felzenszwalb and Huttenlocher (2006) applied the following three techniques.

Reducing number of computed messages. To reduce the number of computed messages, instead of computing the messages explicitly over $f_i(x_i)$, the message update (28) is rewritten into a *infimal convolution* (Maragos (2001)) form:

$$m^t_{i\to j}(f_j(x_j)) = \min_{f_i(x_i)}\{f_{ij}(x_i, x_j) + g(f_i(x_i))\}, \tag{30}$$

where $g(f_i(x_i)) \overset{\text{def}}{=} f_i(x_i) + \sum m^{t-1}_{s\to j}(x_i)$, which enables message minimization over $f_i(x_i)$. When the term $f_i(x_i)$ is computed first, the result can be used to compute the corresponding message value for $f_j(x_j)$ in constant time.

To evaluate the difference magnitude between labels, that is $f_{ij}(x_i, x_j)$, different *cost functions* can be applied. For example, the *Potts model* (Boykov and Funka-Lea (2006)) which enforces the label difference to be piecewise constant:

$$f_{ij}(x_i, x_j) = \begin{cases} 0 & \text{, if } x_i - x_j = 0 \\ d & \text{, else,} \end{cases}$$

---

2 That is, not only restricted to M.A.P estimations.

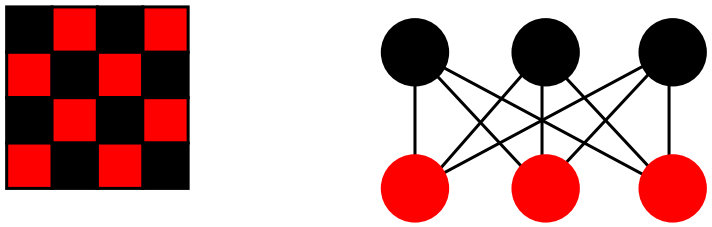
3 Definition 3.7, Appendix 3.2.

where d is some positive constant. When the minimization over $f_i(x_i)$ is done independently from $f_j(x_j)$, the minimization can be done in $\Theta(|\mathcal{L}|)$ operations. That is, if we first compute the minimization $\min_{f_i(x_i)}$, we can use this to compute the message for each $f_i(x_i)$:

$$m^t_{i\to j}(f_j(x_j)) = \min\{g(f_j(x_j), \min_{f_i(x_i)} g(f_i(x_i)) + f_{ij}(x_i, x_j)\}. \tag{31}$$

Instead of enforcing $f_{ij}(x_i, x_j)$ to do "paranoid" label cost selections, we can use the form of the Potts model but measure the label difference differently. For example, a *linear model*, where the absolute difference between the labeling cost is evaluated; or *quadratic cost*, where the quadratic difference is taken between the labeling cost. In both cases the constant d acts as a *ceiling* factor which stops the labeling cost when the difference becomes large. Both linear and quadratic costs can be seen as computing a collection of *lower envelopes* (l.e.) of $|\mathcal{L}|$ cones or parabolas with slope c, rooted at $(f_i(x_i), g(x_i))$. The computation of these envelopes is similar to computing a binary distance transform (Borgefors (1986)), where the l.es are sorted into a combinatorial structure that minimizes (31).

Computing beliefs on a grid. To reduce the number of message computation, a *bipartite network*[4] structure is utilized, which is an analogous to a red-black ordering used for Gauss-Seidel relaxations in M.G solvers (Trottenberg et al. (2000)) (Figure 8). Computing messages in a bipartite network with the max-product (28), the computed messages are only dependent on their respective partitions at each iteration t. That is, if we have a set of nodes partitioned into partitions R, B, at iteration t we can compute the messages from R and then computing the messages from B will be done at iteration $t+1$, and so on. With this alternating ordering, half of messages on the network are updated and sent at each iteration.

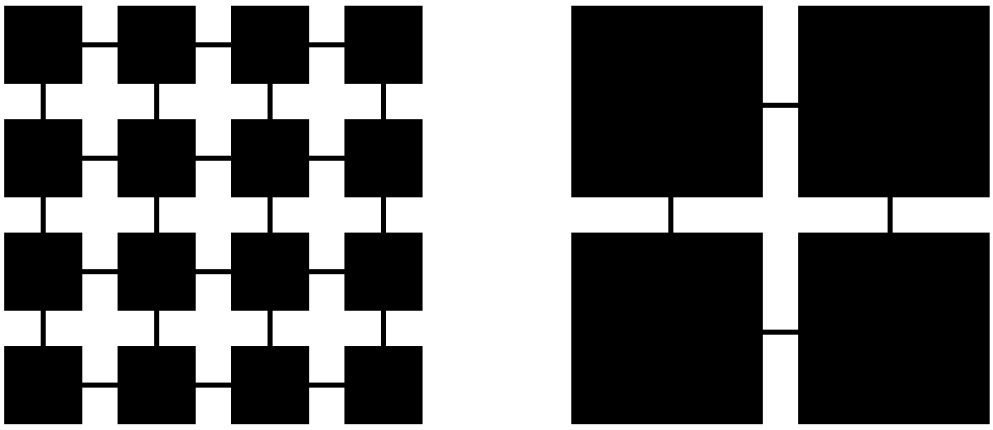


(a) (b)

FIGURE 8 Computing beliefs on a grid. (a) Gauss-Seidel relaxation on a red-black ordering. (b) Analogous bipartite network structure.

Cascadic multigrid data structure. The principal operation of M.Gs is that, solutions are computed, sent and refined through multiple levels. That is, the solutions navigate from a *fine* set of grid points to a *coarser* set of grid points, and

[4] A network where nodes are divided into two disjoint and independent partitions.



(a) (b)

FIGURE 9 Scaling between grid block levels. From a finer grid (a) to a coarser grid (b).

the finally moving back up[5]. For early vision task, the challenge of using a bipartite network structure is that messages has to travel "long" distances in the grid. To minimize the distance between grid points, a *cascadic multigrid* (C.M.G) (Bornemann and Deuflhard (1997)) is applied[6]. The difference between M.Gs and C.M.Gs, is that in C.M.Gs the computing starts from the coarsest level of the grid, and then moves up between levels. With the C.M.G structure, the energy minimization is solved level by level, where each level consists of blocks of $n \times n$ grid points, where $n = 2^l$ and $l = [0..k], k > 0$ is the number of levels used in the structure. With $n = 2^0$, the B.P inference is done only on the fine grid points, that is, the "original" image grid. Figure 9 illustrates a set of grid points in two different levels.

In energy function form, let $\mathbb{G}^l$ be a hierarchy of blocks of $\mathcal{V}$, and f_i^l, f_{ij}^l be the the unary potential and clique potential energy respectively. Then the energy function (27) is expressed as

$$\arg\min_{\mathbf{X}^l} E(\mathbf{X}^l) = \sum_{i \in \mathbb{G}^l} f_i^l(x_i^l) + \sum_{j \in \mathcal{N}_i \in \mathbb{G}^{l=k\times(k-1)}} f_{ij}^l(x_i^l, x_j^l) + \sum_{j \in \mathcal{N}_i \in \mathbb{G}^{l=(k-1)\times k}} f_{ij}^l(x_i^l, x_j^l) \tag{32}$$

In terms of messages, the belief of the messages are initialized with the message values from the previous level, moving from the coarsest level to the finer level.

Next we consider computational speed-ups and improving the quality of belief messages in B.P computations are presented. Specifically:

Belief improvements: Adding a small additional computational cost to evaluate the point energies f_i on the grid.

---

5 In imaging terms, this analogous to adjusting the subsampling level of an image (Section 1, discrete geometry).

6 This type of multi-leveled vision processing was pioneered by Marr (1982) using Gaussian convolutions. Other important related scale-space developments were developed by Witkin (1987) and Koenderink (1984).

Speed-ups: Using H.P.C strategies for the overall C.M.G structure to speed up the overall computation on the grid.

2.2 Belief revision improvement by coherence

First we elaborate improving the unary potential evaluations, following an analysis on the effects. Here we utilize the concepts presented in Section 1.6.

Recall that f_i term evaluates the unary potential on the image grid, and f_{ij} evaluates the level of interaction between adjacent points. We apply the semblance measure directly on the f_i term. The benefit of doing this is that it is computationally feasible and has a direct effect on f_{ij}. That is, we rewrite the energy function (27) into

$$\arg\min_{\mathbf{x}} E(\mathbf{X}) = \sum_{i \in \mathcal{V}} \mathcal{K}^{z(r)}(x_i) \cdot f_i(x_i) + \sum_{j \in \mathcal{N}_i} f_{ij}(x_i, x_j), \tag{33}$$

where $\mathcal{K}^{z(r)}, z(r) = (2r + 1 \times 2r + 1), r > 0$, is a similarity measure kernel (which acts as a probabilistic relaxation of the ith variable). We experiment with the following coherence measure from S.S.A, called *semblance* (Neidell and Taner (1971)), which will be used in $\mathcal{K}_i^{z(s)}$ to improve belief revisions done by the B.P:

$$\frac{1}{m} \frac{\sum_t \sum_i f_{i,t(i)}}{\sum_t \sum_i f_{i,t(i)}^2}, \text{ with range } [0, 1]. \tag{34}$$

See Neidell and Taner (1971) for various other types of coherency measures.

As a reminder from Section 1.6, in the context in velocity analysis, coherence is used to minimize the difference between the observed data and the prior of the data while the velocity changes. In the context of M.A.P estimation and B.P, we are minimizing the observed data and the prior of the data while *preserving context*. The context here is the computed semblance measure (34) of the variable i within a neighborhood defined by the kernel $\mathcal{K}$.

We have to modify $\mathcal{K}$ to work on single-channel images, since we will experiment on gray-scale images. The semblance measure on the grid points is evaluated as follows: given the ith grid point, we construct a $\mathcal{K}$ kernel around the point. Within $\mathcal{K}$, we evaluate all $x_{m,n} \in \mathcal{K}$ to evaluate the ith grid point similarity (or context) within the kernel. The semblance measure (34) computes a similarity measure in four directions: horizontal α_0, 45 degree slope α_1, vertical α_2 and 135 degree slope α_3. That is,

$$\alpha_0 = \frac{(\sum_{k=-|r|}^{|r|} x_{m+k,n})^2}{(2|r|+1)\sum_{k=-|r|}^{|r|} x_{m+k,n}^2} \qquad \alpha_1 = \frac{(\sum_{k=-|r|}^{|r|} x_{m+k,n+k})^2}{(2|r|+1)\sum_{k=-|r|}^{|r|} x_{i+k,j+k}^2}, \tag{35a}$$

$$\alpha_2 = \frac{(\sum_{k=-|r|}^{|r|} x_{m,n+k})^2}{(2|r|+1)\sum_{k=-|r|}^{|r|} x_{m,j+n}^2}, \qquad \alpha_3 = \frac{(\sum_{k=-|r|}^{|r|} x_{m-k,n+k})^2}{(2|r|+1)\sum_{k=-|r|}^{|r|} x_{m-k,n+k}^2} \tag{35b}$$

Now we can rewrite equation (33) into its full form:

$$\arg\min_{\mathbf{x}} E(\mathbf{X}) = \sum_{i\in\mathcal{V}} \mathcal{K}_i^{z(r)}(x_i,\beta)\cdot f_i(x_i) + \sum_{j\in\mathcal{N}_i} f_{ij}(x_i,x_j), \tag{36a}$$

$$\mathcal{K}^{z(r)}(x_i,\beta) =\leftarrow \begin{cases} \alpha_i & \text{, if } \max\{\alpha_0(x_i),\alpha_1(x_i),\alpha_2(x_i),\alpha_3(x_i)\} > \beta \\ 0 & \text{, else.} \end{cases} \tag{36b}$$

where r and $\beta \in [0,1]$ are user-defined parameters for the kernel respectively, and returning the maximum similarity $\alpha_i \in [0,1]$ of the ith variable within the kernel. Obtaining the measure of similarity at the ith variable, we apply the measure if $\alpha_i > \beta$, otherwise we set $\alpha_i = 0$.

Figure 10 illustrates the computing of the semblance measure. Here r denotes the "range" or "span" of the evaluated kernel $\mathcal{K}$.

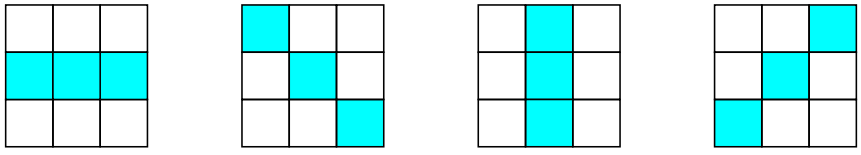


FIGURE 10 Computing semblance with a $\mathcal{K}^{z(r)}$ kernel with $r = 1$, yielding a $(2s+1)\times(2s+1) = 3\times 3$ kernel. Computing in four directions, semblance gives the center grid point's similarity information with respect to its neighboring points given span s by selecting the maximum direction $\alpha_i, i = 0,1,2,3$.

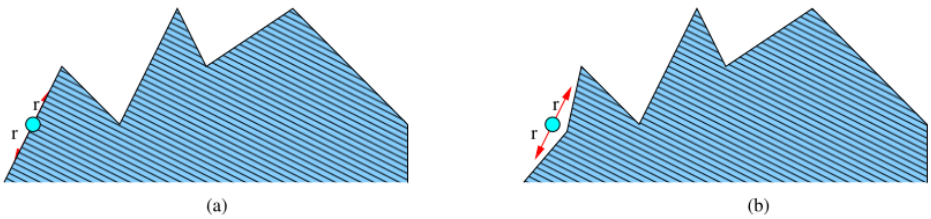


FIGURE 11 Intuitive idea of the semblance measure. Given a point and desired span r, the computed similarity measure can be regarded as how regular the point is with respect to a computed direction.

Figure 11 illustrates a simplified, intuitive idea of the coherency measure: given a point the signal and the span of the coherency computation, the points

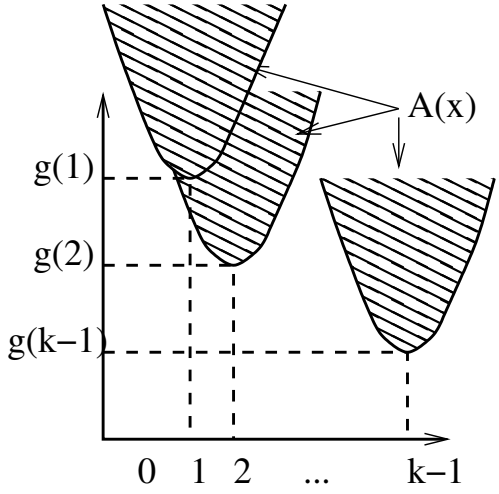


FIGURE 12 Computing lower envelopes of belief messages using infimal convolution (Felzenszwalb and Huttenlocher (2006)).

regularity is measured from the signal profile (Figure 11 (a)) along the given direction. If the span of the evaluation in Figure 11 (a) would exceed beyond the peak, this would indicate irregularity at the evaluated point. After evaluation, the measured point's value is adjusted to a more "true" signal value with respect to its evaluated neighborhood (Figure 11 (b)).

Remember, the idea of the coherency measures in S.S.A is to reveal the true signals from a multi-channelled seismic recording device. (In more complicated cases, the coherency would be computed in multiple directions. In Figure 11, an example only in one direction is given.) The benefit of evaluating the signal with the semblance approach, is that we do not need to interpolate for a more suitable value. The semblance equations (35) is easy to implement and will be addressed Section 3.1.1.

Next the effects of the semblance measure to probabilistic propagation is discussed. The effect is discussed from two different viewpoints:

Section 2.2.1 Topologically, the semblance measure induces a new combinatorial ordering on the propagating messages. With a certain threshold, the semblance measure gives a more minimal energy configuration compared to the "hard choice" done by the Pott's computation model.

Section 2.2.2 The semblance measure acts as a *modification strategy* to solve the M.A.P estimation in vision tasks. The computable justification is given in terms of probabilistic combinatorial optimization.

2.2.1 Inducing new combinatorial structure with semblance measure

Recall the message update rule from Section 2.1.2:

$$m^t_{x_i \to x_j}(x_j) \leftarrow \min_{x_i}\{f_{ij}(x_i, x_j) + g(x_i)\}.$$

This can be seen as solving a sampling problem on a grid of labels (not to be confused to the image grid itself). Also recall that computing f_{ij} can be seen

as a collection of l.es of k upward facing parabolas (Figure 12.), when using a quadratic cost model. Here the parabolas have a slope of c, rooted at $(x_i, g(x_i))$ and bounded by x_i. The minimization then becomes on solving the combinatorial structure over these parabolas using the infimal convolution computation via reordering the parabolas. The new reordering of the parabolas provides the solution to the minimization problem.

We will use the l.e properties (Appendix 2.4) to justify the new minimal ordering induced by the semblance measure.

Corollary 2.1. *The set of k parabolas defines a family of $(x_i)_{i\in[0..k]}$ constraints, where x_i constraints the parabola from above. This family of functions defines a set of points $A(x)$ which is a closed space. By Weierstrass theorem, this space obtains a minimum and maximum value. Denoting these constraints as λ, we have $g(x_i) \leq \lambda$ based on how $g(x_i)$ is defined. If we denote g as the belief messages without the semblance measure, and $\widetilde{g}$ the computed belief messages with the semblance measure, by the properties of l.e and Lipschitz functions we have*

$$\lambda \geq g(x_i) \geq \widetilde{g}(x_i). \tag{37}$$

That is, the messages $\widetilde{g}$ provides a lower energy estimation.

2.2.2 Modification strategy for M.A.P estimation in vision

When considering the M.A.P estimation problem formulation, it is posed as a *deterministic* combinatorial optimization problem. This deterministic approach does not capture the underlying randomness or uncertainty of the underlying optimization problem of P.G.Ms at the level of the grid points. Only when solving the energy minimization with associated labelling costs, which are treated as r.vs, there is an association of uncertainty.

Probabilistic combinatorial optimization attempts to address the limitations of the deterministic approach to combinatorial optimization (Murat and Paschos (2006)). Here, for each node in $\mathcal{G}$ there is an associated probability p, that the particular node belongs to some subnetwork of $\mathcal{G}' \subset \mathcal{G}$. So for solving the energy minimization problem, introduce a probability weight p into the formulation:

$$E(\mathbf{X}) = \sum_{i\in\mathcal{V}} p_i f_i(x_i) + \sum_{j\in\mathcal{N}_i} f_{i,j}(x_i, x_j), \tag{38}$$

where p_i is the probability of x_i belonging to some subnetwork in $\mathcal{G}$. In solving optimization problems, usually only a subinstance of the original problem is solved. To refine the solution, *reoptimization* is applied but this is a resource consuming (time- and spacewise) approach. This comes particularly problematic when dealing with NP-Hard optimization problems, which is usually the case in optimization problems in P.G.Ms.

Let us look closer at the formulation of the M.A.P problem. The energy functional E is divided into two functionals: $\sum_{i\in\mathcal{V}} f_i(x_i)$, which controls the "bending" or contour energy of the grid, and $\sum_{j\in\mathcal{N}_i} f_{ij}(x_i, x_j)$ which governs the relations

between the edges of the nodes in the neighborhood. The second functional is solved using the C.M.G structure by belief revision. The modification strategy is applied to the first energy function, which governs the how much each grid point "costs". Using a constant weight on evaluating the unary potential cost $f_i(x_i)$ and then processed as a $4-$neighborhood can be seen as making conservative decisions about the state of the pixel, because a $4-$neighborhood system cannot account, for example, line breaks.

The modification strategy computes an *a priori* solution for the unary potential cost. In this work, the a priori solution is computed by the semblance measure. After the computation of the a priori solutions, the solutions are then further solved in the message-passing scheme. Thus, this approach is a "greedy" heuristic of solving the M.A.P assignment problem:

1. Set a set of solutions $S \leftarrow \emptyset$.
2. Order the parabolas (messages) in the multi-scale structure in increasing order.
3. Include a minimum solution into S.
4. Remove the minimum solution from the network with any edge incident parabola (messages) to the solution (done in the infimal convolution step).
5. Repeat steps 2 and 3 until there is no more solutions to be removed.
6. return S.

2.3 H.P.C approaches

(In this section, we will refer the multi-level structure presented in Section 2.1.2 as a M.G structure.) Here we consider a more "technical" aspect of solving B.P inferences on an image grid, using the M.G structure in Section 2.1.2. When considering speeding up a M.G structure using parallel computing, some suitable components of the M.G structure are suitable for parallel computation. However the *over all M.G structure* is not fully parallelizable. There are two reasons for this:

1. The levels in a M.G structure are proceeded in a *sequential* manner.
2. The M.G levels limits the degree of parallelism (coarse grids versus fine grids).

Furthermore, the selected M.G structure has an effect on the complexity of parallelism. This is because a number of levels may be solved a number of times ($V-$cycle versus $W-$cycle (Appendix 3.4)).

When solving numerical computations with M.G solvers, there are two approaches for solving the computation in parallel:

1. Use a fast, efficient sequential solver for the problem and parallelize this solver.
2. Domain decomposition, where the problem is decomposed into a number of subproblems, and then each subproblem is solved in parallel.

With the above in mind, there are two different parallelizing approaches which can be considered:

1. **Memory optimization**: an elementary approach to speed up computation, where accessing data from memory also retrieves nearby data which are stored in cache[7]. That is, exploit cache-coherency, which minimizes data retrieval from cache (Loshin (1999)). The exploitation will be done using *loop-blocking*, where the grid is processed in a $n \times n$ or $m \times n$ blocks, in order to speed up the access to data stored in memory[8].
2. **Parallelization via domain decomposition**: the M.G structure in Section2.1.2 takes advantage of a *red-black relaxation* approach, to reduce the number of message computations done in the grid. This is a standard approach in M.G literature, which we will exploit in a parallel, brute-force-manner, using OpenMPI[9].

[7] A small auxilary storage location in the computer, allowing fast data retrieval.

[8] This is an elementary approach for speeding up a *five star stencil* operator in numerical methods. See Appendix 3.4 for an example the stencil operator.

[9] OpenMPI is an open source interface using a set of *compiler directives* for Fortran, C and C++ languages. https://www.open-mpi.org/

3 EXPERIMENTAL RESULTS

> "A thinker sees his own actions as experiments and questions – as attempts to find out something. Success and failure are for him answers above all."
>
> *Friedrich Wilhelm Nietzsche – The Gay Science*

Here three experiments will be covered, using the methods discussed in previous Sections:

Section 3.1.2 : Image denoising using B.P with and without semblance.
Section 3.2 : Image denoising, using B.P with H.P.C-based acceleration.
Section 3.3 : Retrieving and estimating physical parameters from simulated spectra, using a combination of C.N.N and Bayesian Networks.

3.1 Image denoising: problem definition

Understanding image recording and formation processes helps a long way in dealing with any type of image processing and analysis techniques. Biologically speaking, the image formation systems in (most) mammals are done via iris-lens systems and the image sensor or recording systems are done via the retina (Andrews and Hunt (1977)). Given an object $f(\psi, \nu)$, where (ψ, ν) are coordinates of the object, an image formation system transforms the object into an image plane $g(x, y)$, where (x, y) are the coordinates of the image (see Figure 13.). The object-to-image transformation is done by a source of radiant energy (reflected, transmitted or emitted by the object).

There are three image formation principles on which the object-to-image transformation is done. The first principle is **neighborhood processing**. As the object $f(\psi, \nu)$ is transformed into the image plane $g(x, y)$ through radiant energy propagation, the recorded object at some point (ψ, ν) is affected by its neighboring points. So the object-to-image transformation process may be dependent not only a single object point, but (possibly infinite) neighborhood points surrounding the object point.

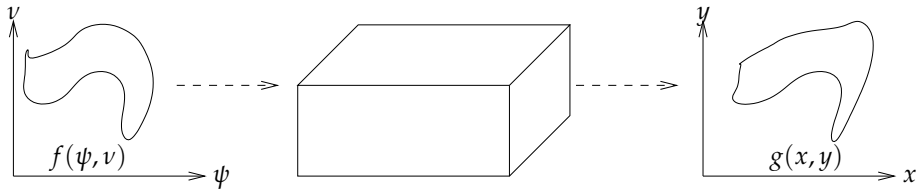


FIGURE 13 Transformation from the object plane to the image plane by an image formation system. The object $f(\psi, \nu)$ is transformed from the (ψ, ν) coordinate system to the image coordinate system (x, y) into an image $g(x, y)$ by some formation system.

The second principle is **nonnegativity**. It is assumed that when the object is transformed into an image, the resulting energy distribution (from the radiant energy propagation) is either positivie or zero.

The third principle is **superposition**. Given two points in the object plane $f_1(\psi, \nu)$ and $f_2(\psi, \nu)$, if only $f_1(\psi, \nu)$ is the radiating energy, then we observe the corresponding measurement in the image plane by the quantity $g_1(\psi, \nu)$ (similarly, if we only observe $f_2(\psi, \nu)$). The image transformation is responsible for the energy distribution in the object plane. Taking this responsibility into account, we can describe the energy distribution in the image plane $g(x, y)$ as a function h of the object $f(\psi, \nu)$, its coordinate points (ψ, ν) and its corresponding points in the image plane (x, y) as

$$g(x, y) = h(x, y, \psi, \nu, f(\psi, \nu)). \tag{39}$$

The function h can be used to represent the energy distribution of points as a superposition, because many transport processes of energy radiation are *additive*. The image formation may be *nonlinear*, in which case the components in the object plane may not be additive, where as the components may be additive in the image plane. In either case of linear or nonlinear image formation, it is possible to describe the image formation process by extending from pointwise to a continuum. This is done by summing up the infinitesimal contributions of points in the image plane due to all contributing points in the object plane. These contributions can be expressed generally as

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y, \psi, \nu, f(\psi, \nu)) d\psi d\nu. \tag{40}$$

3.1.1 Image denoising: program

Now to restore the image using

$$E(\mathbf{X}) = \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{j \in \mathcal{N}_i} f_{i,j}(x_i, x_j),$$

the first step is to compute the initial energies (or data costs) of each pixel state, which the term $\sum_{i \in \mathcal{V}} f_i(x_i)$. After the initial energies have been computed, we construct the multi-scale structure to evaluate $\sum_{j \in \mathcal{N}_i} f_{i,j}(x_i, x_j)$, where the belief messages are computed and propagated from the coarsest level to the finer

level (using the computed unary potentials). Then, from the computed messages, the denoised image is produced. Programs 1 and 2 shows the main body of the denoising program, with the relevant Subprograms are shown in Subprograms 3 and 4.

Progam 1: Image denoising using B.P with C.M.G structure. (Minimization problem (27)).

input : Corrupted image I_0, truncation cost d, scaling level L, number of iterations T, number of states K

output: Restored image $\widetilde{I}$

```
// Define quadratic cost function
```
$f(x,y) \stackrel{\text{def}}{=} \min\{(x-y)^2, d\}$;
```
// Compute unary potentials
```
$E_0 \leftarrow \text{InitialEnergies}(I_0, f, d, K)$;
```
// Approximate posterior with C.M.G structure
according to equation (32), and red-black ordering
(Subprogram 3)
```
$\widetilde{I} \leftarrow \text{ConstructMultiscale}(E_0, f, d, L, T, K)$;

Progam 2: Image denoising using B.P with C.M.G structure and semblance measure (Minimization problem (36)).

input : Corrupted image I_0, semblance radius r, truncation cost d, scaling level L, number of iterations T, number of states K

output: Restored image $\widetilde{I}$

```
// Define quadratic cost function
```
$f(x,y) \stackrel{\text{def}}{=} \min\{(x-y)^2, d\}$;
```
// Define semblance kernel 𝒦 (Subprogram 4)
```
$\mathcal{K}^{z(r)}$;
```
// Compute unary potentials
```
$E_0 \leftarrow \text{InitialEnergies}(I_0, f, \mathcal{K}, d, K)$;
```
// Approximate posterior with C.M.G structure
according to equation (32), and red-black ordering
(Subprogram 3)
```
$\widetilde{I} \leftarrow \text{ConstructMultiscale}(E_0, f, d, L, T, K)$;

Subprogam 3: Computing belief messages.

input : Grid points at level $\mathbb{G}^l$, cost function f, truncation cost d, number of iterations T, number of states K, width $|m^l|$ and height $|n^l|$ of block $\mathbb{G}^l$

output: $m_n^l, m_s^l, m_e^l, m_w^l$

for $t \leftarrow 0$ **to** $T-1$ **do**
  for $x \leftarrow 0$ **to** $|m^l|-1$ **do**
    for $y \leftarrow ((m+t)\ mod\ 2)$ **to** $|n^l|-1$ **do**
      $m_n^l(\mathbb{G}^l(x,y+1), \mathbb{G}^l(x+1,y), \mathbb{G}^l(x-1,y), f, d, K);$
      $m_s^l(\mathbb{G}^l(x,y-1), \mathbb{G}^l(x+1,y), \mathbb{G}^l(x-1,y), f, d, K);$
      $m_e^l(\mathbb{G}^l(x,y+1), \mathbb{G}^l(x1,y-1), \mathbb{G}^l(x-1,y), f, d, K);$
      $m_w^l(\mathbb{G}^l(x,y+1), \mathbb{G}^l(x,y-1), \mathbb{G}^l(x+1,y), f, d, K);$

Subprogam 4: Compute semblance measure α of given pixel $I(x,y)$

input : Image I, semblance radius r, current spatial coordinates (x,y) of I

output: maximum coherency α

// Denumerator variables
$\mathsf{denum}_0 \leftarrow \mathsf{denum}_1 \leftarrow \mathsf{denum}_2 \leftarrow \mathsf{denum}_3 \leftarrow 0;$
// Numerator variables
$\mathsf{num}_0 \leftarrow \mathsf{num}_1 \leftarrow \mathsf{num}_2 \leftarrow \mathsf{num}_3 \leftarrow 0;$
// Semblace variables
$\alpha_0 \leftarrow \alpha_1 \leftarrow \alpha_2 \leftarrow \alpha_3 \leftarrow 0;$
for $i \leftarrow -r$ **to** r **do**
  $\mathsf{num}_0 \leftarrow \mathsf{num}_0 + I(x+i,y);$
  $\mathsf{num}_1 \leftarrow \mathsf{num}_1 + I(x+i,y+i);$
  $\mathsf{num}_2 \leftarrow \mathsf{num}_2 + I(x,y+i);$
  $\mathsf{num}_3 \leftarrow \mathsf{num}_3 + I(x-i,y+i);$
  $\mathsf{denum}_0 \leftarrow \mathsf{denum}_0 + I(x+i,y)^2;$
  $\mathsf{denum}_1 \leftarrow \mathsf{denum}_1 + I(x+i,y+i)^2;$
  $\mathsf{denum}_2 \leftarrow \mathsf{denum}_2 + I(x,y+i)^2;$
  $\mathsf{denum}_3 \leftarrow \mathsf{denum}_3 + I(x-i,y+i)^2;$
$\alpha_0 \leftarrow \frac{\mathsf{num}_0^2}{(2r+1)\mathsf{denum}_0};$
$\alpha_1 \leftarrow \frac{\mathsf{num}_1^2}{(2r+1)\mathsf{denum}_1};$
$\alpha_2 \leftarrow \frac{\mathsf{num}_2^2}{(2r+1)\mathsf{denum}_2};$
$\alpha_3 \leftarrow \frac{\mathsf{num}_3^2}{(2r+1)\mathsf{denum}_3};$
$\alpha \leftarrow max\{\alpha_0, \alpha_1, \alpha_2, \alpha_3\}$

3.1.2 Image denoising: results

Here we compare the B.P by Felzenszwalb and Huttenlocher (2006), against the B.P using the semblance measure. We tested our method with 8bit encoded grayscale images: the well-known Lena and Barbara images. Both images were corrupted with Gaussian random noise with standard deviations $\sigma = \{10, 12, 15\}$. The experiments were carried out using C++11 with an Intel Ivy Bridge i5 processor in a Linux environment (kernel version 5.2.11-1). The code was compiled using native architecture optimization flags. We used two validation metrics to evaluate the denoising results: *Peak-Signal-to-Noise ration* (P.S.N.R) and *Structural Similarity Index* (S.S.I).

The base-line B.P by Felzenszwalb and Huttenlocher (2006) is computed using Program 1 and will be denoted plainly as B.P. Applying the semblance measure on the B.P will denoted as B.P$_r$ and is computed with Program 2, using the following parameters for the semblance kernel $\mathcal{K}^{z(r)}$:

1. semblance radius $r = 1, 2, 3$,
2. threshold level $\beta = 0.3$.

The following parameters for the C.M.G structure, and cost function for unary and pairwise pixel costs, were set for both Programs 1 and 2:

- number of iterations $T = 5$.
- number of levels $L = 5$.
- threshold $d = 200$ for the quadratic truncation cost ($\min\{(x_i - x_j)^2, d\}$).
- number of states $K = 255$ (since we aim to restore a full gray scale image).

Note that the implementer may use any other cost function for evaluating the energy costs. The quadratic cost and parameters r, β, T, L, d were experimentally chosen by trial-and-error.

Figure 14 shows the uncorrupt test images of Lena and Barbara. The P.S.N.R can be found in Tables 1 and 3 for Barbara and Lena images respectively. In Figures 15–17 are shown the denoising experiments using the baseline model and the proposed approach for Barbara, and in Figures 18–20 shows the denoising experiments of Lena.

(a) (b)

FIGURE 14 Original, uncorrupted test images: (a) Barbara; (b) Lena

TABLE 1 Comparing P.S.N.R values between Felzenszwalb and Huttenlocher (2006) and B.P with semblance measure (B.P$_r$). The highlighted quantities denotes the best result column-wise.

		Barbara (512×512)			
		$\sigma = 10$	$\sigma = 12$	$\sigma = 15$	CPU(s)
B.P		22.14	22.13	22.10	10.7
B.P$_r$	$r = 1$	**26.14**	**25.62**	**24.46**	13.7
	$r = 2$	26.10	25.58	24.44	14.3
	$r = 3$	26.01	25.51	24.40	14.8

TABLE 2 Comparing S.S.I values between Felzenszwalb and Huttenlocher (2006) and B.P with semblance measure (B.P$_r$). The highlighted quantities denotes the best result column-wise.

		Barbara (512×512)			
		$\sigma = 10$	$\sigma = 12$	$\sigma = 15$	CPU(s)
B.P		0.586	0.5853	0.5832	10.7
B.P$_r$	$r = 1$	**0.8690**	**0.8184**	**0.7164**	13.7
	$r = 2$	0.8581	0.8177	0.7159	14.3
	$r = 3$	0.8567	0.8167	0.7153	14.8

(a) (b)

(c) (d)

FIGURE 15 (a) Image with additive Gaussian noise $\sigma = 10$. (b) Denoising using Felzenszwalb and Huttenlocher (2006). (c)-(d) Proposed method with span $r = 1, 3$ respectively.

TABLE 3 Comparing P.S.N.R values between Felzenszwalb and Huttenlocher (2006) and B.P with semblance measure (B.P$_r$). The highlighted quantities denotes the best result column-wise.

		Lena (512×512)			
		$\sigma = 10$	$\sigma = 12$	$\sigma = 15$	CPU(s)
B.P		24.30	24.27	24.26	10.4
B.P$_r$	$r = 1$	**26.05**	25.28	24.37	13.4
	$r = 2$	26.00	**25.54**	**25.05**	14.2
	$r = 3$	25.95	25.05	24.32	14.6

(a) (b)

(c) (d)

FIGURE 16 (a) Image with additive Gaussian noise $\sigma = 12$. (b) Denoising using Felzenszwalb and Huttenlocher (2006). (c)-(d) Proposed method with span $r = 1, 3$ respectively.

TABLE 4 Comparing S.S.I values between Felzenszwalb and Huttenlocher (2006) and B.P with semblance measure (B.P$_r$). The highlighted quantities denotes the best result column-wise.

		Lena (512×512)			
		$\sigma = 10$	$\sigma = 12$	$\sigma = 15$	CPU(s)
B.P		0.7237	0.7220	0.**7221**	10.4
B.P$_r$	$r = 1$	**0.8616**	**0.7910**	0.6341	13.4
	$r = 2$	0.8616	0.7909	0.6347	14.2
	$r = 3$	0.8600	0.6347	0.6350	14.6

(a) (b)

(c) (d)

FIGURE 17 (a) Image with additive Gaussian noise $\sigma = 15$. (b) Denoising using Felzenszwalb and Huttenlocher (2006). (c)-(d) Proposed method with span $r = 1, 3$ respectively.

(a) (b)

(c) (d)

FIGURE 18 (a) Image with additive Gaussian noise $\sigma = 10$. (b) Denoising using Felzenszwalb and Huttenlocher (2006). (c)-(d) Proposed method with span $r = 1, 3$ respectively.

(a) (b)

(c) (d)

FIGURE 19 (a) Image with additive Gaussian noise $\sigma = 15$. (b) Denoising using Felzenszwalb and Huttenlocher (2006). (c)-(d) Proposed method with span $r = 1, 3$ respectively.

(a) (b)

(c) (d)

FIGURE 20 (a) Image with additive Gaussian noise $\sigma = 15$. (b) Denoising Felzenszwalb and Huttenlocher (2006). (c)-(d) Proposed method with span $r = 1, 3$ respectively.

3.1.3 Strengths and limitations

When comparing P.S.N.R scores between the proposed approach and the baseline method, the proposed method was able to provide better scores as seen in Tables 1 and 3. Comparing to the respective S.S.I scores in Tables 2 and 4, the proposed method also gave higher scores, with the exception on noise level $\sigma = 15$ with the Lena image. In both Lena and Barbara, there remains noise artifacts with $\sigma = 15$, however qualitatively the textures recovered from both images are satisfactory.

When comparing the Barbara image, the proposed approach was able to reconstruct the diagonal texture stripes from the hood, pants and table cloth. Additionally the chair in the background was reconstructed well preserving the chair's textures. The blurred background in the Lena image provided challenges for the proposed method while the textures from the hat's decorations were well preserved.

The strengths of the proposed approach is that with a small, extra computational overhead, better reconstructions was achieved. The simplicity of the semblance measure is also a strength, since the implementation is simple and can be executed during the unary potential calculation (Program 2, line 6.). Also the proposed method does not require training[1] on many types of images (compared to a generic A.N.N). This makes the proposed B.P method suitable for low-level denoising tasks for arbitrary digital images. However, why the semblance measure works so well could be due to the underlying C.M.G data structure and because of the presence of multiple loops in the network.

The limitations of the proposed approach is that it will not work on higher noise levels. As seen from in Figures 17 and 20, there remains noise artifacts in the image. With higher noise levels the method will leave more artifacts in the image. Comparing to other neurocomputing models, say C.N.Ns, is pointless in the sense because C.N.Ns (or other neural network models) require a lot of data for training. These types of network models also have a large degrees of freedom (the number of parameters) which makes these networks models (trivially) better[2] when a sufficiently large size of training images is provided.

---

1 When using a generic A.N.N architecture, a large and *diverse* set of training images is required, and not only in different scales and rotations.

2 Or paraphrasing Uncle Ben: with great degrees of freedom, comes great (neural) performance.

3.2 High-Performance B.P

3.2.1 Better belief corrections or speeding up computation?

The belief revisions in B.P are done in parallel. However this only means that the beliefs are **updated** in parallel, or simultaneously. For time-critical applications, or machines with limited hardware resources, two considerations must be done if one is to use B.P methods. Either achieving faster convergence to the correct posterior estimate[3], or speeding up the computational steps in the B.P method. In general, the interest in speeding up the computation of the optimization problems in B.Ps is that the number of computed messages is high, which results also in higher resource requirements both in time and space. An interesting observation in solving the combinatorial structure of the parabolas with the quadratic truncation cost, we can still capture meaningful content (such as textures and edges) with fewer pixel states in the problem formulation. (See Figures 21 and 22.) Another interesting effect of the semblance method is that a fewer iterations are needed to obtain a decent result of the image restoration task (Figure 23).

To speed up the convergence of the B.P inference, one approach is to reformulate the optimization problem, such that the number of computed messages is minimized. Another way would be to rewrite the optimization problem, such that the computational problem could be decomposed into independent computational stages and apply *parallel computing*[4] approaches. A third option would be to consider H.P.C approaches (Section 1.7), which is basically combines compiler optimizations, thread and memory optimizations additionally to parallel computing solutions.

All three options stated above are valid and correct in their own right and depends entirely on the situation. One must also consider additional cost of thinking and code implementations to a particular problem when choosing an approach[5].

---

[3] This approach may not be resource friendly.

[4] We separate the terms parallel computing and H.P.C. The reason being, that (in general), parallel computing literature focuses on the *mathematical* formulation (Aki (1989); Greenlaw et al. (1995)) of the problem, while H.P.C focuses more on the implementation and hardware aspects.

[5] Irwing J. Good coins this mentality as *Type 2 rationality* (Good (1983)).

(a) (b)

(c) (d)

FIGURE 21 Reconstruction using 130 states. (a) Regular Lena. (b) Sampling using Felzenszwalb and Huttenlocher (2006). (c)-(d) Proposed method with span $r = 1, 2$ respectively.

(a) (b)

(c) (d)

FIGURE 22 Reconstruction using 190 states (a) Regular Lena. (b) Sampling using Felzenszwalb and Huttenlocher (2006). (c)-(d) Proposed method with span $r = 1, 2$ respectively.

(a) (b)

(c) (d)

FIGURE 23 Resulting restoration with (a) B.P, iteration 1. (b) B.P$_r, r = 1$, iteration 1. (c) B.P, iteration 2.(d)B.P$_r, r = 1$, iteration 2, respectively.

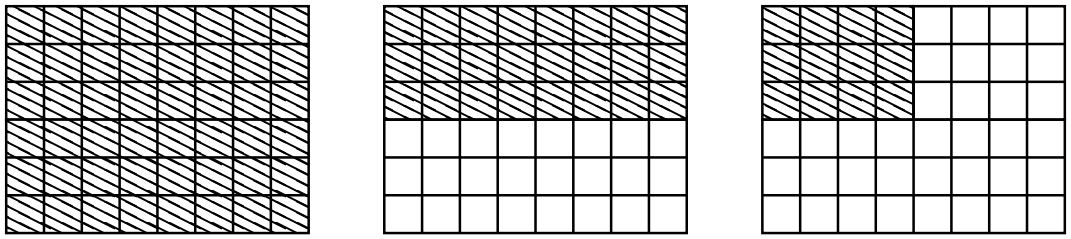


FIGURE 24 Memory optimization approaches. (a) No memory optimization. The whole grid is processed in a brute-force manner with no consideration of data locality. (b) Memory optimization using inner-loop approach: the grid is processed in a particular sized slab, defined by the user. Now grid points in memory are more localized and can be retrieved faster, minimizing cache misses. (c) Memory optimization using double-loop approach: the grid is processed within $k \times k$ blocks. Same principle applies as in (b).

3.2.2 H.P.C: problem formulation

The main bottle-neck is computing the messages on the image grid (Subprogram 3). Here we consider H.P.C approaches which have been applied to solving M.G computations. The main observation (also an exploit) is the fact that the B.P approach by Felzenszwalb and Huttenlocher (2006) utilizes a message computing scheme which is analogous to solving discretized P.D.Es with Gauss-Seidel relaxation with red-black ordering. The aim is to exploit the red-black ordering of the message computation. We consider the following speed-up approaches:

1. **Parallel Brute-force**: A simple "brute-force" approach, where we process the grid points in parallel with no memory optimization.
2. **Inner-loop optimization**: The grid is partitioned into "slabs" with respect to the width of the grid.
3. **Double-loop optimization**: The grid is partitioned into $k \times k$ blocks.

The inner- and double-loop optimization approaches are also parallelized. Figure 24 illustrates the above approaches on a grid of points. To compute the inner- and double-block loop optimization in parallel, one just parallelizes the all loops except the iteration loop.

3.2.3 H.P.C: results

In this experiments, we will consider only speeding up the computation of the message computation section of the B.P inference using the Lena image. Since computing the unary potentials in the model is done separately from the rest of the program, we will omit the B.P inference model with semblance computation. The experiment were carried out with the following hardware:

Subprogam 5: Computing belief messages using inner-loop optimization.

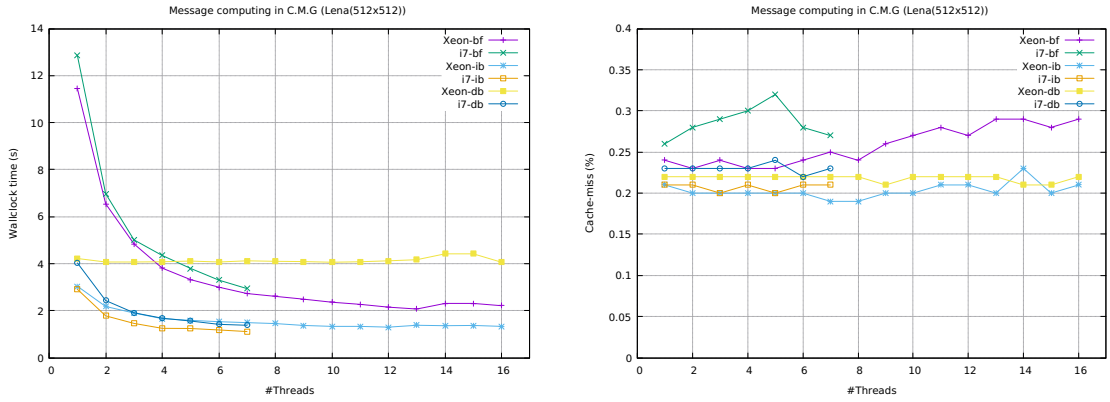
input : Grid points at level $\mathbb{G}^l$, cost function f, number of iterations T, number of states K, width $|m^l|$ and height $|n^l|$ of block $\mathbb{G}^l$

output: Return messages

for $t \leftarrow 0$ **to** $T-1$ **do**
  for $x \leftarrow 1$ **to** $|n^l|-1;\ x = x+4$ **do**
    // Inner block
    for $xb \leftarrow (x+t)\ mod\ 2$ **to** $|m^l|-1;\ xb = xb+8$ **do**
      for $y \leftarrow xb$ **to** $xb+3;\ xb = xb+1$ **do**
        $m_u^l(\mathbb{G}^l(x,y+1),\mathbb{G}^l(x+1,y),\mathbb{G}^l(x-1,y),f,K);$
        $m_d^l(\mathbb{G}^l(x,y-1),\mathbb{G}^l(x+1,y),\mathbb{G}^l(x-1,y),f,K);$
        $m_r^l(\mathbb{G}^l(x,y+1),\mathbb{G}^l(x1,y-1),\mathbb{G}^l(x-1,y),f,K);$
        $m_l^l(\mathbb{G}^l(x,y+1),\mathbb{G}^l(x,y-1),\mathbb{G}^l(x+1,y),f,K);$

– Intel Xeon E5-2670 (2.60GHz)[6], 8 cores with 20MB cache size and 32GB RAM on the PC. Used Linux environment: 4.18.0.
– Intel i7-4910MQ (2.90GHz), 4 cores with cache size 8MB and 32GB RAM on the PC. Used Linux environment: 3.10.0.

All codes were compiled using native architecture optimization flags. Figure 25 shows how increasing the number of threads or computing messages in separate blocks affect the message computation times.



(a) (b)

FIGURE 25 Computation statistics of belief message computations in C.M.G structure. (a) Measuring Wall-clock time (s) vs Threads. (b) Measuring cache-miss vs Threads.

6 Experiments done with this machine had two computing nodes. In other words, two 8-core processors with the potential of using 32 threads. The experiment was limited to 16 threads. It is good practice to reserve at least one thread to handle various background operations done by the operation system, behind the program execution.

Subprogam 6: Computing belief messages using double-loop optimization.

input : Grid points at level $\mathbb{G}^l$, cost function f, number of iterations T, number of states K, width $|m^l|$ and height $|n^l|$ of block $\mathbb{G}^l$

output: Return messages

```
1  for t ← 0 to T − 1 do
2      // Outer block
3      for xb ← 1 to |n^l| − 1; xb = xb + 8 do
4          // Inner block
5          for yb ← ((xb) + t) mod 2 to |m^l| − 1; yb = yb + 8 do
6              for x ← xb to xb + 3; xb = xb + 1 do
7                  for y ← yb to yb + 3; yb = yb + 1 do
8                      m_u^l(G^l(x, y + 1), G^l(x + 1, y), G^l(x − 1, y), f, K);
9                      m_d^l(G^l(x, y − 1), G^l(x + 1, y), G^l(x − 1, y), f, K);
10                     m_r^l(G^l(x, y + 1), G^l(x1, y − 1), G^l(x − 1, y), f, K);
11                     m_l^l(G^l(x, y + 1), G^l(x, y − 1), G^l(x + 1, y), f, K);
```

When adding more threads to the computation, the wallclock time reduced as the number of threads were increased. The only exception was with computing the messages in a double-loop approach, where the wallclock time and cache-misses remained fairly linear. Using the inner-loop optimization, both i7 and Xeon processors managed to reduce the wallclock time and cache-miss when the number of threads were increased. The Xeon processor performed better with the inner-loop optimization compared to the i7.

When it comes to image quality, whatever hardware or computational paralellization should not (ideally) differ from the sequential approach. In Figure 26 the brute-force parallelization does not differ from the sequential approach in both with and without semblance. However when using inner- and double-block loop optimization (Figures 26 and 27) there is a difference when semblance is not applied. When semblance is applied, the restored image is closer to the sequential output but there are block-like artifacts present in the image.

(a) (b)

(c) (d)

FIGURE 26 Image restoration using B.P with and without semblance measure using H.P.C approaches. (a) B.P with brute-force paralellization (P.S.N.R: 24.30). (b) B.P$_{r=1}$, brute-force parallelization (P.S.N.R: 26.05). (c) B.P with inner-block optimization (P.S.N.R: 23.43). (d) B.P$_{r=1}$, with inner-block optimization (P.S.N.R: 24.18).

Subprogam 7: Computing belief messages in parallel in a brute-force manner.

input : Grid points at level $\mathbb{G}^l$, cost function f, number of iterations T, number of states K, width $|m^l|$ and height $|n^l|$ of block $\mathbb{G}^l$

output: Return messages

for $t \leftarrow 0$ **to** $T-1$ **do**
  parfor $x \leftarrow 1$ **to** $|n^l|-1;\ x = x+4$ **do**
    parfor $y \leftarrow (x+t)\ mod\ 2$ **to** $|m^l|-1;\ y = y+4$ **do**
      $m_u^l(\mathbb{G}^l(x,y+1), \mathbb{G}^l(x+1,y), \mathbb{G}^l(x-1,y), f, K);$
      $m_d^l(\mathbb{G}^l(x,y-1), \mathbb{G}^l(x+1,y), \mathbb{G}^l(x-1,y), f, K);$
      $m_r^l(\mathbb{G}^l(x,y+1), \mathbb{G}^l(x1,y-1), \mathbb{G}^l(x-1,y), f, K);$
      $m_l^l(\mathbb{G}^l(x,y+1), \mathbb{G}^l(x,y-1), \mathbb{G}^l(x+1,y), f, K);$

(a) (b)

FIGURE 27 Image restoration using B.P with and without semblance measure using H.P.C approaches. (a) B.P with double-loop optimization (P.S.N.R: 23.04). (b) B.$P_{r=1}$, with double-loop optimization (P.S.N.R: 24.01).

3.2.4 Strength and limitations

Considering data locality in Subprogram 3 can be done in a straight-forward way, and the implementation is easy. The red-black ordering of the Gauss-Seidel approach makes both parallelization and inner-loop (memory) optimization a feasible approach in processing grid points. Double-loop optimization approach is also feasible, however the experiments show, that the choice of processor has an effect in both computation time and the number of cache-misses.

The choice of hardware has an effect on the choice of H.P.C approach when using the red-black ordering on the B.P inference method. Also one should consider border or *halo* effects when processing the grid point in parallel, as seen in Figures 26 and 27. Surprisingly, when using the base-line B.P method, the same cartoonish restoration is not obtained as in the sequential experiments (Section 3.1). This reveals interesting, not yet found, properties of B.Ps on arbitrary networks.

3.3 Retrieving physical parameters from simulated image spectra

In this Section, we combine two interpretations of neurocomputing into a unified tool, namely A.N.Ns and P.G.Ms. This tool consists of two parts:

1. A *neural network architecture* component, which is used to learn and generate data.
2. A P.G.M component, more specifically a Gaussian B.N structure, for doing probabilistic queries on the dependencies between (parameter) propositions.

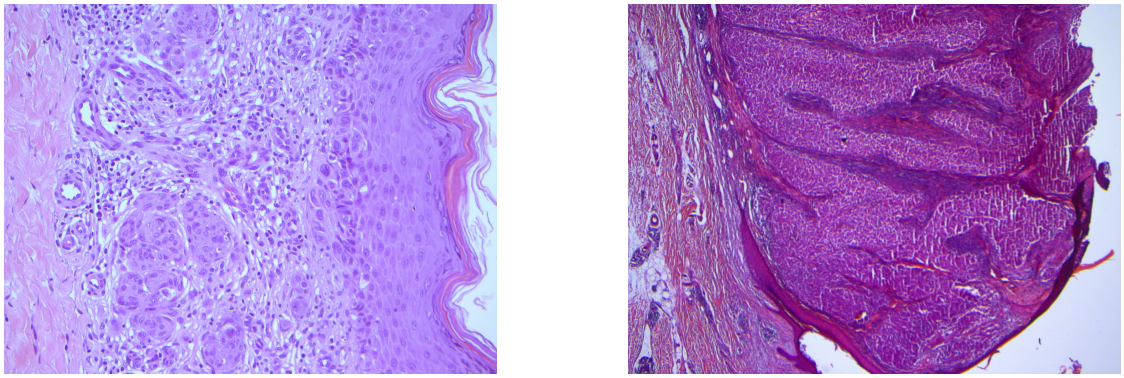
The motivation for the above experiment is to explore how B.Ns could be used as a "modern" expert system (Jones and Graham (1988)). Before the second A.I winter, expert systems were applied in many situations in various industry and medical applications, of which the MYCIN system (Shortliffe (2012); Sotos (1990)) is perhaps the well-known in the medical domain. We will use this expert system framework, in the context of modelling skin cancer from a set of *physical parameters* of the skin. These physical parameters are derived using the Kubelka-Munk light propagation model. From this light propagation model, we retrieve the physical parameters using a C.N.N model, and finally do probabilistic queries for evaluating physical parameter dependencies.

In this experiment the following steps are applied:

1. Simulate physical parameters using the Kubelka-Munk light propagation model (Section 3.3.2).
2. Retrieve the physical parameters by learning an inverse function using a C.N.N architecture (Section 3.3.3).
3. Evaluate physical parameter dependencies with probabilistic queries, using Gaussian B.Ns (Section 3.3.4).

3.3.1 Skin cancer – motivation

Over the past decades there has been an increase of non-melanoma and melanoma incidences, and it is currently estimated that there are 2–3 million non-melanoma and 132000 melanoma skin cancer occurrences every year. One thirds of every cancer diagnosis are diagnosed as skin cancer. Skin cancer also has big societal costs. For example in Sweden in 2005, it was estimated that skin cancer had a total cost of €142.2 million, of which €79.6 million was spent on health services and €62.8 million due to loss of production. Therefore there is a need for establishing and designing effective preventive measures for skin cancer detection to avoid increasing costs and suffering of skin cancer. One possible options for preventive diagnosis of skin cancer is using spectral imaging. Spectral imaging offers both spectral and spatial information of a digital image, obtained by imaging multiple spectrums of the electromagnetic spectrum of a target digital image (Garini et al. (2006)). These spectral and spatial information can be used to detect abnormali-



(a) Melanoma (b) Nodular Melanoma

FIGURE 28 Biopsy samples from H&E stained tissues. Images provided by Noora Neittaanmäki and used with permission.

ties in skin tissue, using hematoxylin eosin (H&E) stained preparations of normal and abnormal skin, benign nevi and melanomas (Figure 28).

3.3.2 Light propagation modelling

We model the major layer structures using the *Kubelka–Munk* (K.M) light propagation model (Kubelka and Munk (1931)), which is based on the relation between scattering and the absorption coefficient of the skin layers, as well as the overall reflectance. The K.M theory describes radiation in diffuse scattering media through an energy transport equation (Jolivot et al. (2013)). Based on this equation, it is possible to make quantitative studies about absorption, scattering and luminescence in diffuse scattering media, as illustrated in Figure 29. A similar approach has been used previously by Jolivot et al. (2013). The optical properties of skin have been studied widely in traditional spectroscopy research[7], where the focus is on the light propagation model. The model consists of the structural properties of several layers, where the most important layers are epidermis, dermis and subcutaneous fat (Anderson et al. (1981)). In principle, it is possible to represent each layer as as a combination of absorption and scattering properties of different chromophores in the skin.

There is a wide range of potential candidates for the model (see for example Jolivot et al. (2013)), but in this experiment, one of the simplest and a numerically inexpensive approach has been chosen. The major chromophore of epidermis is melanin (Norvang et al. (1997)), and in dermis, oxygenated and de-oxygenated haemoglobin.

According to Jacques (2013), the absorption μ_a of each skin layer can be characterized as the following linear mixture of different chromophores:

$$\begin{aligned}\mu_a =& BS\mu_{a,oxy} + B(1-S)\mu_{a,deoxy} + W\mu_{a,water} + \\ & F\mu_{a,fat} + M\mu_{a,mel} + 2.3(C_{bili}\mu_{a,bili} + C_\beta\mu_{a,\beta}), [cm^{-1}]\end{aligned} \tag{41}$$

where $\mu_{a,\_}$ are known absorption coefficients of chromophores. Other parameters

[7] See for examples, the works of Anderson and Parrish (1981) and Jacques (2013)

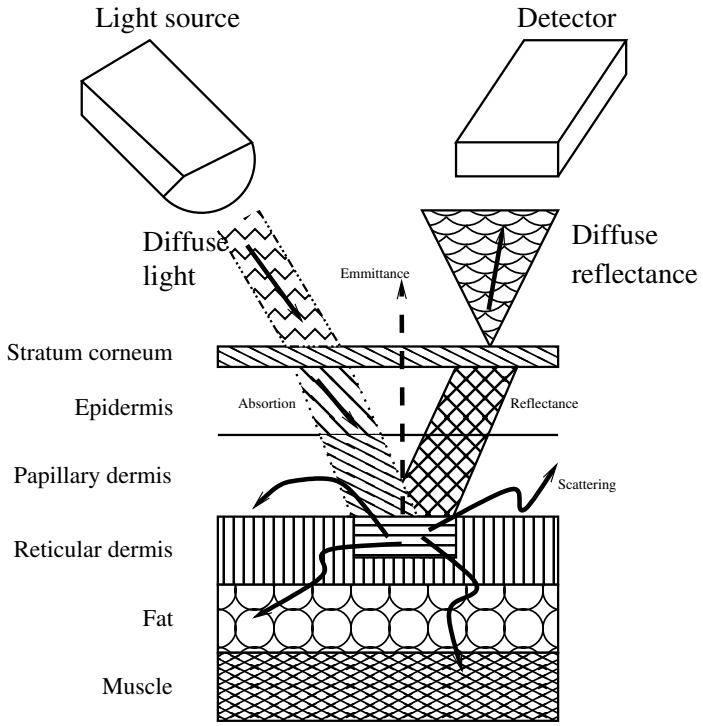


FIGURE 29 Light reflectance of rays in between different skin tissue layers (Pölönen (2013)).

TABLE 5 The seven parameters and their operating range based on literature (Jolivot et al. (2013)).

Skin choromophores and parameters used in the equations	**Symbol**
Hemoglobin oxygen saturation	O
Blood volume fraction	V_B
Scattering fraction	S_F
Scattering power	S_P
Melanosome volume fraction	V_M
Epidermis thickness	T_E
Dermis thickness	T_D

are listed in Table 5[8].

A simplified absorption model is considered: using only the mixture of main chromophores and a baseline spectrum

$$\begin{aligned}\mu_a =& BS\mu_{a,oxy} + B(1-S)\mu_{a,deoxy} \\ &+ M\mu_{a,mel} + (1-M-B)\mu_{a,base}, [cm^{-1}]\end{aligned} \tag{42}$$

where the melanin absorption coefficient is

$$\mu_{a,mel} = 6.6 \times 10^{11}\lambda^{-3.33}[cm^{-1}] \tag{43}$$

and the baseline absorption coefficient is

$$\mu_{a,base} = 0.244 + 85.3\exp^{\frac{-(\lambda-164)}{66.2}}.[cm^{-1}] \tag{44}$$

Both estimates were originally presented by Jacques (1998). When considering non-cancerous skin as a two-layered structure, it follows that $B = 0$ in the epidermal layer, and $M = 0$ in the dermal layer. Cancerous tissue complicates the structure. For example, in advanced malignant melanoma, blood may be found in the epidermal layer, and melanocytes in the dermal layer.

Skin scattering can be approximated using Mie theory (Wriedt (2012)) as follows:

$$\mu_s' = a\left(\frac{\lambda}{500(nm)}\right)^{-b}, [cm^{-1}] \tag{45}$$

where μ_s' is the reduced scattering, a is the scaling factor and b the scattering power. Variance between Mie and Rayleigh scattering is dependent on the diameter of the fibres in the skin. If the majority of skin fibres are very small, then the fraction of Rayleigh scattering f_{Ray} is large. For example, low f_{Ray} could indicate that the amount of collagen is high in a certain spot of skin. The skin anisotropy factor g varies between 0.5 and 0.9. The scattering for each skin layer is $\mu_s = (1-g)\mu_s'$.

The equation for reflectance R for one layer can be represented as a function of layer thickness d_{layer}, absorption μ_a and scattering μ_s (Jolivot et al. (2013)):

$$R_{layer}(\lambda) = \frac{(1-\beta)^2)(\exp^{K_{layer}d_{layer}} - \exp^{-K_{layer}d_{layer}})}{(1+\beta)^2\exp^{-K_{layer}d_{layer}} - (1-\beta)^2\exp^{-K_{layer}d_{layer}}} \tag{46}$$

Transmittance T can be represented as

$$T_{layer}(\lambda) = \frac{4\beta}{(1+\beta)^2\exp^{K_{layer}d_{layer}} - (1-\beta)^2\exp^{-K_{layer}d_{layer}}} \tag{47}$$

where K is the backward flux

$$K_{\text{layer}} = \sqrt{k_{\text{layer}}\left(k_{\text{layer}} + 2 \times s_{\text{layer}}\right)}[cm^{-1}] \tag{48}$$

[8] Haemoglobin absorption coefficients used for this study were downloaded from http://omlc.org/spectra/index.html.

and β is the forward flux

$$\beta_{layer} = \sqrt{\frac{k_{layer}}{k_{layer} + 2s_{layer}}}, \tag{49}$$

and

$$k_{layer} = 2 \times \mu_{a_{layer}}, \qquad s_{layer} = 2 \times \mu_{s_{layer}}.[cm^{-1}] \tag{50}$$

These equations follow from the Kubelka–Munk theory.

The total reflection can be expressed as

$$R_{total} = R_{epidermal} R_{dermal} = R_{epidermal} + \frac{T^2_{dermal} R_{epidermal}}{1 - R_{epidermal} R_{dermal}}. \tag{51}$$

3.3.3 Physical parameter retrieval with C.N.Ns

We follow the modeling approach from Erkkilä et al. (2021). Using the results and parameters of the K.M-model, we aim to train a C.N.N architecture in order to learn an inverse function, which would retrieve parameters from the simulated light propagation model. The training is done in a supervised manner (Section 1.5.2), using a relatively simple architecture model, with *rectified linear activation unit* (ReLU) transfer functions, which is a piecewise function

$$ReLu(\mathbf{w}) = \max\{\mathbf{0}, \mathbf{w}\}, \quad (\mathbf{0} \text{ is a zero vector.}).$$

ReLU functions, particularly ReLu networks (Zou et al. (2020)), are currently popular in practice because of because it makes gradient descent-based optimization easier (Goodfellow et al. (2016); Zou et al. (2020)). Below is a description of the used C.N.N model:

- 1D Convolutional layer, 64 different kernel with kernels size 3 and ReLU activation
- Maximum pooling layer with pooling size 2
- 1D Convolutional layer, 128 different kernel with kernels size 3 and ReLU activation
- Maximum pooling layer with pooling size 2
- 1D Convolutional layer, 256 different kernel with kernels size 3 and ReLU activation
- Maximum pooling layer with pooling size 2
- Fully connected layer with 128 units and ReLU activation
- Dropout with rate 0.5
- Fully connected layer with 64 units and ReLU activation
- Dropout with rate 0.5
- Fully connected layer with 32 units and ReLU activation
- Fully connected layer with 7 unit

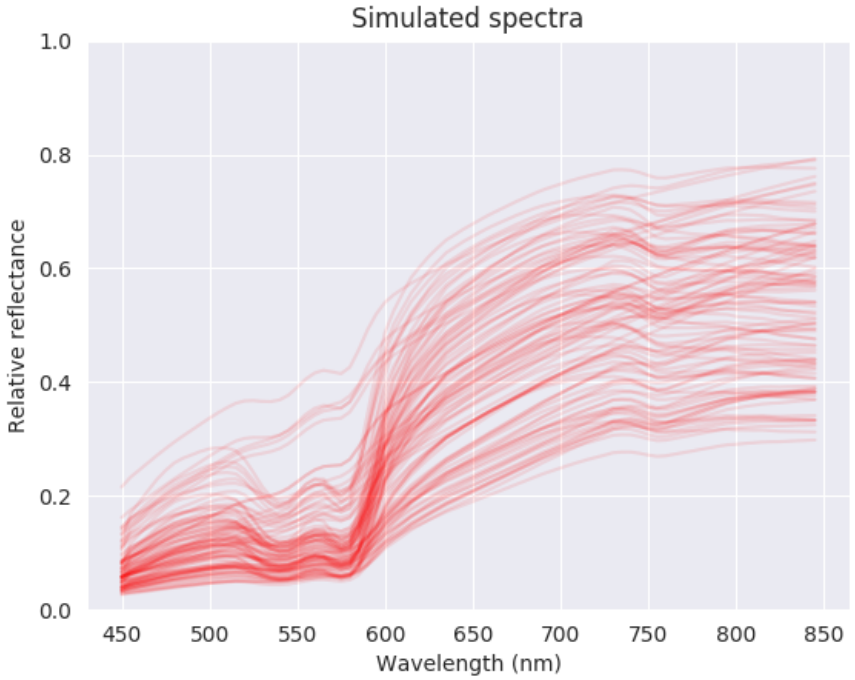


FIGURE 30 Resulting simulated spectra using the K.M light propagation model (Section 3.3.2).

For the experiment, we used the Tensorflow and PgmPy packages. Training of the C.N.N is done with back-propagation (Werbos (1994)) using the Adam stochastic optimization program (Kingma and Ba (2014)). We use a modified loss function for measuring correlation between the true output and the predicted outputs:

$$err = \frac{\sum_i (x_i - y_i)^2}{\sum_i (x_i - \overline{x})^2}, \tag{52}$$

where x_i is the target physical parameter value, y_i is the estimated value and $\overline{x}$ is the average of the output values. The input (Figure 30) and output data are spectra of skin model parameters.

3.3.4 Parameter dependency estimation with Bayesian Networks

After we have modelled the simulated data with the C.N.N model, we assess the dependencies between physical parameters using *Gaussian Bayesian Network* (G.B.N) (instead of discrete states, the proposition are assumed to be from a Gaussian distribution). Instead of variables, we now use the term propositions.

We treat the problem of parameter dependency evaluation as a state-space problem with added stochasticity. The evaluation of the statistical dependencies between propositions and targets are solved in the form of linear regression problem. Because we are dealing with continuous variables (that is, spectral signatures), we will define a *conditional probability distribution* (C.P.D) over the variables. For example, given target Y with parent propositions x_1, x_2 and x_3, then

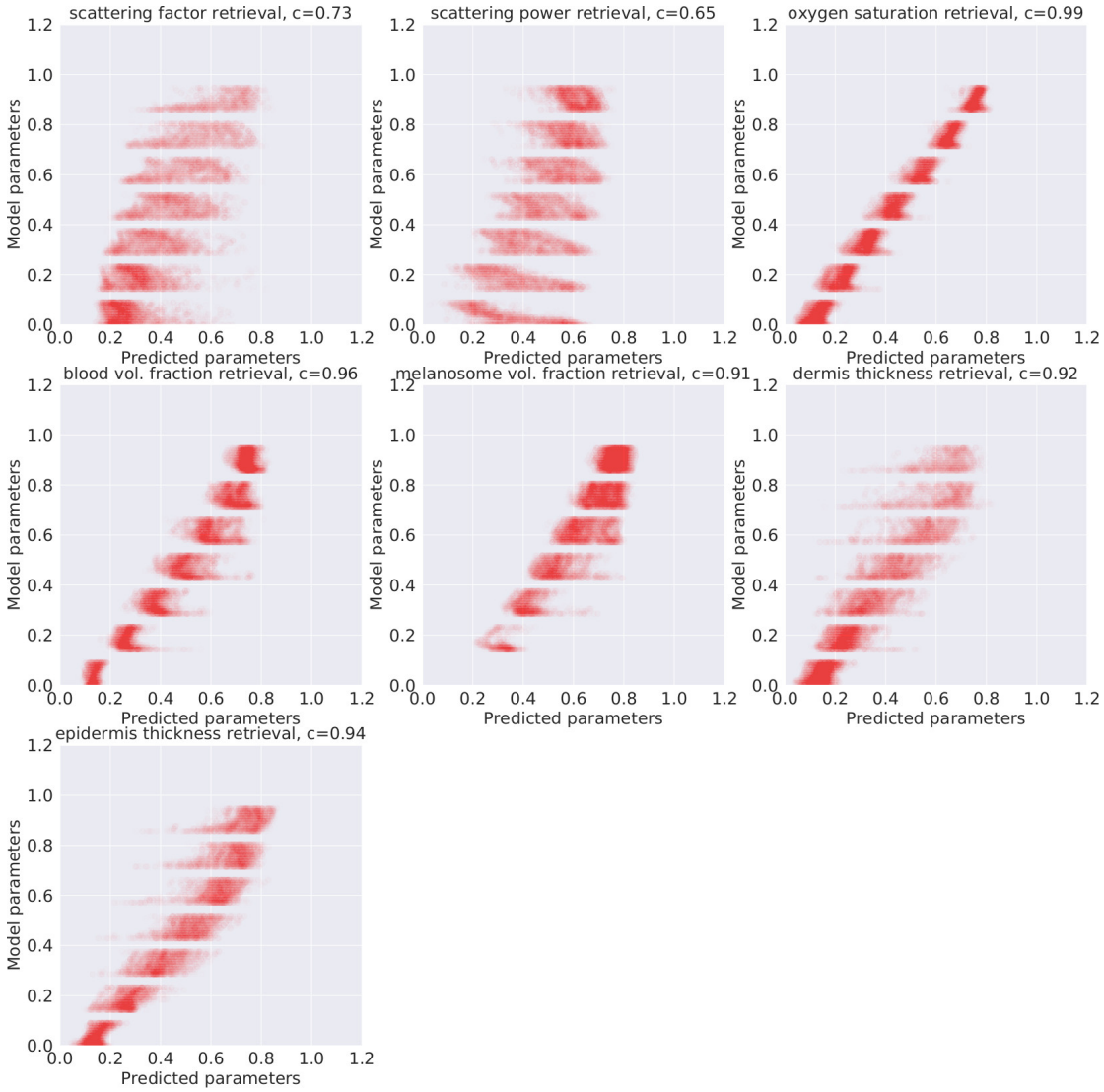


FIGURE 31 Physical parameters retrieved by the C.N.N model from the simulated spectra.

the C.P.D is defined as

$$P(Y \mid x_1, x_2, x_3) = \mathcal{N}(\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3;\ \sigma^2). \tag{53}$$

If Y is a linear Gaussian of its parents x_1, x_2 and x_3, then we have

$$P(Y|\mathbf{x}) = \mathcal{N}(\beta_0 + B^T\mathbf{x};\ \sigma^2). \tag{54}$$

The distribution of Y will be a normal distribution with

$$\bar{y}_Y = \beta_0 + B^T \tag{55a}$$

$$\bar{\sigma}_Y = \sigma^2 + B^T \Sigma B. \tag{55b}$$

The joint distribution over $\mathbf{x}$ and Y is defined by the covariance

$$\Sigma_{x_i Y} = \Sigma_{j=1}^{k} \beta_j \Sigma_{i,j}. \tag{56}$$

With the above equations, a linear G.B.N then defines a joint Gaussian distribution (Koller and Friedman (2009)). The goal here is to solve the latent variables B^T and σ^2, which can be solved using *maximum likelihood estimation* (M.L.E) (Appendix 2.4.1). In order to treat the propositions as continuous variables, a G.N.B requires the propositions to be from a Gaussian distribution. However, C.N.Ns are shown to be connected Gaussian processes[9] (Borovykh (2018)), which makes it possible to solve the latent variables of equations (55)–(56).

Proposition queries. Table 6 shows the resulting probability queries for estimating the dependencies between various physical parameters. The aim is to evaluate the dependencies between the epidermis T_E and dermis T_D layers and the melanosome V_M, to scattering power of the spectra. That is, our target is $Y = S_P$. In order to evaluate Y from both the epidermis and dermis layer as a combination, we have to combine the propositions of each respected layers. That is, we form two new propositions $P_1 \overset{\text{def}}{=} \{T_E, V_M\}$ and $P_2 \overset{\text{def}}{=} \{T_D, V_M\}$.

Figure 32 shows the structure of the G.B.N used to do the queries. Figures 33 – 35 shows the kernel density estimations (K.D.E) of $\{T_E, V_M\}, \{T_D, V_M\}$, $\{T_E, T_D, V_M\}$ propositions respectively from the C.N.N model simulated spectra.

When combining propositions which are at the "same level" in the network (Figure 32), we get stronger dependencies between physical parameters T_E, T_D and V_M for S_P, if we use S_P for inferring melanosome from spectral signatures. That is, observing V_M in either T_E or T_D does not give strong of a correlated dependency, as observed in Table 6.

To exemplify a limitation of the proposed parameter dependency estimation approach, consider the following probability query. Estimating the dependencies between T_D and V_M, with the additional observations of oxygen saturation O. For this query estimation, we must define the following conditional proposition

[9] In fact, given a "large enough" A.N.N, the network will converge to a Gaussian process.

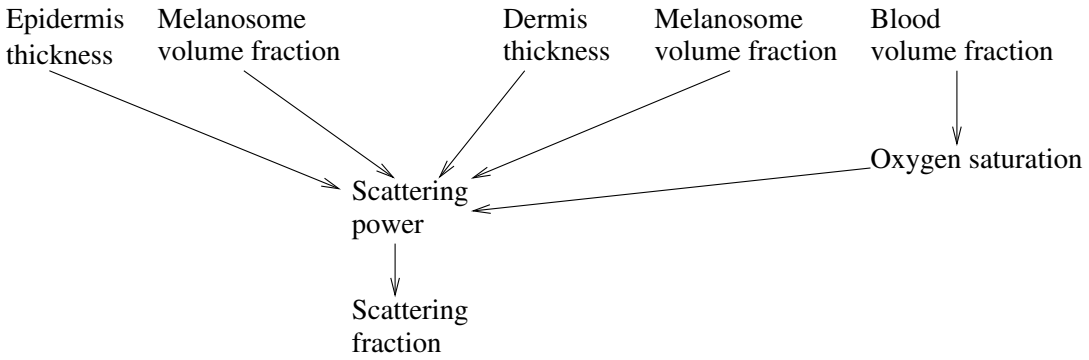


FIGURE 32 The complete G.B.N structure, with all physical parameters, used for solving the example probability queries in Table 6.

TABLE 6 Probabilistic queries results on the parameters using the G.B.N structure in Figure 32.

Parameter dependencies between propositions and target				
Proposition	β_0	β_1	β_2	σ^2
$P(S_P \mid T_E, V_M)$	2.10	0.67	0.21	4.03
$P(S_P \mid T_D, V_M)$	2.15	0.65	0.20	5.01
$P(S_P \mid P_1, P_2)$	1.87	0.88	0.5	4.01
$P(S_P \mid P_3, T_D, V_M)$	8.19	-	-	10.19

$P_3 \stackrel{\text{def}}{=} (O \mid V_B)$ (Figure 32). Afterwards we can solve the query using M.L.E. The result can be found in the last entry in Table 6.

Elaborate propositional queries, such in the case of adding the conditional proposition $(O \mid V_B)$, the estimation is unable to capture the dependencies between propositions[10]. This however expected since we are experimenting with a linear G.B.N to model the variable dependencies. Furthermore our approach is only capable of detecting **linear dependencies** between propositions. Using a B.N model, equipped with a nonlinear estimator, would work better in these types of situations.

[10] That is, to give sensible results.

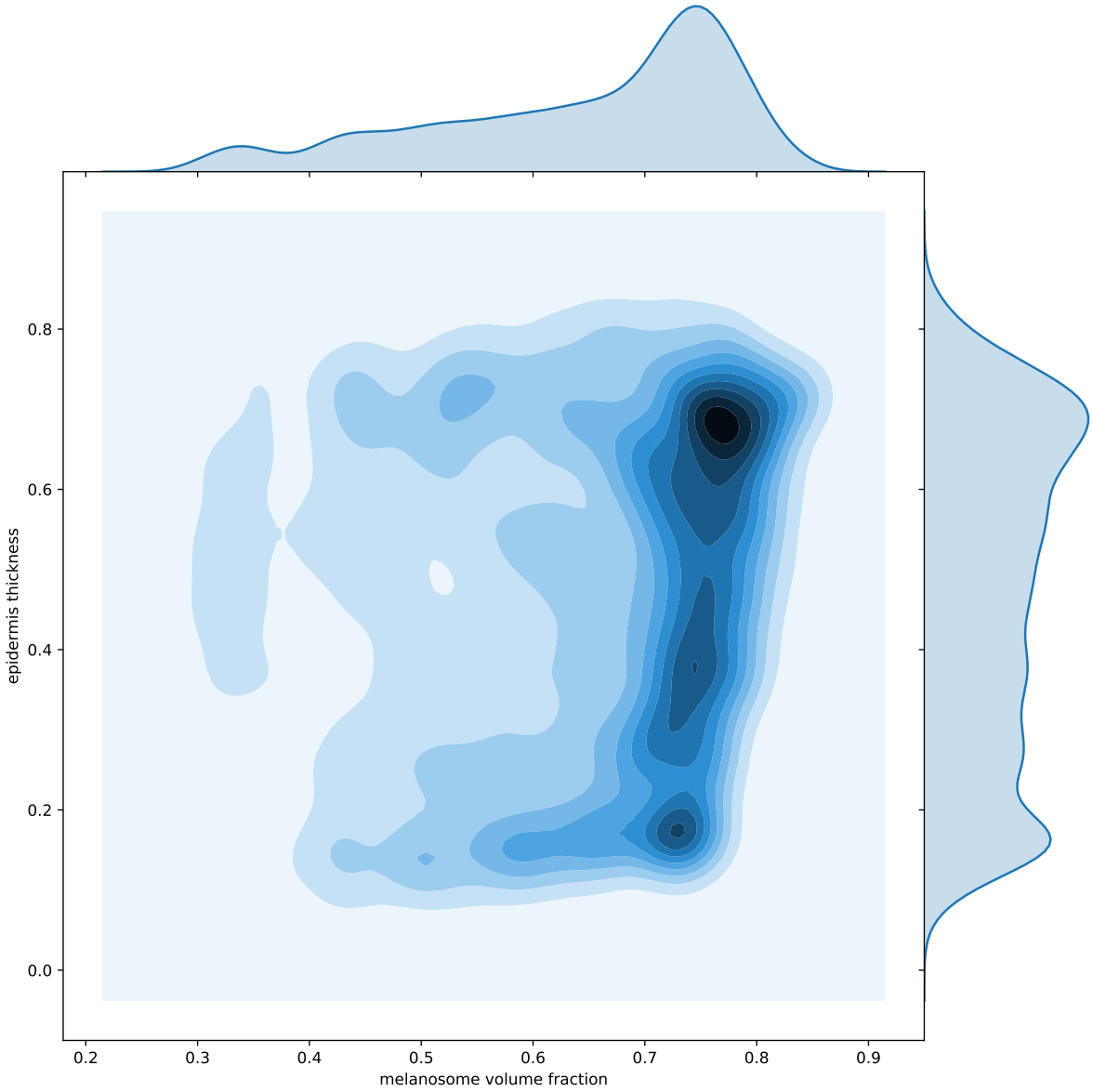


FIGURE 33 K.D.E plots between retrieved parameters epidermis layer thickness and melanosome volume fraction from the C.N.N model, used on the simulated spectra.

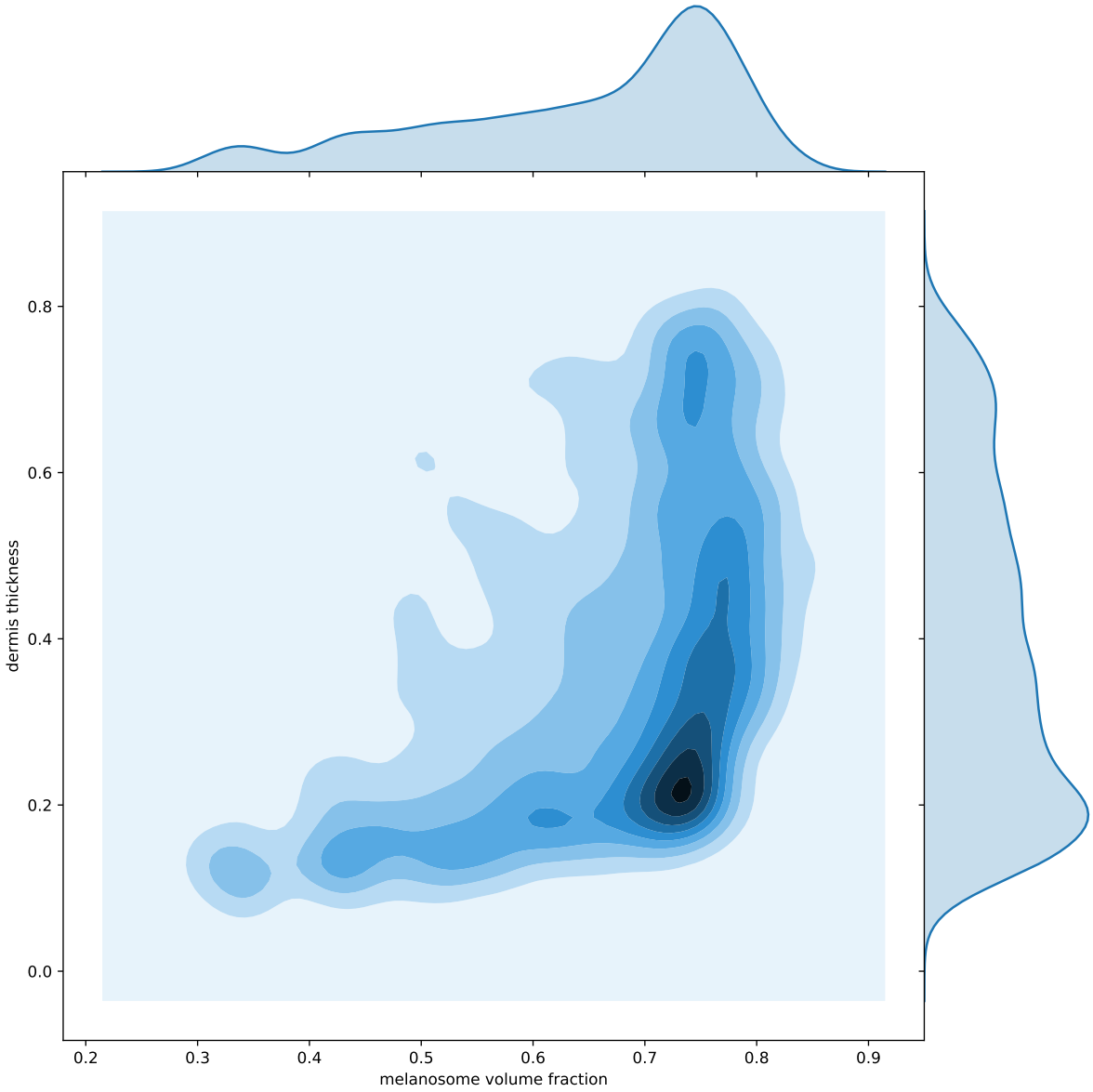


FIGURE 34 K.D.E plots between retrieved parameters dermis layer thickness and melanosome volume fraction from the C.N.N model used on the simulated spectra.

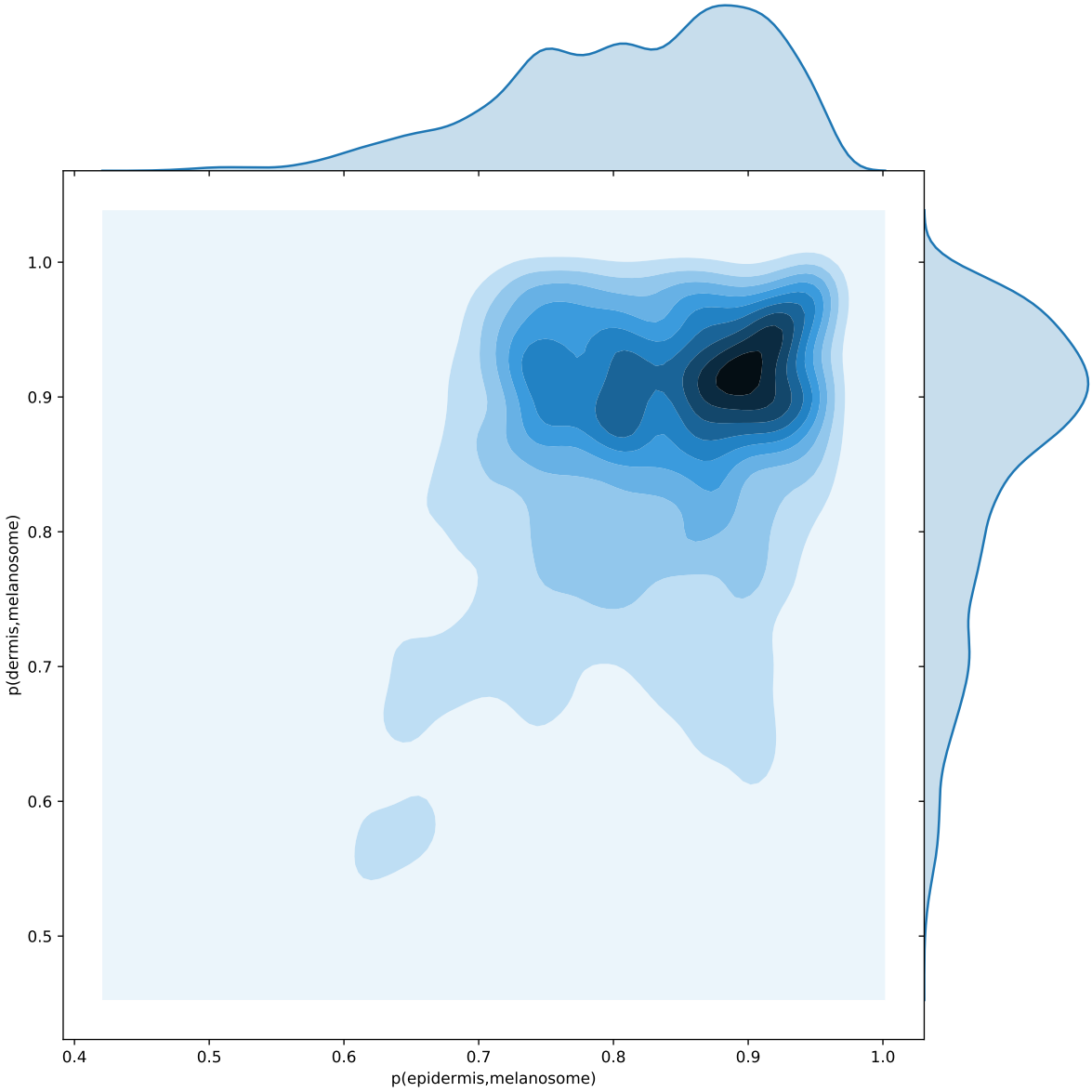


FIGURE 35 K.D.E plot between parameter combinations (epidermis layer thickness, melanosome volume fraction) and (dermis layer thickness, melanosome volume fraction), retrieved with the C.N.N model, used on the simulated spectra.

3.3.5 Strengths and limitations

As mentioned, the main limitation of the proposed approach is that only linear dependencies can be evaluated between propositions. Elaborate queries cannot be solved using this approach. To capture properly the possible nonlinear interactions between propositions, nonlinear solvers should be used for the estimation step. Another limitation is the selection of the A.N.N architecture for doing the parameter dependency estimation.

Note that, the purpose of the proposed approach is not to **explain** the modeling done by the C.N.N. The use of the G.B.N is merely to assist evaluating the used A.N.N model in a certain application domain (medical in this case). There exists methods which are aimed at "explaining" the decision made by a statistical model. Some approaches are for example

- *LIME* (Ribeiro et al. (2016)), which evaluates individual point decision from the given decision surface.
- *Cooperative game theoretic* (Lundberg and Lee (2017); Strumbelj and Kononenko (2014)), where parameters of the model is evaluated using Shapley values (Shapley (1953)).
- Evaluating propagation of learning and decision making in A.N.Ns (Shrikumar et al. (2017); Bach et al. (2015)).

The above approaches have in common that the model parameters or weights are evaluated with respect to target outputs *without taking the context of the application into consideration*[11]. For example, in LIME one can evaluate how individual decisions are made and how the model has weighted the parameters to a given target. If one would make a decision is the used model feasible to a given application, one would have to evaluate all or selectively the decisions made by the model. This is approach could be (possibly) error prone and cumbersome in practice, given that the A.I industry is geared towards "efficiency". The game theoretic approach has the strength of evaluating all parameter pairs or in different permutations (tree structured models only). However the given values of the Shapley values could be difficult to interpret, especially when the combination of the parameters are not taken into the context the application.

The strength of the proposed approach in this Section, is that evaluating variable dependencies to a chosen target variable is model agnostic, and the user can define the dependencies between to a given (known) background information of the application. The approach of using B.Ns to do variable dependencies is well-used in many medical applications (Bucci et al. (2011); Donnat et al. (2020)). That is, using B.Ns for evaluating variable dependencies is not only for simulated data (as we have used). The proposed approach could also be used to model the uncertainties *between classes*: in our skin cancer case, a possible next

[11] Note: feature and *function of a feature* (parameters obtained by the model) are not the same thing. Many of these "model explainer" approaches explain the model parameters, not the feature itself.

step would be to evaluate the parameter dependencies between melanoma and non-melanoma type cancers.

If looking for new, unobserved relations between parameter can also be done with B.Ns in the form of *structured learning* (Koller and Friedman (2009)). The drawback in this approach is that learning a model structure in B.Ns is NP-Complete (Chickering (1996)) (Appendix 3.1), which is done after the user has initially established the structure of the B.N.

4 DISCUSSION & CONCLUSION

> "You have to consider the possibility that God does not like you. He never wanted you. In all probability, he hates you."
>
> *Tyler Durden – Fight Club*

In this research work, algorithmic optimizations in P.G.Ms, more specifically in Belief Propagation, were considered. Applications of B.Ns were experimented in quantifying probabilistically physical parameter dependencies, where the parameters were modelled with a light propagation model and C.N.Ns.

The unary potentials were probabilistically evaluated using a semblance measure. This resulted in improvements in both computational speed (less iterations in B.P inference needed to obtain a solution) and restoration quality, compared to the vanilla approach of using constant weights. H.P.C solutions were also explored to speed-up the message computations in the C.M.G data structure. When using memory optimization techniques such as loop blocking, if one does not use semblance measures in the computation, an incorrect posterior is approximated. This due to not considering the border cases in each block. However, using the semblance measure in the memory optimization one can obtain near correct approximations to the posterior with small artifacts remaining in the solution. With the memory optimization, considerable speed-ups can be obtained if using the appropriate blocking approach.

Such low-level H.P.C approaches have yet to be reported in vision literature. The presented H.P.C approaches are widely used in various high-performance numerical methods. A suitably selected data structure to solve messages on a P.G.M, combined with basic H.P.C approaches, would offer a more pragmatic approach to speed up B.P inferences, compared to laboriously rewrite the optimization problem into a (potentially) more complicated form. Existing parallel computational and H.P.C approaches to B.P on arbitrary networks include computations on computing clusters (Mendiburu et al. (2007)), a MapReduce ¤Lämmel (2008)) approach to B.P inference in parallel (Gonzalez and Guestrin (2009)) and using a shared memory approach to compute B.P inference in parallel (Gonzalez et al. (2011)). Most approaches in the literature are computations on large scale

architectures. The benefit of our approach is that they are suited more for local computation environments and are easier to implement.

A.N.Ns and P.G.Ms were experimented by combining them into evaluate physical parameter dependencies in the context of skin cancer modelling. This type of approach has yet to be reported in the literature (for better or for worse). Initially a C.N.N model was trained to model the physical parameters and then a B.N model was used to evaluate to probabilistic dependency between variables using queries.

4.1 Further considerations

While the presented approaches in this research work does not compete with the state-of-the-art methods in vision tasks, the approaches did reveal some interesting properties. For example, the distance transform sampling done in the C.M.G structure was able to exploit the new combinatorial structure, done by the semblance measure on the unary potentials. The reason the C.M.G structure is a fruitful data structure for solving arbitrary network structures, is that the solutions in an arbitrary network exhibit *periodicity* (Jordan et al. (2001)). The goal of M.G structures is to smooth out the errors between grid layers by filtering out undesirable high and low frequency components. By applying *local fourier analysis*[1] (Trottenberg et al. (2000)), one could design more appropriate M.G or C.M.G structures to aid B.P inference. Also, said local fourier analysis could be another tool to analyze the effects of the coherence measure on vision tasks using the C.M.G structure.

The challenge of using B.Ns to evaluate the variable dependencies from other models, is the construction of an appropriate network structure for the problem. There are computational approaches to solve a suitable structure when given some initial node dependencies, however for continuous variables this is an intractable problem[2]. If in Section 3.3 we would have had discrete state, we could have use for example *Bayes information criterion* (Koller and Friedman (2009)) or *K2* (Cooper and Herskovits (1992)) to evaluate an appropriate structure.

A suitable coherence measure should also be considered when using measurements such as spectral imaging. The simulated K.M model provided spectral signatures for each physical parameter, however if using a spectral imaging device on capturing an image of a skin lesion, other coherence measures should be considered where structural infromation could be retrieved. For example, coherence measure by Gersztenkorn and Marfurt (1999) which is sensitive to lateral changes of the input signals but not to changes in the signal amplitude. Or the measure by Chopra and Marfurt (2007) which can distinguish for example regional, local discontinuities which many coherence measures lack.

---

1 These are tools used to design M.G structures to solve problems in practice.

2 An empirical observation, yielded during experimentation.

YHTEENVETO (SUMMARY IN FINNISH)

Tässä työssä käsiteltiin neurolaskentamenetelmiä konenäkösovelluksissa. Työn tavoitteena oli kehittää laskentamenetelmä hyödyntämällä koherenssimittaa ja rinnakkaislaskentaa todennäköisyysverkkoihin, jolloin todennäköisyysverkkojen lasketansuoritus paranee sekä laadullisesti että laskenta-ajallisesti. Työssä esiteltiin myös menetelmä, jossa todennäköisyysverkkojen ja neuroverkkojen yhdistelmä sovitetaan arvioimaan tilastollisesti neuroverkkojen tuottaman luokittelun tulosta fysikaalisten mallien pohjalta. Sovellusesimerkissä tarkastellaan fysikaalisen mallin muuttujien keskinäistä riippuvuutta ihosyövän tapauksessa. Näitä muuttujia ovat muun muassa hemoglobiinin ja melamiinin määrä sekä ihokerrosten paksuus.

Työssä käsiteltiin myös alemman tason konenäkötehtäviä kuten kohinnan poistoa. Ilman painokerrointa, todennäköisyysverkkoon käytettyjä kustannusfunktiot huomattiin tekevän konservatiivisempia pikselin arvovalintoja verrattuna adaptiivisiin painokerrosmentelmiin. Tavoite oli ymmärtää todennäköisyysverkkojen toimintaa ja löytää halvempia kustannusfunktioita tehtävien hoitamiseksi.

Menetelmän tulokset eivät ole tarkkuudessa kilpailukykyisiä nykyisten tilastollisien menetelmien kanssa, kuten konvoluutioverkkojen tai muiden neuroverkkojen kanssa. Näihin nähden esitetyn menetelmän etuna on, ettei se tarvitse esiopetusta ja se voidaan muokata erilaisiin tehtäviin vaatimattomilla laskentaresursseilla.

Työssä käsiteltiin myös todennäköisyysverkkojen rinnakkaislaskentaa. Haasteena monitasotietorakenteen rinnakkaistamisessa että rinnakkaisuuslaskennan kompleksisuus kasvaa kun mennään hienommasta tasosta karkeampaan verkkotasoon. Tämä vaikuttaa suoraan siihen kuinka montaa laskenta prosessoria voidaan hyödyntää laskentaan. Työssä esiteltiin vaihtoehtoisia rinnakkaistamislaskentatapoja, joita käytetään numeeristen menetelmien laskentojen tehostamiseen

REFERENCES

Aho, A. V. & Hopcroft, J. E. 1974. The Design and Analysis of Computer Algorithms. Addison-Wesley Longman Publishing Co., Inc.

Aki, S. G. 1989. The design and analysis of parallel algorithms. Old Tappan, NJ (USA); Prentice Hall Inc.

Amdahl, G. M. 1967. Validity of the single processor approach to achieving large scale computing capabilities. In Proceedings of the April 18-20, 1967, spring joint computer conference, 483–485.

Anderson, J. A., Rosenfeld, E. & Pellionisz, A. 1988. Neurocomputing, Vol. 2. MIT press.

Anderson, R. R. & Parrish, R. R. 1981. The optics of human skin. Journal of Investigative Dermatology 77, 13–19. doi:10.1111/1523-1747.ep12479191.

Anderson, R. R., Hu, J. & Parrish, J. A. 1981. Optical radiation transfer in the human skin and applications in in vivo remittance spectroscopy. In Bioengineering and the Skin. Springer, 253–265.

Andrews, H. C. & Hunt, B. R. 1977. Digital image restoration. Prentice-Hall.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. & Samek, W. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one 10 (7), e0130140.

Bahorich, M. & Farmer, S. 1995. 3-d seismic discontinuity for faults and stratigraphic features: The coherence cube. The leading edge 14 (10), 1053–1058.

Beeri, C. 1980. On the menbership problem for functional and multivalued dependencies in relational databases. ACM Transactions on Database Systems (TODS) 5 (3), 241–259.

Besag, J. 1974. Spatial interaction and the statistical analysis of lattice systems. Journal of the Royal Statistical Society: Series B (Methodological) 36 (2), 192–225.

Bickhard, M. H. & Terveen, L. 1996. Foundational issues in artificial intelligence and cognitive science: Impasse and solution, Vol. 109. Elsevier.

Bird, R. & De Moor, O. 1996. The algebra of programming. In The algebra of programming.

Bogachev, V. I. 2007. Measure theory, Vol. 1. Springer Science & Business Media.

Borgefors, G. 1986. Distance transformations in digital images. Computer vision, graphics, and image processing 34 (3), 344–371.

Bornemann, F. A. & Deuflhard, P. 1997. Cascadic multigrid methods. In Domain Decomposition Methods in Sciences and Engineering, Procs. 8th International Conference, Beijing, PR China, 205–212.

Borovykh, A. 2018. A gaussian process perspective on convolutional neural networks. arXiv preprint arXiv:1810.10798.

Boykov, Y., V. O. & Zabih, R. 1999. Fast approximate energy minimization via graph cuts. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 1. IEEE, 377–384.

Boykov, Y. & Funka-Lea, G. 2006. Graph cuts and efficient nd image segmentation. International journal of computer vision 70 (2), 109–131.

Brémaud, P. 2013. Markov chains: Gibbs fields, Monte Carlo simulation, and queues, Vol. 31. Springer Science & Business Media.

Bucci, G., Sandrucci, V. & Vicario, E. 2011. Ontologies and bayesian networks in medical diagnosis. In 2011 44th Hawaii International Conference on System Sciences. IEEE, 1–8.

Burt, P. & Adelson, E. 1983. The laplacian pyramid as a compact image code. IEEE Transactions on communications 31 (4), 532–540.

Caetano, T. S., Caelli, T., Schuurmans, D. & Barone, D. A. C. 2006. Graphical models and point pattern matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (10), 1646–1663.

Carlo, C. M. 2004. Markov chain monte carlo and gibbs sampling. Lecture notes for EEB 581.

Chalmond, B. 2003. Modeling and inverse problems in image analysis, Vol. 155. Springer Berlin.

Chan, T. F. & Shen, J. J. 2005. Image processing and analysis: variational, PDE, wavelet, and stochastic methods, Vol. 94. Siam.

Chekuri, C., Khanna, S., Naor, J. & Zosin, L. 2004. A linear programming formulation and approximation algorithms for the metric labeling problem. SIAM Journal on Discrete Mathematics 18 (3), 608–625.

Chickering, D. M. 1996. Learning bayesian networks is NP-Complete. In Learning from data. Springer, 121–130.

Chopra, S. & Marfurt, K. J. 2007. Seismic attributes for prospect identification and reservoir characterization. Society of Exploration Geophysicists and European Association of ....

Choquet, G. 1966. Topology. Academic Press, Series on Pure and Applied Mathematics.

Cohen, D. A., Cooper, M. C., Jeavons, P. G. & Krokhin, A. A. 2006. The complexity of soft constraint satisfaction. Artificial Intelligence 170 (11), 983–1016.

Cooper, G. F. & Herskovits, E. 1992. A bayesian method for the induction of probabilistic networks from data. Machine learning 9 (4), 309–347.

Daubechies, I. 1992. Ten lectures on wavelets, Vol. 61. Society of Industrial and Applied Mathematics.

Davis, M., Sigal, R. & Weyuker, E. J. 1994. Computability, complexity, and languages: fundamentals of theoretical computer science. Elsevier.

Dechter, R. 2003. Constraint processing. Morgan Kaufmann.

Dobkin, D., Lipton, R. J. & Reiss, S. P. 1979. Linear programming is log-space hard for *P*.

Dobkin, D. P. & Reiss, S. P. 1980. The complexity of linear programming. Theoretical Computer Science 11 (1), 1–18.

Donnat, C., Miolane, N., Bunbury, F. d. S. P. & Kreindler, J. 2020. A bayesian hierarchical network for combining heterogeneous data sources in medical diagnoses. arXiv preprint arXiv:2007.13847.

Ehrenfels, C. v. 1890. Über gestaltqualitäten. Vierteljahrsschrift für wissenschaftliche Philosophie 14 (3), 249–292.

Erdös, P. 1959. Graph theory and probability. Canadian Journal of Mathematics 11, 34–38.

Erkkilä, A.-L., Räbinä, J., Pölönen, I., Sajavaara, T., Alakoski, E. & Tuovinen, T. 2021. Using Wave Propagation Simulations and Convolutional Neural Networks to Retrieve Thin Film Thickness from Hyperspectral Images. Springer Verlag. Intelligent Systems, Control and Automation: Science and Engineering. (in press).

Felzenszwalb, P. F. & Huttenlocher, D. P. 2006. Efficient belief propagation for early vision. International journal of computer vision 70 (1), 41–54.

Field, D. J. 1987. Relations between the statistics of natural images and the response properties of cortical cells. Josa a 4 (12), 2379–2394.

Freedman, D. & Drineas, P. 2005. Energy minimization via graph cuts: Settling what is possible. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 2. IEEE, 939–946.

Fujishige, S. 2005. Submodular functions and optimization. Elsevier.

Fukunaga, K. 2013. Introduction to statistical pattern recognition. Elsevier.

Garini, Y., Young, I. T. & McNamara, G. 2006. Spectral imaging: principles and applications. Cytometry Part A: The Journal of the International Society for Analytical Cytology 69 (8), 735–747.

Geiger, A., Roser, M. & Urtasun, R. 2010. Efficient large-scale stereo matching. In Asian conference on computer vision. Springer, 25–38.

Geman, S. & Geman, D. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on pattern analysis and machine intelligence 6, 721–741.

Gersztenkorn, A. & Marfurt, K. J. 1999. Eigenstructure-based coherence computations as an aid to 3-d structural and stratigraphic mapping. Geophysics 64 (5), 1468–1479.

Gibbs, J. W. 1902. Elementary principles in statistical mechanics: developed with especial reference to the rational foundations of thermodynamics. C. Scribner's sons.

Gibson, J. J. 2002. A theory of direct visual perception. Vision and Mind: selected readings in the philosophy of perception, 77–90.

Goldschlager, L. M., Shaw, R. A. & Staples, J. 1982. The maximum flow problem is log space complete for P. Theoretical Computer Science 21 (1), 105–111.

Gonzalez, J., L. Y. & Guestrin, C. 2009. Residual splash for optimally parallelizing belief propagation. In Artificial Intelligence and Statistics, 177–184.

Gonzalez, J., Low, Y. & Guestrin, C. 2011. Parallel belief propagation in factor graphs. Scaling Up Machine Learning: Parallel and Distributed Approaches.

Good, I. J. 1983. Good thinking: The foundations of probability and its applications. U of Minnesota Press.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. 2016. Deep learning, Vol. 1. MIT press Cambridge.

Greenlaw, R., Hoover, H. J., Ruzzo, W. L. et al. 1995. Limits to parallel computation: P-completeness theory. Oxford University Press on Demand.

Gács, P. & Lovász, L. 1981. Khachiyan's algorithm for linear programming. In Mathematical Programming at Oberwolfach. Springer, 61–68.

Hadlock, F. 1975. Finding a maximum cut of a planar graph in polynomial time. SIAM Journal on Computing 4 (3), 221–225.

Hager, G. & Wellein, G. 2010. Introduction to high performance computing for scientists and engineers. CRC Press.

Hecht-Nielsen, R. 1990. Neurocomputing. Addison-Wesley Publishing Co.

Hertz, J. A. 2018. Introduction to the theory of neural computation. CRC Press.

Horn, B. K. P. 1977. Understanding image intensities. Artificial intelligence 8 (2), 201–231.

Hubel, D. H. & Wiesel, T. N. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology 160 (1), 106–154.

Ihler, A. T. & Willsky, A. S. 2005. Loopy belief propagation: Convergence and effects of message errors. Journal of Machine Learning Research 6 (May), 905–936.

Jacques, S. L. 1998. Skin optics. Oregon Medical Laser Center News 1998 (1), 1–9.

Jacques, S. L. 2013. Optical properties of biological tissues: a review. Physics in Medicine and Biology 58 (11), R37. ⟨URL:http://stacks.iop.org/0031-9155/58/i=11/a=R37⟩.

James, W. 1984. Psychology, briefer course, Vol. 14. Harvard University Press.

Johnson-Laird, P. N. 1988. The computer and the mind: An introduction to cognitive science. Harvard University Press.

Jolivot, R., Benezeth, Y. & Marzani, F. 2013. Skin parameter map retrieval from a dedicated multispectral imaging system applied to dermatology/cosmetology. Journal of Biomedical Imaging 2013, 26.

Jones, P. L. & Graham, I. 1988. Expert systems: knowledge, uncertainty and decision. Chapman and Hall.

Jordan, M. I., Sejnowski, T. J. & Poggio, T. A. 2001. Graphical models: Foundations of neural computation. MIT press.

Jordan, M. I. 2004. Graphical models. Statistical science 19 (1), 140–155.

Jähne, B. 2005. Digital image processing, Vol. 4. Springer Berlin.

Khrennikov, A. 2009. Interpretations of probability. Walter de Gruyter.

Kingma, D. P. & Ba, J. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Koenderink, J. J. 1984. The structure of images. Biological cybernetics 50 (5), 363–370.

Koller, D. & Friedman, N. 2009. Probabilistic graphical models: principles and techniques. MIT press.

Kolmogorov, A. N. 1983. On logical foundations of probability theory. In Probability theory and mathematical statistics. Springer, 1–5.

Kolmogorov, V., Thapper, J. & Zivny, S. 2015. The power of linear programming for general-valued CSPs. SIAM Journal on Computing 44 (1), 1–36.

Koster, A. M. C. A., Van Hoesel, S. P. M. & Kolen, A. W. J. 1998. The partial constraint satisfaction problem: Facets and lifting theorems. Operations research letters 23 (3-5), 89–97.

Kozen, D. C. 2012. The design and analysis of algorithms. Springer Science & Business Media.

Kubelka, P. & Munk, F. 1931. Reflection characteristics of paints. Zeitschrift fur Technische Physik 12, 593–601.

Landau, L. D. & Lifshitz, E. M. 1969. Statistical physics, part 2, Vol. 5. Pergamon Press.

Lauritzen, S. L. 2002. Lectures on Contingency Tables.

Lauritzen, S. L. 1996. Graphical models, Vol. 17. Clarendon Press.

Leeuwenberg, E. L. J. & Buffart, H. F. J. M. 1978. Formal theories of visual perception. John Wiley & Sons.

Lengauer, T. & Wagner, K. W. 1990. The binary network flow problem is logspace complete for p. Theoretical Computer Science 75 (3), 357–363.

Li, S. Z. 2009. Markov random field modeling in image analysis. Springer Science & Business Media.

Loshin, D. 1999. Efficient Memory Programming. McGraw-Hill Professional.

Luenberger, D. G. 1973. Introduction to linear and nonlinear programming, Vol. 28. Addison-Wesley Reading, MA.

Lundberg, S. M. & Lee, S.-I. 2017. A unified approach to interpreting model predictions. In Advances in neural information processing systems, 4765–4774.

Lämmel, R. 2008. Google's mapreduce programming model—revisited. Science of computer programming 70 (1), 1–30.

Malfait, M. & Roose, D. 1997. Wavelet-based image denoising using a markov random field a priori model. IEEE Transactions on image processing 6 (4), 549–565.

Mallat, S. 1999. A wavelet tour of signal processing. Elsevier.

Mallat, S. 2001. Applied mathematics meets signal processing. In Challenges for the 21st Century. World Scientific, 138–161.

Maragos, P. 2001. Differential morphology. In Nonlinear Image Processing. Elsevier, 289–329.

Marr, D. 1980. Visual information processing: The structure and creation of visual representations. Philosophical Transactions of the Royal Society of London. B, Biological Sciences 290 (1038), 199–218.

Marr, D. 1982. Vision: A computational investigation into the human representation and processing of visual information. Inc., New York, NY 2 (4.2).

Matheron, G. 1975. Random sets and integral geometry. Wiley, New York.

McCulloch, W. S. & Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics 5 (4), 115–133.

Mendiburu, A., Santana, R., Lozano, J. A. & Bengoetxea, E. 2007. A parallel framework for loopy belief propagation. In Proceedings of the 9th annual conference companion on Genetic and evolutionary computation, 2843–2850.

Mezard, M. & Montanari, A. 2009. Information, physics, and computation. Oxford University Press.

Mine, H. & Osaki, S. 1970. Markovian decision processes. American Elsevier.

Minsky, M. & Papert, S. A. 1969. Perceptrons. MIT Press.

Montanari, U. 1974. Networks of constraints: Fundamental properties and applications to picture processing. Information sciences 7, 95–132.

Murat, C. & Paschos, V. T. 2006. Probabilistic combinatorial optimization on graphs. Wiley Online Library.

Murphy, K., Weiss, Y. & Jordan, M. I. 2013. Loopy belief propagation for approximate inference: An empirical study. arXiv preprint arXiv:1301.6725.

Murphy, K. P., Torralba, A. & Freeman, W. 2003. Using the forest to see the trees: A graphical model relating features, objects, and scenes. Advances in neural information processing systems 16, 1499–1506.

Müller, V. C. & Bostrom, N. 2016. Fundamental issues of artificial intelligence, Vol. 376. Springer.

Neidell, N. S. & Taner, M. T. 1971. Semblance and other coherency measures for multichannel data. Geophysics 36 (3), 482–497.

Norvang, L. T., Milner, T. E., Nelson, J. S., Berns, M. W. & Svaasand, L. O. 1997. Skin pigmentation characterized by visible reflectance measurements. Lasers in Medical Science 12 (2), 99–112.

Oppenheim, A. V. 1978. Applications of digital signal processing. Englewood Cliffs, NJ, Prentice-Hall, Inc., 1978. 510 p.

Pearl, J. 2014. Probabilistic reasoning in intelligent systems: networks of plausible inference. Elsevier.

Pearl, J. & Paz, A. 1985. Graphoids: A graph-based logic for reasoning about relevance relations. University of California (Los Angeles). Computer Science Department.

Pitts, W. & McCulloch, W. S. 1947. How we know universals the perception of auditory and visual forms. The Bulletin of mathematical biophysics 9 (3), 127–147.

Prince, S. 2012. Computer vision: models, learning, and inference. Cambridge University Press.

Pölönen, I. 2013. Discovering knowledge in various applications with a novel hyperspectral imager. University of Jyväskylä.

Quincy, E. & Tomich, D. 1985. Image enhancement using coherence processing with applications to seismograms. In ICASSP'85. IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 10. IEEE, 188–191.

Ribeiro, M. T., Singh, S. & Guestrin, C. 2016. "why should i trust you?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 1135–1144.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review 65 (6), 386.

Sapiro, G. 2006. Geometric partial differential equations and image analysis. Cambridge university press.

Schrijver, A. 2003. Combinatorial optimization: polyhedra and efficiency, Vol. 24. Springer Science & Business Media.

Serra, J. 1983. Image analysis and mathematical morphology. Academic Press, Inc.

Shapley, L. 1953. A value for n-person games. Contributions to the Theory of Games 2 (28), 307-317.

Shlezinger, M. I. 1976. Syntactic analysis of two-dimensional visual signals in the presence of noise. Cybernetics and systems analysis 12 (4), 612–628.

Shortliffe, E. 2012. Computer-based medical consultations: MYCIN, Vol. 2. Elsevier.

Shrikumar, A., Greenside, P. & Kundaje, A. 2017. Learning important features through propagating activation differences. arXiv preprint arXiv:1704.02685.

Simpson Jr, S. M. 1967. Traveling signal-to-noise ratio and signal power estimates. Geophysics 32 (3), 485–493.

Smyth, P., Heckerman, D. & Jordan, M. I. 1997. Probabilistic independence networks for hidden markov probability models. Neural computation 9 (2), 227–269.

Sotos, J. G. 1990. Mycin and neomycin: two approaches to generating explanations in rule-based expert systems. Aviation, space, and environmental medicine 61 (10), 950–954.

Strang, G. & Nguyen, T. 1996. Wavelets and filter banks. SIAM.

Strumbelj, E. & Kononenko, I. 2014. Explaining prediction models and individual predictions with feature contributions. Knowledge and Information Systems 41 (3), 647–665.

Sun, J., Zheng, N.-N. & Shum, H.-Y. 2003. Stereo matching using belief propagation. IEEE Transactions on pattern analysis and machine intelligence 25 (7), 787–800.

Teuscher, C. & Sanchez, E. 2001. A revival of turing's forgotten connectionist ideas: exploring unorganized machines. In Connectionist Models of Learning, Development and Evolution. Springer, 153–162.

Tjelmeland, H. & Besag, J. 1998. Markov random fields with higher-order interactions. Scandinavian Journal of Statistics 25 (3), 415–433.

Torralba, A., Murphy, K. P. & Freeman, W. 2004. Contextual models for object detection using boosted random fields. Advances in neural information processing systems 17, 1401–1408.

Trottenberg, U., Oosterlee, C. W. & Schuller, A. 2000. Multigrid. Elsevier.

Valiant, L. G. 1980. Reducibility by algebraic projections. University of Edinburgh, Department of Computer Science.

Vapnik, V. N. 1998. Statistical learning theory. John Wiley & Sons, Inc.

Vazirani, V. V. 2013. Approximation algorithms. Springer Science & Business Media.

Verma, T. & Pearl, J. 1990. Causal networks: Semantics and expressiveness. In Machine intelligence and pattern recognition, Vol. 9. Elsevier, 69–76.

Warren, W. H. 2012. Does this computational theory solve the right problem? marr, gibson, and the goal of vision. Perception 41 (9), 1053–1060.

Werbos, P. J. 1994. The roots of backpropagation: from ordered derivatives to neural networks and political forecasting, Vol. 1. John Wiley & Sons.

Winkler, G. 2012. Image analysis, random fields and Markov chain Monte Carlo methods: a mathematical introduction, Vol. 27. Springer Science & Business Media.

Witkin, A. P. 1987. Scale-space filtering. In Readings in Computer Vision. Elsevier, 329–332.

Wolpert, D. H. & Macready, W. G. 1997. No free lunch theorems for optimization. IEEE transactions on evolutionary computation 1 (1), 67–82.

Won, C. S. & Derin, H. 1992. Unsupervised segmentation of noisy and textured images using markov random fields. CVGIP: Graphical models and image processing 54 (4), 308–328.

Wriedt, T. 2012. Mie theory: a review. In The Mie Theory. Springer, 53–71.

Xie, J., Xu, L. & Chen, E. 2012. Image denoising and inpainting with deep neural networks. In Advances in neural information processing systems, 341–349.

Yilmaz, Ö. 2001. Seismic data analysis: Processing, inversion, and interpretation of seismic data. Society of exploration geophysicists.

Zhang, L. & Ji, Q. 2009. Image segmentation with a unified graphical model. IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (8), 1406–1425.

Zou, D., Cao, Y., Zhou, D. & Gu, Q. 2020. Gradient descent optimizes over-parameterized deep relu networks. Machine Learning 109 (3), 467–492.

APPENDIX 1 IMAGE MODELS

This appendix gives a brief introduction to different image processing and analysis methods. We follow Chan and Shen (2005) in this Appendix unless stated otherwise.

APPENDIX 1.1 Mathematical Morphology

The study of *mathematical morphology* (M.M) began with Georges Matheron's study of random sets (Matheron (1975); Serra (1983)). Modelling objects in an image can be done in either on a continuous $2D$ plane in $\mathbb{R}^2$, or in a discrete $2D$ lattice in $\mathbb{Z}^2_+$. An object A can be identified with a binary characteristic function:

$$1_A(a) \stackrel{\text{def}}{=} \begin{cases} a & \text{when } a \in A \\ 0 & \text{when } a \notin A \end{cases}$$

A *morphological* transformation is done with an operator T, which gives a mapping between objects A and B, such that

$$T(A) \stackrel{\text{def}}{=} B \leftarrow A. \tag{57}$$

The operator T does a local mapping between A and B: evaluating a pixel a, we determine does a belong to B according the local behaviour of $a \in A$ in some chosen neighborhood of A itself.

There is two basic operation in M.M, from which other operations can be derived from. These two operations are known as *erosion* E and *dilation* D. Both operations depend on a local neighborhood template S, which is known as a *structuring element*. A basic structuring element could be, for example, a 3×3 square template:

$$S \stackrel{\text{def}}{=} \{(i,j) \in \mathbb{Z}^2_+ \ : \ i,j = -1,0,1\}.$$

With the above template S, we can define E and D operations for defining object boundaries in an integer lattice $\mathbb{Z}^2$ as

$$D_S(A) \stackrel{\text{def}}{=} \{a \in \mathbb{Z}^2 \ : \ y + S \cap A \neq \emptyset\}$$
$$E_D(A) \stackrel{\text{def}}{=} \{a \in \mathbb{Z}^2 \ : \ y + S \subset A\}.$$

We can also use morphological operators on grayscale or color images by applying the above principles to their respective *level-sets*.

APPENDIX 1.2 Fourier methods

Fourier and spectral methods are classic tools in signal and image processing (Oppenheim (1978)). Given a continuous image u as a function in a rectangular domain $\Omega \stackrel{\text{def}}{=} (0,1)^2 \subset \mathbb{R}^2$ with periodic extension L. Then we can encode the information of u into coefficients of its Fourier series:

$$c_{n0,n1} = \langle u(\vec{x}), \exp\{i2\pi < x, \vec{n} >\}\rangle_{L^2(\Omega)}, \; n_0, n \in \mathbb{Z}_+^2, \; \vec{x} = (x_0, x_1) \in \Omega.$$

In a discrete case, Ω is defined as a square matrix $(0, n-1) \times (0, n-1)$, $n \in \mathbb{Z}_+^2$. Then we can encode u using a *discrete Fourier transformation*:

$$c_{n0,n1} = \sum_{\vec{j} \in \Omega} u_{\vec{j}} \exp\{\vec{j} \frac{2\pi}{N} \vec{j}, \vec{n} >\}, \vec{j} = (j_0, j_1), \vec{n} = (n_0, n_1) \in \Omega.$$

Fourier methods have wide applicability in many image processing tasks, such as linear filtering and image compression[1].

APPENDIX 1.3 Wavelet and Space-Scale methods

In the 1989s, *Wavelet transforms* challenged Fourier transform methods. In Fourier transformations, long-range spatial information are mixed, causing indiscriminate responses to local visual features. Wavelets organizes the locality of features with respect to different scales (Daubechies (1992); Mallat (2001)).

A wavelet representation of an image u is defined as

$$c_\alpha \stackrel{\text{def}}{=} < u, \psi_\alpha >, \; \alpha\Lambda,$$

where ψ_α are the wavelets indexed by the set Λ. Each wavelet possesses a similar (psychological) differential property as the human vision system[2]:

$$< 1, \psi_\alpha >= 0, \; \forall \alpha \in \Lambda.$$

The above property implies that differentiation is constant for featureless images. Generally, wavelets often satisfies

$$< x_0^{j_0} x_1^{j_1}, \psi_\alpha >= 0, \; \forall j_0 + j_1 \leq m, \; m \in \mathbb{Z}_+.$$

1 The JPEG image compression method is a well-known example.

2 Cognitive and physiological results given by Field (1987) and Hubel and Wiesel (1962).

which is called the *vanishing-moment condition*. This is an annihilation property, which implies the coefficients of the wavelets responds to important visual cues (such as edges) but are negligible to piecewise smooth cues (Strang and Nguyen (1996); Mallat (1999)).

APPENDIX 1.4 Stochastic modelling

Statistical methods are more suitable for images which are stochastic in nature. This stochastic nature of an image u can arise from:

1. The observed image u and given some random effect $\mathcal{X}$ gives the composition u_0 as
$$u_0 = F(u, \mathcal{X}),$$
where F is either a deterministic or non-deterministic function. For example, $F(u, \mathcal{X})$ could represent Gaussian noise or a Poisson process.
2. Treat each composition u_0 as a random field.

Stochastic methods are the ideal choice when dealing with images with statistical nature. For example, statistical pattern recognition or learning theory are important methods for estimating signals and parameters.

Observing an image u_0 with latent variables caused by the (non-)deterministic function F, *a Bayesian inference* of F is solved by the *maximum a posteriori* (M.A.P) probability:
$$p(F|u_0) = \frac{p(F)p(u_0|F)}{p(u_0)}.$$

$p(F)$ is called the *prior* model, specifying independently of u_0 a priori bias among targeted patterns. $p(u_0|F)$ is the conditional probability of how u_0, describing the distribution once F is specified. $p(u_0)$ is a normalizing constant and plays no role in the computation. Estimating F without prior knowledge, M.A.P aims to maximize the *maximum likelihood* of
$$F^* = \arg\max_F p(u_0|F),$$

However in high-dimensional images the number of degrees of freedom is large. For this reason, prior knowledge becomes important for solving the M.A.P estimation effectively. Combining both prior and data knowledge, M.A.P estimation can be done, such that:
$$F^* = \arg\max_F p(F|u_0) = \arg\max_F p(u_0) \times p(u_0|F).$$

The above Bayesian principle emerges frequently in various image processing and analysis models.

APPENDIX 1.5 Variational methods

Variational methods can be viewed as a *deterministic* reflection of the Bayesian approach. This reflection comes from mirroring the Gibb's formula in statistical mechanics (Landau and Lifshitz (1969)):

$$p(F) = \frac{1}{Z}\exp\{-\beta E(F)\},$$

where $\beta \stackrel{\text{def}}{=} 1/(kT)$ denotes the reciprocal of temperature T multiplied by the Boltzmann constant k and Z denotes the Zustandssumme of the system. Gibb's formula expresses directly the likelihood of the prior $p(F)$ of a configuration F, with respect to its "energy" $E(F)$. For any given T, the Bayesian M.A.P estimation becomes the *minimization* of the *posterior* energy:

$$E(F|u_0) = E(F) + E(u_0|F).$$

If F and u_0 belong to a certain functional space, for example *Sobolev* or *bounded variation* space, the posterior minimization leads naturally to variational models.

APPENDIX 1.6 Partial Differential Equations

The power of *partial differential equation* (P.D.E) based models comes from that:

1. P.D.Es offers adequate mathematical descriptions of continuous models in various scientific fields, and can be used to simulate many dynamic (and equilibrium) phenomena in images (for example, diffusion or transport models). From P.D.Es, one can formulate variational models and then apply them to imaging tasks.
2. Many variational problems, or their regularized approximations, can be effectively computed by transforming the problems into an *Euler-Lagrange* form.

For example, using a variational model approach for image restoration could be estimating F from u_0 by solving

$$u^* = \text{argmin } E(u|u_0) = argmin\frac{\alpha}{2}\int_\Omega |\nabla u(x)|^2 dx + \frac{\lambda}{2}\int_\Omega (u(x) - u_0(x))^2 dx,$$

where the weights α and λ are inversely proportional to the variances of the observed data. Using calculus of variation, the above variational model can transformed into the following elliptic boundary value problem:

$$-\alpha\Delta u(\vec{x}) + \lambda u(\vec{x}) = u_0,\ \vec{x} = (x_0, x_1) \in \Omega,\ \frac{\partial u}{\partial v} = 0,\ \text{along } \partial\Omega,$$

with some suitable initial condition $u(\vec{x}, t = 0)$. However, models based on P.D.Es do not always result in a variational model. P.D.E models can also include *geometric* features in the model, resulting into a *geometric P.D.E* (Sapiro (2006)). Geometric P.D.Es can be constructed by optimizing either:

1. Some global geometric quantities in a variational setting (for example, length or area).
2. Geometric invariance under certain transform groups.

APPENDIX 1.7 Intrinsic connection between different methods

The intrinsic connection between different methods will be presented in the case of image restoration. Assume we observe an image u_0, which is corrupted with Gaussian white noise. That is, u_0 is a composition of the form

$$u_0 = F(u, \mathcal{X}) = u + \mathcal{X},\ \mathcal{X} = n \text{ (white Gaussian noise of mean 0.)}$$

To recover the image u using Bayesian M.A.P estimation, the goal is to maximize the posterior distribution (or likelihood):

$$u^* = \underset{u}{argmax}\ p(u, |u_0) = \arg\max p(u)p(u_0|u). \tag{59}$$

The performance of the above M.A.P estimator is mostly dependent on the prior likelihood $p(u)$ of the image.

If we use Gibb's ensemble methods for solving the M.A.P estimation of equation (59), we then dependent on the prior $p(u)$ in the form of an energy function $E(u)$. The M.A.P estimation then becomes a variational model, when we use a standard Cartesian topological structure on the underlying pixel lattice Ω, and the energy $E(u)$ is built upon dipolar quadratic pontentials. The variational model of the M.A.P estimation is then of the form:

$$\min E(u|u_0) = \frac{1}{2}\int_\Omega |\nabla u(x)|^2 dx + \frac{\lambda}{2}\int_\Omega (u(x) - u_0(x))^2 dx. \tag{60}$$

The integrals of equation (60) are to be read as discrete summations over the (Cartesian) pixel lattice Ω.

The variational form of the M.A.P estimation can be converted into a P.D.E model, by applying the first variation $u \to u + \delta u$ to the energy function $E(u|u_0)$. From which we obtain the following Euler-Lagrange equation

$$\frac{\partial E(u|u_0)}{\partial u} = -\Delta u + \lambda(u - u_0) + \left.\frac{\partial u}{\partial n}\right|_{\partial\Omega}, \tag{61}$$

in a distributional sense. In this model, the boundary term $\partial\Omega$ is an element of a Hilbert space $L^2(\partial\Omega, \mathcal{H}^1)$. That is, all boundary functions which are square integrable with respect to the $1D$ Hausdorff measure $\mathcal{H}^1$ of $\partial\Omega$. One can use either gradient descent time evolution or solving the equilibrium equation directly, yielding the P.D.E equations:

$$\begin{cases} u_t = \nabla u + \lambda(u_0 - u) & \text{(Gradient descent time evolution)} \\ -\nabla u + \lambda u = u_0. & \text{(Equilibrium equation)} \end{cases}$$

Here, the boundary term for both equations is the Neumann boundary condition. From the equilibrium equation $-\nabla u + \lambda u = u_0$, the optimally denoised estimator of u is given as:

$$u_0 = u + \frac{-\nabla u}{\lambda} = u + w,\ \lambda > 0, \tag{62}$$

and this can be understood as decomposing the given image u_0 into the components u and w. Here u belongs to the Sobolov space H^1, making u a smooth or regular component. In contrast, w is the distributional Laplacian of u : $w = -\nabla u/\lambda$, making w an oscillatory component containing the discontinuities of u. Generally, w belongs only to $L^2(\Omega)$.

The combination of locality, oscillation and strong responses of discontinuity components, makes w a generalized wavelet projection of u_o, encoding detailed features. The smoother part u behaves similarly to a multiscale projection in the multiresolution setting in wavelet theory[3]. This interpretation leads to the wavelet methods.

To adjust equation (62) into a multiscale or wavelet form, we set

$$\lambda \stackrel{\text{def}}{=} \frac{1}{h} \quad \text{and} \quad u_0 \stackrel{\text{def}}{=} G_{\sqrt{h}} * u,$$

where $G_\sigma \stackrel{\text{def}}{=} G_1(\vec{x}/\sigma)/\sigma^2$, with $G_1(\vec{x}) = G_1(x_0, x_1)$ denoting the canonical $2D$ Gaussian (Burt and Adelson (1983)).

APPENDIX 1.8 M.A.P estimation in P.G.Ms

(The reader is reminded, that this Section uses the concepts defined in Section 1.4.3.) In a computer vision and image processing context, the M.A.P estimation problem becomes an optimization problem (Appendix 1.4): we are interested in finding the most probable configuration of a discrete set of labels $\mathcal{L}$, associated with a hypergraph $\mathcal{H} \stackrel{\text{def}}{=} (\mathcal{V}, \mathcal{C}, \mathcal{E})$, where $\mathcal{V}$ is the set of variables (or nodes) and $\mathcal{C}$ is a set of cliques in $\mathcal{H}$. The most probable configuration is determined over the

[3] The components u and w may not need to be exactly orthogonal.

joint variable distribution of r.vs $\mathbf{X} = (x_i)_{i\in\mathcal{V}}$, which are indexed by $i \in \mathcal{V}$. The notion of a *clique* is important in solving M.A.P problems in P.G.Ms. A clique is a maximal subset of variables, that is, one subset of variables are not contained within any larger subset of propositions. Each clique $c \in \mathcal{C}$ is associated with a potential function $g_c(x_c) \in \mathbb{Q}_+$ over the possible configurations x_c, and can be factorized as:

$$p(\mathbf{X}) = \frac{1}{Z}\prod_{c\in\mathcal{C}} g_c(x_c), \tag{63}$$

where Z is the Zustandssumme of the P.G.M, and $p(\mathbf{X})$ is the joint distribution of the configuration x_c. The cliques of the P.G.M encodes the prior information between the interactions of variable subsets. Thus the M.A.P estimation problem becomes

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} p(\mathbf{X}). \tag{64}$$

The potential function (or clique potential) is usually defined as

$$f_c(x_c) = -\log g_c(x_c),$$

because $g_c(x_c)$ is a positive real-valued function. Defining the potential function in this way, the joint distribution $p(\mathbf{X})$ can now be expressed in a more convenient way:

$$p(\mathbf{X}) = \frac{1}{Z}\exp\{-E(\mathbf{X})\}, \tag{65a}$$

$$E(\mathbf{X}) = \sum_{c\in\mathcal{C}} f_c(x_c), \tag{65b}$$

where $E(\mathbf{X})$ is the *energy* of the P.G.M. Observing that the "-log" transformation between the energy $E(\mathbf{X})$ and the joint distribution $p(\mathbf{X})$ is a monotonic function, then the M.A.P estimation problem is equivalent to minimizing the energy function $E(\mathbf{X})$ in a variational form (Appendix 1.5), instead of maximizing the joint distribution $p(\mathbf{X})$:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} E(\mathbf{X}). \tag{66}$$

Traditionally, the M.A.P problem of equation (16) is solved using unary and pairwise potentials, which are expressed in the following form:

$$E(\mathbf{X}) = \sum_{i\in\mathcal{V}} f_i(x_i) + \sum_{\substack{i\in\mathcal{V},\\ j\in\mathcal{N}_i}} f_{i,j}(x_i, x_j), \tag{67}$$

where $\mathbf{X}$ is a vector of binary variables, defined over a set $i \in \mathcal{V}$ with an appropriately defined neighborhood $\mathcal{N}_i$ of the i'th variables. Unary and pairwise potentials are commonly used in P.G.M applications because they are easier to formulate and certain pairwise models can be solved exactly[4] in polynomial time.

[4] That is, there is no need for approximating the optimization problem. When the optimization problem cannot be solved exactly, such as the case in NP-Hard problems (Appendix 3.1), methods based on approximation should be used.

However, unary and pairwise potentials fails in capturing rich structural contents, accurate spatial dependencies, nor recognizing convex or concave shapes in a digital image (Tjelmeland and Besag (1998)). To this end, higher-order potentials should be considered. The challenge is that higher-order P.G.Ms pose a significant computational challenge, because adding higher-order terms gives an exponential growth of variables in the optimization solution space. Another challenge is the "competition" of two competing functional terms, namely convex and concave functions, which is how higher-ordered P.G.Ms are formulated.

The potentials of x_i and the pair (x_i, x_j) are given by the potential functions $f_i(x_i) : \mathbb{Q} \leftarrow \mathcal{L}$ and $f_{i,j}(x_i, x_j) : \mathbb{Q} \leftarrow \mathcal{L} \times \mathcal{L}$ respectively. The objective is to find the most probable M.A.P configuration via minimizing the energy E with respect to $\mathbf{X}$. The neighborhoods $\mathcal{N}_i$ determines a maximal clique c of the P.G.M and is expressed as $x_c = \{x_i : i \in c\}$. Note that, equation (67) is a special case of the energy function:

$$E(\mathbf{X}) = \sum_{i \in V} f_i(x_i) + \sum_{c \in C} f_c(x_c). \tag{68}$$

P.G.M potentials expressed in the form of equation (67) have tractable algorithms for certain types of networks where $|\mathcal{C}| \leq 2$.

Solving the M.A.P optimization problem can bee seen as minimizing a partially separable function of many discrete variables. From this view, equation (67) has a natural linear programming relaxation formulation (Shlezinger (1976); Koster et al. (1998); Chekuri et al. (2004)). However, the structure of the M.A.P problem determines how effectively or can we even solve the problem. That is, given an arbitrary hypergraph, we try to solve all functions which belongs to a given subset of all possible functions. In theory of computation, this means that we are trying to *recognize* a *language* (see Appendix 3.1 for more detail and consequences).

APPENDIX 2 MATHEMATICAL TOOLS

APPENDIX 2.1 Probability theory

A brief review of relevant concepts of probability theory, regarding image processing and analysis, is covered here. Through out this section, Q is denoted as the level of quantization of an image, f a probability density function (P.D.F) and g a random variable (r.v).

Definition 2.1. *In probability theory, all possible* outcomes *in given problem represents a* sample space*. An* event *is a representation of some subset from the sample space (for example, obtaining odd numbers after rolling a fair die). For each collection of events in a sample space, there is an assigned probability which are not necessarily identical to the collection of all subsets of a sample space.*

An outcome q is a realization of an event A if $q \in A \subset Q$. Two events A and B are said to be *incompatible* if and only if an outcome q realizes only either A or B. This is notated as $A \cup B = \emptyset$. If $A \cap B$ is non-empty, then q is said to realize both A and B. For q to realize $A \cup B$, it needs only to realize at least one event among subsets A and B.

Definition 2.2. *Let Q be our sample space and $\mathcal{F}$ a collection of events in Q. A continuous r.v is a* function $X : \mathbb{Q} \leftarrow Q$*, which assigns a probability to an* event $\{X \leq a\} = \{q; X(q) \leq a\}$*, $a \in \mathbb{Q}$, $q \in Q$. That is, we assign a probability to a r.v, such that*

$$\{X \leq a\} \in \mathcal{F}.$$

If the r.v is discrete, then the function becomes a mapping from a denumerable set E. That is, we have $X : E \leftarrow Q$ if for all $e \in E$ we have

$$\{X = e\} \in \mathcal{F}.$$

Definition 2.3. *Assigning probabilities to events is done by measuring (as in a probability measure) the likeliness of the events occurrence. Let $\mathbb{P}$ be a probability measure (see Appendix 2.2 on measure theory) and $\mathcal{F}$ be a collection of events in Q. A probability is given to the pair $(Q, \mathcal{F})$ as a measure, with the mapping $\mathbb{P} : \mathbb{Q} \leftarrow \mathcal{F}$, by the following* axioms of probability*:*

1. $0 \leq \mathbb{P}(A) \leq 1$, $A \in \mathcal{F}$.
2. $\mathbb{P}(Q) \equiv 1$.
3. $\mathbb{P}(\sum_{k=0}^{\infty} A_k) = \sum_{k=0}^{\infty} \mathbb{P}(A_k)$.

Definition 2.4. *Two events A and B are said to be* independent *if we have the following decomposition*

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B).$$

In the case of r.vs, independence of two r.vs X and Y is expressed as

$$\mathbb{P}(X \leq x, Y \leq y) = \mathbb{P}(X \leq x)\mathbb{P}(Y \leq y),\ \forall x, y \in \mathbb{Q}.$$

Definition 2.5. *To express the expectation of an event A being realized after we have observed the realization of event B, gives us the notion of* conditional probability*:*

$$\mathbb{P}(A|B) \stackrel{def}{=} \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)},\ \mathbb{P}(B) > 0. \tag{69}$$

From the axioms of probability and the conditional probability (68), we can give the well-known Bayes' Rule.

Theorem 2.1. *Given a probability measure $\mathbb{P}$ and events $A, B \in \mathcal{F}$, the* Bayes' Rule *can be stated as the following three rules:*

Bayes' Rule of retrodiction. *If $\mathbb{P}(A) > 0$, then we have*

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A)}. \tag{70}$$

Bayes' Rule of exclusive and exhaustive causes. *Given events $B_0, B_1, \ldots,$ such that*

$$\sum_{k=0}^{\infty} B_k = Q,$$

for all A we have,

$$\mathbb{P}(A) = \sum_{k=0}^{\infty} \mathbb{P}(A \mid B_k)\mathbb{P}(B_k). \tag{71}$$

Bayes' sequential formula. *For any sequence of events $A_0, A_1, \ldots, A_{k-1}$, we have*

$$\mathbb{P}(\cap_{i=0}^{k-1}) = \mathbb{P}(A_0)\mathbb{P}(A_1|A_0)\mathbb{P}(A_2|A_1 \cap A_0)\ldots\mathbb{P}(A_k|\cap_{i=0}^{k-2} A_i). \tag{72}$$

Definition 2.6. *We say that events $A, B \in \mathcal{F}$ are* conditionally independent *given an event $C \in \mathcal{F}$, if we have*

$$\mathbb{P}(A \cap B|C) = \mathbb{P}(A|C)\mathbb{P}(B|C). \tag{73}$$

If we have three (discrete) r.vs $X \in E, Y \in F, Z \in G$, then X and Y are conditionally independent given Z for all $x \in E, y \in F, z \in G$ if the events $\{X = x\}$ and $\{Y = y\}$ are conditionally independent given $\{Z = z\}$.

Definition 2.7. Markov property *defines the following conditional independence for the events $A, B, C \in \mathcal{F}$:*

$$\mathbb{P}(C|A \cap B) = \mathbb{P}(C|B). \tag{74}$$

Theorem 2.2. *The Markov property for r.vs is defined as follows. Assume $X \in E, Y \in F, Z \in G$ are discrete r.vs. If for some function $g : [0,1] \leftarrow E \times F, \mathbb{P}(X = x \,|Y = y, Z = z) = g(x,y)$, for all $x, y, z \in \mathbb{R}$, then we have $\mathbb{P}(X = x \,|Y = y) = g(x,y)$, for all x, y.*

Definition 2.8. *Given quantization level Q, f and g within the interval $[g, dg]$, the P.D.F must meet the requirement*

$$\int_{-\infty}^{\infty} f(g)\, dg = 1.$$

If f is a discrete P.D.F, then for the observed values g_q, $q = 0, 1, \ldots, |Q| - 1$, f_q gives the probability of the observed values, meeting the requirement

$$\sum_{q=0}^{|Q|-1} f_q = 1.$$

Definition 2.9. *The continuous and discrete* expectation value, *$\mathbb{E}_c$ and $\mathbb{E}_d$ respectively, of a P.D.F and its observed r.v are defined as*

$$\mathbb{E}_c \overset{def}{=} \int_{\infty}^{-\infty} gf(g)\, dg, \quad \mathbb{E}_d \overset{def}{=} \sum_{q=0}^{|Q|-1} g_q f_q$$

Alternatively, $\mathbb{E}_d$ can be determined without f by explicitly averaging an infinite number of measurement observations:

$$\mathbb{E}_d \overset{\text{def}}{=} \lim_{P \to \infty} \frac{1}{P} \sum_{p=0}^{P} g_p.$$

Definition 2.10. *A* kernel density estimation *is a non-parametric statistical method way to estimate PDFs. If we have a $\vec{x} \in \mathbb{R}^n$, which is a univariate* independent and identically distributed *sample from an unknown P.D.F f. The* kernel estimator *is defined as*

$$\hat{f}_h(\vec{x}) \overset{def}{=} \frac{1}{n} \sum_{i=0}^{n-1} K_h(x - x_i) = \frac{1}{nh} \sum_{i=0}^{n-1} K(\frac{x - x_i}{h}), \tag{75}$$

where $K(\cdot) > 0$ is a kernel function, *and $h > 0$ is the* bandwidth *of K.*

APPENDIX 2.2 Measure theory

Measure theory is the mathematical study of the length of a set, and (in principle) concern itself on the algebra of sets (Bogachev (2007)). Here we give the main notions of measure theory.

Definition 2.11. *Given some (fixed) set X, $\mathcal{A}$ is called an* algebra *of sets, if*

1. $X, \emptyset \in \mathcal{A}$;
2. *if* $X, Y \in \mathcal{A}$, *then* $X \cup Y \in \mathcal{A},\ X \cap Y \in \mathcal{A}$ *and* $X \setminus Y \in \mathcal{A}$.

Definition 2.12. *If $\mathcal{A}$ is an algebra, then the mapping $\mu : \mathbb{R}_+ \leftarrow \mathcal{A}$ is called* measurable, *if $\mu(X \cup Y) = \mu(X) + \mu(Y),\ \forall X, Y \in \mathcal{A},\ X \cap Y = \emptyset$, and $\mu(\emptyset) = 0$.*

Definition 2.13. *An algebra $\mathcal{A}$ is called a $\sigma -$ algebra, if for any sequence of sets $X_n \in \mathcal{A}$, one has $\cup_{n=0}^{\infty} X_n \in \mathcal{A}$. Additionally, the measure μ is called $\sigma -$ additive if $\mu(\cup_{n=0}^{\infty} X_n) = \sum_{n=0}^{\infty} \mu(X_n)$.*

Definition 2.14. *A* measurable space *consists of a tuple $(X, \mathcal{A})$, where X is a set and collection of subsets of $\mathcal{A}$. More precisely, given arbitrary sets X_1, X_2 and their respective algebras $\mathcal{A}_1, \mathcal{A}_2$, the mapping $\mu : (X_2, \mathcal{A}_2) \leftarrow (X_1, \mathcal{A}_1)$ is called a measurable space.*

Definition 2.15. *A* probability space *is defined as a triple $\mathcal{P} = (W, \mathcal{A}, \mu)$, where W is the set of* elemantary events, *$\mathcal{A}$ an arbitrary $\sigma -$ algebra, and μ is a $\sigma -$ additive measure in the segment $[0, 1]$ with the normalized condition $\mu(W) = 1$.*

r.vs in $\mathcal{P}$ are measurable functions $\mu : (\mathbb{R}, \mathcal{B}) \leftarrow (W, \mathcal{A})$, where $\mathcal{B}$ is a *Borel set*[1]. A probability distribution of a r.v in $\mathcal{P}$ is defined as $\mathbb{P}_S(A) \stackrel{\text{def}}{=} \mathbb{P}(\mu^{-1}(A))$, $\forall A \in \mathcal{B}$.

APPENDIX 2.3 Ensemble definition of probability

The probability of ensembles is the coefficient of the proportions of realized events and all possible events (Khrennikov (2009)). Let S be a finite ensemble with arbitrary properties, denoted as π_S. Each property in π_S cane be described as a mapping $\mu : K_\mu \leftarrow S$, where $K_\mu \stackrel{\text{def}}{=} \{1, 2, \ldots, k_\mu\}$ is a finite set. An *event* can be defined by setting $S(\mu = j) = \{s \in S\ :\ \mu(s) = j\}$, where $\mathcal{A}(\pi_S)$ is the collection of all possible events. The probability of an event is defined as

$$\mathbb{P}(S[\mu = j]) = \frac{|S(\mu = j)|}{|S|}. \tag{76}$$

If $\mathcal{A}(\pi_S)$ is an algebra, then $\mathbb{P} : T_S \subset \mathbb{R}_+ \leftarrow \mathcal{A}(\pi_S)$ is a measure, where $T_S \stackrel{\text{def}}{=} \{x = k/N\ :\ k = 0, 1, \ldots, N\},\ N = |S|$, with $\mathbb{P}(S) = 1$.

Conditional probabilities are an important role in ensemble probabilities. Assume we have $B = S(\mu = l), A(\nu = k), \mu, \nu \in \pi_S$ with the set of events $C = A \cap B \in \mathcal{A}(\pi_S)$. Then there exists a property $\phi \in pi_S$ such that $C = S(\phi = m)$. The conditional probability of event B after observing A is defined as:

$$\mathbb{P}_S(B|A) \stackrel{\text{def}}{=} \mathbb{P}_A(B) = \frac{|B \cap A|}{|A|}. \tag{77}$$

[1] That is, for a measurable function μ we have $\mu^{-1}(A) \in \mathcal{A},\ \forall A \in \mathcal{B}$.

From the conditional probability, we can obtain what is known as the *Bayes formula*[2]:

$$\mathbb{P}_S(B|A) = \frac{\mathbb{P}_S(B \cap A)}{\mathbb{P}_S(A)}, \mathbb{P}_S(A) > 0. \tag{78}$$

We have the following consequence of the Bayes formula:

$$\mathbb{P}_S(A \cap B) = \mathbb{P}_S(B|A)\mathbb{P}_S(A). \tag{79}$$

By symmetry we get

$$\mathbb{P}_S(A \cap B) = \mathbb{P}_S(A|B)\mathbb{P}_S(B), \tag{80}$$

from which we obtain

$$\mathbb{P}_S(A|B) = \frac{\mathbb{P}_S(B \cap A)}{\mathbb{P}_S(A)}, \mathbb{P}_S(B) > 0. \tag{81}$$

One cannot directly generalize equation (77) from ensemble probabilities to infinite ensembles S, using real analysis, because actual infinities do not exist in the field[3] of $\mathbb{R}$. However, a measure theoretical approach can provide some indirect generalizations (Kolmogorov (1983); Khrennikov (2009)). However, it could be that a set of properties π_S can be different from the set of r.vs in $\mathcal{P}_S$. That is, there could exists r.vs which do not share the properties of $s \in S$. On-the-other-hand, there can be properties $\nu \in \pi_S$ which are not r.vs. Also it is important to note, that all probability distributions depend on the ensemble S.

APPENDIX 2.4 Properties of numerical functions

Some properties of numerical functions (Choquet (1966)) are presented that are used in Section 2.1.

Lower envelopes. Let $(f_i)_{i \in I}$ be a family of numerical functions, where I is some interval on a set E. For a numerical function g to have an upper bound for the families of f, a necessary and sufficient condition is that $f_i(x) \leq g(x)$, $\forall x \in E$, $\forall i \in I$. Among the functions g, there exists one function which is smaller than all other functions, namely $f(x) = \sup_{i \in I} f_i(x)$.

---

2 "Standard" textbooks on probability omit the ensemble index.

3 As a reminder: a *field* is a real number system consisting of an *uncountable set*, the binary operations *addition* $+$ and multiplication $\cdot$ and order relations $<, >, =$.

Definition 2.16. *The* lower envelope[4] *of the family* $(f_i)_{i\in I}$*, denoted by* $\inf_{i\in I}$*, is defined as the function* f*, such that*

$$f(x) = \inf_{i\in I} f_i(x)\ \forall x \in E.$$

Definition 2.17. *Let* (X, d_X) *and* (Y, d_Y) *be metric spaces and* d_X, d_Y *denotes the metrics on sets* X, Y *respectively. A function* $f : Y \leftarrow X$ *is called* Lipschitz continuous *if*

$$\exists k \geq 0,\ d_Y(f(x_1), f(x_2)) \leq k \cdot d_X(x_1, x_2),\ \forall x_1, x_1 \in X. \tag{82}$$

Proposition 2.1. *Let* $(f_i)_{i\in I}$ *be a family of arbitrary elements on a topological space* $(E, \mathbb{R})$ *and* f *its lower envelope. If each* f_i *is* Lipschitz continuous *with ratio* k*, and if* f *is finite at least at one point, then* f *is finite every and Lipschitz continuous with ratio* k*.*

Proof of Proposition 2.1 can be found from Choquet (1966).

APPENDIX 2.5 Structure scores in Bayesian networks

For evaluating for structured learning in Bayesian Networks (B.Ns), we can use either *maximum likelihood scoring* or *Bayes scoring* (Koller and Friedman (2009)). Evaluating how well a B.N has learned the structure of the data, the evaluation is posed as an optimization problem. The scoring function helps with scoring candidate structures one wishes to learn from the training data.

Maximum likelihood estimation scoring. Here a structured scoring version of the maxium likelihood estimation (M.L.E) for evaluating B.Ns is presented. In order to make the data more probable for a given model, we aim to maximize the likelihood of the pair $(G, \vec{\theta_G})$, where G is the network and $\vec{\theta_G}$ is the parameter vector of G. That is, we aim to maximize the likelihood of the structure G:

$$\max_{G,\vec{\theta_G}} \mathcal{L}(< G, \vec{\theta_G} > : \mathcal{D}) = \max_{G}[\max_{\vec{\theta_G}} \mathcal{L}(< G, \vec{\theta_G} > : \mathcal{D})], \tag{83}$$

where $\mathcal{D}$ is the data. The M.L.E of the pair $(G, \vec{\theta_G})$ should give the structure of G with the highest likelihood of the given data $\mathcal{D}$. Scoring of a network structure G with a given data $\mathcal{D}$ will be denoted as

$$\text{score}_{\mathcal{L}}(G\ : \mathcal{D}) = l(\hat{\theta_G}\ : \mathcal{D}), \tag{84}$$

where $l(\cdot : \cdot)$ denotes the logarithm of the likelihood function $\mathcal{L}$.

A information-theoretic interpretation of the likelihood score (84) is given as follows. Given a network structure G_0 and two independent propositions X and Y, we get the following score:

$$\text{score}_{\mathcal{L}}(G_0\ : \mathcal{D}) \stackrel{\text{def}}{=} \sum_m \log \hat{\theta}_{x[m]} + \log \hat{\theta}_{y[m]}. \tag{85}$$

[4] We could also define a class of *upper envelope* functions by switching inf to sup.

Now consider a network structure G_1 where the propositions X and Y have an edge connecting them: $X \to Y$. Now the likelihood score (84) is given as

$$\text{score}_{\mathcal{L}}(G_1 \ : \mathcal{D}) \stackrel{\text{def}}{=} \sum_m \log \hat{\theta}_{x[m]} + \log \hat{\theta}_{y[m] \mid x[m]}. \tag{86}$$

Here $\hat{\theta}_x$ is the maximum likelihood estimate for the prior $P(x)$, and $\hat{\theta}_{y \mid x}$ is the maximum likelihood estimate of the likelihood term $P(y \mid x)$. Now between the network structures G_0 and G_1, the scores (85) and (86) share a common component, namely $\log \hat{\theta}_x$. Taking the difference between the two scores, we get

$$\text{score}_{\mathcal{L}}(G_1 \ : \mathcal{D}) - \text{score}_{\mathcal{L}}(G_0 \ : \mathcal{D}) = \log \hat{\theta}_{y[m] \mid x[m]} - \log \hat{\theta}_{x[m]}.$$

Counting how many times each conditional probability parameter appears when taking the difference, we can rewrite the difference as:

$$\text{score}_{\mathcal{L}}(G_1 \ : \mathcal{D}) - \text{score}_{\mathcal{L}}(G_0 \ : \mathcal{D}) = \sum_{x,y} M[x,y] \log \hat{\theta}_{y \mid x} - \sum_y M[y] \log \hat{\theta}_y. \tag{87}$$

If we denote $\hat{P}$ as the empirical distribution (or frequency) in $\mathcal{D}$, then denoting $M[x,y] = M \cdot \hat{P}(x,y)$, $M[y] = M \cdot \hat{P}(y)$, $\hat{\theta}_{y \mid x} = \hat{P}(y \mid x)$, and $\hat{\theta}_x = \hat{P}(x)$, we get the following score:

$$\text{score}_{\mathcal{L}}(G_1 \ : \mathcal{D}) - \text{score}_{\mathcal{L}}(G_0 \ : \mathcal{D}) = M \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(y \mid x)}{\hat{P}(y)} = M \cdot \mathbb{I}_{\hat{P}}(X;Y), \tag{88}$$

where $\mathbb{I}_{\hat{P}}(X;Y) \geq 0$ is the *mutual information* between X and Y in $\hat{P}$. That is, $\mathbb{I}_{\hat{P}}(X;Y)$ tells us how strongly dependent X and Y are to each other in $\hat{P}$. In general, the M.L.E score in a network measures the *strength* of the dependency between propositions and their parents.

Limitations. M.L.E is a good way to estimate the fit of the estimated B.N structure. However for learning the performance of the learned B.N to new unseen data, from the same underlying distribution P^*, M.L.E will run into problems.

As an example, if we have a network G_α where X and Y are independent, and G_β where X is the parent of Y. Now from the score (88) gives that $\text{score}_{\mathcal{L}}(G_\beta \ : \mathcal{D}) \geq \text{score}_{\mathcal{L}}(G_\alpha \ : \mathcal{D})$ for any $\mathcal{D}$. This tells us that the maximum likelihood score never prefers the simpler network structure over a complex one. Moreover, the maximum likelihood score gives identical scores to any network structure where (in rare cases) X and Y are *truly* independent in $\mathcal{D}$.

APPENDIX 3 THEORY OF COMPUTATION & DATA STRUCTURES

APPENDIX 3.1 Languages

The theory of computation (or computability) has helped in distinguishing which computational problems are solvable and which are not (Davis et al. (1994)). With *solvable* problems, it is said to be *tractable* (or *feasible*[1]) in practical terms. Those problems which, on the contrary, are solvable "in principle" but not in practical terms are called *intractable*.

Determining the tractability of a program is posed as a *decision problem*, where from a given instance Π, we are seeking for a *yes* or *no* answer. Some classical decision problems are

1. *The travelling salesman problem*: Given an undirected network $G = (V, E)$, is there a *tour* of the network of length k, such that each node is visited *once* (except for the starting node).
2. *SAT*: Let f be a Boolean formula in *conjuctive normal form*. Is there a way to satisfy the truth assignment of f?
3. *Cardinality vertex cover*: Given an undirected network $G = (V, E)$ and $k \geq \mathbb{Z}_+$, can we find a vertex cover of size $\leq k$?.

Answering these decision problems are viewed as recognizing a *language* L, which is a subset of $\{0,1\}^*$ (Vazirani (2013)). All elements in L are elements which can be encoded with a "yes" answer of an given instance Π. The recognition can be viewed as asking a question (an instance from Π) from a *verifier* (a suitable abstract computing model) , which then answers "yes" or "no" in some time. For "efficient" programs, this "time" is usually meant to be bounded by some polynomial $p(|n|) \stackrel{\text{def}}{=} n^{O(1)}$, with respect to the programs input length. If the question can answered in polynomial time, we say that answer belongs to the class NP. Before formally stating polynomial time solvability, we refresh our memory on the *Turing machine*[2] computing model.

Definition 3.1. *A* deterministic Turing machine *M is defined by the* $10-$*tuple*

$$M \stackrel{def}{=} (Q, \Sigma, \Gamma, \sqcup, \vdash, \dashv, \delta, s, t, r),$$

where

- Q *a finite set of* states.
- Σ *is a finite* input alphabet.

[1] With feasibility, it is also meant that we can solve the problem with the given computing resources (for example, processor speed, memory size, degree of decomposition.)

[2] Examples of other computing machines: Kleene's μ^* recursive model, $WHILE$-programs, Markov algorithms and Thue systems. All such machines are equivalent under the Church's thesis.

- Γ *is a finite* worktape alphabet.
- $\sqcup \in \Gamma$ *is a* blank symbol.
- $\vdash \in \Gamma \setminus \Sigma$ *is the* left endmarker.
- $\dashv \notin \Sigma$ *is the* right endmarker.
- $\delta : Q \times \Gamma \times \{-1, 0, 1\}^2 \leftarrow Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \Gamma$ *is the* transition function.
- $s \in Q$ *is the* start state.
- $t \in Q$ *is the* accept state.
- $r \in Q$ *is the* reject state $r \neq t$.

That is, M reads an input string from the input tape Σ*, beginning at s, operates between the endmarkers* $\vdash, \dashv$*, and is a read-only operation. The worktape* Γ *can read and write between the endmarkers and moves according to the transfer function* δ *according the directions* $\{-1$ *(left)*$, 0$ *(do not move)*$, 1$ *(right)*$\}$*. The machine M halts when it reaches t. The number of operations of the machine M does on the worktape* Γ *determines the* complexity *of the task.*

The Turing machine can be made *nondeterministic* by using a nondeterministic transfer function, which is more important when dealing with NP-Complete problems.

Now we can formally state answering questions in polynomial time:

Definition 3.2. *A language L is recognizable, that is* $L \in NP$*, if there is a polynomial p and a polynomially bounded Turing machine M for each string* $x \in \{0, 1\}^*$*, such that:*

- *if* $x \in L$*, then there is a string y which M accepts in polynomial time. That is, given* $M(x, y)$*, we have* $|y| \leq n^{p(|x|)}$*.*
- *if* $x \notin L$ *for any string y such that* $|y| \leq n^{p(|x|)}$*, then M rejects x.*

Next we define the concept of a NP-Complete problem. First we need the following definition.

Definition 3.3. *Given two languages* $L_1, L_2 \in NP$*, we say* L_1 *can be* reduced in polynomial time *to* L_2*, denoted as* $L_1 \preceq L_2$*, if there exists a polynomial time Turing machine T, with an input string* $x \in \{0, 1\}^*$*, the machine outpouts a string y such that* $x \in L_1$ *if and only if* $y \in L_2$*.*

Definition 3.4. *A language L is NP-hard if* $\forall L' \in NP$*,* $L' \preceq L$*. A language L is NP-complete if* $L \in NP$ *and L is NP-hard.*

APPENDIX 3.2 Asymptotic complexity classes

This Section reviews rudimentary *asymptotic complexity classes* (Kozen (2012)).

Let $(f, g) : \mathbb{N} \leftarrow \mathbb{N}$.

Definition 3.5. Worst-case complexity $\mathcal{O}(\cdot)$ *is the asymptotic* upper bound *of an operation f if $f \in \mathcal{O}(g)$. That is,*

$$c \in \mathbb{N} \overset{\infty}{\forall} n \; f(n) \leq c \cdot g(n),$$

where $\overset{\infty}{\forall}$ means "for almost all".

Definition 3.6. Lower-case complexity $\Omega(\cdot)$ *is the asymptotic* lower bound *of an operation f if $f \in \Omega(g)$ and $g \in \mathcal{O}(f)$. That is,*

$$c \in \mathbb{N} \overset{\infty}{\forall} n \; f(n) \geq \frac{1}{c} \cdot g(n), \; c > 0.$$

Definition 3.7. Exact-case complexity $\Theta(\cdot)$ *is the asymptotic* exact bound *of an operation f if $f \in \mathcal{O}(g) \wedge f \in \Omega(g)$.*

APPENDIX 3.3 Linear programming relaxation for M.A.P inference

Many computer vision and image analysis problems can be casted into a M.A.P estimation problem (Appendix 1.4 and 1.5), and frequently come up as NP-Hard combinatorial optimization problem. These arising NP-Hard optimization problems have a natural linear programming (L.P) relaxation (Shlezinger (1976); Koster et al. (1998)) and are often included as a subroutine in solvers for many practical problems.

We now reiterate the discrete energy minimization (or *valued constrained satisfaction problem* (Cohen et al. (2006); Kolmogorov et al. (2015)) for the M.A.P inference problem. We are interested in solving the probable configuration of a discrete set of labels $\mathcal{L}$ in an associated hypergraph $\mathcal{H} \overset{\text{def}}{=} (\mathcal{V}, \mathcal{C}, \mathcal{E})$, where $\mathcal{V}$ is the set of variables (or nodes) and $\mathcal{C}$ is a set of cliques in $\mathcal{H}$. For each variable $x_i \in \mathcal{V}, i = 0, \cdots |\mathcal{V}| - 1$, there is an associated state $\mathcal{L}$. Define $\bar{\mathbb{Q}} \overset{\text{def}}{=} \mathbb{Q} \cup \{\infty\}$ as the extended set of rational numbers[3]. Let $\Phi : \bar{\mathbb{Q}} \leftarrow \mathcal{L}^{\mathcal{V}}$ be a *partially separable function* if it can be expressed as

$$\Phi(\vec{x}) = \sum_{\mathcal{C} \in \mathcal{H}} \phi_{\mathcal{C}}(\vec{x}_{\mathcal{C}}), \; \phi_{\mathcal{C}} : \bar{\mathbb{Q}} \leftarrow \mathcal{L}^{|\mathcal{C}|}. \tag{89}$$

If we do a L.P relaxation on equation (89), we obtain the following *Basic L.P relaxation* (B.L.P) (Shlezinger (1976); Koster et al. (1998)) :

[3] Discrete computers still cannot handle infinite precision numbers.

$$\min \leftarrow \sum_{\mathcal{C}\in\mathcal{H}} \sum_{\vec{x}\in\mathcal{L}^{|\mathcal{C}|}} \phi_{\mathcal{C}}(\vec{x})\mu_{\mathcal{C}}(\vec{x}), \tag{90a}$$

$$\sum_{\vec{y}\in\mathcal{L}^{|\mathcal{C}|} \mid y_c = x} \mu_{\mathcal{C}}(\vec{y}) = \mu_c(x),\ c \in \mathcal{C} \in \mathcal{H},\ x \in \mathcal{V}, \tag{90b}$$

$$\sum_{x\in\mathcal{L}} \mu_i(x) = 1,\ i \in \mathcal{V}, \tag{90c}$$

$$\text{subject to } \mu_{\mathcal{C}}(\vec{x}) \geq 0,\ \mathcal{C} \in \mathcal{H},\ \vec{x} \in \mathcal{L}^{|\mathcal{C}|} \tag{90d}$$

$$\mu_i(x) \geq 0,\ i \in \mathcal{V},\ x \in \mathcal{L}. \tag{90e}$$

The aim is to minimize the functions[4] $\mu_{\mathcal{C}} : \mathbb{Q} \leftarrow \mathcal{L}^{|\mathcal{C}|}$ and $\mu_i : \mathbb{Q} \leftarrow \mathcal{L}$. The B.L.P (90) can be equivalently understood as a *dual decomposition* (or *Lagrangian relaxation*).

Languages that can be solved with B.L.P. The interest of solving the discrete energy minimization cost (89) is essentially to establish hardness results to the *constrained satisfaction problem* (C.S.P), where we seek solutions to functions with *zero-labelling cost* $\{0, \infty\}$ (Montanari (1974)). C.S.P problems are generally NP-Hard problems, so it is natural to study the restrictions on the general framework that guarantee tractability. This study is done by studying language restrictions that restricts the allowed constraints in the problem instance. Complexity results for the language restricted C.S.Ps are known for binary element domains, three element domains and few others. *Structural* restrictions on C.S.Ps do not impose constraint conditions but restrict how constraints *interact* in the hypergraph.

The hardness results for C.S.Ps apply for the (more general) discrete energy minimization cost problem (89). All known tractable solutions (or bounded arity) for structural, restricted C.S.Ps can be easily extended to the problem (89) (Dechter (2003)). However, there are only a certain classes of (89) which are known to be tractable in terms of BLP (90). The tractability is established by being able to recognize a language (Appendix 3.1). The most well-known tractability result for (90) is the concept of *submodularity* (Schrijver (2003); Fujishige (2005)). Given a totally ordered set $\mathcal{L}$ and an $r-$ary function $\phi : \bar{\mathbb{Q}} \leftarrow \mathcal{L}^r$, ϕ is submodular if and only if for all $\vec{x}, \vec{y}, \in \mathcal{L}^r$ we have

$$\phi(\vec{x}) + \phi(\vec{y}) \geq \phi(\min\{\vec{x}, \vec{y}\}) + \phi(\max\{\vec{x}, \vec{y}\}), \tag{91}$$

where min and max are component-wise minimum and maximum operations respectively, with respect to the total order on $\mathcal{L}$. Another well-known tractable result is the *Potts model*[5] (Mezard and Montanari (2009)), contains all

[4] These functions can be seen as probability distributions on $\mathcal{L}^{|\mathcal{C}|}$ and $\mathcal{L}$ respectively.

[5] A model from statistical mechanics with external field.

unary functions and a single binary function $\phi : \mathbb{Q} \leftarrow \mathcal{L}^2$, ϕ, defined as

$$\phi(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y. \end{cases}$$

When $r = 2$, the Potts model is submodular and therefore tractable. For $r > 2$ the Potts model is intractable (Vazirani (2013)).

APPENDIX 3.3.1 P-complete problems

P-complete problems are a set of (parallel) computational problems for which there are no fast, feasible *parallel* solutions to every problem instance[6] (Greenlaw et al. (1995)). In parallel computing, processors are considered as a resource, just like time and space requirements in sequential computing. When viewing processors as a resource, we can address intrinsic issues in a computational task, such as problem decomposition. Identifying P-complete problems is done in a similar way as in identifying NP-complete problems, that is doing a language reduction from a harder problem into a easier one.

The interest in identifying and analyzing P-complete problems is the same as identifying recognizable languages: by identifying and characterizing computational problems as P-complete, the hope is to identify common characteristics of problems which are inherently sequential. As seen in Appendix 3.2, that certain classes of discrete energy minimization problems have tractable solutions, the same principle applies to solving parallel computational problems. For example, L.P (Dobkin et al. (1979); Dobkin and Reiss (1980); Gács and Lovász (1981); Valiant (1980)) and maximum flow (Goldschlager et al. (1982); Lengauer and Wagner (1990)) are P-complete problems, while a two variable L.P and $0 - 1$ maximum flow have feasible parallel solutions.

APPENDIX 3.4 Multigrids

Here we give a very short and informal description of the multi-level data structures known as *multigrids* (M.Gs), which are mainly used as efficient numerical solvers of P.D.Es. Refer to Trottenberg et al. (2000) for a more in-depth treatment. M.Gs operate by creating a *discretized version* of the P.D.E on a mesh grid of points, where the problem is solved iteratively using various approximation schemes. The ordering of the grid points characterizes what type of discretization we are using. We first define a "standard" M.G structure and then define the cascadic M.G structure.

We illustrate the M.G principle on a *Poisson P.D.E model problem*[7]: $Lu = -\Delta u = -u_{xx} - u_{yy} = f$, where L is a partial differential operator.

---

[6] Or at least it is likely that no fast parallel solutions exists. That is, the problem is inherently sequential.

[7] M.Gs are well suited for elliptic P.D.E problems, which is why the *toy problem*. Incidentally,

Definition 3.8. *A discrete Poisson model with Dirichlet boundary conditions is defined as*

$$-\Delta_h u_h(x,y) = f_h^{\Omega}(x,y), \quad ((x,y) \in \Omega_h) \tag{92a}$$

$$u_h(x,y) = f_h^{\Gamma}(x,y), \quad ((x,y) \in \Gamma_h = \partial\Omega_h) \tag{92b}$$

where $\Omega = (0,1)^2 \subset \mathbb{R}^2$ *is a unit square,* $h = \frac{1}{n}, n \in \mathbb{N}$ *the mesh size on a square grid. The differential operator* L *is approximated using a* smoothing kernel*, for example the so-called* five point stencil *approximation kernel which defines the point ordering*

$$L_h u_h(x,y) = \frac{1}{h^2} \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} u_h(x,y).$$

The two main principles of M.G are

Smoothing principle : Chosen a suitable discretization for the P.D.E, the P.D.E is solved by *smoothing* the error of the approximation.

Coarse grid principle : Computationally, smoothing the error on coarser mesh points is cheaper, and provides better smoothing of the error.

With these principles, the M.G structure aims at smoothing the *low and high frequency* error terms in the approximation. There are several smoothing strategies, or *cycles*, which are applied iteratively on the mesh grids. Figure 36 illustrates three strategies: $V-$cycle, $W-$cycle and *cascadic cycle* (to be explained later below). A M.G cycle operates iteratively in the following, general way:

1. Compute a *pre-smoothing* solution to (u_h, L_h, f_h) at level h.
2. Do a coarse grid correction by $i)$ computing the solution's defect, $ii)$ restricting the defect, $iii)$ interpolate the correction between levels, $iv)$ compute the new corrected approximate solution on Ω_h.
3. Compute a *post-smoothing* solution to (u_h, L_h, f_h).

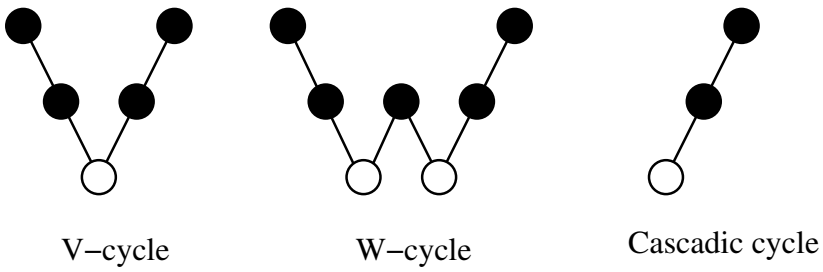


FIGURE 36 Examples of M.G structure cycles, where filled circles are smoothed points, hollow circles exact solutions, \ fine-to-coarse propagation, / coarse-to-fine propagation (Trottenberg et al. (2000)).

In this work, the *cascadic M.G* approach is used to solve early vision problems. The same principles apply as in M.Gs, but the error smoothing is done only in *one direction*, namely coarse-to-fine.

image processing & analysis problems are of elliptical P.D.E variety (if casted into P.D.E form that is).