Institute of Chemical Technology, Prague
Faculty of Chemical Engineering
Department of Computing and Control Engineering

IMAGE PROCESSING BY MEANS OF ŁUKASIEWICZ ALGEBRA WITH SQUARE ROOT

Ph.D. Thesis

Dana Majerová

Prague **March 2004**

Declaration

This doctoral thesis was elaborated at the Department of Computing and Control Engineering at the Institute of Chemical Technology, Prague. Here I declare that the thesis presented has been elaborated by the author exclusively, and only those information resources have been used, which are provided in the list of the cited literature.

Dana Majerová

Acknowledgements

I would like to thank Assoc. Prof. Jaromír Kukal, Ph.D. for his valuable guidance and precious advices throughout the past years. Furthermore I thank him for his unlimited support to find the right solution of many problems in relation to my work.

I thank all members of the Department of Computing and Control Engineering at the Institute of Chemical Technology, Prague for our suggestive and helpful discussions.

My thanks also go to Otto Lang, MD from 3rd Faculty of Medicine at Charles University, Prague for his support.

Keywords

Digital Image Processing, Digital Image Enhancement, Digital Image De-Noising, 2D Image, Lukasiewicz Algebra with Square Root, Fuzzy Logic Function, Constrained Sensitivity, Linear and Nonlinear Filter, Fuzzy Logic Network, Biomedical Image, Magnetic Resonance Imaging

Abstract

This doctoral thesis is oriented to digital 2D image de-noising as part of digital 2D image processing. A good mathematical model for image processing background represents Łukasiewicz algebra with square root. After its definition, terms of fuzzy logic function and its sensitivity were introduced. The sensitivity propositions and their proofs follow.

Basic terms of digital image processing and description of several image processing techniques are subjects of second part of this doctoral thesis. A quality of image processing can be measured using Minkowski distance of processed and ideal images. A multicriterial approach can be applied if several quality measures are used.

A sample of traditional image filters, which are not realizable in Łukasiewicz algebra with square root, is included in the third part first. Then a set of filters, which are realizable in Łukasiewicz algebra with square root, is discussed. Finally, a list of fuzzy logic function filters is introduced.

The image de-noising can be performed by hierarchical structures for data processing. This structure is called fuzzy logic network if it contains fuzzy logic functions only in hidden nodes. Five fuzzy logic networks including their sensitivity are introduced. Their outputs can be better than results of individual fuzzy filters.

Both individual and hierarchical fuzzy filter properties are subject of experimental part. The set of seven artificial images is used in tests. Four quality measures are used for fuzzy filter comparing. Then the multicriterial approach is applied and the set of eight prime filters is selected. This set is used as first hidden layer of fuzzy logic networks. Their structures are optimized in the second step. The results of three types of fuzzy logic networks are better than result of the best individual fuzzy filter. Finally, real biomedical images are processed via selected individual fuzzy filters and all fuzzy logic networks.

All algorithms were realized in MATLAB environment.

Abstrakt

Tato disertační práce je zaměřena na potlačování šumu ve 2D obrazech v kontextu digitálního zpracování 2D obrazů. Vhodným matematickým modelem pro zpracování obrazů je Łukasiewiczova algebra s odmocninou. Po její definici jsou uvedeny pojmy fuzzy logická funkce a její citlivost. Následují tvrzení o citlivosti a jejich důkazy.

Základní pojmy z oblasti digitálního zpracování obrazu a popis několika technik pro zpracování obrazu jsou předmětem druhé části této disertační práce. Kvalitu zpracování obrazu lze měřit pomocí Minkowského vzdálenosti mezi zpracovaným a ideálním obrazem. V případě použití více různých měřítek kvality současně je možno aplikovat vícekriteriální přístup.

Ve třetí části je nejprve uvedeno několik tradičních obrazových filtrů, které nejsou realizovatelné v Łukasiewiczově algebře s odmocninou. Poté je diskutována množina filtrů, jež v Łukasiewiczově algebře s odmocninou realizovat lze. Nakonec je uveden seznam fuzzy filtrů.

Potlačování šumu v obrazech je možno provádět pomocí hierarchických struktur pro zpracování dat. Jestliže ve skrytých uzlech jsou obsaženy pouze fuzzy logické funkce, nazveme tuto strukturu fuzzy logickou sítí. Je uvedeno pět typů fuzzy logických sítí včetně jejich citlivosti. Jejich výstupy mohou být lepšími než výsledky jednotlivých fuzzy filtrů.

Předmětem experimentální části jsou vlastnosti individuálních i hierarchických filtrů. Při testování je použita množina sedmi umělých obrazů. Pro porovnání fuzzy filtrů je užito čtyř měřítek kvality. Pomocí vícekriteriálního přístupu je vybráno osm nejlepších filtrů, které jsou použity v první skryté vrstvě fuzzy logických sítí. Ve druhém kroku je optimalizována jejich struktura. Výsledky tří typů fuzzy logických sítí jsou lepší než výsledek nejlepšího individuálního fuzzy filtru. Nakonec jsou individuální i hierarchické fuzzy filtry použity pro odstranění šumu v reálných biomedicínských obrazech.

Všechny algoritmy byly realizovány v prostředí MATLABu.

Contents

List of Symbols **1**

List of Abbreviations **2**

Introduction **3**

1 ŁA$_{\text{sqrt}}$ as Algebraic Model **5**

1.1 Basic Algebraic Structures 5

1.1.1 Semilattice and Lattice 5

1.1.2 Residuated Lattice 6

1.1.3 Basic Logic Algebra 7

1.1.4 Many-Valued Logic Algebra 7

1.2 Łukasiewicz Algebra with Square Root 8

1.2.1 ŁA$_{\text{sqrt}}$ operators as basic functions 11

1.2.2 Fuzzy Logic Expression 11

1.2.3 Fuzzy Logic Function 11

2 Properties of FLFs in Metric Space **13**

2.1 Metric Space on $\mathbf{L}^n$ 13

2.1.1 Metric Space and Minkowski Metric 13

2.1.2 Minkowski Metric Space 15

2.1.3 Unit Vector and Normalization 15

2.1.4 Metric Space on $\mathbf{L}^n$ 16

2.1.5 Distance and Normalization as FLFs 17

2.2 Properties of FLFs 17

2.2.1 Lipschitz Continuity 17

2.2.2 Sensitivity 19

2.2.3 Upper Bounds of Sensitivity 22

3 Processing of 2D Gray Image **25**

3.1 Basic Terms 25

3.1.1 Pixel 25

3.1.2 2D Image 25

3.1.3 Pixel Neighborhood 26

3.1.4 Mask . . . 27
3.1.5 List and Weighted Lists . . . 28
3.1.6 Local Processing . . . 29
3.1.7 2D Image Processing . . . 31
3.2 De-Noising of 2D Image . . . 32
3.2.1 Noise . . . 32
3.2.2 Ideal, Real and Artificial Images . . . 32
3.2.3 Image De-Noising . . . 32
3.2.4 Filter . . . 33
3.3 Local FLF Processing for Image De-Noising . . . 33
3.3.1 Individual FLF Processing . . . 33
3.3.2 Hierarchical FLF Processing . . . 34

4 Quality of Image Processing **35**
4.1 Image Similarity . . . 35
4.1.1 Image Difference . . . 35
4.1.2 Minkowski Distance of Images . . . 35
4.1.3 Signal to Noise Ratio . . . 36
4.1.4 Other Quality Measure . . . 37
4.2 Multicriterial Approach . . . 37
4.2.1 Pareto Optimality . . . 37
4.2.2 AIA Technique . . . 38

5 Traditional Non-FLF Filters for Image De-Noising **39**
5.1 Mean Filter . . . 39
5.2 Alpha Trimmed Mean Filter . . . 40
5.3 Integer Weighted Average Filter . . . 40
5.4 Gaussian Filter . . . 41
5.5 Wiener Filter . . . 41
5.6 IIR Filter . . . 42

6 Traditional FLF Filters for Image De-Noising **43**
6.1 Statistics Functions as FLFs . . . 43
6.1.1 Lists . . . 43
6.1.2 Median Filter . . . 45
6.1.3 Another Order Statistics as FLF . . . 46
6.2 Morphological Operators as FLFs . . . 47
6.2.1 Erosion . . . 48
6.2.2 Dilation . . . 48
6.2.3 Opening . . . 49
6.2.4 Closing . . . 50

6.3 Dyadic FIR Filter 50
6.4 Useful FLFs 52

7 FLF Networks for Image De-Noising 53
7.1 Modus Ponens Fuzzy Network 54
7.2 Min-Max Fuzzy Network 56
7.3 Modular FLF Network 58
7.3.1 Similarity 58
7.3.2 Modular FLF Network 60
7.4 Constrained Referential Neural Network 62
7.5 Dyadic Weight Neural Network 64

8 Biomedical Image De-Noising 66
8.1 Biomedical Imaging Techniques 66
8.1.1 Magnetic Resonance Imaging (MRI) 66
8.1.2 Positron Emission Tomography (PET) 66
8.1.3 Computed Tomography (CT) 67
8.1.4 Single-Photon Emission Computed Tomography (SPECT) 67
8.2 Experimental Scheme 68

Conclusions 75

References 77

List of the Author's Publications 79

Appendices 83

A Individual FLF Filters and Their Results on Artificial Images 84

B Results of Filtering of Artificial MRI Image 100

C Results of Filtering of Real MRI Image (Human Brain) 102

D Results of Filtering of Real MRI Image (Human Brain) 104

E Results of Filtering of Real MRI Image (Human Genu) 106

F MATLAB Codes of FLF Networks 108

Selected Paper Published 117

List of Figures

3.1 Two types of digital images: (a) gray, (b) binary 26
3.2 Pixel neighborhood for $R = 1$ with a central pixel $\mathcal{X}_{i,j}$ 27
3.3 Image extension for $R = 5$: (a) zero padding, (b) constant replication, (c) periodic replication, (d) mirror replication 27
3.4 Mask as vector of weighs for $R = 1$ 28
3.5 Various types of masks 28
3.6 Pixel neighborhood as list 29
3.7 Weighted list forming from mask $\mathbf{M}$ ($R = 1$) and neighborhood of pixel $\mathcal{X}_{i,j}$ 29
3.8 Local processing scheme 30
3.9 Local FLF processing scheme 33
3.10 Pixel neighborhood (a) before and (b) after local FLF processing 34

7.1 Three-layered FLF network 53
7.2 Four-layered FLF network 53
7.3 Modus ponens fuzzy network for $m = 1$ 54
7.4 Example of MMFN structure ($n = 9$, $F = 4$, $H = 5$) 57
7.5 Typical architecture of modular networks 58
7.6 Architecture of MFLFN 61
7.7 Structure of CRNN 63
7.8 Structure of DWNN 64

8.1 Ideal technical image 68
8.2 Ideal MRI image 68
8.3 Ideal SPECT images 69
8.4 Artificial technical images 69
8.5 Artificial MRI image (I_5) 69
8.6 Artificial SPECT images 69
8.7 Final set of masks 70

B.1 Ideal MRI image 100
B.2 Artificial MRI image I_5 100
B.3 I_5 filtered by F_{13} 100
B.4 I_5 filtered by F_{14} 100
B.5 Artificial MRI image I_5 101

B.6 I_5 filtered by MPFN . . . 101
B.7 I_5 filtered by MMFN . . . 101
B.8 I_5 filtered by MFLFN . . . 101
B.9 I_5 filtered by CRNN . . . 101
B.10 I_5 filtered by DWNN . . . 101

C.1 Real MRI image RI_1 . . . 102
C.2 RI_1 filtered by F_1 . . . 102
C.3 RI_1 filtered by F_{13} . . . 102
C.4 RI_1 filtered by F_{14} . . . 102
C.5 Real MRI image RI_1 . . . 103
C.6 RI_1 filtered by MPFN . . . 103
C.7 RI_1 filtered by MMFN . . . 103
C.8 RI_1 filtered by MFLFN . . . 103
C.9 RI_1 filtered by CRNN . . . 103
C.10 RI_1 filtered by DWNN . . . 103

D.1 Real MRI image RI_2 . . . 104
D.2 RI_2 filtered by F_1 . . . 104
D.3 RI_2 filtered by F_{13} . . . 104
D.4 RI_2 filtered by F_{14} . . . 104
D.5 Real MRI image RI_2 . . . 105
D.6 RI_2 filtered by MPFN . . . 105
D.7 RI_2 filtered by MMFN . . . 105
D.8 RI_2 filtered by MFLFN . . . 105
D.9 RI_2 filtered by CRNN . . . 105
D.10 RI_2 filtered by DWNN . . . 105

E.1 Real MRI image RI_3 . . . 106
E.2 RI_3 filtered by F_1 . . . 106
E.3 RI_3 filtered by F_{13} . . . 106
E.4 RI_3 filtered by F_{14} . . . 106
E.5 Real MRI image RI_3 . . . 107
E.6 RI_3 filtered by MPFN . . . 107
E.7 RI_3 filtered by MMFN . . . 107
E.8 RI_3 filtered by MFLFN . . . 107
E.9 RI_3 filtered by CRNN . . . 107
E.10 RI_3 filtered by DWNN . . . 107

List of Tables

1.1 LA_{sqrt} operators for $a, b \in \mathbf{L}$ and $n \in \mathbf{N}_0$ 9

2.1 Upper bounds of basic FLF sensitivities 22

3.1 Examples of local processing 30

6.1 Useful FLFs for $\boldsymbol{x} \in \mathbf{L}^n$ 52

8.1 Final set of FLFs 70
8.2 List of used quality measures 70
8.3 Multicriterial approach for image processing 71
8.4 Selected Pareto optimum filters 71
8.5 Final MPFN results 72
8.6 Final MFLFN results 72
8.7 MMFN results 73
8.8 Final CRNN results 73
8.9 Final DWNN results 73

A.1 Individual FLF filters 85
A.2 Quality of I_1 filtering 87
A.3 Quality of I_2 filtering 89
A.4 Quality of I_3 filtering 91
A.5 Quality of I_4 filtering 93
A.6 Quality of I_5 filtering 95
A.7 Quality of I_6 filtering 97
A.8 Quality of I_7 filtering 99

List of Symbols

$\mathbf{N}$	set of natural numbers
$\mathbf{N}_0$	set of natural numbers including zero
$\mathbf{Z}$	set of integer numbers
$\mathbf{R}$	set of real numbers
$\mathbf{L}$	[0,1] interval
$\mathcal{X}$	pixel
x	pixel value/vector
r	row number
c	column number
$\mathcal{P} = (\mathcal{X}_{i,j})_{\substack{i=1,\ldots,r \\ j=1,\ldots,c}}$	image
$X = (x_{i,j})_{\substack{i=1,\ldots,r \\ j=1,\ldots,c}}$	image represented as matrix of pixel values/vectors

List of Abbreviations

AIA	absolute value of distance from ideal alternative
BES	best easy systematic estimation
BL-algebra	basic logic algebra
CRNN	constrained referential neural network
CT	computed tomography
DFT	discrete Fourier transform
DWNN	dyadic weights neural network
FIR	finite impulse response
FLE	fuzzy logic expression
FLF	fuzzy logic function
IDFT	inverse discrete Fourier transform
IIR	infinite impulse response
ŁA_{sqrt}	Łukasiewicz algebra with square root
MAE	mean of absolute error
MAX	maximum of absolute error
MED	median of absolute error
MFLFN	modular FLF network
MMFN	min-max fuzzy network
MPFN	modus ponens fuzzy network
MRI	magnetic resonance imaging
MSE	mean square of error
MV-algebra	many-valued logic algebra
PET	positron emission tomography
SNR	signal/noise ratio
SPECT	single-photon emission computed tomography

Introduction

A 2D image processing is an important instrument for many fields of human activities. Biomedical, environmental, meteorological and technological structures are typical data sources. Their analysis has a wide application. The 2D image processing includes various operations, for example image reconstruction, filtering, sharpening and edge detection. Several techniques can be chosen for this purpose. One of them is a fuzzy logic.

The first aim of my doctoral thesis is to demonstrate that basic logic algebra—Łukasiewicz algebra with square root is useful mathematical background for image processing. The second aim is to develop hierarchical structures as fuzzy logic networks and then compare them with individual fuzzy filters for noise suppressing.

The first chapter is oriented to the algebraic model for data processing. Łukasiewicz algebra with square root was chosen from basic logic algebras as a tool for realization of fuzzy logic expressions and fuzzy logic functions.

Basic properties of fuzzy logic functions, which were studied in the metric space on $[0,1]^n$, are included in the second chapter. The main result of this chapter is a sensitivity definition and computing of upper bounds of operator and function sensitivities. All the sensitivity propositions and their proofs are original and they were a subject of my previous publications activity.

The third chapter describes basic terms of 2D image processing. Beginning with pixel, 2D image, neighborhood and mask, the role of list and weighted lists in image processing is defined. After the definition of noise, ideal, real and artificial images, the image de-noising is defined and the role of fuzzy logic functions in local image processing is discussed.

The quality of image processing is a subject of the fourth chapter. The similarity of images can be measured using Minkowski distance of them, signal/noise ratio or using other quality measures. The multicriterial approach can be also used in case of optimum de-noising of a set of images. The Pareto optimality and AIA technique are introduced in this case. The application of multicriterial approach to image de-noising was also subject of my previous publication activity.

A sample of traditional image filters, which are not fuzzy logic functions, is included in the fifth chapter. There is only a list of popular filters which are not in the focus of my work.

Several types of traditional image filters are realizable as fuzzy logic functions. Their list is included in the sixth chapter. They are based on weighted and sorted lists. First subclass of them is related to median and other order statistics. The traditional morphological operators form the second subclass. The third subclass is related to the finite impuls response filters with dyadic weights, which are realizable in Łukasiewicz algebra with square root.

The new ideas how to built hierarchical fuzzy networks in Łukasiewicz algebra with square root are discussed in the seventh chapter. Five approaches to hierarchical fuzzy logic function image processing are introduced. First of them is direct extension of modus

ponens rule from propositional logic. The second approach produces optimum min-max network. Similarity measures and vector normalization are used in the modular approach as third one. The fourth way how to built hierarchical network is based on constraining result of referential method into floating interval between lower and upper bounds. The last approach is based on linear mixing with dyadic weights. These five approaches generates five types of hierarchical fuzzy networks and their structures or parameters are subject of optimization with respect to quality measures.

The last chapter is specialized to biomedical image de-noising. The MRI, PET, CT and SPECT techniques are described first. Then the artificial and ideal MRI and SPECT images are prepared for a testing of fuzzy logic function filters. Various fuzzy logic functions, quality measures and masks are used individually first. Finally, the five types of new hierarchical networks from previous chapter are used and optimized on artificial and ideal images. The resulting filters are compared on real MRI images.

The principles of data processing are realized in MATLAB environment and the source code examples are also included.

Chapter 1

ŁA$_{\text{sqrt}}$ as Algebraic Model

1.1 Basic Algebraic Structures

1.1.1 Semilattice and Lattice

DEFINITION 1.1. *An algebraic structure $\langle S, \triangle \rangle$ is called a* ***semilattice*** *if S is a nonempty set and $\triangle$ is a binary operation such that*

$$\begin{aligned} \triangle \text{ is associative:} &\quad (x \triangle y) \triangle z = x \triangle (y \triangle z) \text{ for all } x, y, z \in S, \\ \triangle \text{ is commutative:} &\quad x \triangle y = y \triangle x \text{ for all } x, y \in S \text{ and} \\ \triangle \text{ is idempotent:} &\quad x \triangle x = x \text{ for all } x \in S. \end{aligned}$$

DEFINITION 1.2. *An algebraic structure $\langle S, \wedge, \vee \rangle$ is called* ***lattice*** *if S is a nonempty set, $\wedge$ and $\vee$ are binary operations on S, $\langle S, \wedge \rangle$ and $\langle S, \vee \rangle$ are semmilattices and $\wedge, \vee$ are absorbtive:*

$$(\forall x, y \in S)\ x \wedge (x \vee y) = x,$$
$$(\forall x, y \in S)\ x \vee (x \wedge y) = x.$$

Using the basic lattice operators, an ordering can be defined.

DEFINITION 1.3. *Let $\langle S, \wedge, \vee \rangle$ be any lattice and $x, y \in S$. The binary relation $\leq$ which is defined by formula*

$$(\forall x, y \in S)\ x \leq y \iff x \vee y = y$$

is called an ***ordering****.*

Remark 1: The ordering can be defined by an equivalent formula

$$(\forall x, y \in S)\ x \leq y \iff x \wedge y = x.$$

THEOREM 1.1. *The ordering is reflexive, antisymmetric and transitive relation:*

$$\begin{aligned} (\forall x \in S) &\quad x \leq x, \\ (\forall x, y \in S) &\quad (x \leq y \wedge y \leq x) \Rightarrow x = y, \\ (\forall x, y, z \in S) &\quad (x \leq y \wedge y \leq z) \Rightarrow x \leq z. \end{aligned}$$

Proof. See [14] □

RESOURCES: [14], [25]

1.1.2 Residuated Lattice

DEFINITION 1.4. *An algebraic structure* $\langle S, \wedge, \vee, \otimes, \rightarrow, 0, 1\rangle$ *is called a* ***residuated lattice*** *if* $\langle S, \wedge, \vee\rangle$ *is a lattice containing the least element* 0 *and the greatest element* 1*, and endowed with two binary operations* $\otimes$ *(called* ***multiplication*** *or* ***t-norm****) and* $\rightarrow$ *(called* ***residuum****) such that* $\otimes$ *is associative, commutative and following identities hold:*

$$\begin{aligned} (\forall x \in S) \quad & x \otimes 1 = x, \\ (\forall x, y, z \in S) \quad & x \otimes y \leq z \Leftrightarrow x \leq y \rightarrow z. \end{aligned}$$

Remark 1: The second formula is called the *Galois correspondence*.

Remark 2: Any multiplication determines exactly one residuum such that the Galois correspondence holds.

Remark 3: The multiplication is isotone in any lattice:

$$(\forall x, y, u, v \in S)\, x \leq y, u \leq v \Rightarrow x \otimes u \leq y \otimes v.$$

EXAMPLE 1: **Gödel Algebra**

DEFINITION 1.5. *A residuated lattice* $\langle [0,1], \wedge, \vee, \otimes, \rightarrow, 0, 1\rangle$ *where*

$$\begin{aligned} a \otimes b &= a \wedge b, \\ a \rightarrow b &= \begin{cases} 1 & \text{if } a \leq b, \\ b & \text{if } a > b \end{cases} \end{aligned}$$

is called ***Gödel algebra****.*

EXAMPLE 2: **Goguen Algebra**

DEFINITION 1.6. *A residuated lattice* $\langle [0,1], \wedge, \vee, \otimes, \rightarrow, 0, 1\rangle$ *where*

$$\begin{aligned} a \otimes b &= a \cdot b, \\ a \rightarrow b &= \begin{cases} 1 & \text{if } a \leq b, \\ \frac{b}{a} & \text{if } a > b \end{cases} \end{aligned}$$

is called ***Goguen algebra****.*

Remark 1: Goguen algebra is also called a *product algebra*.

EXAMPLE 3: **Łukasiewicz Algebra**

DEFINITION 1.7. *A residuated lattice* $\langle [0,1], \wedge, \vee, \otimes, \rightarrow, 0, 1\rangle$ *where*

$$\begin{aligned} a \otimes b &= \max(a + b - 1, 0), \\ a \rightarrow b &= \begin{cases} 1 & \text{if } a \leq b, \\ 1 - a + b & \text{if } a > b \end{cases} \end{aligned}$$

is called ***Łukasiewicz algebra****.*

NUMERIC EXAMPLE:

				Gödel		Goguen		Łukasiewicz	
a	b	$a \wedge b$	$a \vee b$	$a \otimes b$	$a \to b$	$a \otimes b$	$a \to b$	$a \otimes b$	$a \to b$
0	0	0	0	0	1	0	1	0	1
0	1	0	1	0	1	0	1	0	1
1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
0.3	0.3	0.3	0.3	0.3	1	0.09	1	0	1
0.1	0.6	0.1	0.6	0.1	1	0.06	1	0	1
0.6	0.1	0.1	0.6	0.1	0.1	0.06	$0.1\overline{6}$	0	0.5
0.6	0.6	0.6	0.6	0.6	1	0.36	1	0.2	1
0.6	0.8	0.6	0.8	0.6	1	0.48	1	0.4	1
0.8	0.6	0.6	0.8	0.6	0.6	0.48	0.75	0.4	0.8

RESOURCES: [26], [25], [24], [20]

1.1.3 Basic Logic Algebra

DEFINITION 1.8. *A residuated lattice $\langle S, \wedge, \vee, \otimes, \to, 0, 1\rangle$ is called a **basic logic algebra** (BL-algebra) if the following identities hold for all $x, y \in S$:*

$$x \wedge y = x \otimes (x \to y), \tag{1.1}$$

$$x \vee y = ((x \to y) \to y) \wedge ((y \to x) \to x), \tag{1.2}$$

$$(x \to y) \vee (y \to x) = 1. \tag{1.3}$$

Remark 1: Gödel, Goguen and Łukasiewicz algebras are BL-algebras.

RESOURCES: [24], [25], [26]

1.1.4 Many-Valued Logic Algebra

DEFINITION 1.9. *A BL-algebra $\langle S, \wedge, \vee, \otimes, \to, 0, 1\rangle$ is called a **many-valued logic algebra** (MV-algebra) if the following identity holds for all $x \in S$:*

$$(x \to 0) \to 0 = x.$$

Remark 1: The $x \to 0$ operation is called *negation* and it is denoted as $\neg x$.

Remark 2: Any MV-algebra is BL-algebra.

Remark 3: Łukasiewicz algebra is MV-algebra.

RESOURCES: [24], [25], [26]

1.2 Łukasiewicz Algebra with Square Root

Łukasiewicz algebra [20] is an MV-algebra operating on $[0,1]$ interval using conjunction, disjunction, multiplication and residuum as basic logic operators. Having no more than these operators it is impossible to construct low sensitivity systems and compromise data processing. That is why the Łukasiewicz algebra was enriched by a square root function. This function is defined as extended inversion of Łukasiewicz square function.

DEFINITION 1.10. *Let* $\mathbf{L} = [0,1]$. *Let* $a, b \in \mathbf{L}$. *Let*

$$\begin{aligned} a \wedge b &= \min(a,b), \\ a \vee b &= \max(a,b), \\ a \otimes b &= \max(a+b-1,0), \\ a \to b &= \min(1-a+b,1), \\ \mathrm{sqrt}(a) &= (1+a)/2. \end{aligned}$$

Then the ***Łukasiewicz algebra with square root*** *(*LA_{sqrt}*) is defined as*

$$LA_{sqrt} = \langle \mathbf{L}, \wedge, \vee, \otimes, \to, \mathrm{sqrt}, 0, 1 \rangle.$$

THEOREM 1.2. *The* LA_{sqrt} *is a BL-algebra.*

Proof. Let $x, y \in \mathbf{L}$. It suffices to prove that (1.1)–(1.3) hold.
1a) Let $x \leq y$. Then $(x \to y) = 1$ and

$$x \otimes (x \to y) = \max\big(x + (x \to y) - 1, 0\big) = \max(x + 1 - 1, 0) = x = x \wedge y.$$

1b) Let $x > y$. Then

$$x \otimes (x \to y) = \max\big(x + (1 - x + y) - 1, 0\big) = y = y \wedge x = x \wedge y.$$

2a) Let $x \leq y$. Then $(x \to y) = 1$ and

$$\begin{aligned} \big((x \to y) \to y\big) \wedge \big((y \to x) \to x\big) &= (1 \to y) \wedge \big(1 - (1 - y + x) + x\big) = \\ &= (1 - 1 + y) \wedge y = y = x \vee y. \end{aligned}$$

2b) Let $x > y$. Then $(y \to x) = 1$ and

$$\begin{aligned} \big((x \to y) \to y\big) \wedge \big((y \to x) \to x\big) &= \big(1 - (1 - x + y) + y\big) \wedge (1 - 1 + x) = \\ &= x \wedge x = x = y \vee x = x \vee y. \end{aligned}$$

3a) Let $x \leq y$. Then $(x \to y) = 1$ and

$$(x \to y) \vee (y \to x) = 1 \vee (1 - y + x) = 1.$$

3b) Let $x > y$. Then $(y \to x) = 1$ and

$$(x \to y) \vee (y \to x) = (1 - x + y) \vee 1 = 1.$$ □

It is useful to define some derived operators for the simplification of expression and function description. Some of them have a linguistic meaning, for example negation, equivalence, 'very' (the square) and 'roughly' (the square root).

DEFINITION 1.11. *Let $n \in \mathbf{N}_0$. Let $a, b \in \mathbf{L}$. Then* ***derived operators*** *are defined as*

$$\begin{aligned}
\neg a &= a \to 0, \\
a \leftrightarrow b &= (a \to b) \wedge (b \to a), \\
a \circ b &= \neg(a \leftrightarrow b), \\
a \oplus b &= \neg(\neg a \otimes \neg b), \\
a \ominus b &= a \otimes \neg b, \\
n \odot a &= \bigoplus_{k=1}^{n} a = \underbrace{a \oplus a \oplus \cdots \oplus a}_{n}, \ 0 \odot a = 0, \\
a^n &= \bigotimes_{k=1}^{n} a = \underbrace{a \otimes a \otimes \cdots \otimes a}_{n}, \ a^0 = 1.
\end{aligned}$$

Remark 1: The $\mathrm{f}(a) = a^2$ is called *Łukasiewicz square.*

The basic and derived operators are collected in the Tab. 1.1.

operator	**name**	**definition**
$\wedge$	conjunction	$a \wedge b = \min(a, b)$
$\vee$	disjunction	$a \vee b = \max(a, b)$
$\otimes$	Łukasiewicz multiplication	$a \otimes b = \max(a + b - 1, 0)$
$\to$	residuum	$a \to b = \min(1 - a + b, 1)$
$\neg$	negation	$\neg a = a \to 0$
$\leftrightarrow$	equivalence (biresiduum)	$a \leftrightarrow b = (a \to b) \wedge (b \to a)$
$\circ$	non-equivalence (distance)	$a \circ b = \neg(a \leftrightarrow b)$
$\oplus$	addition	$a \oplus b = \neg(\neg a \otimes \neg b)$
$\ominus$	subtraction	$a \ominus b = a \otimes \neg b$
$\odot$	multiplication by integer	$n \odot a = \underbrace{a \oplus a \oplus \cdots \oplus a}_{n}, \ 0 \odot a = 0$
a^n	integer power	$a^n = \underbrace{a \otimes a \otimes \cdots \otimes a}_{n}, \ a^0 = 1$

Table 1.1: $\mathrm{ŁA}_{\mathrm{sqrt}}$ operators for $a, b \in \mathbf{L}$ and $n \in \mathbf{N}_0$

The square root function is extended inversion of Łukasiewicz square as proven in following lemma.

LEMMA 1.3. *Let $a \in \mathbf{L}$. Then* $\mathrm{sqrt}(a)^2 = a$.

Proof. Let $a \in \mathbf{L}$. Then

$$\begin{aligned}
\mathrm{sqrt}(a)^2 &= \mathrm{sqrt}(a) \otimes \mathrm{sqrt}(a) = \max\left(\frac{1+a}{2} + \frac{1+a}{2} - 1, 0\right) = \\
&= \max\left(2 \cdot \frac{1+a}{2} - 1, 0\right) = a. \qquad \square
\end{aligned}$$

LEMMA 1.4. *Let $a \in \mathbf{L}$. Then $\neg\neg a = a$.*

Proof. Let $a \in \mathbf{L}$. Then

$$\begin{aligned} \neg\neg a &= (a \to 0) \to 0 = \min(1 - \min(1 - a + 0, 1) + 0, 1) = \\ &= \min(1 - (1 - a), 1) = \min(a, 1) = a. \end{aligned}$$ □

COROLLARY 1.5. *The LA_{sqrt} is a MV-algebra.*

LEMMA 1.6. *Let $n \in \mathbf{N}$. Let $a, b \in \mathbf{L}$. Then*

$$\begin{aligned} \neg a &= 1 - a, \\ a \leftrightarrow b &= 1 - |a - b|, \\ a \circ b &= |a - b|, \\ a \oplus b &= \min(a + b, 1), \\ a \ominus b &= \max(a - b, 0). \end{aligned}$$

Proof. Let $a, b \in \mathbf{L}$. It is trivial to prove

$$\neg a = a \to 0 = \min(1 - a + 0, 1) = 1 - a,$$

$$\begin{aligned} a \leftrightarrow b &= (a \to b) \wedge (b \to a) = \min(\min(1 - a + b, 1), \min(1 - b + a, 1)) = \\ &= \min(1 - a + b, 1, 1 - b + a) = 1 - |a - b|, \end{aligned}$$

$$a \circ b = \neg(a \leftrightarrow b) = 1 - (1 - |a - b|) = |a - b|,$$

$$\begin{aligned} a \oplus b &= \neg(\neg a \otimes \neg b) = 1 - \max(1 - a + 1 - b - 1, 0) = \\ &= 1 - \max(1 - a - b, 0) = -\max(-a - b, -1) = \\ &= \min(a + b, 1) \text{ and finally} \end{aligned}$$

$$a \ominus b = a \otimes \neg b = \max(a + 1 - b - 1, 0) = \max(a - b, 0).$$ □

LEMMA 1.7. *Let $n \in \mathbf{N}$. Let $a \in \mathbf{L}$. Then*

$$n \odot a = \min(n \cdot a, 1).$$

Proof. Let $a \in \mathbf{L}$ and $n \in \mathbf{N}$. Let $n = 1$ in the first induction step. Then

$$n \odot a = a = \min(a, 1) = \min(n \cdot a, 1).$$

The second induction step leads to

$$\begin{aligned} (n + 1) \odot a &= (n \odot a) \oplus a = \min(n \cdot a, 1) \oplus a = \min(\min(n \cdot a, 1) + a, 1) = \\ &= \min((n + 1) \cdot a, 1). \end{aligned}$$ □

LEMMA 1.8. *Let $n \in \mathbf{N}$. Let $a \in \mathbf{L}$. Then*

$$a^n = \max(n \cdot a - n + 1, 0)$$

Proof. Let $a \in \mathbf{L}$ and $n \in \mathbf{N}$. Let $n = 1$ in the first induction step. Then

$$a^n = a = \max(a, 0) = \max(n \cdot a - n + 1, 0).$$

The second induction step leads to

$$\begin{aligned} a^{n+1} &= a^n \otimes a = \max(n \cdot a - n + 1, 0) \otimes a = \\ &= \max(\max(n \cdot a - n + 1, 0) + a - 1, 0) = \\ &= \max((n + 1) \cdot a - (n + 1) + 1, 0). \end{aligned}$$ □

1.2.1 ŁA_{sqrt} operators as basic functions

All the operators in ŁA_{sqrt} can be represented by functions of one or two variables.

DEFINITION 1.12. *Let $x, y \in \mathbf{L}$, $n \in \mathbf{N}_0$. Then*

$$\begin{aligned}
\mathrm{f}_1(x) &= \neg x \\
\mathrm{f}_2(x) &= \mathrm{sqrt}(x) \\
\mathrm{f}_3(x,y) &= x \wedge y \\
\mathrm{f}_4(x,y) &= x \vee y \\
\mathrm{f}_5(x,y) &= x \otimes y \\
\mathrm{f}_6(x,y) &= x \to y \\
\mathrm{f}_7(x,y) &= x \leftrightarrow y \\
\mathrm{f}_8(x,y) &= x \circ y \\
\mathrm{f}_9(x,y) &= x \oplus y \\
\mathrm{f}_{10}(x,y) &= x \ominus y \\
\mathrm{f}_{11}(x,n) &= n \odot x \\
\mathrm{f}_{12}(x,n) &= x^n
\end{aligned}$$

*are defined as **basic functions** in LA_{sqrt}.*

1.2.2 Fuzzy Logic Expression

DEFINITION 1.13. ***Fuzzy logic expression** (FLE) is defined by the rules:*

(i) *Any free variable $x \in \mathbf{L}$ is FLE.*

(ii) *Any constant $a \in \mathbf{L}$ is FLE.*

(iii) $\mathrm{f}_i(FLE)$ *is FLE for* $i = 1, 2$,

(iv) $\mathrm{f}_j(FLE, FLE)$ *is FLE for* $j = 3, \ldots, 10$,

(v) $\mathrm{f}_k(FLE, n)$ *is FLE for* $k = 11, 12$ *and* $n \in \mathbf{N}_0$

where f_m $(m = 1, \ldots, 12)$ are the basic functions in LA_{sqrt} (see definition 1.12).

1.2.3 Fuzzy Logic Function

DEFINITION 1.14. *Let $n \in \mathbf{N}$, $\boldsymbol{x} \in \mathbf{L}^n$ and $\varphi : \mathbf{L}^n \to \mathbf{L}$. If $\varphi(\boldsymbol{x})$ is FLE then φ is called a **fuzzy logic function** (FLF) in LA_{sqrt}.*

Thus, fuzzy logic function in ŁA_{sqrt} is composed from constants and free variables from $\mathbf{L}$ and finite number of basic ŁA_{sqrt} operators and functions. Properties of FLFs are described in the section 2.2.

EXAMPLE 1: average of two: $\mathrm{AV2} : \mathbf{L}^2 \to \mathbf{L}$,

$$\mathrm{AV2}(\boldsymbol{x}) = \mathrm{sqrt}(x_1) \otimes \mathrm{sqrt}(x_2)$$

Let $\boldsymbol{x} = (0.87, 0.35)$. Then $\mathrm{AV2}(\boldsymbol{x}) = \max\left(\frac{1+0.87}{2} + \frac{1+0.35}{2} - 1, 0\right) = 0.61$.

EXAMPLE 2: average of four: $\mathrm{AV4} : \mathbf{L}^4 \to \mathbf{L}$,

$$\mathrm{AV4}(\boldsymbol{x}) = \mathrm{sqrt}(\mathrm{sqrt}(x_1) \otimes \mathrm{sqrt}(x_2)) \otimes \mathrm{sqrt}(\mathrm{sqrt}(x_3) \otimes \mathrm{sqrt}(x_4)).$$

Let $\boldsymbol{x} = (0.2, 1, 0.6, 0.3)$. Then

$$\mathrm{AV4}(\boldsymbol{x}) = \max\left(\frac{1 + \max\left(\frac{1.2}{2} + \frac{2}{2} - 1, 0\right)}{2} + \frac{1 + \max\left(\frac{1.6}{2} + \frac{1.3}{2} - 1, 0\right)}{2} - 1, 0\right) =$$
$$= \max\left(\frac{1 + 0.6}{2} + \frac{1 + 0.45}{2} - 1, 0\right) = 0.525.$$

EXAMPLE 3: median of three: $\mathrm{ME3} : \mathbf{L}^3 \to \mathbf{L}$,

$$\mathrm{ME3}(\boldsymbol{x}) = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3)$$

Let $\boldsymbol{x} = (0.5, 0.28, 0.321)$. Then $\mathrm{ME3}(\boldsymbol{x}) = 0.28 \vee 0.321 \vee 0.321 = 0.321$.

EXAMPLE 4: median of four: $\mathrm{ME4} : \mathbf{L}^4 \to \mathbf{L}$,

$$\begin{aligned}\mathrm{ME4}(\boldsymbol{x}) &= \mathrm{sqrt}\big((x_1 \wedge x_2 \wedge (x_3 \vee x_4)\big) \vee \big((x_1 \vee x_2) \wedge x_3 \wedge x_4)\big) \otimes \\ &\quad \otimes \mathrm{sqrt}\big((x_1 \vee x_2 \vee (x_3 \wedge x_4)\big) \wedge \big((x_1 \wedge x_2) \vee x_3 \vee x_4)\big)\end{aligned}$$

Let $\boldsymbol{x} = (0.2, 1, 0.36, 0.54)$. Then

$$\begin{aligned}\mathrm{ME4}(\boldsymbol{x}) &= \mathrm{sqrt}(0.2 \vee 0.36) \otimes \mathrm{sqrt}(1 \wedge 0.54) = \\ &= \max\left(\frac{1 + 0.36}{2} + \frac{1 + 0.54}{2} - 1, 0\right) = 0.45.\end{aligned}$$

Chapter 2

Properties of FLFs in Metric Space

2.1 Metric Space on $\mathbf{L}^n$

2.1.1 Metric Space and Minkowski Metric

DEFINITION 2.1. *Let M be a nonempty set. Let $\mathrm{d} : M \times M \to \mathbf{R}_0^+$ be a function which satisfies following properties for all $x, y, z \in M$:*

(i) $\mathrm{d}(x, y) \geq 0$,

(ii) $\mathrm{d}(x, y) = 0 \Leftrightarrow x = y$,

(iii) $\mathrm{d}(x, y) = \mathrm{d}(y, x)$,

(iv) $\mathrm{d}(x, y) \leq \mathrm{d}(x, z) + \mathrm{d}(z, y)$.

*Then $\langle M, \mathrm{d}\rangle$ is called a **metric space** and the function d is called a **metric** (or "distance function").*

THEOREM 2.1 (Minkowski). *Let $n \in \mathbf{N},\ p \in \mathbf{R},\ 1 \leq p < +\infty$. Let $\boldsymbol{f}, \boldsymbol{g} \in \mathbf{R}^n$. Then*

$$\left(\sum_{k=1}^{n} |f_k + g_k|^p\right)^{\frac{1}{p}} \leq \left(\sum_{k=1}^{n} |f_k|^p\right)^{\frac{1}{p}} + \left(\sum_{k=1}^{n} |g_k|^p\right)^{\frac{1}{p}}.$$

Proof. See [6]. □

THEOREM 2.2. *Let $p \in \mathbf{R}, 1 \leq p < +\infty$. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$. Then*

$$\mathrm{d}_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{k=1}^{n} |x_k - y_k|^p\right)^{\frac{1}{p}}$$

is a metric on $\mathbf{L}^n$.

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$ and $p \in \mathbf{R}, 1 \leq p < +\infty$.

(i) Clearly, $\mathrm{d}_p \geq 0$.

(ii) Let $\mathrm{d}_p(\boldsymbol{x},\boldsymbol{y})=0$. Then

$$\left(\sum_{k=1}^{n}|x_k-y_k|^p\right)^{\frac{1}{p}}=0,\quad \text{i.e. } (\forall k=1,...,n)\ x_k=y_k.\ \text{Thus, } \boldsymbol{x}=\boldsymbol{y}.$$

Vice versa, let $\boldsymbol{x}=\boldsymbol{y}$. Then

$$\mathrm{d}_p(\boldsymbol{x},\boldsymbol{x})=\left(\sum_{k=1}^{n}|x_k-x_k|^p\right)^{\frac{1}{p}}=0.$$

(iii) Let $\boldsymbol{x},\boldsymbol{y}\in\mathbf{L}^n$. Then

$$\mathrm{d}_p(\boldsymbol{x},\boldsymbol{y})=\left(\sum_{k=1}^{n}|x_k-y_k|^p\right)^{\frac{1}{p}}=\left(\sum_{k=1}^{n}|y_k-x_k|^p\right)^{\frac{1}{p}}=\mathrm{d}_p(\boldsymbol{y},\boldsymbol{x}).$$

(iv) Let $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\in\mathbf{L}^n$. Then

$$\begin{aligned}\mathrm{d}_p(\boldsymbol{x},\boldsymbol{z}) &= \left(\sum_{k=1}^{n}|x_k-z_k|^p\right)^{\frac{1}{p}}=\left(\sum_{k=1}^{n}|x_k-y_k+y_k-z_k|^p\right)^{\frac{1}{p}}\leq\\ &\leq \left(\sum_{k=1}^{n}|x_k-y_k|^p\right)^{\frac{1}{p}}+\left(\sum_{k=1}^{n}|y_k-z_k|^p\right)^{\frac{1}{p}}=\mathrm{d}_p(\boldsymbol{x},\boldsymbol{y})+\mathrm{d}_p(\boldsymbol{y},\boldsymbol{z}).\end{aligned}$$

So, d_p is a metric on $\mathbf{L}^n$. □

COROLLARY 2.3. *$\langle\mathbf{L}^n,\mathrm{d}_p\rangle$ is a metric space for $p\in[1,\infty)$.*

THEOREM 2.4. *Let $\boldsymbol{x},\boldsymbol{y}\in\mathbf{L}^n$. Then $\mathrm{d}_\infty:\mathbf{L}^n\times\mathbf{L}^n\to\mathbf{R}_0^+$, where*

$$\mathrm{d}_\infty(\boldsymbol{x},\boldsymbol{y})=\bigvee_{k=1}^{n}|x_k-y_k|,$$

is a metric on $\mathbf{L}^n$.

Proof. Let $\boldsymbol{x},\boldsymbol{y}\in\mathbf{L}^n$.

(i) $|x_k-y_k|\geq 0$ for all $x_k,y_k\in\mathbf{L}$. Therefore $\mathrm{d}_\infty(\boldsymbol{x},\boldsymbol{y})\geq 0$.

(ii) Let $\mathrm{d}_\infty(\boldsymbol{x},\boldsymbol{y})=0$. Then

$$\bigvee_{k=1}^{n}|x_k-y_k|=0,\quad \text{i.e. } (\forall k=1,\ldots,n)\ x_k=y_k \text{ and } \boldsymbol{x}=\boldsymbol{y}.$$

Vice versa, let $\boldsymbol{x}=\boldsymbol{y}$. Then

$$\mathrm{d}_\infty(\boldsymbol{x},\boldsymbol{x})=\bigvee_{k=1}^{n}|x_k-x_k|=0.$$

(iii) Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$. Then

$$\mathrm{d}_\infty(\boldsymbol{x}, \boldsymbol{y}) = \bigvee_{k=1}^{n} |x_k - y_k| = \bigvee_{k=1}^{n} |y_k - x_k| = \mathrm{d}_\infty(\boldsymbol{y}, \boldsymbol{x}).$$

(iv) Let $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbf{L}^n$. Then

$$\begin{aligned} \mathrm{d}_\infty(\boldsymbol{x}, \boldsymbol{y}) &= \bigvee_{k=1}^{n} |x_k - y_k| = \bigvee_{k=1}^{n} |x_k - z_k + z_k - y_k| \leq \\ &\leq \bigvee_{k=1}^{n} |x_k - z_k| + \bigvee_{k=1}^{n} |z_k - y_k| = \mathrm{d}_\infty(\boldsymbol{x}, \boldsymbol{z}) + \mathrm{d}_\infty(\boldsymbol{z}, \boldsymbol{y}). \end{aligned}$$

So, d_∞ is a metric on $\mathbf{L}^n$. □

COROLLARY 2.5. *$\langle \mathbf{L}^n, \mathrm{d}_\infty \rangle$ is a metric space.*

DEFINITION 2.2. *Let $n \in \mathbf{N}$. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$. Let $p \in [1, \infty) \cup \{\infty\}$ be an exponent. Then a function*

$$\mathrm{d}_p(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \left(\sum_{k=1}^{n} |x_k - y_k|^p \right)^{1/p}, & \text{for } p \in [1, \infty), \\ \lim_{p \to \infty} \mathrm{d}_p(\boldsymbol{x}, \boldsymbol{y}) = \max_{k=1,\ldots,n} |x_k - y_k|, & \text{for } p = \infty. \end{cases}$$

*is called **Minkowski metric**.*

Remark 1: The d_1 metric is called a *city-block-metric*.

Remark 2: The d_2 metric is called an *Euclidean metric*.

Remark 3: The d_∞ metric is called a *maximum metric*.

RESOURCES: [4], [21], [6]

2.1.2 Minkowski Metric Space

DEFINITION 2.3. *Let $n \in \mathbf{N}$ and $p \in [1; \infty) \cup \{\infty\}$ be a real exponent. Then the pair $\langle \mathbf{R}^n, \mathrm{d}_p \rangle$ is called **Minkowski metric space**.*

2.1.3 Unit Vector and Normalization

DEFINITION 2.4. *Let $n \in \mathbf{N}$, $p \in [1; \infty) \cup \{\infty\}$ be parameters of Minkowski metric space $\langle \mathbf{R}^n, \mathrm{d}_p \rangle$. Let $\mathbf{0} \in \mathbf{R}^n$ be zero vector. Then $\boldsymbol{u} \in \mathbf{R}^n$ is called **unit vector** just when $\mathrm{d}_p(\boldsymbol{u}, \mathbf{0}) = 1$.*

DEFINITION 2.5. *Let $\langle \mathbf{R}^n, \mathrm{d}_p \rangle$ be the Minkowski metric space. Let*

$$U_p = \{\boldsymbol{x} \in \mathbf{R}^n | \ \mathrm{d}_p(\boldsymbol{x}, \mathbf{0}) = 1\}$$

*be a set of all unit vectors. The application of any function $\mathrm{t} : \mathbf{R}^n \to U_p$ to vector $\boldsymbol{x} \in \mathbf{R}^n$ is called **free normalization** in Minkowski metric space.*

Using free normalization, we obtain $\boldsymbol{u} = \mathrm{t}(\boldsymbol{x})$ as unit vector. The unit vector $\boldsymbol{u}$ should be near the original vector $\boldsymbol{x}$. This additive condition leads to precious definition of normalization.

DEFINITION 2.6. *The function* t^* *which is a special case of free normalization* t *satisfying the condition*

$$\mathrm{d}_p(\boldsymbol{x}, \mathrm{t}^*(\boldsymbol{x})) = \mathrm{d}_p^*$$

where

$$\mathrm{d}_p^* = \min_{\boldsymbol{u} \in U_p} \mathrm{d}_p(\boldsymbol{x}, \boldsymbol{u})$$

is called a ***normalization*** *in Minkowski metric space.*

In the case of $\boldsymbol{x} \neq \mathbf{0}$ and $p \in (1; \infty)$, the global optimum is unique and easy to obtain as

$$\mathrm{t}^*(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\mathrm{d}_p(\boldsymbol{x}, \mathbf{0})} = \frac{\boldsymbol{x}}{\left(\sum_{k=1}^{n} |x_k|^p \right)^{1/p}}$$

which is discontinuous function for $\boldsymbol{x} = \mathbf{0}$.

In the case of city-block distance we have $p = 1$ and the global optimum for any $\boldsymbol{x} \notin U_1$ is not unique. When $\boldsymbol{x} \neq \mathbf{0}$ then

$$\mathrm{t}^*(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\mathrm{d}_1(\boldsymbol{x}, \mathbf{0})} = \frac{\boldsymbol{x}}{\sum_{k=1}^{n} |x_k|}$$

is one of global optimum. For $p = \infty$ and $\boldsymbol{x} \in U_\infty$, the global optimum is also multiple. There are two simple solutions. First of them for $\boldsymbol{x} \neq \mathbf{0}$ is trivial

$$\mathrm{t}^*(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\mathrm{d}_\infty(\boldsymbol{x}, \mathbf{0})} = \frac{\boldsymbol{x}}{\max_{k=1,\ldots,n} |x_k|}.$$

But the second one is better for the realization inside the $[0, 1]^n$ domain. Every coordinate is normalized by formula

$$u_k = \mathrm{t}_k^*(\boldsymbol{x}) = \begin{cases} x_k + 1 - \mathrm{d}_\infty(\boldsymbol{x}, \mathbf{0}), & \text{for } x_k \geq 0 \\ x_k - 1 + \mathrm{d}_\infty(\boldsymbol{x}, \mathbf{0}), & \text{for } x_k < 0 \end{cases}$$

This normalization is not continuous in $\mathbf{R}^n$ but defined everywhere and continuous in $[0, 1]^n$ domain.

There are three traditional values of Minkowski exponent: $p \in \{1, 2, \infty\}$. For $p = 1$ (city-block distance) we obtain the normalization condition $\sum_{k=1}^{n} |u_k| = 1$ which fulfills on the shape of n-dimensional unit 2^n-hedron. For $p = 2$ (Euclidean distance d_2) we have the normalization condition $\sum_{k=1}^{n} u_k^2 = 1$ which is true on the shape of n-dimensional unit sphere. Finally, for $p = \infty$ (maximum distance d_∞) we obtain the normalization condition $\max_{k=1,\ldots,n} |u_k| = 1$ which corresponds with the shape of n-dimensional unit cube.

2.1.4 Metric Space on $\mathbf{L}^n$

Applying any metrics to $\mathbf{L}^n$ space, we obtain the pair $\langle \mathbf{L}^n, \mathrm{d} \rangle$. Using the Minkowski metric, the pair $\langle \mathbf{L}^n, \mathrm{d}_p \rangle$ is obtained. This space will be used for the analysis of properties of any FLF.

2.1.5 Distance and Normalization as FLFs

THEOREM 2.6. *Let $n \in \mathbf{N}$. Then d_∞ distance function is FLF on $\mathbf{L}^n$.*

Proof. Let $n \in \mathbf{N}$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$. Then $\mathrm{d}_\infty(\boldsymbol{x}, \boldsymbol{y}) = \bigvee_{k=1}^{n} |x_k - y_k| = (x_1 \circ y_1) \vee \ldots \vee (x_n \circ y_n)$. Thus d_∞ is FLF according to definition 1.14. □

Remark 1: The Minkowski distances for $p \in [1, \infty)$ are not FLFs.

THEOREM 2.7. *There is a FLF normalization in $\langle \mathbf{L}^n, \mathrm{d}_\infty \rangle$.*

Proof. Let $n \in \mathbf{N}$ and $\boldsymbol{x} \in \mathbf{L}^n$. Then $x_k \geq 0$ for $k = 1, 2, \ldots, n$. Using the normalization t_k^* for $\boldsymbol{x} \in \mathbf{L}^n$, we obtain a function $\mathrm{u} : \mathbf{L}^n \to \mathbf{L}$ such as

$$u_k = \mathrm{u}_k(\boldsymbol{x}) = \mathrm{t}_k^*(\boldsymbol{x}) = x_k + 1 - \mathrm{d}_\infty(\boldsymbol{x}, \boldsymbol{0}) = x_k \oplus \neg \mathrm{d}_\infty(\boldsymbol{x}, \boldsymbol{0}).$$

Clearly, u is FLF in $\mathbf{L}$ because of d_∞ is FLF (theorem 2.6). Thus, there is a FLF normalization in $\langle \mathbf{L}^n, \mathrm{d}_\infty \rangle$. □

2.2 Properties of FLFs

2.2.1 Lipschitz Continuity

DEFINITION 2.7. *Let $n \in \mathbf{N}$ and $\varphi : \mathbf{L}^n \to \mathbf{L}$. The φ is called **Lipschitz continuous function on** L if*

$$(\exists \lambda \in \mathbf{R}_0^+)(\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n)\ \mathrm{d}(\varphi(\boldsymbol{x}), \varphi(\boldsymbol{y})) \leq \lambda \cdot \mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) \tag{2.1}$$

where $\mathrm{d}(\cdot, \cdot)$ denotes some metric.

Remark 1: It is possible to use the city-block distance d_1 as the metric and then write the formula (2.1) as

$$(\exists \lambda \in \mathbf{R}_0^+)(\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n)\ \varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) \leq \lambda \cdot \sum_{k=1}^{n} |x_k - y_k|.$$

Remark 2: The number λ is called *Lipschitz constant of* φ. It is independent on x and y.

Remark 3: The Lipschitz continuity implies a continuity of function.

RESOURCES: [18]

THEOREM 2.8. *Let $\varphi : \mathbf{L}^n \to \mathbf{L}$ be the FLF. Then φ is the Lipschitz continuous function.*

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n, a \in \mathbf{L}, \boldsymbol{x} \neq \boldsymbol{y}$. Let $\mathrm{p}(\boldsymbol{x}), \mathrm{q}(\boldsymbol{x})$ be FLFs satisfying

$$|\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| \leq \lambda_\mathrm{p} \cdot \mathrm{d}_1(\boldsymbol{x}, \boldsymbol{y}) \quad \text{and} \quad |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})| \leq \lambda_\mathrm{q} \cdot \mathrm{d}_1(\boldsymbol{x}, \boldsymbol{y}).$$

(i) Let $\varphi(\boldsymbol{x}) = a$. Then

$$|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})| \;=\; |a - a| = 0 = 0 \cdot \sum_{k=1}^{n} |x_k - y_k|. \tag{2.2}$$

(ii) Let $\varphi(\boldsymbol{x}) = x_j$ for any $1 \leq j \leq n$. Then

$$|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})| \;=\; |x_j - y_j| \;\leq\; 1 \cdot \sum_{k=1}^{n} |x_k - y_k|. \tag{2.3}$$

(iii) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \wedge \mathrm{q}(\boldsymbol{x})$. Then

$$\begin{aligned}
|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})| &= |(\mathrm{p}(\boldsymbol{x}) \wedge \mathrm{q}(\boldsymbol{x})) - (\mathrm{p}(\boldsymbol{y}) \wedge \mathrm{q}(\boldsymbol{y}))| = \\
&= |\min(\mathrm{p}(\boldsymbol{x}), \mathrm{q}(\boldsymbol{x})) - \min(\mathrm{p}(\boldsymbol{y}), \mathrm{q}(\boldsymbol{y}))| \leq \\
&\leq \max(|\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})|, |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})|) \leq \\
&\leq \max(\lambda_{\mathrm{p}}, \lambda_{\mathrm{q}}) \cdot \sum_{k=1}^{n} |x_k - y_k|.
\end{aligned} \tag{2.4}$$

(iv) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \vee \mathrm{q}(\boldsymbol{x})$. Then

$$\begin{aligned}
|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})| &= |\mathrm{p}(\boldsymbol{x}) \vee \mathrm{q}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y}) \vee \mathrm{q}(\boldsymbol{y})| = \\
&= |\max(\mathrm{p}(\boldsymbol{x}), \mathrm{q}(\boldsymbol{x})) - \max(\mathrm{p}(\boldsymbol{y}), \mathrm{q}(\boldsymbol{y}))| \leq \\
&\leq \max(|\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})|, |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})|) \leq \\
&\leq \max(\lambda_{\mathrm{p}}, \lambda_{\mathrm{q}}) \cdot \sum_{k=1}^{n} |x_k - y_k|.
\end{aligned} \tag{2.5}$$

(v) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \otimes \mathrm{q}(\boldsymbol{x})$. Then

$$\begin{aligned}
|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})| &= |(\mathrm{p}(\boldsymbol{x}) \otimes \mathrm{q}(\boldsymbol{x})) - (\mathrm{p}(\boldsymbol{y}) \otimes \mathrm{q}(\boldsymbol{y}))| = \\
&= |\max(\mathrm{p}(\boldsymbol{x}) + \mathrm{q}(\boldsymbol{x}) - 1, 0) - \max(\mathrm{p}(\boldsymbol{y}) + \mathrm{q}(\boldsymbol{y}) - 1, 0)| \leq \\
&\leq |\mathrm{p}(\boldsymbol{x}) + \mathrm{q}(\boldsymbol{x}) - 1 - (\mathrm{p}(\boldsymbol{y}) + \mathrm{q}(\boldsymbol{y}) - 1)| = \\
&= |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y}) + \mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})| \leq \\
&\leq |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| + |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})| \leq \\
&\leq \lambda_{\mathrm{p}} \cdot \sum_{k=1}^{n} |x_k - y_k| + \lambda_{\mathrm{q}} \cdot \sum_{k=1}^{n} |x_k - y_k| = \\
&= (\lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}) \cdot \sum_{k=1}^{n} |x_k - y_k|.
\end{aligned} \tag{2.6}$$

(vi) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \to \mathrm{q}(\boldsymbol{x})$. Then

$$\begin{aligned}
|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})| &= |(\mathrm{p}(\boldsymbol{x}) \to \mathrm{q}(\boldsymbol{x})) - (\mathrm{p}(\boldsymbol{y}) \to \mathrm{q}(\boldsymbol{y}))| = \\
&= |\min(1 - \mathrm{p}(\boldsymbol{x}) + \mathrm{q}(\boldsymbol{x}), 1) - \min(1 - \mathrm{p}(\boldsymbol{y}) + \mathrm{q}(\boldsymbol{y}), 1)| \leq \\
&\leq |1 - \mathrm{p}(\boldsymbol{x}) + \mathrm{q}(\boldsymbol{x}) - (1 - \mathrm{p}(\boldsymbol{y}) + \mathrm{q}(\boldsymbol{y}))| = \\
&= |-\mathrm{p}(\boldsymbol{x}) + \mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{x}) + \mathrm{q}(\boldsymbol{y})| \leq \\
&\leq |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| + |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})| \leq \\
&\leq \lambda_{\mathrm{p}} \cdot \sum_{k=1}^{n} |x_k - y_k| + \lambda_{\mathrm{q}} \cdot \sum_{k=1}^{n} |x_k - y_k| = \\
&= (\lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}) \cdot \sum_{k=1}^{n} |x_k - y_k|.
\end{aligned} \tag{2.7, 2.8}$$

(vii) Let $\varphi(\boldsymbol{x}) = \mathrm{sqrt}(\mathrm{p}(\boldsymbol{x}))$. Then

$$\begin{aligned} |\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{y})| &= \left| \frac{1+\mathrm{p}(\boldsymbol{x})}{2} - \frac{1+p(\boldsymbol{y})}{2} \right| = \\ &= \frac{1}{2} \cdot |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| \leq \\ &\leq \frac{\lambda_\mathrm{p}}{2} \cdot \sum_{k=1}^{n} |x_k - y_k|. \end{aligned} \tag{2.9}$$

So, any FLF composed from Lipschitz continuous functions using the basic $\mathrm{\L A}_{\mathrm{sqrt}}$ operators is a Lipschitz continuous function. Thus, according to definitions 1.13 (FLE) and 1.14 (FLF), any FLF is Lipschitz continuous function. □

2.2.2 Sensitivity

DEFINITION 2.8. *Let $\varphi : \mathbf{L}^n \to \mathbf{L}$ be FLF. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$. Then*

$$\lambda = \max_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|}$$

*is called a **sensitivity**.*

THEOREM 2.9. *Let $a \in \mathbf{L}$ be a constant, $\boldsymbol{x} \in \mathbf{L}^n$ and $\varphi(\boldsymbol{x})$ is FLF. Then the sensitivity of $\varphi(\boldsymbol{x}) = a$ is $\lambda_a = 0$ while sensitivity of $\varphi_j(\boldsymbol{x}) = x_j$ is $\lambda_{x_j} \leq 1$.*

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n, a \in \mathbf{L}, \boldsymbol{x} \neq \boldsymbol{y}$.
First, let $\varphi(\boldsymbol{x}) = a$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = |a - a| = 0$ and

$$\lambda_a = \max_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{0}{\sum_{k=1}^{n} |x_k - y_k|} = 0.$$

Second, let $\varphi(\boldsymbol{x}) = x_j$ for any $1 \leq j \leq n$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = |x_j - y_j| \leq \sum_{k=1}^{n} |x_k - y_k|$ and

$$\lambda_{x_j} = \max_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{|x_j - y_j|}{\sum_{k=1}^{n} |x_k - y_k|} \leq 1.$$

□

THEOREM 2.10. *Let $\mathrm{p}, \mathrm{q} : \mathbf{L}^n \to \mathbf{L}$ be a Lipschitz continuous functions with sensitivities λ_p and λ_q. Then*

$$\begin{aligned} \lambda_{\neg \mathrm{p}} &= \lambda_\mathrm{p} \\ \lambda_{\mathrm{sqrt(p)}} &= \lambda_\mathrm{p}/2 \\ \lambda_{\mathrm{p} \wedge \mathrm{q}} &\leq \max(\lambda_\mathrm{p}, \lambda_\mathrm{q}) \\ \lambda_{\mathrm{p} \vee \mathrm{q}} &\leq \max(\lambda_\mathrm{p}, \lambda_\mathrm{q}) \\ \lambda_{\mathrm{p} \otimes \mathrm{q}} &\leq \lambda_\mathrm{p} + \lambda_\mathrm{q} \\ \lambda_{\mathrm{p} \to \mathrm{q}} &\leq \lambda_\mathrm{p} + \lambda_\mathrm{q} \end{aligned}$$

$$\begin{aligned}
\lambda_{\mathrm{p}\leftrightarrow\mathrm{q}} &\leq \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}} \\
\lambda_{\mathrm{p}\circ\mathrm{q}} &\leq \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}} \\
\lambda_{\mathrm{p}\oplus\mathrm{q}} &\leq \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}} \\
\lambda_{\mathrm{p}\ominus\mathrm{q}} &\leq \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}} \\
\lambda_{n\odot\mathrm{p}} &\leq n \cdot \lambda_{\mathrm{p}} \\
\lambda_{\mathrm{p}^n} &\leq n \cdot \lambda_{\mathrm{p}}
\end{aligned}$$

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$, $\boldsymbol{x} \neq \boldsymbol{y}$. Let $\mathrm{p}(\boldsymbol{x}), \mathrm{q}(\boldsymbol{x})$ be Lipschitz continuous FLFs with the sensitivities λ_{p} and λ_{q}. Thus

$$\lambda_{\mathrm{p}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|} \text{ and } \lambda_{\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|}.$$

(i) Let $\varphi(\boldsymbol{x}) = \neg\mathrm{p}(\boldsymbol{x}) = 1 - \mathrm{p}(\boldsymbol{x})$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = |1 - \mathrm{p}(\boldsymbol{x}) - (1 - \mathrm{p}(\boldsymbol{y}))| = |-\mathrm{p}(\boldsymbol{x}) + \mathrm{p}(\boldsymbol{y})| = |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| = \mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})$ and the sensitivity of φ is

$$\lambda_{\neg\mathrm{p}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|} = \lambda_{\mathrm{p}}.$$

(ii) Let $\varphi(\boldsymbol{x}) = \mathrm{sqrt}(\mathrm{p}(\boldsymbol{x}))$. Then

$$\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = \left| \frac{1 + \mathrm{p}(\boldsymbol{x})}{2} - \frac{1 + p(\boldsymbol{y})}{2} \right| = \frac{1}{2} \cdot |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| = \frac{1}{2} \cdot \mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})$$

and the sensitivity of φ is

$$\lambda_{\mathrm{sqrt}(p)} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\frac{1}{2} \cdot (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}))}{\sum\limits_{k=1}^{n} |x_k - y_k|} = \frac{1}{2} \cdot \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|} = \frac{\lambda_{\mathrm{p}}}{2}.$$

(iii) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \wedge \mathrm{q}(\boldsymbol{x})$. According to the (2.4), the $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) \leq \max(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}), \mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))$ and the sensitivity of φ is

$$\lambda_{\mathrm{p}\wedge\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|} \leq \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\max(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}), \mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))}{\sum\limits_{k=1}^{n} |x_k - y_k|} = \max(\lambda_{\mathrm{p}}, \lambda_{\mathrm{q}}).$$

(iv) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \vee \mathrm{q}(\boldsymbol{x})$. According to the (2.5), the $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) \leq \max(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}), \mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))$ and the sensitivity of φ is

$$\lambda_{p\vee\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|} \leq \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\max(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}), \mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))}{\sum\limits_{k=1}^{n} |x_k - y_k|} = \max(\lambda_{\mathrm{p}}, \lambda_{\mathrm{q}}).$$

(v) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \otimes \mathrm{q}(\boldsymbol{x})$. According to the (2.6), the $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) \leq (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))$ and the sensitivity of φ is

$$\lambda_{p\otimes\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum\limits_{k=1}^{n} |x_k - y_k|} \leq \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))}{\sum\limits_{k=1}^{n} |x_k - y_k|} = \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}.$$

(vi) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \to \mathrm{q}(\boldsymbol{x})$. According to the (2.8), the $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) \le (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))$ and the sensitivity of φ is

$$\lambda_{p\to\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \le \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))}{\sum_{k=1}^{n} |x_k - y_k|} = \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}.$$

(vii) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \leftrightarrow \mathrm{q}(\boldsymbol{x})$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = (\mathrm{p}(\boldsymbol{x}) \leftrightarrow \mathrm{q}(\boldsymbol{x})) \circ (\mathrm{p}(\boldsymbol{y}) \leftrightarrow \mathrm{q}(\boldsymbol{y})) = \big|1 - |\mathrm{p}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{x})| - \big(1 - |\mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{y})|\big)\big| = \big||\mathrm{p}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{x})| - |\mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{y})|\big| \le |\mathrm{p}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y}) + \mathrm{q}(\boldsymbol{y})| \le |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| + |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})| = (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))$ and

$$\lambda_{p\leftrightarrow\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \le \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))}{\sum_{k=1}^{n} |x_k - y_k|} = \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}.$$

(viii) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{x})$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = \big||\mathrm{p}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{x})| - |\mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{y})|\big| \le |\mathrm{p}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y}) + \mathrm{q}(\boldsymbol{y})| \le |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| + |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})| = (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))$ and

$$\lambda_{p\leftrightarrow\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \le \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))}{\sum_{k=1}^{n} |x_k - y_k|} = \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}.$$

(ix) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \oplus \mathrm{q}(\boldsymbol{x})$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = |\min(\mathrm{p}(\boldsymbol{x}) + \mathrm{q}(\boldsymbol{x}), 1) - \min(\mathrm{p}(\boldsymbol{y}) + \mathrm{q}(\boldsymbol{y}), 1)| \le \big||\mathrm{p}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{x})| - |\mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{y})|\big| \le |\mathrm{p}(\boldsymbol{x}) + \mathrm{q}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{y})| \le |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| + |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})| = (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))$ and

$$\lambda_{p\oplus\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \le \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))}{\sum_{k=1}^{n} |x_k - y_k|} = \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}.$$

(x) Let $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x}) \ominus \mathrm{q}(\boldsymbol{x})$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = |\max(\mathrm{p}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{x}), 0) - \max(\mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{y}), 0))| \le \big||\mathrm{p}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{x})| - |\mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{y})|\big| \le |\mathrm{p}(\boldsymbol{x}) + \mathrm{q}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y}) - \mathrm{q}(\boldsymbol{y})| \le |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| + |\mathrm{q}(\boldsymbol{x}) - \mathrm{q}(\boldsymbol{y})| = (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))$ and

$$\lambda_{p\oplus\mathrm{q}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \le \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{(\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y})) + (\mathrm{q}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{y}))}{\sum_{k=1}^{n} |x_k - y_k|} = \lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}.$$

(xi) Let $n \in \mathbf{N}$ and $\varphi(\boldsymbol{x}) = n \odot \mathrm{p}(\boldsymbol{x})$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = |\min(n \cdot \mathrm{p}(\boldsymbol{x}), 1) - \min(n \cdot \mathrm{p}(\boldsymbol{y}), 1)| \le |n \cdot \mathrm{p}(\boldsymbol{x}) - n \cdot \mathrm{p}(\boldsymbol{y})| \le n \cdot |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| = n \cdot (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}))$ and

$$\lambda_{n\odot\mathrm{p}} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \le \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{n \cdot (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}))}{\sum_{k=1}^{n} |x_k - y_k|} = n \cdot \lambda_{\mathrm{p}}.$$

(xii) Let $n \in \mathbf{N}$ and $\varphi(\boldsymbol{x}) = \mathrm{p}(\boldsymbol{x})^n$. Then $\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y}) = |\max(n \cdot \mathrm{p}(\boldsymbol{x}) - n + 1, 0) - \max(n \cdot \mathrm{p}(\boldsymbol{y}) - n + 1, 0)| \le \big||n \cdot \mathrm{p}(\boldsymbol{x}) - n + 1| - |n \cdot \mathrm{p}(\boldsymbol{y}) - n + 1|\big| \le |n \cdot \mathrm{p}(\boldsymbol{x}) - n + 1 - n \cdot \mathrm{p}(\boldsymbol{y}) + n - 1| \le |n \cdot \mathrm{p}(\boldsymbol{x}) - n \cdot \mathrm{p}(\boldsymbol{y})| = n \cdot |\mathrm{p}(\boldsymbol{x}) - \mathrm{p}(\boldsymbol{y})| = n \cdot (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}))$ and the sensitivity of φ is

$$\lambda_{p^n} = \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\varphi(\boldsymbol{x}) \circ \varphi(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \le \max_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{n \cdot (\mathrm{p}(\boldsymbol{x}) \circ \mathrm{p}(\boldsymbol{y}))}{\sum_{k=1}^{n} |x_k - y_k|} = n \cdot \lambda_{\mathrm{p}}. \qquad \square$$

2.2.3 Upper Bounds of Sensitivity

It is very useful to realize the data processing systems with low sensitivity from a pragmatic point of view. The knowledge of upper bounds of basic FLF sensitivities can help to design a compound FLFs with constrained sensitivity. The upper bounds of sensitivity of useful FLFs are presented in Tab. 2.1.

FLF	λ_{upper}
$\mathrm{p}(\boldsymbol{x})$	λ_{p}
$\mathrm{q}(\boldsymbol{x})$	λ_{q}
$\neg(\mathrm{p}(\boldsymbol{x}))$	λ_{p}
$\mathrm{sqrt}(\mathrm{p}(\boldsymbol{x}))$	$\lambda_{\mathrm{p}}/2$
$\mathrm{p}(\boldsymbol{x}) \wedge \mathrm{q}(\boldsymbol{x})$	$\max(\lambda_{\mathrm{p}}, \lambda_{\mathrm{q}})$
$\mathrm{p}(\boldsymbol{x}) \vee \mathrm{q}(\boldsymbol{x})$	$\max(\lambda_{\mathrm{p}}, \lambda_{\mathrm{q}})$
$\mathrm{p}(\boldsymbol{x}) \otimes \mathrm{q}(\boldsymbol{x})$	$\lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}$
$\mathrm{p}(\boldsymbol{x}) \rightarrow \mathrm{q}(\boldsymbol{x})$	$\lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}$
$\mathrm{p}(\boldsymbol{x}) \leftrightarrow \mathrm{q}(\boldsymbol{x})$	$\lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}$
$\mathrm{p}(\boldsymbol{x}) \circ \mathrm{q}(\boldsymbol{x})$	$\lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}$
$\mathrm{p}(\boldsymbol{x}) \oplus \mathrm{q}(\boldsymbol{x})$	$\lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}$
$\mathrm{p}(\boldsymbol{x}) \ominus \mathrm{q}(\boldsymbol{x})$	$\lambda_{\mathrm{p}} + \lambda_{\mathrm{q}}$
$n \odot \mathrm{p}(\boldsymbol{x})$	$n \cdot \lambda_{\mathrm{p}}$
$(\mathrm{p}(\boldsymbol{x}))^n$	$n \cdot \lambda_{\mathrm{p}}$

Table 2.1: Upper bounds of basic FLF sensitivities

The sqrt function decreases the FLF sensitivity, the operators $\neg$, $\wedge$, $\vee$ do not increase the sensitivity of FLF while the remaining operators increase the sensitivity of FLF.

EXAMPLE 1: Let $\mathrm{p}_1, \mathrm{p}_2$ be FLFs with sensitivities λ_1, λ_2. Then the sensitivity of mean of two is $\lambda_{\mathrm{AV2}} \leq \frac{\lambda_1+\lambda_2}{2}$.

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}$, $\boldsymbol{x} \neq \boldsymbol{y}$ and p_1 and p_2 be FLFs with sensitivities λ_1 and λ_2. Then $\mathrm{AV2}(\boldsymbol{x}) \circ \mathrm{AV2}(\boldsymbol{y}) = \Big|\Big(\mathrm{sqrt}\big(\mathrm{p}_1(\boldsymbol{x})\big) \otimes \mathrm{sqrt}\big(\mathrm{p}_2(\boldsymbol{x})\big)\Big) - \Big(\mathrm{sqrt}\big(\mathrm{p}_1(\boldsymbol{y})\big) \otimes \mathrm{sqrt}\big(\mathrm{p}_2(\boldsymbol{y})\big)\Big)\Big| = \frac{1}{2} \cdot \big|\mathrm{p}_1(\boldsymbol{x}) + \mathrm{p}_2(\boldsymbol{x}) - \mathrm{p}_1(\boldsymbol{y}) - \mathrm{p}_2(\boldsymbol{y})\big| \leq \frac{1}{2} \cdot \big(\big|\mathrm{p}_1(\boldsymbol{x}) - \mathrm{p}_1(\boldsymbol{y})\big| + \big|\mathrm{p}_2(\boldsymbol{x}) + \mathrm{p}_2(\boldsymbol{y})\big|\big) = \frac{1}{2} \cdot \Big(\big(\mathrm{p}_1(\boldsymbol{x}) \circ \mathrm{p}_1(\boldsymbol{y})\big) + \big(\mathrm{p}_2(\boldsymbol{x}) \circ \mathrm{p}_2(\boldsymbol{y})\big)\Big)$ and the λ_{AV2} is

$$\lambda_{\mathrm{AV2}} = \max_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{\mathrm{AV2}(\boldsymbol{x}) \circ \mathrm{AV2}(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \leq \frac{\lambda_1 + \lambda_2}{2} \qquad \square$$

EXAMPLE 2: Let $\mathrm{p}_1, \ldots, \mathrm{p}_4$ be FLFs with sensitivities $\lambda_1, \ldots, \lambda_4$. Then the sensitivity of mean of four is $\lambda_{\mathrm{AV4}} \leq \frac{\lambda_1+\lambda_2+\lambda_3+\lambda_4}{4}$.

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}$, $\boldsymbol{x} \neq \boldsymbol{y}$ and $\mathrm{p}_1, \ldots, \mathrm{p}_4$ be FLFs with sensitivities $\lambda_1, \ldots, \lambda_4$. Then $\mathrm{AV4}(\boldsymbol{x}) \circ \mathrm{AV4}(\boldsymbol{y}) = \Big|\mathrm{sqrt}\Big(\mathrm{sqrt}\big(\mathrm{p}_1(\boldsymbol{x})\big) \otimes \mathrm{sqrt}\big(\mathrm{p}_2(\boldsymbol{x})\big)\Big) \otimes \mathrm{sqrt}\Big(\mathrm{sqrt}\big(\mathrm{p}_3(\boldsymbol{x})\big) \otimes \mathrm{sqrt}\big(\mathrm{p}_4(\boldsymbol{x})\big)\Big) -$

$\mathrm{sqrt}\Big(\mathrm{sqrt}\big(\mathrm{p}_1(\boldsymbol{y})\big) \otimes \mathrm{sqrt}\big(\mathrm{p}_2(\boldsymbol{y})\big)\Big) \otimes \mathrm{sqrt}\Big(\mathrm{sqrt}\big(\mathrm{p}_3(\boldsymbol{y})\big) \otimes \mathrm{sqrt}\big(\mathrm{p}_4(\boldsymbol{y})\big)\Big)\Big| = \frac{1}{4} \cdot \big|\mathrm{p}_1(\boldsymbol{x}) + \mathrm{p}_2(\boldsymbol{x}) + \mathrm{p}_3(\boldsymbol{x}) + \mathrm{p}_4(\boldsymbol{x}) - \mathrm{p}_1(\boldsymbol{y}) - \mathrm{p}_2(\boldsymbol{y}) - \mathrm{p}_3(\boldsymbol{y}) - \mathrm{p}_4(\boldsymbol{y})\big| \leq \frac{1}{4} \cdot \big(\big|\mathrm{p}_1(\boldsymbol{x}) - \mathrm{p}_1(\boldsymbol{y})\big| +$

$\big|\mathrm{p}_2(\boldsymbol{x}) - \mathrm{p}_2(\boldsymbol{y})\big| + \big|\mathrm{p}_3(\boldsymbol{x}) - \mathrm{p}_3(\boldsymbol{y})\big| + \big|\mathrm{p}_4(\boldsymbol{x}) - \mathrm{p}_4(\boldsymbol{y})\big|\big) = \frac{1}{4} \cdot \Big(\big(\mathrm{p}_1(\boldsymbol{x}) \circ \mathrm{p}_1(\boldsymbol{y})\big) + \big(\mathrm{p}_2(\boldsymbol{x}) \circ \mathrm{p}_2(\boldsymbol{y})\big) + \big(\mathrm{p}_3(\boldsymbol{x}) \circ \mathrm{p}_3(\boldsymbol{y})\big) + \big(\mathrm{p}_4(\boldsymbol{x}) \circ \mathrm{p}_4(\boldsymbol{y})\big)\Big)$ and the λ_{AV4} is

$$\lambda_{\mathrm{AV4}} = \max_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{\mathrm{AV4}(\boldsymbol{x}) \circ \mathrm{AV4}(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \leq \frac{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4}{4} \qquad \square$$

EXAMPLE 3: Let $\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3$ be FLFs with sensitivities $\lambda_1, \lambda_2, \lambda_3$. Then the sensitivity of median of three is $\lambda_{\mathrm{ME3}} \leq \max(\lambda_1, \lambda_2, \lambda_3)$.

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}$, $\boldsymbol{x} \neq \boldsymbol{y}$ and $\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3$ be FLFs with sensitivities $\lambda_1, \lambda_2, \lambda_3$. Then $\mathrm{ME3}(\boldsymbol{x}) \circ \mathrm{ME3}(\boldsymbol{y}) = \Big|\Big(\big(\mathrm{p}_1(\boldsymbol{x}) \wedge \mathrm{p}_2(\boldsymbol{x})\big) \vee \big(\mathrm{p}_1(\boldsymbol{x}) \wedge \mathrm{p}_3(\boldsymbol{x})\big) \vee \big(\mathrm{p}_2(\boldsymbol{x}) \wedge \mathrm{p}_3(\boldsymbol{x})\big)\Big) - \Big(\big(\mathrm{p}_1(\boldsymbol{y}) \wedge \mathrm{p}_2(\boldsymbol{y})\big) \vee \big(\mathrm{p}_1(\boldsymbol{y}) \wedge \mathrm{p}_3(\boldsymbol{y})\big) \vee \big(\mathrm{p}_2(\boldsymbol{y}) \wedge \mathrm{p}_3(\boldsymbol{y})\big)\Big)\Big| = \Big|\max\Big(\min\big(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_2(\boldsymbol{x})\big), \min\big(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_3(\boldsymbol{x})\big), \min\big(\mathrm{p}_2(\boldsymbol{x}), \mathrm{p}_3(\boldsymbol{x})\big)\Big) - \max\Big(\min\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{y})\big), \min\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_3(\boldsymbol{y})\big), \min\big(\mathrm{p}_2(\boldsymbol{y}), \mathrm{p}_3(\boldsymbol{y})\big)\Big)\Big| \leq \max\big(\big|\mathrm{p}_1(\boldsymbol{x}) - \mathrm{p}_1(\boldsymbol{y})\big|, \big|\mathrm{p}_2(\boldsymbol{x}) - \mathrm{p}_2(\boldsymbol{y})\big|, \big|\mathrm{p}_3(\boldsymbol{x}) - \mathrm{p}_3(\boldsymbol{y})\big|\big)$, i.e. $\mathrm{ME3}(\boldsymbol{x}) \circ \mathrm{ME3}(\boldsymbol{y}) \leq \max\big(\mathrm{p}_1(\boldsymbol{x}) \circ \mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{x}) \circ \mathrm{p}_2(\boldsymbol{y}), \mathrm{p}_3(\boldsymbol{x}) \circ \mathrm{p}_3(\boldsymbol{y})\big)$ and the λ_{ME3} is

$$\lambda_{\mathrm{ME3}} = \max_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{\mathrm{ME3}(\boldsymbol{x}) \circ \mathrm{ME3}(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \leq \max(\lambda_1, \lambda_2, \lambda_3) \qquad \square$$

EXAMPLE 4: Let $\mathrm{p}_1, \ldots, \mathrm{p}_4$ be FLFs with sensitivities $\lambda_1, \ldots, \lambda_4$. Then the sensitivity of median of four is $\lambda_{\mathrm{ME4}} \leq \max(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$.

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}$, $\boldsymbol{x} \neq \boldsymbol{y}$ and $\mathrm{p}_1, \ldots, \mathrm{p}_4$ be FLFs with sensitivities $\lambda_1, \ldots, \lambda_4$. Then $\mathrm{ME4}(\boldsymbol{x}) \circ \mathrm{ME4}(\boldsymbol{y}) = \Big|\mathrm{sqrt}\Big(\big(\mathrm{p}_1(\boldsymbol{x}) \wedge \mathrm{p}_2(\boldsymbol{x}) \wedge (\mathrm{p}_3(\boldsymbol{x}) \vee \mathrm{p}_4(\boldsymbol{x}))\big) \vee \big((\mathrm{p}_1(\boldsymbol{x}) \vee \mathrm{p}_2(\boldsymbol{x})) \wedge \mathrm{p}_3(\boldsymbol{x}) \wedge \mathrm{p}_4(\boldsymbol{x})\big)\Big) \otimes \mathrm{sqrt}\Big(\big(\mathrm{p}_1(\boldsymbol{x}) \vee \mathrm{p}_2(\boldsymbol{x}) \vee (\mathrm{p}_3(\boldsymbol{x}) \wedge \mathrm{p}_4(\boldsymbol{x}))\big) \wedge \big((\mathrm{p}_1(\boldsymbol{x}) \wedge \mathrm{p}_2(\boldsymbol{x})) \vee \mathrm{p}_3(\boldsymbol{x}) \vee \mathrm{p}_4(\boldsymbol{x})\big)\Big) - \Big(\mathrm{sqrt}\Big(\big(\mathrm{p}_1(\boldsymbol{y}) \wedge \mathrm{p}_2(\boldsymbol{y}) \wedge (\mathrm{p}_3(\boldsymbol{y}) \vee \mathrm{p}_4(\boldsymbol{y}))\big) \vee \big((\mathrm{p}_1(\boldsymbol{y}) \vee \mathrm{p}_2(\boldsymbol{y})) \wedge \mathrm{p}_3(\boldsymbol{y}) \wedge \mathrm{p}_4(\boldsymbol{y})\big)\Big) \otimes \mathrm{sqrt}\Big(\big(\mathrm{p}_1(\boldsymbol{y}) \vee \mathrm{p}_2(\boldsymbol{y}) \vee (\mathrm{p}_3(\boldsymbol{y}) \wedge \mathrm{p}_4(\boldsymbol{y}))\big) \wedge \big((\mathrm{p}_1(\boldsymbol{y}) \wedge \mathrm{p}_2(\boldsymbol{y})) \vee \mathrm{p}_3(\boldsymbol{y}) \vee \mathrm{p}_4(\boldsymbol{y})\big)\Big)\Big)\Big| = \max\Big(\frac{1}{2} \cdot \Big(1 + \max\Big(\min\big(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_2(\boldsymbol{x}), \max(\mathrm{p}_3(\boldsymbol{x}), \mathrm{p}_4(\boldsymbol{x}))\big), \min\big(\max(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_2(\boldsymbol{x})), \mathrm{p}_3(\boldsymbol{x}), \mathrm{p}_4(\boldsymbol{x})\big)\Big)\Big) + \frac{1}{2} \cdot \Big(1 + \min\Big(\max\big(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_2(\boldsymbol{x}), \min(\mathrm{p}_3(\boldsymbol{x}), \mathrm{p}_4(\boldsymbol{x}))\big), \max\big(\min(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_2(\boldsymbol{x})), \mathrm{p}_3(\boldsymbol{x}), \mathrm{p}_4(\boldsymbol{x})\big)\Big)\Big) - 1, 0\Big) - \max\Big(\frac{1}{2} + \frac{1}{2} \max\Big(\min\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{y}), \max\big(\mathrm{p}_3(\boldsymbol{y}), \mathrm{p}_4(\boldsymbol{y})\big)\big), \min\big(\max\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{y})\big), \mathrm{p}_3(\boldsymbol{y}), \mathrm{p}_4(\boldsymbol{y})\big)\Big) + \frac{1}{2} + \frac{1}{2} \min\Big(\max\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{y}), \min(\mathrm{p}_3(\boldsymbol{y}), \mathrm{p}_4(\boldsymbol{y}))\big), \max\Big(\min\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{y})\big), \mathrm{p}_3(\boldsymbol{y}), \mathrm{p}_4(\boldsymbol{y})\big)\Big) - 1, 0\Big) \leq \max\Big(\frac{1}{2} \cdot \Big(\max\big(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_2(\boldsymbol{x}), \mathrm{p}_3(\boldsymbol{x}), \mathrm{p}_4(\boldsymbol{x})\big) + \max\big(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_2(\boldsymbol{x}), \mathrm{p}_3(\boldsymbol{x}), \mathrm{p}_4(\boldsymbol{x})\big)\Big) - \frac{1}{2}\Big(\max\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{y}), \mathrm{p}_3(\boldsymbol{y}), \mathrm{p}_4(\boldsymbol{y})\big) + \max\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{y}), \mathrm{p}_3(\boldsymbol{y}), \mathrm{p}_4(\boldsymbol{y})\big)\Big), 0\Big) = \max\Big(\max\big(\mathrm{p}_1(\boldsymbol{x}), \mathrm{p}_2(\boldsymbol{x}), \mathrm{p}_3(\boldsymbol{x}), \mathrm{p}_4(\boldsymbol{x})\big) - \max\big(\mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{y}), \mathrm{p}_3(\boldsymbol{y}), \mathrm{p}_4(\boldsymbol{y})\big), 0\Big) \leq$

$\max\Big(\big|\mathrm{p}_1(\boldsymbol{x}) - \mathrm{p}_1(\boldsymbol{y})\big|, \big|\mathrm{p}_2(\boldsymbol{x}) - \mathrm{p}_2(\boldsymbol{y})\big|, \big|\mathrm{p}_3(\boldsymbol{x}) - \mathrm{p}_3(\boldsymbol{y})\big|, \big|\mathrm{p}_4(\boldsymbol{x}) - \mathrm{p}_4(\boldsymbol{y})\big|\Big)$. Thus,

$\mathrm{ME4}(\boldsymbol{x}) \circ \mathrm{ME4}(\boldsymbol{y}) \leq \max\big(\mathrm{p}_1(\boldsymbol{x}) \circ \mathrm{p}_1(\boldsymbol{y}), \mathrm{p}_2(\boldsymbol{x}) \circ \mathrm{p}_2(\boldsymbol{y}), \mathrm{p}_3(\boldsymbol{x}) \circ \mathrm{p}_3(\boldsymbol{y}), \mathrm{p}_4(\boldsymbol{x}) \circ \mathrm{p}_4(\boldsymbol{y})\big)$

and the λ_{ME4} is

$$\lambda_{\mathrm{ME4}} = \max_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{\mathrm{ME4}(\boldsymbol{x}) \circ \mathrm{ME4}(\boldsymbol{y})}{\sum_{k=1}^{n} |x_k - y_k|} \leq \max(\lambda_1, \lambda_2, \lambda_3, \lambda_4) \qquad \square$$

Chapter 3

Processing of 2D Gray Image

3.1 Basic Terms

3.1.1 Pixel

DEFINITION 3.1. *The smallest discrete rectangular element of picture is called a* ***pixel****.*

Remark 1: The pixel has two properties:

- a size—this property is dependent on resolution of the apparatus which was used for picture digitizing,
- a color—a property visible on a screen.

Remark 2: The size is common property of all pixels in given picture, while the color is typically different for each pixel.

DEFINITION 3.2. *Let $\mathcal{X}$ be a given pixel, $n \in \mathbf{N}$ and $M \subset \mathbf{R}$. Then a vector $x \in M^n$, which describes the color property of given pixel, is called a* ***pixel vector*** *(related to $\mathcal{X}$). In case of $n = 1$, the pixel vector is called a* ***pixel value*** *or pixel intensity.*

Remark 1: The length of pixel vector and range of its element (a set M) determines a type of picture (see a page 26).

Remark 2: The resulting color of given pixel is produced as visual combination of all values in its pixel vector.

Remark 3: Each pixel is represented by its pixel vector in digital 2D image processing.

3.1.2 2D Image

DEFINITION 3.3. *Let $r, c \in \mathbf{N}$. Then the matrix of size (r, c), which consists of $r \cdot c$ pixels, is called a 2D digital* ***image****.*

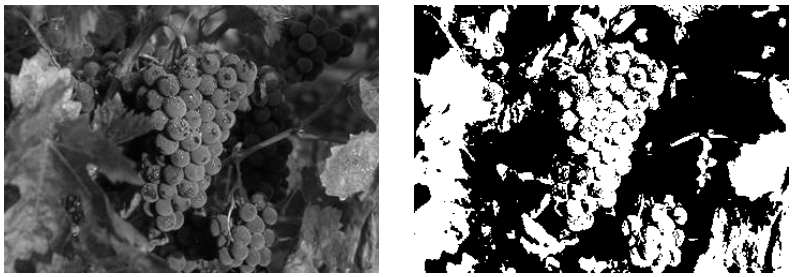
Remark 1: An image is a discrete 2D signal.

Remark 2: From a mathematical point of view, the image is described as matrix with r rows and c columns where each element $x_{i,j} \in M^n$ ($n \in \mathbf{N}$, $M \subset \mathbf{R}$, $1 \leq i \leq r$, $1 \leq j \leq c$) is the pixel vector. Therefore, the image can be stored in a computer memory, and manipulated by a processor.

Remark 3: There are several kinds of images (depending on length of pixel vector n and range of its values M), for example:

- *RGB color image* is the image where $n = 3$ and $M = [0, 1]$. Each element of pixel vector means the intensity of red, green and blue colors. Resulting color of each pixel is produced as visual combination of the three primary colors.
- *Gray image* (graylevel or grayscale image) is the image where $n = 1$ and $M = [0, 1]$. Thus, each pixel value represents intensity of gray scale and each pixel color is any shade of gray.
  Grayscale images are very known as a subject of image processing (chapter 8).
- *Binary image* is the image where $n = 1$, $M = \{0, 1\}$. Thus, the pixel values are one or zero only, i.e. the picture consists of foreground color (typically white) and background one (typically black).

The Fig. 3.1 shows gray and binary images.



(a) (b)

Figure 3.1: Two types of digital images: (a) gray, (b) binary

In following text, the all terms and definitions will be concerned with gray images.

3.1.3 Pixel Neighborhood

DEFINITION 3.4. *Let $R \in \mathbf{N}_0$, $\mathcal{P}$ be any 2D gray image of size (r, c), $i, j \in \mathbf{N}$, $1 \leq i \leq r$ and $1 \leq j \leq c$. Let $\mathcal{X}_{i,j}$ be a given pixel from $\mathcal{P}$. Then a subset*

$$\mathcal{N}_R(\mathcal{X}_{i,j}) = \{\mathcal{X}_{u,v} \in \mathcal{P}; |u - i| \leq R \wedge |v - j| \leq R\}$$

*is called a **pixel neighborhood** and $\mathcal{X}_{i,j}$ is called a **central pixel**.*

From a graphical point of view, the pixel neighborhood is a small square matrix of odd size $(2R + 1)^2$ where the original pixel is just in the matrix center (central pixel). An example of 3-by-3 pixel neighborhood ($R = 1$) is demonstrated in the Fig. 3.2.

Remark 1: The pixel neighborhood is also called a *window*.

Remark 2: The R is called a *radius of pixel neighborhood*. There is $R \leq 3$ in many applications.

Remark 3: For $R = 0$, the pixel neighborhood is degenerated to the central pixel.

Remark 4: In the case of $R > 0$, there is necessary to extend the original image by left and right columns and top and bottom rows to prevent the degeneration of the neighborhood. The extension should be realized by:

- zero padding
- constant replication
- periodic replication
- mirror replication

as depicted in Fig. 3.3

Remark 5: The pixel neighborhood can be represented as list (see Fig. 3.6) of pixel values.

Remark 6: The pixel neighborhood plays an important role in the pixel-by-pixel enhancement of the whole image.

$\mathcal{X}_{i-1,j-1}$	$\mathcal{X}_{i-1,j}$	$\mathcal{X}_{i-1,j+1}$
$\mathcal{X}_{i,j-1}$	$\mathcal{X}_{i,j}$	$\mathcal{X}_{i,j+1}$
$\mathcal{X}_{i+1,j-1}$	$\mathcal{X}_{i+1,j}$	$\mathcal{X}_{i+1,j+1}$

Figure 3.2: Pixel neighborhood for $R = 1$ with a central pixel $\mathcal{X}_{i,j}$

(a) (b) (c) (d)

Figure 3.3: Image extension for $R = 5$: (a) zero padding, (b) constant replication, (c) periodic replication, (d) mirror replication

3.1.4 Mask

DEFINITION 3.5. *Let $R \in \mathbf{N}_0$ and $r = 2R + 1$. Then the square matrix $\mathbf{M} = (m_{i,j})$ of size (r, r), which consists of non-negative integer numbers, is called a* ***mask****.*

Let $k \in \mathbf{N}_0$, $k = \sum_{i=1}^{r} \sum_{j=1}^{r} m_{i,j}$. Then k is called a ***mask capacity****.*

Remark 1: The R is called a *mask radius*.

Remark 2: The mask $\mathbf{M} = (m_{i,j})$ of type (r, r) can be represented as vector of weights $\boldsymbol{w}$ of size $n = r^2$ using formula $w_k = m_{i,j}$ where $k = (i-1) \cdot r + j$ $(i, j = 1, \ldots, r)$. The Fig. 3.4 demonstrates a connection between mask and vector of weights.

Remark 3: In case of local processing of gray image, the R determines a radius of pixel neighborhood. Then a weighted list (section 3.1.5) of k pixel values where k is mask capacity can be formed from a pixel neighborhood using each mask element as frequency of appropriate pixel in the list.

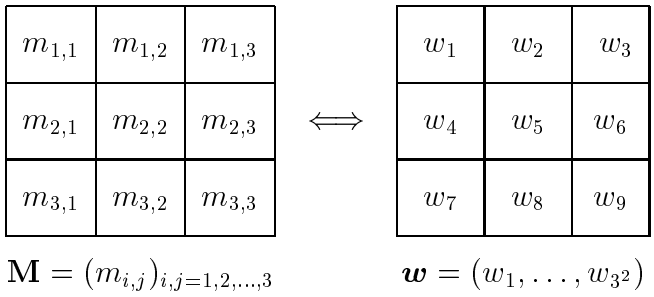


Figure 3.4: Mask as vector of weighs for $R = 1$

DEFINITION 3.6. *Let $R \in \mathbf{N}_0$, $r = 2R + 1$ and $\mathbf{M} = (m_{i,j})$ be a mask of size (r, r).*

(a) *The $\mathbf{M}$ is called a **box mask**, iff $m_{i,j} = 1$ for $i, j = 1, 2, \ldots, r$.*

(b) *The $\mathbf{M}$ is called a **binomial mask**, iff $m_{i,j} = \binom{r-1}{i-1} \cdot \binom{r-1}{j-1}$ for $i, j = 1, 2, \ldots, r$.*

The Fig. 3.5 shows various types of masks. The mask symmetry is not necessary but useful. The $\mathbf{M}_1$ is a box mask with $R = 1$ and the $\mathbf{M}_2$ is a binomial mask with the same radius. The symmetric and circular masks (e.g. $\mathbf{M}_4$, $\mathbf{M}_1$ and $\mathbf{M}_2$) are useful for image de-noising.

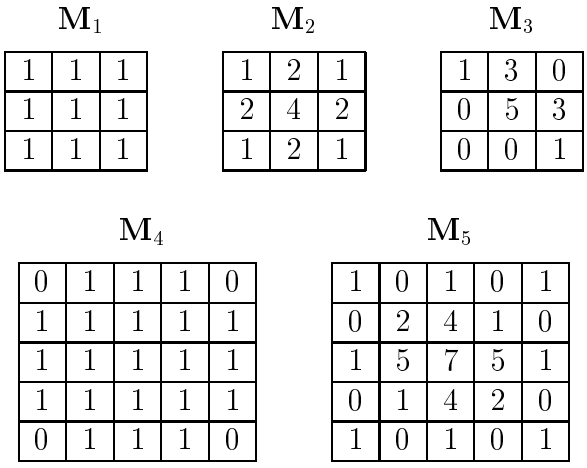


Figure 3.5: Various types of masks

3.1.5 List and Weighted Lists

DEFINITION 3.7. *Let $n \in \mathbf{N}$ and $\boldsymbol{x} \in \mathbf{L}^n$. Then n-tuple $L = (x_1, \ldots, x_n)$ is called a **list**. The list can be also denoted as $L = \overset{n}{\underset{k=1}{\Diamond}} x_k$.*

Remark 1: The pixel vectors in given pixel neighborhood can be denoted as the list of pixel vectors. An example of a list formed from the $\mathcal{X}_{4,3}$ neighborhood with $R = 1$ is depicted in Fig. 3.6.

$\mathcal{N}_1(\mathcal{X}_{4,3})$:

$\mathcal{X}_{3,2}$	$\mathcal{X}_{3,3}$	$\mathcal{X}_{3,4}$
$\mathcal{X}_{4,2}$	$\mathcal{X}_{4,3}$	$\mathcal{X}_{4,4}$
$\mathcal{X}_{5,2}$	$\mathcal{X}_{5,3}$	$\mathcal{X}_{5,4}$

$$\iff \quad L = (x_{3,2}, x_{3,3}, x_{3,4}, x_{4,2}, x_{4,3}, x_{4,4}, x_{5,2}, x_{5,3}, x_{5,4})$$

Figure 3.6: Pixel neighborhood as list

DEFINITION 3.8. *Let $n \in \mathbf{N}$, $\boldsymbol{x} \in \mathbf{L}^n$ and $\boldsymbol{w} \in \mathbf{N}_0^n$. Then a **weighted list** is defined as $L = \overset{n}{\underset{k=1}{\Diamond}} w_k x_k$ where $w_k x_k = (\underbrace{x_k, x_k, \ldots, x_k}_{w_k})$. The $\boldsymbol{w}$ is called a **vector of weights**.*

EXAMPLE: Let $\boldsymbol{w} = (1, 0, 3, 2, 1)$ and $\boldsymbol{x} = (0.2, 0.15, 0.31, 0.1, 0.5)$. Then the weighted list $L = (0.2, 0.31, 0.31, 0.31, 0.1, 0.1, 0.5)$ is obtained.

Remark 1: The weighted list can be formed from given pixel neighborhood using any mask with the same radius as depicted in the Fig. 3.7.

Remark 2: The weighted lists are used in local FLF processing of 2D image.

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 3 & 2 \\ 0 & 2 & 0 \end{pmatrix} \quad \searrow$$

$\mathcal{N}_1(\mathcal{X}_{i,j})$:

$\mathcal{X}_{i-1,j-1}$	$\mathcal{X}_{i-1,j}$	$\mathcal{X}_{i-1,j+1}$
$\mathcal{X}_{i,j-1}$	$\mathcal{X}_{i,j}$	$\mathcal{X}_{i,j+1}$
$\mathcal{X}_{i+1,j-1}$	$\mathcal{X}_{i+1,j}$	$\mathcal{X}_{i+1,j+1}$

$\nearrow$

$$L = (x_{i-1,j-1}, x_{i-1,j}, x_{i,j-1}, x_{i,j}, x_{i,j}, x_{i,j}, x_{i,j+1}, x_{i,j+1}, x_{i+1,j}, x_{i+1,j})$$

Figure 3.7: Weighted list forming from mask $\mathbf{M}$ ($R = 1$) and neighborhood of pixel $\mathcal{X}_{i,j}$

3.1.6 Local Processing

DEFINITION 3.9. *Let $\mathcal{P}$ be any 2D gray image of size (r, c), $\mathcal{X}_{i,j}$ be a given pixel from $\mathcal{P}$, $R \in \mathbf{N}_0$ be a radius of $\mathcal{X}_{i,j}$ neighborhood, $\mathbf{M} = (m_{i,j})$ be any mask with radius R and*

mask capacity $k > 1$. *Then a* ***local processing*** *is equivalent to a mapping* $\mathrm{f} : \mathbf{L}^k \to \mathbf{L}$ *where* $x^*_{i,j} = \mathrm{f}(\boldsymbol{w})$ *is new intensity (value) of central pixel* $\mathcal{X}_{i,j}$ *and* $\boldsymbol{w} = (w_1, \ldots, w_k)$ *is weighted list, which was formed from the* $\mathcal{X}_{i,j}$ *neighborhood using the mask* $\mathbf{M}$.

Function f *is called a* ***local processing function***.

The Fig. 3.8 illustrates a scheme of local processing with $R = 1$.

Remark 1: In case of any gray image enhancement, the data from each pixel neighborhood are typically proceed to obtain a new pixel intensities $x^*_{i,j}$. A pixel-by-pixel processing leads to the resulting enhanced gray image.

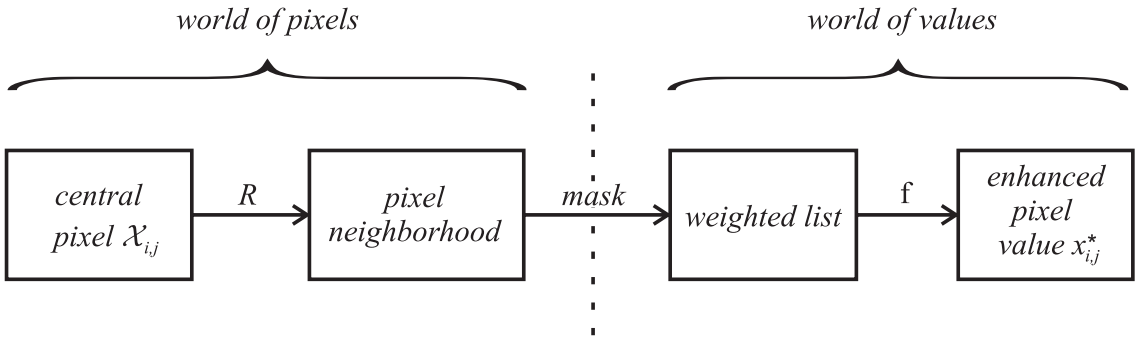


Figure 3.8: Local processing scheme

values of pixel neighborhood	resulting value of central pixel		
	median	mean	maximum
0.49 0.51 0.52 0.47 1.00 0.50 0.50 0.52 0.51	0.51	0.5578	1.00
1.00 0.50 0.48 0.48 0.47 0.53 0.50 0.51 0.48	0.50	0.5533	1.00
0.49 0.52 0.50 0.48 0.49 1.00 0.46 0.51 0.48	0.49	0.5478	1.00

Table 3.1: Examples of local processing

The Tab. 3.1 shows examples of local processing, which use box mask for three different neighborhoods and three differential local processing functions:

median: $\mathrm{f}(\boldsymbol{x}) = \underset{k=1,\ldots,9}{\mathrm{median}}\, x_k$, mean: $\mathrm{f}(\boldsymbol{x}) = \frac{1}{9} \cdot \sum_{k=1}^{9} x_k$ and maximum: $\mathrm{f}(\boldsymbol{x}) = \max_{k=1,\ldots,9} x_k$.

3.1.7 2D Image Processing

The image processing is a discipline of computer vision dealing with transformations of image data into image data. The design of an image transformation follows special aims, e.g. image segmentation, image enhancement, or image restoration.

The *image enhancement* is the processing of images to improve their appearance to human viewers or to enhance other image processing modules' performance. The objective of image enhancement is dependent on the application context and criteria for enhancement is often subjective or too complex to be easily converted to useful objective measures. Enhancement tasks are typical for the analysis of microscope images in medicine or biology, for material inspection, or for remote sensing.

Image enhancement is closely related to image restoration but they are in principle quite different things. In image enhancement there is no assumption on a known degradation, the objective is just to make the processed image better in some sense than the unprocessed image.

Image enhancement is used in all image processing applications, it includes

- contrast manipulation,
- noise reduction (noise suppression, image de-noising),
- image smoothing,
- edge sharpening,
- pseudocoloring.

Image Processing Techniques

An *image operator*, or *transform*, is a function which takes an image for its input and produces an image for its output. The domain of a transform operator is often called the spatial or space domain. The range of the operator is often called the transform domain.

A *neighborhood operator* is an image operator in which the output image value at each pixel position depends only on the input image values in a neighborhood containing the corresponding input pixel position.

A *point operator* is an image operator in which the output image value at each pixel position depends only on the input image value at the corresponding pixel position.

Thresholding is an image point operation, which produces a binary image from a grayscale image. Level slicing or density slicing is a point operation, which employs two thresholds and produces a binary image.

Contrast stretching refers to any monotonically increasing point operator whose effect is to increase or enhance the visibility of an images detail.

An *edge operator* is a neighborhood operation which determines the extent to which each pixels neighborhood can be partitioned by a simple arc passing through the pixel where pixels in the neighborhood on one side of the arc have one predominant value and pixels in the neighborhood on the other side of the arc have a different predominant value. There are four classes of edge operators: gradient operators, Laplacian operators, zero-crossing operators and morphologic edge operators.

Morphology is the study of form and shape. On this basis digital morphology means topology or structure of objects on a given image.

3.2 De-Noising of 2D Image

3.2.1 Noise

DEFINITION 3.10. ***Image noise*** *is defined as any unwanted disturbance in image data.*

Remark 1: The noise is produced from the signal digitization, data recording and data transmission. Image noise may also result in "holes" in the image data.

Remark 2: The noise can generally be grouped into two classes:

- independent noise,
- noise which is dependent on the image data.

Image independent noise can often be described by an additive noise model, where the recorded image is the sum of the true image and the noise.

In many cases, additive noise is uniformly distributed over the frequency domain (*white noise*), whereas an image contains mostly low frequency information. Hence, the noise is dominant for high frequencies.

In the second case of *data-dependent noise*, it is possible to model noise with a multiplicative, or nonlinear, model. These models are mathematically more complicated; hence, if possible, the noise is assumed to be data independent.

Another common form of noise is *data drop-out noise*. The corrupted pixels are set to the maximum value (which looks like snow in the image—*impulse noise*) In some cases, single pixels are set alternatively to zero or to the maximum value, giving the image a 'salt and pepper' like appearance (*salt and pepper noise*).

3.2.2 Ideal, Real and Artificial Images

DEFINITION 3.11. *The model of 2D signal is called an* ***ideal image****.*

Remark 1: The ideal image can be approximated by any real image with minimum noise level.

DEFINITION 3.12. *Any image obtained by measurement is called a* ***real image****.*

DEFINITION 3.13. *Any ideal image modified by any noise signal is called an* ***artificial image****.*

Remark 1: The additive operation is recommended for this modification.

3.2.3 Image De-Noising

DEFINITION 3.14. *The special case of image enhancement, aim of which is the noise reduction and image structure saving, is called the* ***image de-noising****.*

Remark 1: The noise reduction and image structure saving are contradictory aims.

3.2.4 Filter

DEFINITION 3.15. *Any neighborhood operator (image transformation), which decrease the noise level, is called a* ***filter****.*

Remark 1: Many filters can be described as local processing with given mask and given local processing function.

Remark 2: A set of filters can be classified using several criteria, for example:

- by local processing function (linear × nonlinear filters),
- by filter impulse response (FIR × IIR filters),
- by filter frequency response (high-pass × low-pass filters).

RESOURCES: [13]

3.3 Local FLF Processing for Image De-Noising

3.3.1 Individual FLF Processing

DEFINITION 3.16. *Any local processing, where local processing function is FLF, is called a* ***FLF processing****.*

Thus, FLF processing is any FLF application to the weighted list formed on given image using given mask. A scheme of FLF processing is depicted in Fig. 3.9.

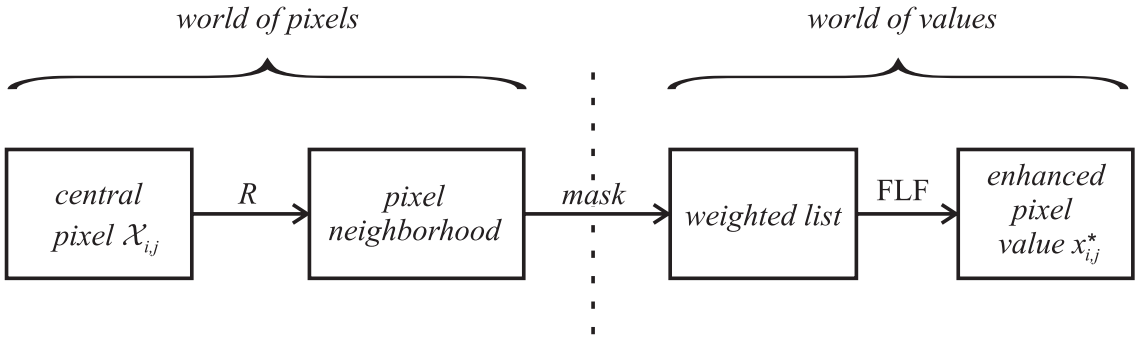


Figure 3.9: Local FLF processing scheme

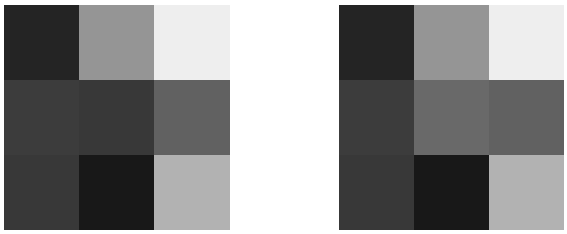
EXAMPLE:
A local processing can be demonstrated for gray image and following mask and FLF:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \mathrm{f}(\boldsymbol{x}) = \frac{1}{8}\sum_{i=1}^{8} x_i.$$

Then $R = 1$ and mask capacity $k = 8$. The neighborhood of $\mathcal{X}_{37,36}$ is depicted in the Fig. 3.10, original value of central pixel is $x_{37,36} = 0.23$, weighted list is

$$\boldsymbol{w} = (0.15, 0.6, 0.95, 0.24, 0.39, 0.23, 0.1, 0.7)$$

and the enhanced central pixel value is $x^*_{37,36} = \mathrm{f}(\boldsymbol{w}) = 0.42$.



(a) (b)

Figure 3.10: Pixel neighborhood (a) before and (b) after local FLF processing

3.3.2 Hierarchical FLF Processing

We can define the local FLF processing as a hierarchical structure of data processing. Resulting structures are called FLF networks. They consist of independent layers with interconnections. The signals from pixel neighborhood come into the first input layer. The enhanced pixel is produced by output node in output layer. It is necessary to use one hidden layer at least for advanced pixel processing. The three-layered network is able to realize disjunctive normal form or conjunctive normal form of logical expressions. This property is helpful for median realization for example. The four-layered network contains two hidden layers. This structure is useful for preprocessing realization, sub-concept forming and advanced pixel processing.

Realizing FLF networks, ŁA_{sqrt} can be used for node processing. The useful FLFs and FLF networks are described in the chapters 6–7.

Chapter 4

Quality of Image Processing

4.1 Image Similarity

4.1.1 Image Difference

DEFINITION 4.1. *Let X and Y are 2D images of the same type where X is the ideal image and Y is noised one (it is identified as subject of image processing). Then $D = Y - X$ is called an* ***image difference****.*

The aim of image processing is to construct a new image Y^* such that $Y^* - X$ is approximately equal to zero.

4.1.2 Minkowski Distance of Images

DEFINITION 4.2. *Let $r, c \in \mathbf{N}$. Let X, Y be the images of size (r, c) and $p \in [1, \infty) \subset \mathbf{R}$. Then*

$$\mathrm{d}_p(X,Y) = \left(\sum_{i=1}^{r} \sum_{j=1}^{c} |y_{ij} - x_{ij}|^p \right)^{\frac{1}{p}}$$

is called a ***Minkowski distance of images*** *X, Y.*

Remark 1: The standard measures of image processing quality are derived from Minkowski distance of images:

- *mean of absolute error* (MAE):

$$\mathrm{MAE} = \frac{\mathrm{d}_1(X,Y)}{r \cdot c} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} |y_{ij} - x_{ij}|}{r \cdot c}.$$

  MATLAB source code

```
function y = critMAE(X,Y)
y = mean(abs(X(:)-Y(:)));
```

- *mean square of error* (MSE):

$$\mathrm{MSE} = \frac{\mathrm{d}_2(X,Y)}{\sqrt{r \cdot c}} = \sqrt{\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} (y_{ij} - x_{ij})^2}{r \cdot c}}.$$

MATLAB source code

```
function y = critMSE(X,Y)
y = sqrt(mean((X(:)-Y(:)).^2));
```

Remark 2: The *maximum of absolute error* (MAX) is based on d_∞ distance function:

$$\mathrm{MAX} = \lim_{p\to\infty} \mathrm{d}_p(X,Y) = \mathrm{d}_\infty(X,Y) = \max_{\substack{i=1,\ldots,r\\ j=1,\ldots,c}} |y_{ij} - x_{ij}|.$$

MATLAB source code

```
function y = critMAX(X,Y)
y = max(abs(X(:)-Y(:)));
```

4.1.3 Signal to Noise Ratio

The signal to noise ratio (SNR) criterion corresponds with digital filtering standard. It is defined by a formula

$$\mathrm{SNR} = 10 \cdot \log \frac{\mathrm{Var}(X)}{\mathrm{Var}(Y-X)}$$

or by an alternative formula

$$\mathrm{SNR}^* = 10 \cdot \log \frac{\mathrm{Var}(X)}{\mathbf{E}(Y-X)^2}$$

where $\mathrm{Var}(A) = \mathbf{E}\{(A - \mathbf{E}A)^2\}$, where $\mathbf{E}$ is a symbol of average value.

MATLAB source code

```
function [y]=critSNR(X,Y)
% critSNR - Signal/noise ratio - SNR for 2D image pair
% [y]=critSNR(X,Y);
% y  ... signal/noise ratio [dB]
% X  ... original image matrix (m,n) with minimum noise level
% Y  ... corrupted image matrix (m,n)
Y=Y-X;
X=X-mean(mean(X));
Y=Y-mean(mean(Y));
a=sum(sum(X.^2));
b=sum(sum(Y.^2));
if a==0
   y=-inf;
else
   if b==0
      y=inf;
   else
      y=10*log10(a/b);
   end
end
```

4.1.4 Other Quality Measure

Various criteria can be used as measure of de-noising quality. Four of them were described above: the MAE criterion corresponds with two-sided exponential noise, the MSE criterion agrees with Gaussian noise, the MAX criterion agrees with limited noise while the SNR criterion corresponds with digital filtering standard. Fifth criterion is a *median of absolute error* (MED). This criterion corresponds with robust variance estimation and it is defined as

$$\mathrm{MED} = \underset{\substack{i=1,\ldots,r \\ j=1,\ldots,c}}{\mathrm{median}} |y_{ij} - x_{ij}|.$$

MATLAB source code

```
function y = critMED(X,Y)
y = median(abs(X(:)-Y(:)));
```

4.2 Multicriterial Approach

Supposing an existence of single image in both noised and ideal form, the noised form of image can be passed through any filter and the quality of de-noising can be measured by various criteria. The SNR or SNR*, MSE and MAE can be recommended. Then the quality of given filter is a vector of selected criteria values. The results of various filters can be collected to quality matrix. There is a possibility to find optimum filter using multicriteria decision making (MDM) technique.

Let m be a number of filters, n_1 be a number of processing noised images ($n_1 = 1$ in case of single image de-noising) and n_2 be a number of quality measures. Then $n = n_1 \cdot n_2$ is a total number of collected properties of objects. The matrix of x^*_{ij} consists of m rows and n columns. The main problem of optimum filtering is a selection of training images representing the real world and a choice of quality measure. The optimum filter should be designed to be efficient on any training image with respect to any given quality measure. This is a kind of multicriteria decision task.

4.2.1 Pareto Optimality

DEFINITION 4.3. *Let $m \in \mathbf{N}$ be a number of objects. Let $n \in \mathbf{N}$ be a number of properties. Let $x_{ij} \in \mathbf{R}$ be a j-th property of i-th object for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Let $s_j \in \{-1, +1\}$ be a signum of j-th property for $j = 1, \ldots, n$.*

*Then the **ideal alternative** is defined as a fictive object I with properties*

$$I_j = \max_{i=1,\ldots,m} s_j x_{ij} \text{ for } j = 1, \ldots, n,$$

*while the **base alternative** is defined as a fictive object B with properties*

$$B_j = \min_{i=1,\ldots,m} s_j x_{ij} \text{ for } j = 1, \ldots, n.$$

*The **object normalization** is defined as a mapping $\mathcal{N} : \mathbf{R}^{m \times n} \to [0,1]^{m \times n}$ using*

$$x^*_{ij} = \begin{cases} \frac{x_{ij} - B_j}{I_j - B_j} & \text{for } I_j > B_j, \\ 0 & \text{for } I_j = B_j. \end{cases}$$

DEFINITION 4.4. *Let $n \in \mathbf{N}$ be a number of properties. Let $I = \{1, 2, \ldots, n\}$. The i-th object* ***is dominated*** *by j-th one when $(\forall k \in I)\ x_{ik}^* \leq x_{jk}^*$ and $(\exists u \in I)\ x_{iu}^* < x_{ju}^*$.*

Any object which is not dominated by another one is called ***Pareto optimum object****.*

Remark 1: The main advantage of Pareto optimum object is its independence on metric and weights.

Remark 2: Any object which is not Pareto optimum one, is worse then another Pareto optimum one.

RESOURCES: [23], [27]

4.2.2 AIA Technique

Special MDM technique minimizes the **a**bsolute value of distance from **i**deal **a**lternative (AIA technique).

DEFINITION 4.5. *Let $m \in \mathbf{N}$ be a number of objects. Let $n \in \mathbf{N}$ be a number of properties. Let $\lambda \geq 1$ be a real parameter. Then the* ***AIA compromise quality*** *is defined as*

$$q_i^{(\lambda)} = \sum_{j=1}^{n} |1 - x_{ij}^*|^{\lambda}.$$

The traditional AIA technique is AIA with the value $\lambda = 1$.

DEFINITION 4.6. *Let $m \in \mathbf{N}$ be a number of objects. Let $n \in \mathbf{N}$ be a number of properties. Let $\lambda \geq 1$ be a real parameter. Let $O_k = (x_{k1}, \ldots, x_{kn})$ be k-th object for $k = 1, \ldots, n$. Then the* ***set of AIA optimum objects*** *is defined as*

$$\mathcal{S}_{(\lambda)} = \{O_k \mid q_k^{(\lambda)} = \min_{i=1,\ldots,m} q_i^{(\lambda)}\}.$$

The parameter $\lambda \in [1; \infty)$ can cause small changes in the set of AIA optimum objects.

Chapter 5

Traditional Non-FLF Filters for Image De-Noising

The image processing is a traditional application of various mathematical methods. This chapter describes several non-FLFs typically used for image de-noising.

5.1 Mean Filter

DEFINITION 5.1. *Let $\mathcal{P}$ be an image of size (r, c). Let $\mathbf{M}$ be a box mask with a radius $R \in \mathbf{N}$, $n = (2R+1)^2$ and $\boldsymbol{v} \in \mathbf{L}^n$ be a weighted list formed using $\mathbf{M}$. Let*

$$\mathrm{AVERAGE}(\boldsymbol{v}) = \frac{1}{n} \cdot \sum_{k=1}^{n} v_k.$$

Then a local processing, which use the mask $\mathbf{M}$ and the local processing function AVERAGE*, is called a **mean filter**.*

A mean filter (also called a *box filter*) is a simple smoothing operator performing an unweighted arithmetic averaging. This filter works fairly well for Gaussian noise, is poor for 'salt and pepper' noise, and has the side-effect of reducing high frequencies. It is basically a low-pass filter. The larger radius R, the more pronounced its effects.

There are two main problems with mean filtering, which are:

(a) A single pixel with a very unrepresentative value can significantly affect the mean value of all the pixels in its neighborhood.

(b) When the filter neighborhood straddles an edge, the filter will interpolate new values for pixels on the edge and so will blur that edge. This may be a problem if sharp edges are required in the output.

Remark 1: The mean filter can not be realized in $\mathrm{LA}_{\mathrm{sqrt}}$ because of only a division into power of two is allowed.

RESOURCES: [9], [19], [7]

5.2 Alpha Trimmed Mean Filter

The median filter (section 6.1.2) preserves edges but wipes out fine details. The mean filter blurs too much. These limitations motivated the development of the adaptive trimmed mean filter.

DEFINITION 5.2. *Let* $\mathbf{M}$ *be a box mask with radius* R *and* $r = 2R + 1$. *Let* $d \in \mathbf{N}_0$ *be an even number. Let* $\mathrm{f} : \mathbf{L}^{r^2} \to \mathbf{L}$ *such as*

$$\mathrm{f}(\boldsymbol{x}) = \frac{1}{r^2 - d} \sum_{i=1+d/2}^{r^2-d/2} x_{(i)},$$

where $x_{(i)}$ *denotes i-th item in ordered list of* x_i *values. Then a local processing, which use box mask* $\mathbf{M}$ *and local processing function* f, *is called* ***alpha trimmed mean filter****.*

Remark 1: The alpha trimmed mean filter is the average of the pixel values within the local neighborhood, but with some of the endpoint-ranked values excluded.

Remark 2: This filter is suitable for combination of different types of noise.

Remark 3: The mean filter is obtained for $d = 0$.

Remark 4: The value $d = r^2 - 1$ produces a median filter.

Remark 5: The alpha trimmed mean filter is not realizable in $\mathrm{LA}_{\mathrm{sqrt}}$ in general.

RESOURCES: [22]

5.3 Integer Weighted Average Filter

The disadvantage of smoothing linear filters is attenuating, besides the noise, also the high spatial frequencies. This leads to a loss of details and to unsharp edges. The aim of many nonlinear filtering approaches is to reduce noise without blurring edges. This is also the aim of averaging in a selected neighborhood.

DEFINITION 5.3. *Let* $\mathbf{M}$ *be any mask with radius* R *and capacity* k. *Let* $\mathrm{f} : \mathbf{L}^k \to \mathbf{L}$ *such as*

$$\mathrm{f}(\boldsymbol{v}) = \frac{1}{k} \sum_{i=1}^{k} v_i.$$

Then a local processing, which use local processing function f *for weighted list* $\boldsymbol{v}$ *formed from the mask* $\mathbf{M}$ *and pixel neighborhood, is called* ***integer weighted average filter****.*

Remark 1: The weighted average filter is linear filter with finite impulse response—it is also called the FIR filter.

Remark 2: This filter is not realizable in $\mathrm{LA}_{\mathrm{sqrt}}$ in general, but it is realizable for mask capacity $k = 2^N$ where $N \in \mathbf{N}$ (section 6.3).

5.4 Gaussian Filter

Gaussian filter is a linear low pass filter. It is equivalent to filtering with a mask of radius R, whose weights are computed according to a Gaussian function ($r = 2R + 1$):

$$m_{i,j} = \frac{1}{2\pi\sigma^2} \cdot \mathrm{e}^{-\frac{(i-R-1)^2+(j-R-1)^2}{2\sigma^2}} \quad (i, j = 1, 2, \ldots, r).$$

The Gaussian outputs a 'weighted average' of each pixel's neighborhood, with the average weighted more towards the value of the central pixels. This is in contrast to the mean filter's uniformly weighted average. Because of this, a Gaussian provides gentler smoothing and preserves edges better than a similarly sized mean filter.

Gaussian smoothing is very effective for removing Gaussian noise. The effect of Gaussian smoothing is to blur an image, in a similar fashion to the mean filter. The degree of smoothing is controlled by σ (standard deviation of the Gaussian; large σ for more intensive smoothing). The weights give higher significance to pixels near the edge (reduces edge blurring). Gaussian filter also turns out to be very similar to the optimal smoothing filter for edge detection under the criteria used to derive the Canny edge detector.

Gaussian filter not only has utility in engineering applications. It is also attracting attention from computational biologists because it has been attributed with some amount of biological plausibility, e.g. some cells in the visual pathways of the brain often have an approximately Gaussian response.

Remark 1: Gaussian filter can not be realized in $\mathrm{LA}_{\mathrm{sqrt}}$ because of the mask weights are not dyadic numbers (i.e. $m_{i,j} \neq \frac{x}{2^y}$, $x, y \in \mathbf{N_0}$).

RESOURCES: [8], [7]

5.5 Wiener Filter

Wiener filter is the MSE-optimal stationary linear low-pass filter for images degraded by additive noise and blurring. Calculation of Wiener filter requires the assumption that the signal and noise processes are second-order stationary (in the random process sense).

Wiener filters are usually applied in the frequency domain. Given a degraded image X, one takes the Discrete Fourier Transform (DFT) to obtain X^*. The original image spectrum is estimated by taking the product of X^* with the Wiener filter W^*:

$$Y^* = W^* \cdot X^*$$

The inverse DFT is then used to obtain the image estimate Y from its spectrum.

For the case of additive white noise and no blurring, the Wiener filter simplifies to:

$$W^* = \frac{P_s(u, v)}{P_s(u, v) + \sigma_n^2}$$

where σ_n^2 is the noise variance and $P_s(u, v)$ is power spectrum of the signal process, obtained by taking the Fourier transform of the signal autocorrelation.

Wiener filters are unable to reconstruct frequency components which have been degraded by noise. They can only suppress them.

Remark 1: Wiener filters are not realizable in $\mathrm{LA}_{\mathrm{sqrt}}$ because of both DFT and the division are not FLFs.

RESOURCES: [8]

5.6 IIR Filter

IIR filter is a recursive linear filter with infinite impulse response Recursive filters are filters, which use the result of the processing for input of the next processing again. The disadvantage of this calculation is the loss of a linear phase.

These filters are not stable in all combinations and may oscillate in the output.

The IIR filters are typically implemented using DFT and IDFT (inverse discrete Fourier transform).

Remark 1: IIR filter advantages

- infinite impulse response, no oscillation at the edges
- low order although meeting the tolerance scheme
- short signal runtime and low calculation time for coefficients

Remark 2: IIR filter disadvantages

- not always stable because of recursive character, stability check required
- no linear signal runtime, dependent on frequency of the signal
- output signal may be falsified

Remark 3: Butterworth and Metz filters are very popular IIR filters used in biomedical applications.

Remark 4: IIR filters are not realizable in ŁA_{sqrt}.

RESOURCES: [13]

Chapter 6

Traditional FLF Filters for Image De-Noising

This chapter describes several filters useful for image de-noising, which are realizable in $\mathrm{LA}_{\mathrm{sqrt}}$.

6.1 Statistics Functions as FLFs

6.1.1 Lists

The intensities of pixels from neighborhood are collected in various lists. Each term is defined first and then its algorithm is realized as MATLAB function.

DEFINITION 6.1. *Let $n \in \mathbf{N}$ and $\boldsymbol{x} \in \mathbf{L}^n$. Then n-tuple $L = (x_1, \ldots, x_n)$ is called a **list**. The list can be also denoted as $L = \bigdiamond_{k=1}^{n} x_k$.*

MATLAB source code

```
function L=list(x)
L=x;
```

DEFINITION 6.2. *Let $n, k \in \mathbf{N}$, $k \leq n$ and $\boldsymbol{x} \in \mathbf{L}^n$. Let $L = (x_1, \ldots, x_n)$ be a list. Then $\mathrm{SORT}(L) = (x_{(1)}, \ldots, x_{(n)})$ where $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$ is called a **sorted list** and $x_{(k)}$ is called the **k-th value** of L.*

MATLAB source codes

```
function L=sort_list(x)
L=sort(x);

function y=k_value(x,k)
if k>length(x) | k<1
  error('k-th value does not exist!');
end
x=sort(x);
y=x(k);
```

THEOREM 6.1. *Let $k, n \in \mathbf{N}$, $k \leq n$ and $\boldsymbol{x} \in \mathbf{L}^n$. Let $\psi_k : \mathbf{L}^n \to \mathbf{L}$ such that $\psi_k(\boldsymbol{x}) = x_{(k)}$ where $x_{(k)}$ is the k-th value from sorted list $\mathrm{SORT}(L) = (x_{(1)}, \ldots, x_{(n)})$. Then ψ_k is FLF.*

Proof. Let $k, n \in \mathbf{N}$, $k \leq n$ and $\boldsymbol{x} \in \mathbf{L}^n$. Let $\mathrm{f} : \mathbf{L}^n \to \mathbf{L}$ and

$$\mathrm{f}(\boldsymbol{x}) = \bigwedge_{i=1}^{\binom{n}{k}} \left(\bigvee_{j \in S_{i,k,n}} x_j \right),$$

where $S_{i,k,n}$ is i-th subset of $\{1, \ldots, n\}$ of size k. There are $\binom{n}{k}$ such subsets. Clearly, $\mathrm{f}(\boldsymbol{x})$ is FLF. Now, we prove that $\mathrm{f}(\boldsymbol{x})$ is k-th value from sorted list $(x_{(1)}, \ldots, x_{(n)})$:
First, let $\varphi_i = \bigvee_{j \in S_{i,k,n}} x_j$ for $i = 1, \ldots, \binom{n}{k}$. Each subset $S_{i,k,n}$ contains k elements. The number of all elements, which are less than $x_{(k)}$, is at most $k-1$. Consequently $\varphi_i \geq x_{(k)}$ and $\mathrm{f}(\boldsymbol{x}) = \bigwedge_{i=1}^{\binom{n}{k}} \varphi_i \geq \bigwedge_{i=1}^{\binom{n}{k}} x_{(k)} = x_{(k)}$, i.e. $\mathrm{f}(\boldsymbol{x}) \geq x_{(k)}$.
Second, suppose that $\mathrm{f}(\boldsymbol{x}) > x_{(k)}$, i.e. $\varphi_i = \bigvee_{j \in S_{i,k,n}} x_j > x_{(k)}$ for all $i \in \{1, \ldots, \binom{n}{k}\}$. But there is a subset $S_{1,k,n} = \{x_{(1)}, \ldots, x_{(k)}\}$ with $\varphi_i = \varphi_1 = x_{(k)}$. This contradiction causes $\mathrm{f}(\boldsymbol{x}) \not> x_{(k)}$.
Thus $\mathrm{f}(\boldsymbol{x}) = x_{(k)} = \psi_k(\boldsymbol{x})$. □

Remark 1: This theorem is useful for extended data processing (construction of the filters based on sorted value lists).

Several pixels from neighborhood can be more significant than the others. Therefore it is useful to introduce a term "weighted list".

DEFINITION 6.3. *Let $n \in \mathbf{N}$ and $\boldsymbol{x} \in \mathbf{L}^n$. Let $\boldsymbol{w} \in \mathbf{N}_0^n$ be a vector of weights. Then a **weighted list** is defined as $L = \mathop{\Diamond}_{k=1}^{n} w_k x_k$ where $w_k x_k = (\underbrace{x_k, x_k, \ldots, x_k}_{w_k}) = \mathop{\Diamond}_{j=1}^{w_k} x_k$.*

EXAMPLE: Let $\boldsymbol{w} = (1, 0, 3, 2, 1)$ and $\boldsymbol{x} = (a, b, c, d, e)$. Then the weighted list $L = (a, c, c, c, d, d, e)$ is obtained.

MATLAB source code

```
function L=w_list(x,w)
if length(x)~=length(w)
  error('vectors must have the same lengths!')
end
L=zeros(1,sum(w)); % prepare
k=1;
for i=1:length(w)
  for j=1:w(i)
    L(k)=x(i);
    k = k+1;
  end
end
```

THEOREM 6.2. *Any element of weighted list is FLF of original list.*

Proof. Trivial application of FLE and FLF definitions (1.13, 1.14). □

Remark 1: Weighted lists are used in local FLF processing as described in subsection 3.3.1.

The alternative data processing techniques prefer large lists of compromise values. The Walsh list is one of them.

DEFINITION 6.4. *Let $n \in \mathbf{N}$ and $\boldsymbol{x} \in \mathbf{L}^n$. Then the* ***Walsh list*** *[11] is defined as*

$$\mathrm{W}(\boldsymbol{x}) = \underset{i \le j}{\Diamond} \left(\frac{x_i + x_j}{2} \right).$$

MATLAB source code

```
function WL=Walsh_list(x)
d = length(x);
WL=zeros(1,(d*(d+1))/2); % prepare
k=1;
for i=1:d
  for j=1:i
    WL(k) = (x(i)+x(j))/2;
    k = k+1;
  end
end
```

LEMMA 6.3. *Let $a, b \in \mathbf{L}$. Then the average value of a, b is FLF.*

Proof. Let $a, b \in \mathbf{L}$. Let $\mathrm{f} : \mathbf{L}^2 \to \mathbf{L}$ and $\mathrm{f}(a,b) = \mathrm{sqrt}(a) \otimes \mathrm{sqrt}(b)$.
Clearly, f is FLF according to the definition 1.14 and

$$\begin{aligned} \mathrm{f}(a,b) &= \mathrm{sqrt}(a) \otimes \mathrm{sqrt}(b) = \max\left(\frac{1+a}{2} + \frac{1+b}{2} - 1, 0 \right) = \max\left(\frac{a+b}{2}, 0 \right) = \\ &= \frac{a+b}{2}. \end{aligned}$$

□

THEOREM 6.4. *Any element of Walsh list is FLF of original list.*

Proof. Any element of Walsh list is an average of two inputs. The average of two values is FLF (lemma 6.3). Then the elements of Walsh list are FLFs. □

6.1.2 Median Filter

DEFINITION 6.5. *Let $L = \overset{n}{\underset{k=1}{\Diamond}} x_k$ be a list, $x_k \in \mathbf{L}$. Then the* ***median*** *of list L is defined as*

$$\mathrm{M}(\boldsymbol{x}) = \underset{i=1,\ldots,n}{\mathrm{median}}\, x_i = \frac{x_{(\lfloor \frac{n+1}{2} \rfloor)} + x_{(\lceil \frac{n+1}{2} \rceil)}}{2},$$

where $\lfloor a \rfloor = \max\{n \in \mathbf{N}; n \le a\}$ (the nearest integer less than or equal to a),
$\lceil a \rceil = \min\{n \in \mathbf{N}; n \ge a\}$ (the nearest integer greater than or equal to a) and
$x_{(k)}$ is the k–th value of L.

The local processing, which use any mask $\mathbf{M}$ with radius R and median as local processing function, is called a ***median filter****.*

MATLAB source code

```
function m=median(x)
n = length(x);
x = sort(x);
m = (x(floor((n+1)/2)) + x(ceil((n+1)/2)))/2;
```

The median filter is normally used to reduce noise in an image, somewhat like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image.

By calculating the median value of a neighborhood rather than the mean filter, the median filter has two main advantages over the mean filter:

- The median is a more robust average than the mean and so a single very unrepresentative pixel in a neighborhood will not affect the median value significantly.
- Since the median value must actually be the value of one of the pixels in the neighborhood, the median filter does not create new unrealistic pixel values when the filter straddles an edge. For this reason the median filter is much better at preserving sharp edges than the mean filter.

In general, the median filter allows a great deal of high spatial frequency detail to pass while remaining very effective at removing noise on images where less than half of the pixels in a smoothing neighborhood have been effected. (As a consequence of this, median filtering can be less effective at removing noise from images corrupted with Gaussian noise.)

One of the major problems with the median filter is that it is relatively expensive and complex to compute. To find the median it is necessary to sort all the values in the neighborhood into numerical order and this is relatively slow, even with fast sorting algorithms such as quicksort.

6.1.3 Another Order Statistics as FLF

DEFINITION 6.6. *Let* $L = \underset{k=1}{\overset{n}{\Diamond}} w_k x_k$ *be a weighted list,* $x_k \in \mathbf{L}$ *and* $w_k \in \mathbf{N}_0$*. Then the* ***weighted median*** *is defined as median of weighted list.*

MATLAB source code

```
function wm=w_median(x,w)
if length(x)~=length(w)
  error('vectors must have the same lengths!')
end
wm = median(w_list(x,w));
```

DEFINITION 6.7. *Let* $L = \underset{k=1}{\overset{n}{\Diamond}} x_k$ *be a list,* $x_k \in \mathbf{L}$*. Then the* ***quasi median*** *of list* L *is defined as*

$$\mathrm{Q}_k(\boldsymbol{x}) = \frac{x_{(\lfloor \frac{n+1}{2} \rfloor - k)} + x_{(\lceil \frac{n+1}{2} \rceil + k)}}{2},$$

where $k \in \mathbf{N}$ *and* $k < \lfloor \frac{n+1}{2} \rfloor$*.*

MATLAB source code

```
function qm=quasi_median(x,q)
n = length(x);
x = sort(x);
qm = (x(floor((n+1)/2-q)) + x(ceil((n+1)/2+q)))/2;
```

DEFINITION 6.8. *The median of Walsh list is called* ***Hodges-Lehmann median****:*

$$\mathrm{HL}(\boldsymbol{x}) = \mathrm{median}\left\{\frac{x_i + x_j}{2}, 1 \le i \le j < n\right\}.$$

MATLAB source code

```
function HLm=HL_median(x)
HLm = median(Walsh_list(x));
```

DEFINITION 6.9. *Let* $L = \overset{n}{\underset{k=1}{\Diamond}} x_k$ *be a list,* $x_k \in \mathbf{L}$*. Then the* ***Best Easy Systematic Estimation*** *(BES) is defined as*

$$\mathrm{BES}(\boldsymbol{x}) = \frac{1}{4} \cdot \left(x_{(\lceil \frac{n}{4} \rceil)} + x_{(\lfloor \frac{n+1}{2} \rfloor)} + x_{(\lceil \frac{n+1}{2} \rceil)} + x_{(\lfloor \frac{3n+4}{4} \rfloor)}\right).$$

MATLAB source code

```
function bes=BES(x)
n = length(x);
x = sort(x);
bes = (x(ceil(n/4)) + x(floor((n+1)/2)) + x(ceil((n+1)/2))...
    + x(floor((3*n+4)/4))) / 4;
```

DEFINITION 6.10. *The BES estimation of Walsh list is called* ***Walsh-BES estimation****.*

THEOREM 6.5. *The medians, quasi medians, Hodges-Lehmann median, BES and Walsh-BES are FLFs.*

Proof. Trivial application of previous theorems. □

RESOURCES: [11], [23]

6.2 Morphological Operators as FLFs

The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set theory. The operators are particularly useful for the analysis of binary or gray images and common usages include edge detection, noise removal, image enhancement and image segmentation.

Virtually all other mathematical morphology operators can be defined in terms of combinations of erosion and dilation.

6.2.1 Erosion

Erosion is one of the two basic operators in the area of mathematical morphology, the other being dilation. It is typically applied to binary images, but there are versions that work on grayscale images. The basic effect of the operator on a binary image is to erode away the boundaries of regions of foreground pixels. Thus areas of foreground pixels shrink in size, and holes within those areas become larger.

Grayscale erosion will generally darken the image. Bright regions surrounded by dark regions shrink in size, and dark regions surrounded by bright regions grow in size. Small bright spots in images will disappear as they are eroded away down to the surrounding intensity value, and small dark spots will become larger spots. The effect is most marked at places in the image where the intensity changes rapidly, and regions of fairly uniform intensity will be left more or less unchanged except at their edges.

Following definition introduces a grayscale erosion:

DEFINITION 6.11. *Let $\mathcal{P}$ be an image, $\mathbf{M}$ be a mask with radius R and $k \in \mathbf{N}$ be a capacity of $\mathbf{M}$. Let* ERO $: \mathbf{L}^k \to \mathbf{L}$,

$$\mathrm{ERO}(\boldsymbol{x}) = \bigwedge_{i=1}^{k} x_i.$$

Then a local processing using the mask $\mathbf{M}$ and local processing function ERO *is called an **erosion**.*

Remark 1: The mask is called a *structuring element* in a field of mathematical morphology. A typical value of radius is $R = 1$. A larger mask produces a more extreme erosion effect.

Remark 2: The weights in mask greater than one have no effect in any morphological operators.

Remark 3: Erosion can be used to remove an impulse noise ('salt noise') in images.

Remark 4: There is possible to use erosion for edge detection by taking the erosion of an image and then subtracting it away from the original image, thus highlighting just those pixels at the edges of objects that were removed by the erosion.

Remark 5: Erosion is also used as the basis for many other operators in mathematical morphology.

RESOURCES: [13], [8], [10]

6.2.2 Dilation

The second basic operation in mathematical morphology is dilation. It is typically applied to binary images, but there are versions that work on grayscale images.

The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels. Thus areas of foreground pixels grow in size while holes within those regions become smaller. Dilation is the dual of erosion i.e. dilating foreground pixels is equivalent to eroding the background pixels.

Grayscale dilation will generally brighten the image. Bright regions surrounded by dark regions grow in size, and dark regions surrounded by bright regions shrink in size. Small dark spots in images will disappear as they are 'filled in' to the surrounding intensity value. Small bright spots will become larger spots. The effect is most marked at places in the image where the intensity changes rapidly and regions of fairly uniform intensity will be largely unchanged except at their edges.

The definition of grayscale dilation follows.

DEFINITION 6.12. *Let $\mathcal{P}$ be an image, $\mathbf{M}$ be a mask with radius R and k be a capacity of $\mathbf{M}$. Let* $\mathrm{DIL} : \mathbf{L}^k \to \mathbf{L}$,

$$\mathrm{DIL}(\boldsymbol{x}) = \bigvee_{i=1}^{k} x_i.$$

Then a local processing using the mask $\mathbf{M}$ and local processing function DIL *is called a **dilation**.*

Remark 1: There are many specialist uses for dilation. For instance it can be used to fill in small spurious holes ('pepper noise') in images.

Remark 2: Dilation can be used for edge detection by taking the dilation of an image and then subtracting away the original image, thus highlighting just those new pixels at the edges of objects that were added by the dilation.

Remark 3: Dilation is also used as the basis for many other operators in mathematical morphology.

RESOURCES: [13], [8], [10]

6.2.3 Opening

Very simply, an opening is defined as an erosion followed by a dilation using the same structuring element (mask) for both operations.

Opening results in removal of narrow peaks. The initial erosion removes the small details and darkens the image. The following dilation increases the brightness but does not reintroduce the details removed by erosion.

Similarly, opening can be used to remove 'salt noise' in images. The noise has been entirely removed with relatively little degradation of the underlying image. However, if the noise consists of dark points ('pepper noise') no noise has been removed.

Remark 1: Opening is idempotent operator, i.e. consecutive openings with the same element do not change an image.

Remark 2: Opening can be very useful for separating out particularly shaped objects from the background, but it is far from being a universal 2D object recognizer (segmenter).

RESOURCES: [13], [8], [10]

6.2.4 Closing

Closing is opening performed in reverse. It is defined simply as a dilation followed by an erosion using the same structuring element (mask) for both operations.

For binary images, closing is the dual of opening, i.e. closing the background pixels with an individual structuring element is equivalent to opening the foreground pixels with the same element.

Closing is similar in some ways to dilation in that it tends to enlarge the boundaries of foreground (bright) regions in an image (and shrink background color holes in such regions), but it is less destructive of the original boundary shape. The effect of the operator is to preserve background regions that have a similar shape to this structuring element, or that can completely contain the structuring element, while eliminating all other regions of background pixels.

Closing is used to remove dark details ('pepper noise') from an image. The initial dilation removes dark details and makes the image brighter. The erosion that follows darkens the image but does not reintroduce the details removed by dilation.

Remark 1: Closing is idempotent operator, i.e. consecutive closings with the same element do not change an image.

RESOURCES: [13], [8], [10]

6.3 Dyadic FIR Filter

LEMMA 6.6. *The division into two is FLF.*

Proof. Let $a \in \mathbf{L}$. Let $\phi(a) = \neg\text{sqrt}(\neg a)$. Clearly, ϕ is FLF. Then

$$\phi(a) = 1 - \text{sqrt}(1-a) = 1 - \frac{1+1-a}{2} = \frac{a}{2}.$$

Therefore the division into two is FLF. □

LEMMA 6.7. *The division into any power of two is FLF.*

Proof. Let $a \in \mathbf{L}$, $N \in \mathbf{N}$. Let $\text{f}(a,N) = \frac{a}{2^N}$. Then

$$\text{f}(a,N) = \frac{a}{\underbrace{2 \cdot 2 \cdot \ldots \cdot 2}_{N}} = \phi(\phi(\ldots\phi(\phi(a))\ldots))$$

and f is FLF according to definition 1.14 and lemma 6.6. □

DEFINITION 6.13. *Let $x \in \mathbf{L}$, $N \in \mathbf{N}_0$. Then a* ***division into N-th power of two*** *is defined by a rule:*

$$\frac{x}{2^N} = \begin{cases} x & \text{for } N = 0, \\ \phi\left(\frac{x}{2^{N-1}}\right) & \text{for } N \geq 1. \end{cases}$$

THEOREM 6.8. *Let $n \in \mathbf{N}$. Let $\boldsymbol{x} \in \mathbf{L}^n$. Let $N, m_k \in \mathbf{N}_0$ for $k = 1, \ldots, n$. Let $w_k = m_k/2^N$ be dyadic weights for $k = 1, \ldots, n$ and $\sum_{k=1}^{n} w_k \leq 1$. Then any function*

$$\text{f}(\boldsymbol{x}) = \sum_{k=1}^{n} w_k \cdot x_k$$

is a FLF.

Proof. Let $n \in \mathbf{N}$ and $N \in \mathbf{N}_0$. Let $\boldsymbol{x} \in \mathbf{L}^n$ and $\boldsymbol{m} \in \mathbf{N}_0^n$. Let $\boldsymbol{w} = \boldsymbol{m}/2^N$ and $\sum_{k=1}^n w_k \leq 1$. Then

$$w_k \cdot x_k = \frac{m_k}{2^N} \cdot x_k = m_k \odot \frac{x_k}{2^N}$$

and

$$\sum_{k=1}^{n} w_k \cdot x_k \leq \sum_{k=1}^{n} w_k \leq 1.$$

Then the traditional summation can by substituted by the $\oplus$ operator and

$$\mathrm{f}(\boldsymbol{x}) = \sum_{k=1}^{n} w_k \cdot x_k = \bigoplus_{k=1}^{n} \left(m_k \odot \frac{x_k}{2^N} \right)$$

are FLF. □

DEFINITION 6.14. *Let $n \in \mathbf{N}$, $N \in \mathbf{N}_0$, $\boldsymbol{x} \in \mathbf{L}^n$, $\boldsymbol{m} \in \mathbf{N}_0^n$, $\boldsymbol{w} = \boldsymbol{m}/2^N$ and $\sum\limits_{k=1}^{n} w_k \leq 1$. Then the FLF*

$$\mathrm{f}(\boldsymbol{x}) = \sum_{k=1}^{n} m_k/2^N \cdot x_k$$

is called ***dyadic FIR filter****.*

THEOREM 6.9. *Any dyadic FIR filter satisfies the Lipschitz condition with the sensitivity*

$$\lambda \leq \max_{k=1,\ldots,n} w_k.$$

Proof. Let $n \in \mathbf{N}$. Let $\boldsymbol{x} \in \mathbf{L}^n$. Let $N, m_k \in \mathbf{N}_0$ for $k = 1, \ldots, n$. Let $w_k = m_k/2^N$ be dyadic weights for $k = 1, \ldots, n$ and $\sum_{k=1}^n w_k \leq 1$. Then

$$\begin{aligned} |\mathrm{f}(\boldsymbol{x}) - \mathrm{f}(\boldsymbol{y})| &= \left| \sum_{k=1}^{n} w_k \cdot x_k - \sum_{k=1}^{n} w_k \cdot y_k \right| \leq \\ &\leq \sum_{k=1}^{n} w_k \cdot |x_k - y_k| \leq \max_{k=1,\ldots,n} (w_k) \cdot \sum_{k=1}^{n} |x_k - y_k| \end{aligned}$$

and

$$\lambda \leq \max_{k=1,\ldots,n} w_k.$$

□

Remark 1: Applying dyadic FIR filter to ordered list, some L-estimates are obtained in $\mathrm{LA}_{\mathrm{sqrt}}$.

Remark 2: Applying dyadic FIR filter to ordered Walsh list is an inspirational way how to construct new generation of FLF filters.

6.4 Useful FLFs

The summary of useful FLFs is described in Tab. 6.1.

function name	**processing**
Erosion	$y = x_1 \wedge x_2 \wedge \ldots \wedge x_n$
Dilation	$y = x_1 \vee x_2 \vee \ldots \vee x_n$
k-th value	$y = x_{(k)} = \bigwedge_{i=1}^{\binom{n}{k}} \left(\bigvee_{j \in S_{i,k,n}} x_j \right)$
Median	$y = \mathrm{M}\left(\Diamond_{k=1}^{n} x_k \right)$
Weighted median	$y = \mathrm{M}\left(\Diamond_{k=1}^{n} w_k x_k \right)$
Quasi median	$y = \mathrm{Q}_k\left(\Diamond_{i=1}^{n} x_i \right)$, $1 \leq k < \lfloor \frac{n+1}{2} \rfloor$
H-L median	$y = \mathrm{M}\left(\mathrm{W}\left(\Diamond_{k=1}^{n} x_k \right) \right)$
Weighted H-L median	$y = \mathrm{M}\left(\mathrm{W}\left(\Diamond_{k=1}^{n} w_k x_k \right) \right)$
BES	$y = \frac{1}{4} \cdot \left(x_{(\lceil \frac{n}{4} \rceil)} + x_{(\lfloor \frac{n+1}{2} \rfloor)} + x_{(\lceil \frac{n+1}{2} \rceil)} + x_{(\lfloor \frac{3n+4}{4} \rfloor)} \right)$
Mean of two	$y = \mathrm{sqrt}(x_1) \otimes \mathrm{sqrt}(x_2)$
Mean of four	$y = \mathrm{sqrt}(\mathrm{sqrt}(x_1) \otimes \mathrm{sqrt}(x_2)) \otimes \mathrm{sqrt}(\mathrm{sqrt}(x_3) \otimes \mathrm{sqrt}(x_4))$
Dyadic FIR filter	$y = \sum_{k=1}^{n} m_k / 2^N \cdot x_k$, where $\sum_{k=1}^{n} m_k / 2^N \leq 1$

Table 6.1: Useful FLFs for $\boldsymbol{x} \in \mathbf{L}^n$

Chapter 7

FLF Networks for Image De-Noising

Previous section summed up important FLFs. The image processing using the local FLF one can be perceived as a hierarchical process without loops. Its representation by oriented acyclic graphs is recommended. A resulting FLF structure can be called *fuzzy logic function network*. It consists of independent layers with interconnections. The signals from pixel neighborhood come into the first input layer. The enhanced pixel signal is produced by output node in output layer. It is necessary to use one hidden layer at least for advanced signal processing.

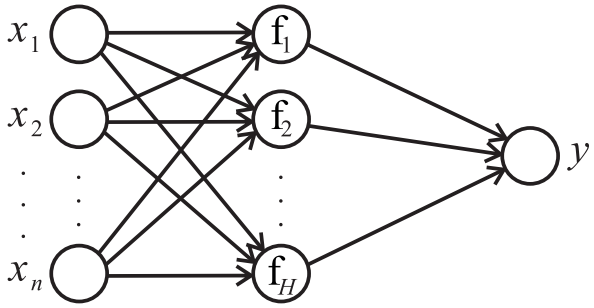


Figure 7.1: Three-layered FLF network

The three-layered network is depicted in Fig. 7.1. It contains one hidden layer only. This kind of network is able to realize disjunctive normal form or conjunctive normal form of logical expressions. It is helpful for median realization or for creating the final concept from sub-concepts.

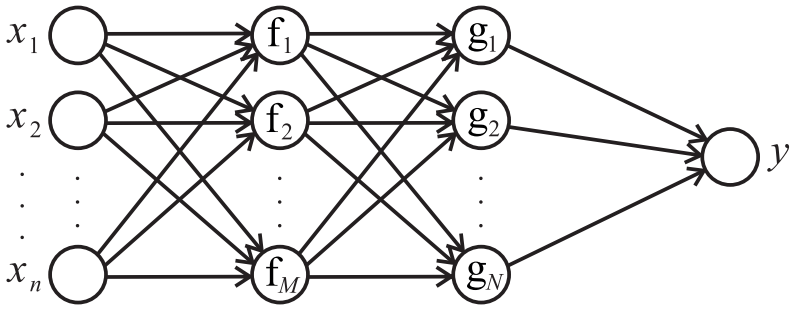


Figure 7.2: Four-layered FLF network

The second useful structure consists of four layers where two of them are hidden ones. This structure is useful for preprocessing, sub-concept forming and advanced data processing. It is depicted in Fig. 7.2.

7.1 Modus Ponens Fuzzy Network

The modus ponens rule as the basic principle of logic motivates a four-layered fuzzy network architecture. A resulting structure is called a ***Modus Ponens Fuzzy Network*** (MPFN) [15]. The first MPFN layer contains n input nodes. The second layer consist of H hidden FLF nodes which make the FLF preprocessing with constrained sensitivity. The third layer realizes the modus ponens law using $2m$ hidden nodes and learnable weights. The fourth layer with m output nodes produces the compromise solution of given task. The MPFN structure is described in the Fig. 7.3.

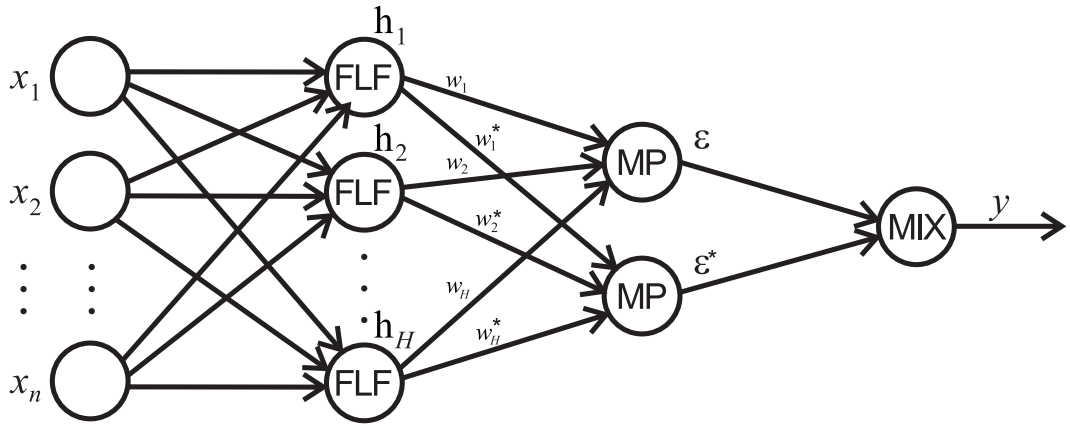


Figure 7.3: Modus ponens fuzzy network for $m = 1$

MPFN Learning Rule

The MPFN is able to learn from the pattern set.

DEFINITION 7.1. *Let $m, n, M \in \mathbf{N}$, $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{in}) \in \mathbf{L}^n$, $\boldsymbol{y}'_i = (y'_{i1}, \ldots, y'_{im}) \in \mathbf{L}^m$ for $i = 1, \ldots, M$. Then the couple $(\boldsymbol{x}_i, \boldsymbol{y}'_i)$ is called a **pattern** and the set*

$$\mathscr{P} = \{(\boldsymbol{x}_i, \boldsymbol{y}'_i),\ i = 1, \ldots, M\}$$

*is called a **pattern set** ($\boldsymbol{y}'_i$ is ideal output vector for given $\boldsymbol{x}_i$).*

*Let $H \in \mathbf{N}$ and $\mathscr{H} = \{\mathrm{h}_1, \ldots, \mathrm{h}_H\}$ be a set of FLFs and $w_{ij}, w^*_{ij} \in \mathbf{L}$. Then the process of w_{ij}, w^*_{ij} estimation using $\mathscr{P}$ and $\mathscr{H}$ is called **MPFN learning**.*

The learning process changes the weights from unit values down, while the system of fuzzy logic functions in the second layer is given before learning and then permanent. The MPFN learning has two phases:

I) initialization phase

$$\begin{aligned} w_{i,j} &= 1, \text{ for } 1 \le i \le m,\ 1 \le j \le H \\ w^*_{i,j} &= 1, \text{ for } 1 \le i \le m,\ 1 \le j \le H \end{aligned}$$

II) one pattern learning

Let $(\boldsymbol{x}_k, \boldsymbol{y}'_k)$ be the k-th pattern. Let M be the number of patterns. Then the learning is repeated for $1 \le k \le M$:

$$\begin{aligned} {w_{i,j}}^{NEW} &= w_{i,j} \wedge (\mathrm{h}_j(\boldsymbol{x}_k) \to y'_{k,i}), \\ {w^*_{i,j}}^{NEW} &= w^*_{i,j} \wedge (\mathrm{h}_j(\boldsymbol{x}_k) \to \neg y'_{k,i}) \end{aligned}$$

for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, H$.

Remark 1: More sophisticated approach is based on weight optimization—the weights $(\boldsymbol{w}, \boldsymbol{w}^*) \in [0,1]^{2H}$ are set to minimum sum of squares: $\mathrm{ssq} = \sum_{k=1}^{M} \sum_{i=1}^{m} \left(y'_{k,i} - \mathrm{y}_i(\boldsymbol{x}_k)\right)^2$.

MPFN Processing Rule

The MPFN produces m outputs and its processing rule is defined as

$$\begin{aligned}
&\mathrm{h}_j = \mathrm{FLF}_j(\boldsymbol{x}) && \text{for } j = 1, \ldots, H,\\
&\mathcal{E}_i(\boldsymbol{x}) = \bigvee_{j=1}^{H} w_{i,j} \otimes \mathrm{h}_j(\boldsymbol{x}) && \text{for } i = 1, \ldots, m,\\
&\mathcal{E}_i^*(\boldsymbol{x}) = \bigvee_{j=1}^{H} w_{i,j}^* \otimes \neg\mathrm{h}_j(\boldsymbol{x}) && \text{for } i = 1, \ldots, m,\\
&\mathrm{y}_i(\boldsymbol{x}) = \mathrm{sqrt}(\mathcal{E}_i(\boldsymbol{x})) \otimes \mathrm{sqrt}(\neg\mathcal{E}_i^*(\boldsymbol{x})) && \text{for } i = 1, \ldots, m.
\end{aligned}$$

MATLAB source code

The MPFN was realized in MATLAB system using three general functions. The `MPFNresp` produces the MPFN output. The `MPFNlearn` realizes the learning process using given pattern set while the `MPFNopt` optimizes $\boldsymbol{w}$ and $\boldsymbol{w}^*$. MPFN library codes you can see in appendix F.1.

MPFN Properties

THEOREM 7.1. *Let $H \in \mathbf{N}$ be number of hidden FLFs in given MPFN. Let λ_j be their sensitivities for $j = 1, \ldots, H$. Then any MPFN output is FLF of input variables with the sensitivity*

$$\lambda \le \max_{j=1,\ldots,H} \lambda_j.$$

Proof. Let $H, m \in \mathbf{N}$. Let λ_j be sensitivity of h_j for all $j = 1, \ldots, H$. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n, \boldsymbol{x} \ne \boldsymbol{y}$. Let $1 \le i \le m$. Then any output function y_i satisfies

$$\begin{aligned}
|\mathrm{y}_i(\boldsymbol{x}) - \mathrm{y}_i(\boldsymbol{y})| &= \left|\mathrm{sqrt}\big(\mathcal{E}_i(\boldsymbol{x})\big) \otimes \mathrm{sqrt}\big(\neg\mathcal{E}_i^*(\boldsymbol{x})\big) - \Big(\mathrm{sqrt}\big(\mathcal{E}_i(\boldsymbol{y})\big) \otimes \mathrm{sqrt}\big(\neg\mathcal{E}_i^*(\boldsymbol{y})\big)\Big)\right| =\\
&= \left|\frac{1 + \mathcal{E}_i(\boldsymbol{x}) - \mathcal{E}_i^*(\boldsymbol{x})}{2} - \frac{1 + \mathcal{E}_i(\boldsymbol{y}) - \mathcal{E}_i^*(\boldsymbol{y})}{2}\right| \le\\
&\le \frac{1}{2} \cdot \left|\mathcal{E}_i(\boldsymbol{x}) - \mathcal{E}_i(\boldsymbol{y})\right| + \frac{1}{2} \cdot \left|\mathcal{E}_i^*(\boldsymbol{x}) - \mathcal{E}_i^*(\boldsymbol{y})\right| = \frac{1}{2} \cdot \left|\bigvee_{j=1}^{H} \big(w_{i,j} \otimes \mathrm{h}_j(\boldsymbol{x})\big) -\right.\\
&\quad \left. - \bigvee_{j=1}^{H} \big(w_{i,j} \otimes \mathrm{h}_j(\boldsymbol{y})\big)\right| + \frac{1}{2} \cdot \left|\bigvee_{j=1}^{H} \big(w_{i,j}^* \otimes \neg\mathrm{h}_j(\boldsymbol{x})\big) - \bigvee_{j=1}^{H} \big(w_{i,j}^* \otimes \neg\mathrm{h}_j(\boldsymbol{y})\big)\right| \le\\
&\le \frac{1}{2} \cdot \left(\bigvee_{j=1}^{H} \left|\mathrm{h}_j(\boldsymbol{x}) - \mathrm{h}_j(\boldsymbol{y})\right|\right) + \frac{1}{2} \cdot \left(\bigvee_{j=1}^{H} \left|\mathrm{h}_j(\boldsymbol{x}) - \mathrm{h}_j(\boldsymbol{y})\right|\right) =\\
&= \max_{j=1,\ldots,H} \left|\mathrm{h}_j(\boldsymbol{x}) - \mathrm{h}_j(\boldsymbol{y})\right| = \max_{j=1,\ldots,H} \big(\mathrm{h}_j(\boldsymbol{x}) \circ \mathrm{h}_j(\boldsymbol{y})\big).
\end{aligned}$$

and then

$$\lambda \le \max_{j=1,\ldots,H} \lambda_j.$$

□

THEOREM 7.2. *MPFN weights are FLFs of pattern properties.*

Proof. It implies from the definition 1.14 and one pattern learning of MPFN weights. □

Summary

The MPFN produces the FLFs and it contains learnable weights. Three basic properties of MPFN are guaranteed:

- MPFN outputs are FLFs (each output satisfies Lipschitz condition)
- MPFN weights are FLFs of pattern properties
- MPFN sensitivity is not higher than hidden FLF sensitivities

The MPFN properties are useful for image processing as described in chapter 8. In case of image processing, the MPFN output is just one ($m = 1$).

7.2 Min-Max Fuzzy Network

The ***min-max fuzzy network*** (MMFN) is a four-layered network which is based on fuzzy logic functions and operators of $\mathrm{LA}_{\mathrm{sqrt}}$. The first layer consists of $n \in \mathbf{N}$ inputs $x_1, \ldots, x_n \in \mathbf{L}$. The second and third layers are hidden, the first of them contains $F \in \mathbf{N}$ selected fuzzy logic functions $\varphi_1, \ldots, \varphi_F$ and the second one consists of $H \in \mathbf{N}$ neurons $p_1, \ldots, p_H$. Each neuron p_i is a subset of $\{\varphi_1, \ldots, \varphi_F\}$ and it produces the conjunction of its inputs. The fourth layer of MMFN produces one output y using disjunction of the third layer.

From the mathematical point of view, the output of MMFN is described as

$$y = \bigvee_{k=1}^{H} \left(\bigwedge_{j \in p_k} \varphi_j(\boldsymbol{x}) \right) \text{ where } p_k \subseteq \{\varphi_1, \ldots, \varphi_F\}.$$

Remark 1: The number H is called a *complexity* of MMFN.

Remark 2: It is clearly, the maximal H is 2^F.

Remark 3: The selection of H subsets can be a subject of discrete optimization.

The MMFN structure for $n = 9$, $F = 4$, $H = 5$ where $p_1 = \{\varphi_1, \varphi_2\}$, $p_2 = \{\varphi_1, \varphi_3\}$, $p_3 = \{\varphi_2, \varphi_3\}$, $p_4 = \{\varphi_1, \varphi_4\}$ and $p_5 = \{\varphi_2, \varphi_4\}$ is depicted in the Fig. 7.4.

MMFN Structure Optimization

If we know the ideal output, we can optimize the MMFN structure using a following learning task. Let $m \in \mathbf{N}$ be a number of patterns. Then $(x_{i1}, \ldots, x_{in}, y_i^*) \in \mathbf{L}^{n+1}$ is called i-th *pattern* for $i = 1, \ldots, m$ where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{in})$ is the input vector and y^* is the ideal output for $\boldsymbol{x}_i$. Let $y_i = \bigvee_{k=1}^{H} (\bigwedge_{j \in p_k} \varphi_j(\boldsymbol{x}_i))$ is the *response* of MMFN for i-th pattern. Let $Q = \mathrm{Q}(\boldsymbol{y}, \boldsymbol{y}^*)$ be a given quality measure where $\boldsymbol{y} = (y_1, \ldots, y_m)$, $\boldsymbol{y}^* = (y_1^*, \ldots, y_m^*)$. Then $Q = \mathrm{f}(\boldsymbol{p})$ for $\boldsymbol{p} = (p_1, \ldots, p_H)$.

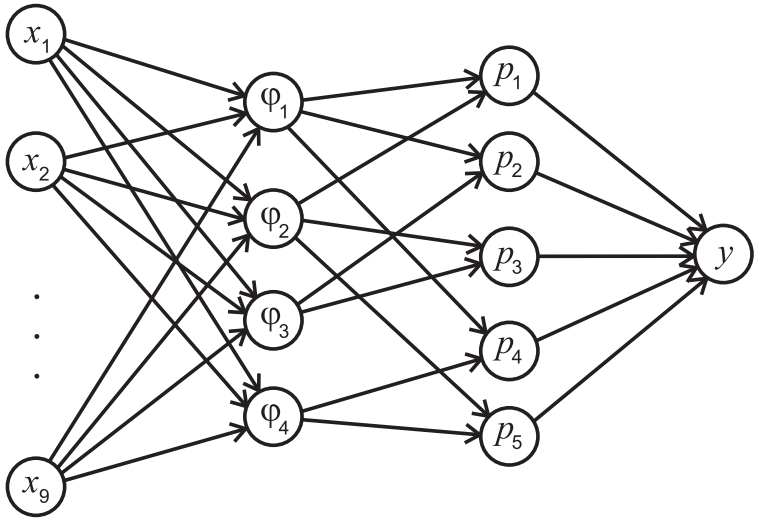


Figure 7.4: Example of MMFN structure ($n = 9$, $F = 4$, $H = 5$)

So the learning of the MMFN is converted to finding the optimum vector of parameters

$$\boldsymbol{p}_{\text{opt}} = \arg\min_{\boldsymbol{p}} \mathrm{f}(\boldsymbol{p}).$$

The optimization task is too difficult to be solved exactly in finite and acceptable time horizon. From the pragmatic point of view, any sub-optimum solution $\boldsymbol{p}_{\text{sub}}$ is also acceptable result satisfying the condition

$$\mathrm{f}(\boldsymbol{p}_{\text{sub}}) \leq f^*$$

where f^* is the given quality threshold. The parallel 3R [16] heuristic (stochastic discrete optimization) can be used for $\boldsymbol{p}_{\text{sub}}$ finding.

Remark 1: The SNR criterion can be used as quality measure Q.

MATLAB Source Code

The MMFN was realized in MATLAB system using two general functions. The `MMFNresp` produces the MMFN output. The `MMFNopt` realizes the optimization of MMFN structure using objective function `MMFNobj` and given pattern set. MMFN library codes you can see in appendix F.2.

Useful Properties of MMFN

- The MMFN output is FLF.
- The MMFN output is a compromise function of individual inputs $x_1, \ldots, x_n$, i.e. it satisfies

$$\min_{k=1,\ldots,n} x_k \leq \mathrm{y}(x_1, \ldots, x_n) \leq \max_{k=1,\ldots,n} x_k.$$

- The third MMFN layer is a subject of discrete optimization.

7.3 Modular FLF Network

Modular networks consist of N independent networks (subnetworks) and a *gating network*, which determines how much of each network's output is applied to the final output. The single network is also called an *expert network* [2]. The output of modular network is computed as weighted compromise (frequently as weighted sum of subnetworks outputs). Figure 7.5 shows a typical architecture of modular networks with single output.

When the gating network produces a normalized weight vector from $\mathbf{L}^N$ and each subnetwork is FLF, the output weighted compromise is also FLF. This motivates a development of modular FLF network. The gating network can be constructed using each subnetwork output similarity with given, or recommended value.

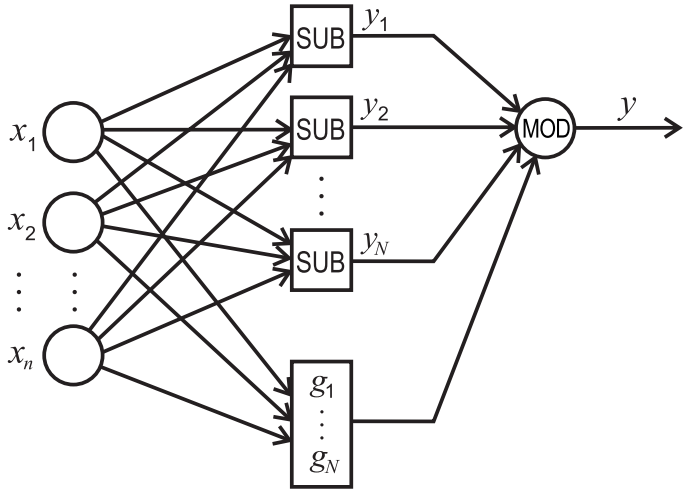


Figure 7.5: Typical architecture of modular networks

7.3.1 Similarity

DEFINITION 7.2. *Let $x, y \in \mathbf{L}$ be values of given property and its ideal. Let $k \in \mathbf{N}$. Then a* ***similarity*** *of x, y values is defined as FLF function* $\mathrm{s} : \mathbf{L} \times \mathbf{L} \to \mathbf{L}$ *where*

$$\mathrm{s}(x, y) = (x \leftrightarrow y)^k = \max(0, 1 - k \cdot |x - y|).$$

Thus the similarity $\mathrm{s}(x, y)$ is only a power of the biresiduum. The recommended values are $1 \leq k \leq 1000$. In the case of image processing, there are some useful similarities of

- filter value and original value (central pixel),
- filter value and recommended value (favorite but not use filter),
- filter variability and zero value (for various submasks).

EXAMPLE 1: Let $\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ be a mask and $X = \begin{pmatrix} 0.1 & 0.2 & 0.4 \\ 0.5 & 0.4 & 0.2 \\ 0.6 & 0.2 & 0.3 \end{pmatrix}$ be a gray intensity values from the neighborhood of central value $x_c = 0.4$. The result of median filtering with the box mask is

$$f_{\mathrm{MED}} = 0.3.$$

The result of dyadic FIR filtering with the ring mask $\mathbf{M}$ is

$$f_{\mathrm{RING}} = \frac{0.1 + 0.2 + 0.4 + 0.5 + 0.2 + 0.6 + 0.2 + 0.3}{8} = 0.3125.$$

The result of binomial FIR filtering is

$$f_{\mathrm{BIN}} = \frac{0.1 + 0.2 \cdot 2 + 0.4 + 0.5 \cdot 2 + 0.4 \cdot 4 + 0.2 \cdot 2 + 0.6 + 0.2 \cdot 2 + 0.3}{16} = 0.325.$$

Using the similarities with original value x_c, the results for $k = 1$ are

$$\begin{aligned} s_{\mathrm{MED}} &= \mathrm{s}(f_{\mathrm{MED}}, x_c) = \mathrm{s}(0.3, 0.4) = 0.9, \\ s_{\mathrm{RING}} &= \mathrm{s}(f_{\mathrm{RING}}, x_c) = \mathrm{s}(0.3125, 0.4) = 0.9125, \\ s_{\mathrm{BIN}} &= \mathrm{s}(f_{\mathrm{BIN}}, x_c) = \mathrm{s}(0.325, 0.4) = 0.925. \end{aligned}$$

EXAMPLE 2: Using the same intensity matrix X and three filters from previous example plus one favorite BES filter as a generator of recommended value, there is

$$f_{\mathrm{BES}} = \frac{0.2 + 0.3 + 0.3 + 0.5}{4} = 0.325.$$

Then the similarity values ($k = 1$) are

$$\begin{aligned} s_{\mathrm{MED}} &= \mathrm{s}(f_{\mathrm{MED}}, f_{\mathrm{BES}}) = \mathrm{s}(0.3, 0.325) = 0.975, \\ s_{\mathrm{RING}} &= \mathrm{s}(f_{\mathrm{RING}}, f_{\mathrm{BES}}) = \mathrm{s}(0.3125, 0.325) = 0.9875, \\ s_{\mathrm{BIN}} &= \mathrm{s}(f_{\mathrm{BIN}}, f_{\mathrm{BES}}) = \mathrm{s}(0.325, 0.325) = 1. \end{aligned}$$

EXAMPLE 3: The same matrix X is used with four rotating masks

$$\mathbf{R}_1 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{R}_2 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{R}_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \mathbf{R}_4 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

and the median filter. Then the medians are $f_1 = 0.3$, $f_2 = 0.3$, $f_3 = 0.25$, $f_4 = 0.45$. The variability of median filter can be described using MED (median of absolute error) which is also FLF function. The median variabilities are

$$\begin{aligned} v_1 &= \mathrm{median}(0.2, 0.1, 0.2, 0.1) = 0.15, \\ v_2 &= \mathrm{median}(0.1, 0.1, 0.1, 0.1) = 0.10, \\ v_3 &= \mathrm{median}(0.15, 0.05, 0.05, 0.05) = 0.05, \\ v_4 &= \mathrm{median}(0.05, 0.05, 0.15, 0.25) = 0.10 \end{aligned}$$

and the similarities with zero ideal ($k = 2$) are

$$\begin{aligned} s_1 &= (v_1 \leftrightarrow 0)^2 = \max(1 - 2 \cdot 0.15, 0) = 0.7, \\ s_2 &= (v_2 \leftrightarrow 0)^2 = \max(1 - 2 \cdot 0.10, 0) = 0.8, \\ s_3 &= (v_3 \leftrightarrow 0)^2 = \max(1 - 2 \cdot 0.05, 0) = 0.9, \\ s_4 &= (v_4 \leftrightarrow 0)^2 = \max(1 - 2 \cdot 0.10, 0) = 0.8. \end{aligned}$$

EXAMPLE 4: The same matrix X is used with four rotating masks

$$\mathbf{R}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \mathbf{R}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \mathbf{R}_3 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{R}_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and the median filter. Then the medians are $f_1 = 0.2$, $f_2 = 0.4$, $f_3 = 0.4$, $f_4 = 0.3$. The variability of median filter is also described using MED criterion. The median variabilities are

$$\begin{aligned} v_1 &= \text{median}(0, 0.2, 0) = 0, \\ v_2 &= \text{median}(0, 0, 0.2) = 0, \\ v_3 &= \text{median}(0.1, 0, 0.2) = 0.1, \\ v_4 &= \text{median}(0.2, 0.1, 0) = 0.1. \end{aligned}$$

and the similarities with zero ideal ($k = 2$) are

$$\begin{aligned} s_1 &= (v_1 \leftrightarrow 0)^2 = \max(1 - 2 \cdot 0, 0) = 1, \\ s_2 &= (v_2 \leftrightarrow 0)^2 = \max(1 - 2 \cdot 0, 0) = 1, \\ s_3 &= (v_3 \leftrightarrow 0)^2 = \max(1 - 2 \cdot 0.1, 0) = 0.8, \\ s_4 &= (v_4 \leftrightarrow 0)^2 = \max(1 - 2 \cdot 0.1, 0) = 0.8. \end{aligned}$$

Now, a modular FLF network can be defined.

7.3.2 Modular FLF Network

DEFINITION 7.3. *Let $n, N \in \mathbf{N}$ are number of inputs and number of subnetworks. Let each subnetwork is FLF and y_k be output of k-th subnetwork for $k = 1, \ldots, N$. Let a gating network produces a normalized vector $\boldsymbol{g} \in \mathbf{L}^N$ using similarities of subnetwork outputs with given value x^*. Let $y_{\text{REF}} = \text{r}(\boldsymbol{y})$ where* r *is a FLF. Then a modular network, whose output*

$$y = \text{r}(\boldsymbol{y}) \oplus \left(\bigvee_{k=1}^{N} \left(g_k \otimes (y_k \ominus \text{r}(\boldsymbol{y})) \right) \right) \ominus \left(\bigvee_{k=1}^{N} \left(g_k \otimes (\text{r}(\boldsymbol{y}) \ominus y_k) \right) \right),$$

is called a ***modular FLF network*** *(MFLFN) where* r *is called a* ***referential function****.*

Remark 1: Normalized similarity vector is used as $\boldsymbol{g}$ (normalization is FLF—theorem 2.7).

Remark 2: The useful referential function is a median.

The MFLFN can be used for the local image processing when input vector is weighted list of neighborhood values. The MFLFN enables to combine several FLF filters to obtain an output compromise FLF filter with constrained sensitivity. An example of MFLFN structure is depicted in the Fig. 7.6.

Using examples from the section 7.3.1, the similarities are normalized to obtain the gating vector $\boldsymbol{g}$.

EXAMPLE 1: Having $\boldsymbol{y} = (0.3, 0.325, 0.3125)$, $\boldsymbol{s} = (0.9, 0.9125, 0.925)$, $y_{\text{REF}} = \text{median}(\boldsymbol{y}) = 0.3125$, we obtain $\boldsymbol{g} = \text{u}(\boldsymbol{s}) = (0.975, 0.9875, 1)$ and the MFLFN output is

$y = 0.3125 \oplus \max\big(0.975 \otimes (0.3 \ominus 0.3125), 0.9875 \otimes (0.325 \ominus 0.3125), 1 \otimes (0.3125 \ominus 0.3125)\big) \ominus \max\big(0.975 \otimes (0.3125 \ominus 0.3), 0.9875 \otimes (0.3125 \ominus 0.325), 1 \otimes (0.3125 \ominus 0.3125)\big) = 0.3125 \oplus 0 \ominus 0 = 0.3125.$

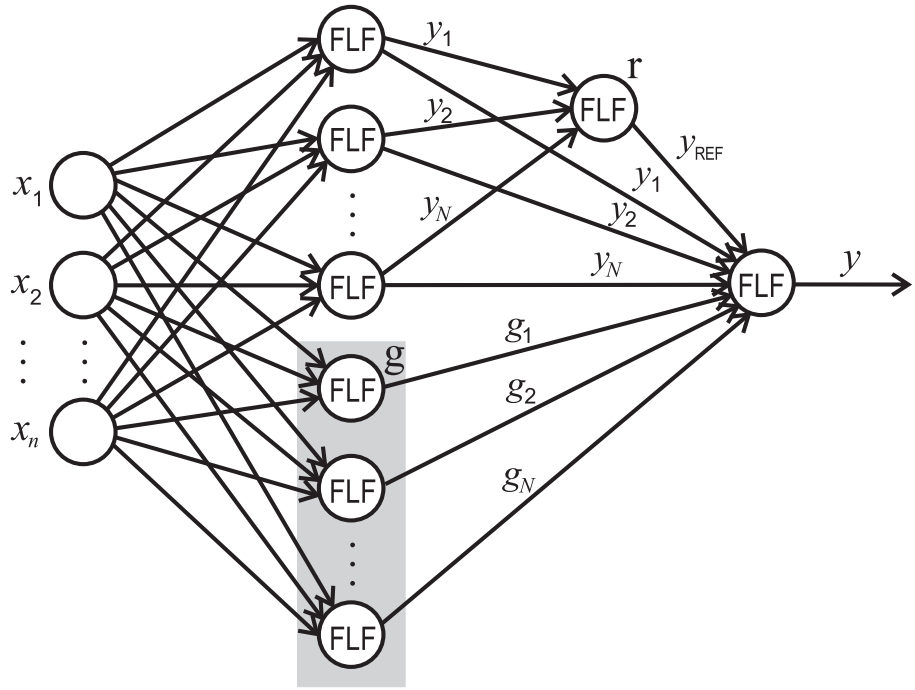


Figure 7.6: Architecture of MFLFN

EXAMPLE 2: Having $\boldsymbol{y} = (0.3, 0.325, 0.3125)$, $\boldsymbol{s} = (0.975, 0.9875, 1)$, $y_{\text{REF}} = f_{\text{BES}} = 0.325$, we obtain $\boldsymbol{g} = \text{u}(\boldsymbol{s}) = \boldsymbol{s}$ and the MFLFN output is

$y = 0.325 \oplus \max\big(0.975 \otimes (0.3 \ominus 0.325), 0.9875 \otimes (0.325 \ominus 0.325), 1 \otimes (0.3125 \ominus 0.325)\big) \ominus \max\big(0.975 \otimes (0.325 \ominus 0.3), 0.9875 \otimes (0.325 \ominus 0.325), 1 \otimes (0.325 \ominus 0.3125)\big) = 0.325 \oplus 0 \ominus 0.0125 = 0.3125.$

EXAMPLE 3: Having $\boldsymbol{y} = (0.3, 0.3, 0.25, 0.45)$, $\boldsymbol{s} = (0.7, 0.8, 0.9, 0.8)$, $y_{\text{REF}} = \text{median}(\boldsymbol{y}) = 0.3$, we obtain $\boldsymbol{g} = \text{u}(\boldsymbol{s}) = (0.8, 0.9, 1, 0.9)$ and the MFLFN output is

$y = 0.3 \oplus \max\big(0.8 \otimes (0.3 \ominus 0.3), 0.9 \otimes (0.3 \ominus 0.3), 1 \otimes (0.25 \ominus 0.3), 0.9 \otimes (0.45 \ominus 0.3)\big) \ominus \max\big(0.8 \otimes (0.3 \ominus 0.3), 0.9 \otimes (0.3 \ominus 0.3), 1 \otimes (0.3 \ominus 0.25), 0.9 \otimes (0.3 \ominus 0.45)\big) = 0.3 \oplus 0.05 \ominus 0.05 = 0.3.$

EXAMPLE 4: Having $\boldsymbol{y} = (0.2, 0.4, 0.4, 0.3)$, $\boldsymbol{s} = (1, 1, 0.8, 0.8)$, $y_{\text{REF}} = \text{median}(\boldsymbol{y}) = 0.35$, we obtain $\boldsymbol{g} = \boldsymbol{s} = (1, 1, 0.8, 0.8)$ and the MFLFN output is

$y = 0.35 \oplus \max\big(1 \otimes (0.2 \ominus 0.35), 1 \otimes (0.4 \ominus 0.35), 0.8 \otimes (0.4 \ominus 0.35), 0.8 \otimes (0.3 \ominus 0.35)\big) \ominus \max\big(1 \otimes (0.35 \ominus 0.2), 1 \otimes (0.35 \ominus 0.4), 0.8 \otimes (0.35 \ominus 0.4), 0.8 \otimes (0.35 \ominus 0.3)\big) = 0.35 \oplus 0.05 \ominus 0.15 = 0.25.$

MFLFN Properties

THEOREM 7.3. *Let $N \in \mathbf{N}$ be number of FLF subnetworks in given MFLFN. Let λ_k be their sensitivities for $k = 1, \ldots, N$. Let λ_{r} be a sensitivity of referential FLF and λ_k^* be sensitivities of g_k where $\boldsymbol{g} = (g_1, \ldots, g_N)$, $\boldsymbol{g} = \text{g}(\boldsymbol{x})$ and g is FLF. Then MFLFN output is FLF of input variables with the sensitivity*

$$\lambda \leq \lambda_{\text{r}} + 2 \cdot \max_{k=1,\ldots,N} \lambda_k + 2 \cdot \max_{k=1,\ldots,N} \lambda_k^*.$$

Proof. Let $y = \mathrm{y}(\boldsymbol{x}) = \mathrm{r}(\boldsymbol{y}) \oplus \left(\bigvee_{k=1}^{N} \left(g_k \otimes (y_k \ominus \mathrm{r}(\boldsymbol{y})) \right) \right) \ominus \left(\bigvee_{k=1}^{N} \left(g_k \otimes (\mathrm{r}(\boldsymbol{y}) \ominus y_k) \right) \right)$, where r is FLF, $(g_1, \ldots, g_N) = \mathrm{g}(\boldsymbol{x})$, $y_k = \mathrm{f}_k(\boldsymbol{x})$ and f, g are FLFs.
Then y is FLF according to definition 1.14. For any $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$, $\boldsymbol{x} \neq \boldsymbol{y}$ is

$$\begin{aligned}
\mathrm{y}(\boldsymbol{x}) \circ \mathrm{y}(\boldsymbol{y}) &= \left| \mathrm{r}(\boldsymbol{x}) \oplus \left(\bigvee_{k=1}^{N} \left(\mathrm{g}_k(\boldsymbol{x}) \otimes (\mathrm{f}_k(\boldsymbol{x}) \ominus \mathrm{r}(\boldsymbol{x})) \right) \right) \ominus \left(\bigvee_{k=1}^{N} \left(\mathrm{g}_k(\boldsymbol{x}) \otimes (\mathrm{r}(\boldsymbol{x}) \ominus \mathrm{f}_k(\boldsymbol{x})) \right) \right) - \right. \\
&- \left. \left(\mathrm{r}(\boldsymbol{y}) \oplus \left(\bigvee_{k=1}^{N} \left(\mathrm{g}_k(\boldsymbol{y}) \otimes (\mathrm{f}_k(\boldsymbol{y}) \ominus \mathrm{r}(\boldsymbol{y})) \right) \right) \ominus \left(\bigvee_{k=1}^{N} \left(\mathrm{g}_k(\boldsymbol{y}) \otimes (\mathrm{r}(\boldsymbol{y}) \ominus \mathrm{f}_k(\boldsymbol{y})) \right) \right) \right) \right| \leq \\
&\leq |\mathrm{r}(\boldsymbol{x}) - \mathrm{r}(\boldsymbol{y})| + 2 \cdot \left| \bigvee_{k=1}^{N} \mathrm{g}_k(\boldsymbol{x}) + \bigvee_{k=1}^{N} \mathrm{f}_k(\boldsymbol{x}) - \left(\bigvee_{k=1}^{N} \mathrm{g}_k(\boldsymbol{y}) + \bigvee_{k=1}^{N} \mathrm{f}_k(\boldsymbol{y}) \right) \right| \leq \\
&\leq |\mathrm{r}(\boldsymbol{x}) - \mathrm{r}(\boldsymbol{y})| + 2 \cdot \max_{k=1,\ldots,N} |\mathrm{g}_k(\boldsymbol{x}) - \mathrm{g}_k(\boldsymbol{y})| + 2 \cdot \max_{k=1,\ldots,N} |\mathrm{f}_k(\boldsymbol{x}) - \mathrm{f}_k(\boldsymbol{y})|
\end{aligned}$$

and then

$$\lambda \leq \lambda_{\mathrm{r}} + 2 \cdot \max_{k=1,\ldots,N} \lambda_k + 2 \cdot \max_{k=1,\ldots,N} \lambda_k^*.$$

□

MATLAB Source Code

The MFLFN was realized as a function `MFLFNresp` in MATLAB system—see appendix F.3.

7.4 Constrained Referential Neural Network

It is not prohibited to have a favorite procedure and use it as referential one. The other procedures can represents a traditional approaches to given task and then typically make a frame or acceptable range for the favoured procedure which should not offer the results out of the range.

DEFINITION 7.4. *Let $n, H \in \mathbf{N}$ be number of inputs and number of standard nodes. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbf{L}^n$, $\boldsymbol{f} = (f_1, \ldots, f_H) \in \mathbf{L}^H$. Let $f_i = \mathrm{f}_i(\boldsymbol{x})$ be a FLF for $i = 1, \ldots, H$. Let $f_{\mathrm{R}} = \mathrm{f}_{\mathrm{REF}}(\boldsymbol{x})$, $f_{\mathrm{L}} = \mathrm{f}_{\mathrm{LOW}}(\boldsymbol{f})$, $f_{\mathrm{U}} = \mathrm{f}_{\mathrm{UPP}}(\boldsymbol{f})$ be FLFs. Let $y = (f_{\mathrm{R}} \vee f_{\mathrm{L}}) \wedge f_{\mathrm{U}}$. Then the structure producing y from $\boldsymbol{x}$ via $\boldsymbol{f}$, f_{R}, f_{L}, f_{U} is called* ***constrained referential neural network*** *(CRNN).*

The structure of CRNN is depicted in the Fig. 7.7.

CRNN Properties

THEOREM 7.4. *The output y of CRNN is FLF of its input $\boldsymbol{x}$.*

Proof. The output is a composition of individual FLFs. Then $y = \mathrm{y}(\boldsymbol{x})$ is also FLF. □

THEOREM 7.5. *Let λ_{REF}, λ_{LOW} and λ_{UPP} be sensitivities of $\mathrm{f}_{\mathrm{REF}}$, $\mathrm{f}_{\mathrm{LOW}}$ and $\mathrm{f}_{\mathrm{UPP}}$. Then the sensitivity of CRNN is*

$$\lambda \leq \max(\lambda_{\mathrm{REF}}, \lambda_{\mathrm{LOW}}, \lambda_{\mathrm{UPP}}).$$

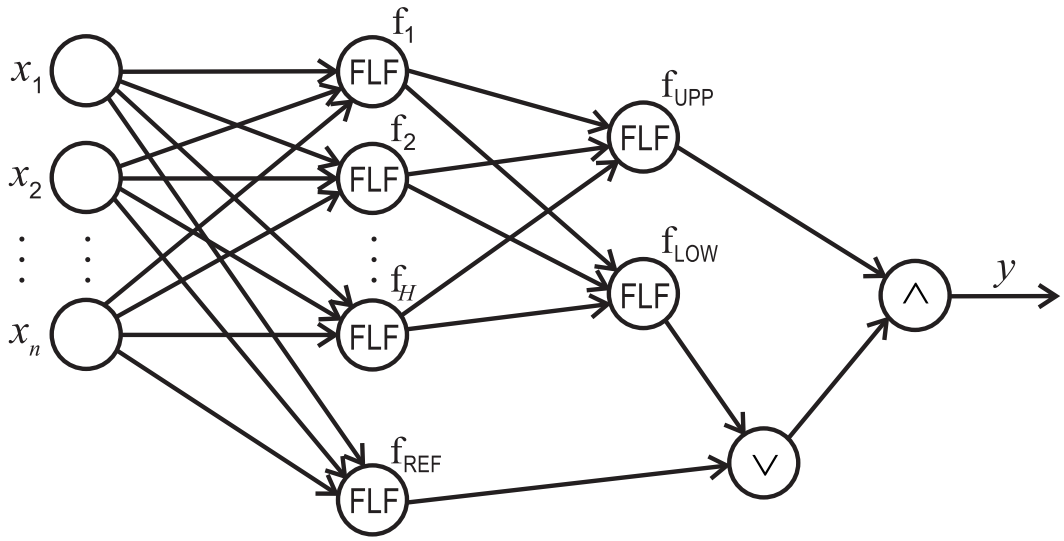


Figure 7.7: Structure of CRNN

Proof. Let λ_{REF}, λ_{LOW} and λ_{UPP} be sensitivities of $\mathrm{f}_{\mathrm{REF}}$, $\mathrm{f}_{\mathrm{LOW}}$ and $\mathrm{f}_{\mathrm{UPP}}$. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$, $\boldsymbol{x} \neq \boldsymbol{y}$. Then the output of CRNN fulfills

$$\begin{aligned}
\mathrm{y}(\boldsymbol{x}) \circ \mathrm{y}(\boldsymbol{y}) &= \Big|\big(\mathrm{f}_{\mathrm{REF}}(\boldsymbol{x}) \vee \mathrm{f}_{\mathrm{LOW}}(\boldsymbol{x})\big) \wedge \mathrm{f}_{\mathrm{UPP}}(\boldsymbol{x}) - \Big(\big(\mathrm{f}_{\mathrm{REF}}(\boldsymbol{y}) \vee \mathrm{f}_{\mathrm{LOW}}(\boldsymbol{y})\big) \wedge \mathrm{f}_{\mathrm{UPP}}(\boldsymbol{y})\Big)\Big| \leq \\
&\leq \Big|\max\big(\mathrm{f}_{\mathrm{REF}}(\boldsymbol{x}), \mathrm{f}_{\mathrm{LOW}}(\boldsymbol{x}), \mathrm{f}_{\mathrm{UPP}}(\boldsymbol{x})\big) - \max\big(\mathrm{f}_{\mathrm{REF}}(\boldsymbol{y}), \mathrm{f}_{\mathrm{LOW}}(\boldsymbol{y}), \mathrm{f}_{\mathrm{UPP}}(\boldsymbol{y})\big)\Big| \leq \\
&\leq \max\big(\big|\mathrm{f}_{\mathrm{REF}}(\boldsymbol{x}) - \mathrm{f}_{\mathrm{REF}}(\boldsymbol{y})\big|, \big|\mathrm{f}_{\mathrm{LOW}}(\boldsymbol{x}) - \mathrm{f}_{\mathrm{LOW}}(\boldsymbol{y})\big|, \big|\mathrm{f}_{\mathrm{UPP}}(\boldsymbol{x}) - \mathrm{f}_{\mathrm{UPP}}(\boldsymbol{x})\big|\big)
\end{aligned}$$

(whereas $\mathrm{f}_{\mathrm{LOW}}$ and $\mathrm{f}_{\mathrm{UPP}}$ are composite functions) and then

$$\lambda \leq \max(\lambda_{\mathrm{REF}}, \lambda_{\mathrm{LOW}}, \lambda_{\mathrm{UPP}}). \qquad \square$$

Remark 1: CRNN is a special case of fixed MIN-MAX structure [3] applied to f_{R}, f_{L} and f_{U} signals.

Remark 2: A trivial but useful range can be obtained by using $f_{\mathrm{L}} = \bigwedge_{k=1}^{H} f_k$, $f_{\mathrm{U}} = \bigvee_{k=1}^{H} f_k$.

Remark 3: More sophisticated range can be obtained from ordered list $(f_{(1)}, \ldots, f_{(H)})$ using $f_{\mathrm{L}} = f_{(\mathrm{KEY})}$, $f_{\mathrm{U}} = f_{(H-\mathrm{KEY}+1)}$ for $\mathrm{KEY} \leq \lfloor H/2 \rfloor$.

Remark 4: In the case of $f_{\mathrm{L}} = f_{(\mathrm{KEY})}$, and $f_{\mathrm{U}} = f_{(H-\mathrm{KEY}+1)}$, the sensitivity of CRNN is

$$\lambda \leq \max(\lambda_{\mathrm{REF}}, \lambda_{\mathrm{LOW}}, \lambda_{\mathrm{UPP}})$$

where λ_k is sensitivity of f_k.

MATLAB Source Code

The CRNN was realized as a function `CRNNresp` in MATLAB system—see appendix F.4.

7.5 Dyadic Weight Neural Network

The last hierarchical approach to image processing is based on a compromise in ŁA_{sqrt} using weighted average with fixed dyadic weights.

DEFINITION 7.5. *Let $n, H \in \mathbf{N}$ be numbers of input and hidden nodes. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbf{L^n}$, $\boldsymbol{f} = (f_1, \ldots, f_H) \in \mathbf{L}^H$. Let $f_i = \mathrm{f}_i(\boldsymbol{x})$ where f_i be FLF for $i = 1, \ldots, H$. Let $y = \sum_{k=1}^{H} w_k \mathrm{f}_k(\boldsymbol{x})$ be an output signal. Let $w_k = m_k/2^N$ where $m_k, N \in \mathbf{N}_0$ and $\sum_{k=1}^{H} w_k = 1$. Then the structure producing y from $\boldsymbol{x}$ by $\boldsymbol{f}$ is called* ***dyadic weight neural network*** *(DWNN).*

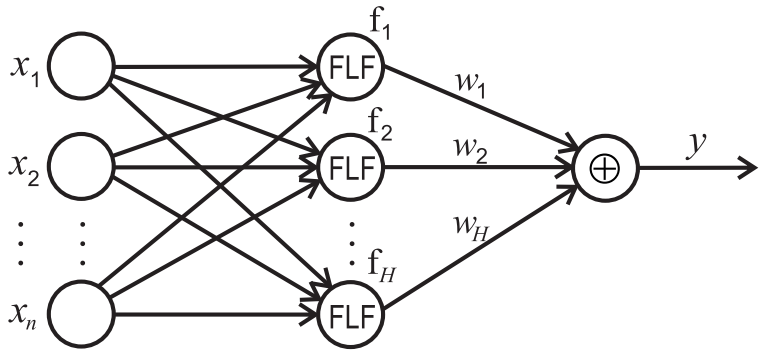


Figure 7.8: Structure of DWNN

DWNN Properties

THEOREM 7.6. *The output of DWNN is FLF of its input $\boldsymbol{x}$.*

Proof. Any f_k is FLF of $\boldsymbol{x}$ and y is a weighted average of $f_1, \ldots, f_H$ with positive dyadic weights. According to theorem 6.8 and definition 1.14, the function y is FLF (whereas $y = \mathrm{y}(\boldsymbol{x})$ is output of DWNN). □

Remark 1: The structure of DWNN is depicted in the Fig. 7.8.

Remark 2: The weights can be subject of discrete optimization for fixed exponent N.

Remark 3: A good initial estimate of weights can be obtained using linear regression in three steps:

(i) non-dyadic weights are obtained by the minimization of the sum of squares:

$$\mathrm{ssq}(w_1, \ldots, w_H) = \min$$

$$\sum_{k=1}^{H} w_k = 1$$

$$w_k \geq 0 \text{ for } k = 1, \ldots, H$$

(ii) rounding of weights to the nearest dyadic values

(iii) final dyadic normalization

THEOREM 7.7. *Let $H \in \mathbf{N}$ and λ_k be a sensitivity of f_k for $k = 1, \ldots, H$. Then the sensitivity of DWNN is*

$$\lambda \leq \max_{1 \leq k \leq H} (\lambda_k).$$

Proof. Let $H \in \mathbf{N}$ and λ_k be a sensitivity of f_k for $k = 1, \ldots, H$. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n$, $\boldsymbol{x} \neq \boldsymbol{y}$. Then

$$\begin{aligned}
\mathrm{y}(\boldsymbol{x}) \circ \mathrm{y}(\boldsymbol{y}) &= \left| \sum_{k=1}^{H} w_k \mathrm{f}_k(\boldsymbol{x}) - \sum_{k=1}^{H} w_k \mathrm{f}_k(\boldsymbol{y}) \right| = \left| \sum_{k=1}^{H} w_k (\mathrm{f}_k(\boldsymbol{x}) - \mathrm{f}_k(\boldsymbol{y})) \right| \leq \\
&\leq \sum_{k=1}^{H} w_k |\mathrm{f}_k(\boldsymbol{x}) - \mathrm{f}_k(\boldsymbol{y})| \leq \sum_{k=1}^{H} w_k \lambda_k \cdot \sum_{j=1}^{n} |x_k - y_k| \leq \\
&\leq \max_{1 \leq k \leq H} (\lambda_k) \cdot \sum_{k=1}^{H} w_k \cdot \sum_{j=1}^{n} |x_k - y_k| = \max_{1 \leq k \leq H} (\lambda_k) \cdot \sum_{j=1}^{n} |x_k - y_k|
\end{aligned}$$

and then

$$\lambda \leq \max_{1 \leq k \leq H} (\lambda_k).$$

□

MATLAB Source Code

The DWNN was realized in MATLAB system using two general functions. The `DWNNresp` produces the DWNN output. The `DWNNlearn` realizes the optimization of DWNN weights using objective function `DWNNobj` and given pattern set. DWNN library codes you can see in appendix F.5.

Chapter 8

Biomedical Image De-Noising

8.1 Biomedical Imaging Techniques

8.1.1 Magnetic Resonance Imaging (MRI)

MRI has evolved into one of the most powerful non-invasive techniques in diagnostic clinical medicine and biomedical research. The technique is an application of nuclear magnetic resonance, a well known analytical method of chemistry, physics and molecular structural biology. MRI is primarily used as a technique for producing anatomical images, but as described below, MRI also gives information on the physical-chemical state of tissues, flow diffusion and motion information.

Most elements have at least one reasonably abundant isotope whose nucleus is magnetic. The hydrogen nucleus is abundant in the body due to the high water content of non-bony tissues. When the body is immersed in a static magnetic field, slightly more protons become aligned with the magnetic field than against the static field. A rapidly alternating magnetic field at an appropriate resonant frequency in the radio frequency range, applied by a coil near the subject or specimen in the static magnetic field, changes the orientation of the nuclear spins relative to the direction of the static magnetic field.

These changes are accompanied by the absorbtion of energy from the alternating magnetic field by nuclei which undergo the transition from a lower energy state to a higher one. When the alternating field is turned off, the nuclei return to the equlibrium state, emitting energy at the same frequency as was previously absorbed. The nuclei of different elements, and even of different isotopes of the same element, have very different resonance frequencies.

The principal components of the MRI machine are the magnet, radiofrequency coils and the gradient coils. The majority of MR systems use superconducting magnets which provide fields of high strength and stability.

RESOURCES: [1], [5]

8.1.2 Positron Emission Tomography (PET)

PET imaging begins with the injection of a metabolically active tracer—a biological molecule that carries with it a positron-emitting isotope. Within minutes, the isotope accumulates in an area of the body for which the molecule has an affinity. The radioactive nuclei then decay by positron emission. The emitted positron collides with a free electron usually within less than 1 mm from the point of emission. The interaction of the two subatomic

particles results in a conversion of matter to energy in the form of two gamma rays. These high-energy gamma rays emerge from the collision point in opposite directions, and are detected by an array of detectors which surround the patient.

When the two photons are recorded simultaneously by a pair of detectors, the collision that gave rise to them must have occurred somewhere along the line connecting the detectors. Of course, if one of the photons is scattered then the line of coincidence will be incorrect. After 500,000 or more annihilation events are detected, the distribution of the positron emitting tracer is calculated by tomographic reconstruction procedures. PET then reconstructs a two-dimensional image. Three dimensional reconstructions can also be done using 2D projections from multiple angles.

Applications to the study of epilepsy, brain tumors, stroke and Alzheimer's disease have occupied the interest of neurologists for nearly 20 years. A major attribute of PET is its ability to show activity of neuroreceptors such as the dopamine, serotonin and noradrenergic receptor systems. The concentrations of neurochemical sites in the body are generally too low for MRI studies.

RESOURCES: [1], [5]

8.1.3 Computed Tomography (CT)

CT is considered one of the major technological advances of medical science. X-ray CT gives anatomical information on the positions of air, soft tissues, and bone. Three dimensional imaging is achieved by rotating an x-ray emitter around the patient, and measuring the intensity of transmitted rays from different angles.

X-ray CT does not use film to detect the transmitted gamma rays. Instead, the photons are collected by an electronic device which converts the x-ray photons into an electric current. Either a solid state device or a crystal scintillator is used to make visible light from the x-rays. Then the light photons are converted to electrons in a photoelectron multiplyer tube. The result is an electric current whose magnitude is proportional to the initial energy of the x-ray photon.

RESOURCES: [1], [5]

8.1.4 Single-Photon Emission Computed Tomography (SPECT)

SPECT, like PET, acquires information on the concentration of radionuclides introduced to the patient's body. As in x-ray CT, SPECT imaging involves the rotation of a photon detector array around the body to acquire data from multiple angles. Using this technique, we seek the position and concentration of radionuclide distribution. Because the emission sources (injected radionuclides) are inside the body cavity, this task is far more difficult than for x-ray CT, where the source position and strength (outside the body) are known at all times. That is, in x-ray CT the attenuation is measured, not the transmission source. To compensate for the attenuation experienced by emission photons from injected tracers in the body, contemporary SPECT machines use mathematical reconstruction algorithms to increase resolution.

SPECT imaging is inferior to PET because of attainable resolution and sensitivity. Because only a single photon is emitted from the radionuclides used for SPECT, a special lens known as a collimator is used to acquire the image data from multiple views around the body. The use of a collimator results in a tremendous decrease in detection efficiency as compared to PET. For positron emission tomographs, collimation is achieved naturally by the fact that

a pair of detected photons can be traced back to their origin since they travel along the same line after being produced. In PET, there might be as many as 500 detectors that could "see" a PET isotope at any one time where as in SPECT, there may be only 1 or 3 collimators. Higher resolutions require better sensitivity and the resolution of SPECT is many times less than that of PET. The resulting useful resolution (about 7 mm) for SPECT is inferior to PET resolutions by a factor of 3 or 4.

Although SPECT imaging resolution is not that of PET, the availability of new SPECT radiopharmaceuticals, particularly for the brain and head, and the practical and economic aspects of SPECT instrumentation make this mode of emission tomography attractive for clinical studies of the brain.

RESOURCES: [1], [5]

8.2 Experimental Scheme

Both traditional and hierarchical FLF filters were applied to the set of artificial and real images. The filters were tested on artificial images first. The ideal technical image is depicted in the Fig. 8.1. The other testing images are biomedical ones. The ideal MRI image [12] is depicted in the Fig. 8.2. The ideal SPECT images [17] obtained using standard built-in Metz filter are depicted in the Fig. 8.3. The artificial technical images were obtained from ideal one applying uniform (I_1), gaussian (I_2), laplacean (I_3) and cauchian (I_4) noise (Figs. 8.4–8.6). The artificial biomedical image I_5 was obtained using addition of noise from top left corner of real MRI image, while I_6 and I_7 were obtained using gaussian noise.

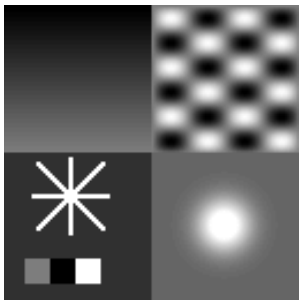


Figure 8.1: Ideal technical image

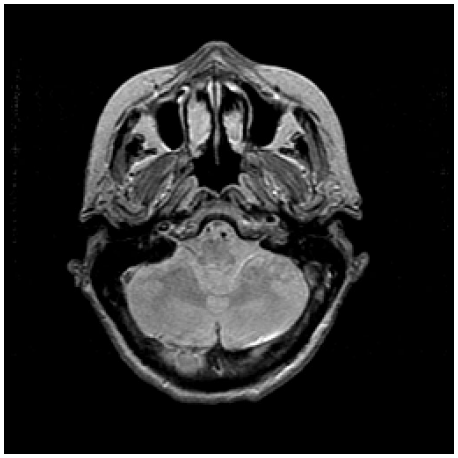
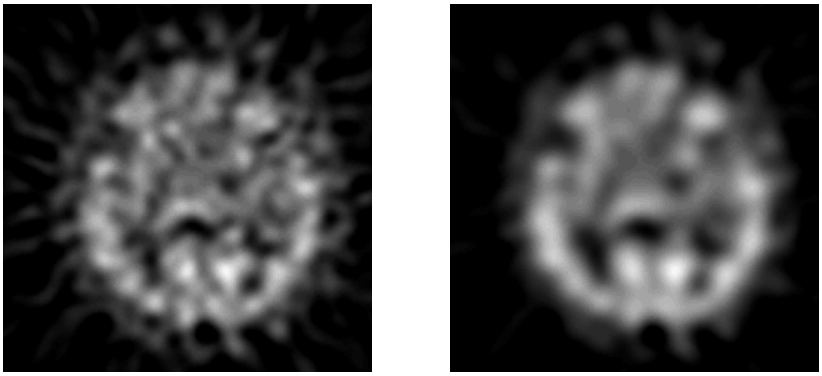
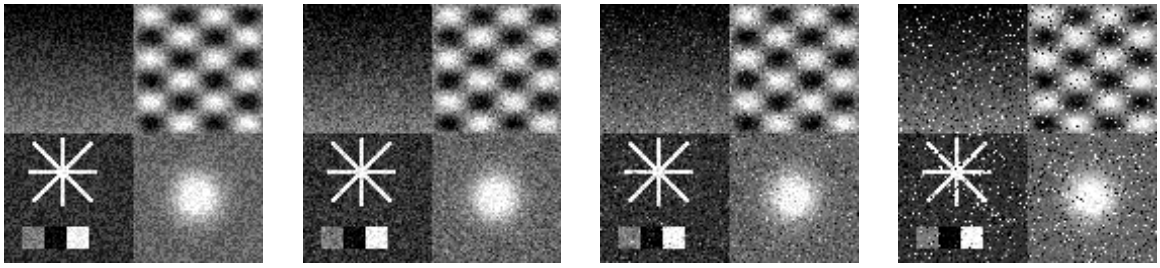


Figure 8.2: Ideal MRI image



(a) (b)

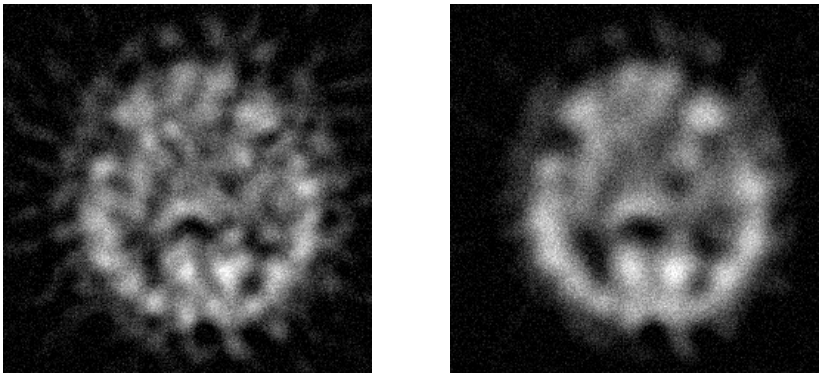
Figure 8.3: Ideal SPECT images



(a) I_1 (b) I_2 (c) I_3 (d) I_4

Figure 8.4: Artificial technical images

Figure 8.5: Artificial MRI image (I_5)



(a) I_6 (b) I_7

Figure 8.6: Artificial SPECT images

Final sets of masks, FLFs and quality measures were used for traditional FLF filter testing. The final set of masks is depicted in the Fig. 8.7, the final list of traditional FLF filters is collected in the Tab. 8.1 and the list of quality measures is in the Tab. 8.2.

$\mathbf{M}_1$

1	1	1
1	1	1
1	1	1

$\mathbf{M}_2$

1	2	1
2	4	2
1	2	1

$\mathbf{M}_3$

0	1	0
1	4	1
0	1	0

$\mathbf{M}_4$

0	1	0
1	1	1
0	1	0

$\mathbf{M}_5$

0	1	0
1	0	1
0	1	0

$\mathbf{M}_6$

1	1	1
1	8	1
1	1	1

Figure 8.7: Final set of masks

FLF	**formula** applied to given list
median	$\mathrm{M}(\boldsymbol{x}) = \underset{k=1,\ldots,n}{\text{median}}\, x_k$
quasi median	$\mathrm{Q}_k(\boldsymbol{x}) = \frac{1}{2} \cdot \left(x_{(\lceil \frac{n+1}{2} \rceil + k)} + x_{(\lfloor \frac{n+1}{2} \rfloor - k)} \right)$
BES estimation	$\mathrm{BES}(\boldsymbol{x}) = \frac{1}{4} \cdot \left(x_{(\lceil \frac{n}{4} \rceil)} + x_{(\lfloor \frac{n+1}{2} \rfloor)} + x_{(\lceil \frac{n+1}{2} \rceil)} + x_{(\lfloor \frac{3n+4}{4} \rfloor)} \right)$
erosion	$\mathrm{ERO}(\boldsymbol{x}) = x_1 \wedge x_2 \wedge \ldots \wedge x_n$
dilation	$\mathrm{DIL}(\boldsymbol{x}) = x_1 \vee x_2 \vee \ldots \vee x_n$
dyadic average	$\mathrm{DA}(\boldsymbol{x}) = \frac{1}{n} \sum_{k=1}^{n} x_k,\ n = 2^N$

Table 8.1: Final set of FLFs

criterion	**formula**
signal to noise ratio	$\mathrm{SNR} = 10 \cdot \log \frac{\mathrm{Var}(X)}{\mathrm{Var}(Y-X)}$
mean of square error	$\mathrm{MSE} = \sqrt{\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} (y_{ij} - x_{ij})^2}{r \cdot c}}$
mean of absolute error	$\mathrm{MAE} = \frac{1}{r \cdot c} \cdot \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \lvert y_{ij} - x_{ij} \rvert \right)$
median of absolute error	$\mathrm{MED} = \underset{\substack{i=1,\ldots,r \\ j=1,\ldots,c}}{\text{median}} \lvert y_{ij} - x_{ij} \rvert$

Table 8.2: List of used quality measures

The individual FLF filters with various masks and local processing function were tested in the first step. The list of individual FLF filters is included in the Tab. A.1 (appendix A.1).

The quality of filtering was computed on the artificial technical images I_1, I_2, I_3, I_4 (Fig. 8.4) as shown in the Tabs. A.2–A.5 (appendix A.2). Analogous experiments were made on artificial MRI (Fig. 8.5) and SPECT (Fig. 8.6) images. The results are collected in the Tab. A.6 (appendix A.3) and Tabs. A.7–A.8 (appendix A.4). The best individual filters for MRI image I_5 are F_{14} (SNR), F_{13} (MSE), F_{47}–F_{58} (MED) and F_1 (MAE), i.e. quasimedians or BES estimation applied to weighted lists are suitable individual filters for MRI or SPECT images according to SNR while the binomial filter (F_{43}) is not recommended for biomedical images.

The multicriterial approach (section 4.2) was applied to the results of individual filtering. Both AIA method for $\lambda = 1$, $\lambda = 2$ and Pareto optimality was used for multicriterial selection. Four criteria (SNR, MSE, MED, MAE) from Tab. 8.2 were used for the individual artificial images. The results are collected in the Tab. 8.3. A set of all artificial images was used with the same criteria in the second step and the results are also in the Tab. 8.3. Finally, the subset of artificial biomedical images was used for the multicriterial selection as shown in the Tab. 8.3, too. The set of Pareto optimum filters for artificial biomedical images consists of eight filters collected in the Tab. 8.4.

Image	**AIA optimum filter** ($\lambda = 1$)	**AIA optimum filter** ($\lambda = 2$)	**Pareto optimum filters**
I_1	F_{28}	F_{28}	F_{18}, F_{24}, F_{27}, F_{28}, F_{43}
I_2	F_{13}	F_{13}	F_{13}, F_{14}, F_{24}, F_{27}, F_{33}, F_{37}
I_3	F_{14}	F_{14}	F_1, F_2, F_{13}, F_{14}
I_4	F_2	F_2	F_1, F_2, F_{13}, F_{14}, F_{49}, F_{52}, F_{55}, F_{58}
I_5	F_1	F_1	F_{27}, F_{37}
I_6	F_{27}	F_{27}	F_{27}, F_{33}, F_{37}
I_7	F_{33}	F_{33}	F_1, F_{13}, F_{27}, F_{33}, F_{37}
I_1–I_7	F_{13}	F_1	F_1, F_2, F_7, F_8, F_{13}, F_{14}, F_{17}, F_{18}, F_{24}, F_{27}, F_{28}, F_{33}, F_{37}, F_{38}, F_{43}, F_{48}, F_{49}, F_{51}, F_{52}, F_{54}, F_{55}, F_{57}, F_{58}
I_5–I_7	F_{13}	F_{13}	F_1, F_{13}, F_{14}, F_{24}, F_{27}, F_{28}, F_{33}, F_{37}

Table 8.3: Multicriterial approach for image processing

Filter	**Mask**	1st **level**	2nd **level**
F_1	$\mathbf{M}_1$	M	
F_{13}	$\mathbf{M}_1$	Q_1	
F_{14}	$\mathbf{M}_2$	Q_1	
F_{24}	$\mathbf{M}_2$	Q_2	
F_{27}	$\mathbf{M}_1$	Walsh list	Q_2
F_{28}	$\mathbf{M}_2$	Walsh list	Q_2
F_{33}	$\mathbf{M}_1$	BES	
F_{37}	$\mathbf{M}_1$	Walsh list	BES

Table 8.4: Selected Pareto optimum filters

Image	SNR [dB]	MSE	MED	MAE
I_1	12.5696	0.0552	0.0292	0.0382
I_2	12.4951	0.0556	0.0272	0.0376
I_3	12.6898	0.0545	0.0250	0.0356
I_4	11.9484	0.0592	0.0211	0.0346
I_5	15.3123	0.0395	0.0181	0.0270
I_6	19.7095	0.0222	0.0145	0.0175
I_7	21.0256	0.0214	0.0137	0.0168

Table 8.5: Final MPFN results

Image	SNR [dB]	MSE	MED	MAE
I_1	13.6071	0.0487	0.0274	0.0349
I_2	13.4806	0.0494	0.0266	0.0347
I_3	13.6424	0.0485	0.0237	0.0324
I_4	12.6196	0.0546	0.0184	0.0303
I_5	15.1654	0.0401	0.0175	0.0265
I_6	19.7715	0.0222	0.0146	0.0174
I_7	21.0854	0.0216	0.0139	0.0167

Table 8.6: Final MFLFN results

The architecture of hierarchical filter can be built on previous set of Pareto optimum filters. The main result of this work is the comparing of various hierarchical principles. Five types of networks (MPFN, MMFN, MFLFN, CRNN, DWNN) according to the chapter 7 were applied to artificial images. The weights of MPFN and DWNN, the structure of MMFN and the parameters of MFLFN and CRNN were subject of optimization for I_5 with SNR criterion. The results of network optimization are:

- MPFN: $w = (0, 0, 0.5416, 1, 0, 0, 1, 1)$, $w^* = (1, 1, 1, 1, 1, 1, 1, 1)$
- MMFN: 120 subsets in first hidden layer
- MFLFN: $k = 2$, r is Hodges–Lehmann median of individual filters, $\boldsymbol{g}$ represents similarity between individual filters and their median
- CRNN: $\mathrm{f}_{\mathrm{REF}} = F_{13}$, KEY = 2
- DWNN: $w = (0, 17/32, 15/32, 0, 0, 0, 0, 0)$ (i.e. the output is affected by F_{13} and F_{14} only)

The quality of outputs from final networks are collected in the Tabs. 8.5–8.8.

The quality measures of MPFN, CRNN and DWNN outputs are better than quality of the best individual filter F_{14} on I_5 (SNR=15.2262 dB). Thus, this three types of networks are recommended for MRI 2D image de-noising.

Image	**SNR** [dB]	**MSE**	**MED**	**MAE**
I_1	13.3338	0.0516	0.0298	0.0375
I_2	13.3779	0.0513	0.0288	0.0367
I_3	13.6246	0.0496	0.0247	0.0336
I_4	13.0102	0.0530	0.0190	0.0304
I_5	15.0042	0.0429	0.0200	0.0293
I_6	19.2994	0.0250	0.0166	0.0197
I_7	20.5948	0.0246	0.0162	0.0192

Table 8.7: MMFN results

Image	**SNR** [dB]	**MSE**	**MED**	**MAE**
I_1	13.6017	0.0487	0.0304	0.0366
I_2	13.6355	0.0485	0.0284	0.0357
I_3	13.8716	0.0473	0.0244	0.0327
I_4	13.1650	0.0513	0.0187	0.0295
I_5	15.3301	0.0394	0.0169	0.0258
I_6	19.1297	0.0238	0.0153	0.0185
I_7	20.4326	0.0231	0.0142	0.0175

Table 8.8: Final CRNN results

Image	**SNR** [dB]	**MSE**	**MED**	**MAE**
I_1	13.5952	0.0488	0.0306	0.0366
I_2	13.7301	0.0480	0.0276	0.0349
I_3	14.1082	0.0460	0.0238	0.0317
I_4	13.3753	0.0500	0.0177	0.0283
I_5	15.5375	0.0381	0.0140	0.0245
I_6	19.3728	0.0231	0.0147	0.0178
I_7	20.7623	0.0221	0.0134	0.0167

Table 8.9: Final DWNN results

Conclusions

Noisy 2D images are typical results of technical monitoring, meteorological observations or biomedical measurements. Noise reduction and structure saving are contradictory but useful aims. Łukasiewicz algebra with square root was used as mathematical model for image processing background. Łukasiewicz algebra with square root as algebraic structure was defined first. This is a simple algebraic model but it has some useful properties:

- the number of operators and functions is minimum possible,
- there is possible to construct infinite number of different expressions and functions,
- the Lipschitz continuity and constrained sensitivity of functions are guaranteed.

Fuzzy logic function and its sensitivity were defined next. Upper bounds of sensitivity of operators and basic functions were inferred. Then the basic terms of 2D image processing were defined and a sample of traditional filters, which are not realizable in Łukasiewicz algebra with square root, was described. The set of fuzzy logic function filters and proofs of their realization in Łukasiewicz algebra with square root were introduced.

The second aim of my work was to define the image enhancement networks as a hierarchical structures of data processing and compare their outputs with results of individual fuzzy logic functions. Łukasiewicz algebra with square root was used for node processing in fuzzy logic function networks. More complex activities can be performed for advanced processing and five network were introduced:

- modus ponens fuzzy network (MPFN),
- min-max fuzzy network (MMFN),
- modular fuzzy logic function network (MFLFN),
- constrained referential neural network (CRNN) and
- dyadic weights neural network (DWNN).

Their sensitivities were also inferred. Finally, the previous hierarchical and individual fuzzy logic function filters were tested and compared on artificial images. The set of them also includes 2D biomedical images with natural and artificial noises. The filtering principles were compared on the basis of traditional quality measures (SNR, MSE, MAE, MED). The multicriterial approach were also used. The best individual and optimized hierarchical fuzzy logic function filters are recommended for the de-noising of biomedical 2D MRI images.

The algorithms of data processing were realized in MATLAB environment and the source code examples are included.

References

[1] C.J. Aine. A conceptual overview and critique of functional neuro-imaging techniques in humans: I. MRI/fMRI and PET. *Critical Reviews in Neurobiology*, 9(2–3):229–309, 1995.

[2] C. Anderson and Z. Hong. Reinforcement Learning with Modular Neural Networks for Control. *Proceedings of NNACIP'94, the IEEE International Workshop on Neural Networks Applied to Control and Image Processing*, 1994.

[3] S. Armstrong, A.C. Kokaram, and P.J.W. Rayner. Non-linear interpolation of missing image data using min-max functions. *Proc. IEEE Nonlinear Signal and Image Processing Conference NSIP'97*, 1997.

[4] F. Bubeník, M. Pultar, and I. Pultarová. *Matematické vzorce a metody.* ČVUT, Praha, 1997.

[5] T.F. Budinger. Imaging technique tutorial, 2003.

[6] M. Demlová and J. Nagy. *Algebra.* MVŠT. SNTL, Praha, 1985.

[7] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Image processing learning resources, 2000.

[8] R.C. Gonzalez and R.E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, Hardcover, 1992.

[9] A.B. Hamza and H. Krim. Nonlinear Image Filtering: Trade-off between Optimality and Practicality. *Proc. IEEE International Conference on Image Processing, Greece*, 2001.

[10] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision*, volume 1. Addison-Wesley, 1992.

[11] J.L. Hodges and E.L. Lehmann. On Medians and Quasi Medians. *Journal of the American Statistical Association*, 62:926–931, 1967.

[12] K.A. Johnson and J.A. Becker. The whole brain atlas, 1997.

[13] R. Klette and P. Zamperoni. *Handbook of Image Processing Operators.* John Wiley and Sons, Chichester, 1996.

[14] J. Kopka. *Svazy a Booleovy algebry.* Univerzita J.E.Purkyně, Ústí nad Labem, 1991.

[15] J. Kukal. Fuzzy envelope and modus ponens fuzzy network. In *Procceedings of the Mendel 2000 Conference*, pages 1–1. FSI VUT Brno, 2000.

[16] J. Kukal and D. Majerová. Parallel random rank roam method in MATLAB. In *Proc. of 8th Int. Conference on Soft Computing Mendel 2002*, pages 335–340. Tech. Univ. of Brno, 2002.

[17] O. Lang. Unpublished results of experiments, 2002.

[18] O. L. Mangasarian. Lipschitz Continuity, Rates of Convergence & Related Results, 2003.

[19] S.K. Mitra and J.F. Kaiser. *Handbook for Digital Signal Processing.* John Wiley and Sons, New York, 1993.

[20] V. Novák, I. Perfilieva, and J. Močkoř. *Mathematical Principles of Fuzzy Logic.* Kluwer Academic Publishers, Boston, 1999.

[21] J. O'Connor. Metric and topological spaces, 2002.

[22] A. Restrepo and A.C. Bovik. Adaptive trimmed mean filters for image restoration. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 36:1326–1337, 1988.

[23] J.W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, New York, 1977.

[24] E. Turunen. BL-algbras of Basic Fuzzy Logic. *Mathware and Soft Computing*, 6(1):49–61, 1999.

[25] Chapman University. Mathematical structures, 2003.

[26] V. Vychodil. Shlukování založené na fuzzy ekvivalenci, 2002.

[27] M. Zeleny. *Multiple Criteria Decision Making.* McGraw-Hill, New York, 1982.

List of the Author's Publications

D. Majerová, J. Kukal. Multicriteria Approach to 2D Image De-Noising by Means of Lukasiewicz Algebra with Square Root. *Neural Network World*, 2002, 12, 4, p. 333–348. ISSN 1210-0552.

J. Kukal, D. Majerová, A. Procházka. Fuzzy Prediction Based on Łukasiewicz Algebra. *Neural Network World*, 2002, 12, 1, p. 45–66. ISSN 1210-0552.

D. Majerová, J. Kukal. Fuzzy řízení v ŁA_{sqrt}. *Automatizace*, 2003, 46, 12, s. 809–813. ISSN 0005-125X

D. Majerová, J. Kukal. Modulární sítě v ŁA_{sqrt}. *Automatizace*, 2003, 46, 5, s. 302–304. ISSN 0005-125X

D. Majerová, J. Kukal. Etalony a umělý intelekt v ŁA_{sqrt}. *Automatizace*, 2003, 46, 2, s. 97–99. ISSN 0005-125X

D. Majerová, J. Kukal. Expertní systém v ŁA_{sqrt}. *Automatizace*, 2002, 45, 12, s. 756–758. ISSN 0005-125X.

D. Majerová, J. Kukal. Algebra a praxe: báječné časy s odmocninou. *Automatizace*, 2002, 45, 10, s. 615–619. ISSN 0005-125X.

D. Majerová, J. Kukal. Algebra a praxe: cesta k Łukasiewiczově algebře. *Automatizace*, 2002, 45, 9, s. 562–566. ISSN 0005-125X.

D. Majerová, J. Kukal. Algebra a praxe: co je dovoleno ve svazu. *Automatizace*, 2002, 45, 8, s. 466–469. ISSN 0005-125X.

D. Majerová, J. Kukal. Zpracování obrazu — rutinní technologie nebo neuskutečnitelný sen. *Automatizace*, 2002, 45, 5–6, s. 348–350. ISSN 0005-125X.

J. Kukal, D. Majerová, V. Musoko, A. Pavelka, A. Procházka. Nonlinear Filtering of 3D MRI in MATLAB. *Sborník 11. konference MATLAB 2003.* Praha, 25. listopad 2003. Ed. HUMUSOFT, s.r.o., 2003, p. 336–340. ISBN 80-7080-526-9.

K. Nováková, J. Kukal, D. Majerová. Invariant Moments for 2D Object Categorization. *Sborník 11. konference MATLAB 2003.* Praha, 25. listopad 2003. Ed. HUMUSOFT, s.r.o., 2003, p. 412–415. ISBN 80-7080-526-9.

D. Bártová, J. Kukal, D. Majerová. Compromise Discrete Nonlinear Control and Its Stability. *Sborník 11. konference MATLAB 2003.* Praha, 25. listopad 2003. Ed. HUMUSOFT, s.r.o., 2003, p. 46–49. ISBN 80-7080-526-9.

D. Majerová, J. Kukal, A. Procházka. Łukasiewicz Neural Network for Hierarchical Image Processing. *Proc. of 14th Int. Conf. on Process Control PC '03.* Štrbské Pleso, June 8–11, 2003. Ed. Slovak University of Technology in Bratislava, 2003, p. PC045/1–4. ISBN 80-227-1902-1.

M. Kubíček, J. Kukal, D. Majerová. Fractal Analysis of Laminar and Turbulent Flow Patterns. *Proc. of 14th Int. Conf. on Process Control PC '03.* Štrbské Pleso, June 8–11, 2003. Ed. Slovak University of Technology in Bratislava, 2003, p. PC030/1–4. ISBN 80-227-1902-1.

M. Kmínek, J. Kukal, D. Majerová. Simple Mixed Predictor in Feedback. *Proc. of 14th Int. Conf. on Process Control PC '03.* Štrbské Pleso, June 8–11, 2003. Ed. Slovak University of Technology in Bratislava, 2003, p. PC078/1–5. ISBN 80-227-1902-1.

D. Bártová, J. Kukal, D. Majerová. Feedback Control of Reactor with Axial Dispersion. *Proc. of 14th Int. Conf. on Process Control PC '03.* Štrbské Pleso, June 8–11, 2003. Ed. Slovak University of Technology in Bratislava, 2003, p. PC028/1–4. ISBN 80-227-1902-1.

J. Kukal, D. Majerová. Modular Network in Łukasiewicz Algebra. *Proc. of 9th Int. Conf. on Soft Computing MENDEL 2003.* Brno, June 4–6, 2003. Ed. VUT Brno, 2003, p. 270–274. ISBN 80-214-2411-7.

J. Kukal, D. Majerová. Optimum Min-Max Network for 2D Image Processing. *Proc. of 9th Int. Conf. on Soft Computing MENDEL 2003.* Brno, June 4–6, 2003. Ed. VUT Brno, 2003, p. 275–278. ISBN 80-214-2411-7.

D. Majerová. Fuzzy filtrace obrazu s užitím Łukasiewiczovy algebry. *Sborník semináře Digitální mikroskopie a analýza obrazu.* Praha, 24.–25. leden 2002. CD-ROM. Ed. VŠCHT Praha, 2002. ISBN 80-7080-477-7.

M. Kukalová, D. Majerová, J. Kukal. Minimum OCR Representation Using Pruning Techniques. *Proceedings of 8th International Conference on Soft Computing MENDEL 2002.* Brno June 5–7, 2002. Ed. VUT Brno, 2002, p. 211–216. ISBN 80-214-2135-5.

J. Kukal, D. Majerová, M. Kukalová. Coming Back to Oja Learning. *Proceedings of 8th International Conference on Soft Computing MENDEL 2002.* Brno June 5–7, 2002. Ed. VUT Brno, 2002, p. 217–220. ISBN 80-214-2135-5.

D. Majerová, J. Kukal. Multicriteria Choice of Optimum Filter. *Proceedings of 8th International Conference on Soft Computing MENDEL 2002.* Brno June 5–7, 2002. Ed. VUT Brno, 2002, p. 281–286. ISBN 80-214-2135-5.

D. Majerová, K. Sýkorová, J. Kukal. Optimum SOM Distribution in Metric Spaces. *Proceedings of 8th International Conference on Soft Computing MENDEL 2002.* Brno June 5–7, 2002. Ed. VUT Brno, 2002, p. 287–291. ISBN 80-214-2135-5.

D. Majerová, I. Křivý, J. Kukal. Parallel Random Rank Roam Method in MATLAB. *Proceedings of 8th International Conference on Soft Computing MENDEL 2002.* Brno June 5–7, 2002. Ed. VUT Brno, 2002, p. 335–340. ISBN 80-214-2135-5.

J. Kukal, M. Záhrubský, D. Majerová, D. Bártová, O. Schmidt. Transfer Function of Chemical Reactor with Boundary Conditions. *Proceedings of 5th International Scientific—Technical Conference on Process Control '02.* Kouty nad Desnou, June

9–12, 2002. CD-ROM. Ed. University of Pardubice, 2002, p. R188/1–7. ISBN 80-7194-452-1.

D. Majerová, L. Hainc, J. Kukal, A. Procházka. Optimum Filtering as Multicriteria Decision Making Task. *Proceedings of 5th International Scientific—Technical Conference on Process Control '02.* Kouty nad Desnou, June 9–12, 2002. CD-ROM. Ed. University of Pardubice, 2002, p. R192a/1–8. ISBN 80-7194-452-1.

D. Majerová, K. Sýkorová, D. Bártová, J. Kukal. Optimum SOM Distribution in Łukasiewicz Algebra. *Proceedings of 5th International Scientific—Technical Conference on Process Control '02.* Kouty nad Desnou, June 9–12, 2002. CD-ROM. Ed. University of Pardubice, 2002, p. R192b/1–5. ISBN 80-7194-452-1.

J. Kukal, M. Kmínek, D. Majerová, I. Nachtigalová. Non-linear Extension of PSD Controller in Łukasiewicz Algebra with sqrt. *Proceedings of 5th International Scientific–Technical Conference on Process Control '02.* Kouty nad Desnou, June 9–12, 2002. CD-ROM. Ed. University of Pardubice, 2002, p. R201a/1–7. ISBN 80-7194-452-1.

L. Hainc, J. Kukal, D. Majerová, A. Procházka. Bootstrap Filtering of 2D Images. *Proceedings of 5th International Scientific–Technical Conference on Process Control '02. Kouty nad Desnou, June 9–12, 2002.* CD-ROM. Ed. University of Pardubice, 2002, p. R201c/1–4. ISBN 80-7194-452-1.

J. Kukal, PÁNEK Martin, D. Majerová, A. Procházka. Cross Wavelet Denoising of 2D Image. *Proceedings of 5th International Scientific—Technical Conference on Process Control '02.* Kouty nad Desnou, June 9–12, 2002. CD-ROM. Ed. University of Pardubice, 2002, p. R201d/1–7. ISBN 80-7194-452-1.

J. Kukal, M. Mudrová, D. Majerová, A. Procházka. Standard and Fuzzy Edge Operators in Biomedical Image Analysis. *Proceedings of the 16th International Conference Biosignal 2002.* Brno, June 26–28, 2002. Ed. VUT Brno, 2002, p. 338–340. ISBN 80-214-2120-7.

D. Majerová, O. Lang, J. Kukal, A. Procházka. Objective Approach to SPECT Filtering. *Sborník příspěvků 10. ročníku konference MATLAB 2002.* Praha, 7. listopad 2002. Ed. VŠCHT Praha, 2002, p. 295–300. ISBN 80-7080-500-5.

J. Kukal, D. Majerová, M. Pánek, A. Procházka. Lukasiewicz Algebra and Remote Fuzzy Image Enhancement. *Proc. of 1st CISIC Conference 2002*, vol. 2. Madrid, March 2002. Ed. McGraw Hill, 2002, p. 195–199. ISBN 84-481-3603-9.

J. Kukal, D. Majerová, A. Procházka. Fuzzy Approach to Image Enhancement. *Proceedings of the 13th International Conference on Process Control '01.* Štrbské Pleso, June 11–14, 2001. CD-ROM. Ed. STU Bratislava, 2001, p. P113/1–3. ISBN 80-227-1542-5.

D. Majerová, A. Procházka, J. Kukal. Sunspot Number Prediction Using Łukasiewicz Algebra with Square Root. *Proceedings of the Conference on Information Engineering and Process Control 2001.* Praha, September 3–4, 2001. Ed. VŠCHT Praha, 2001, p. 51–52. ISBN 80-214-1609-2.

J. Kukal, D. Majerová, L. Hainc. Regularized Approach to Linear Fitting. *Sborník příspěvků 9. ročníku konference MATLAB 2001.* Praha, 11. říjen 2001. Ed. VŠCHT Praha, 2001, p. 200–203. ISBN 80-7080-446-7.

J. Kukal, I. Křivý, D. Majerová. MATLAB Realization of Parallel Random Rank Roam Method. *Sborník příspěvků 9. ročníku konference MATLAB 2001.* Praha, 11. říjen 2001. Ed. VŠCHT Praha, 2001, p. 193–199. ISBN 80-7080-446-7.

D. Majerová, A. Procházka, J. Kukal. Fuzzy Prediction of Sunspot Number. *Sborník příspěvků 9. ročníku konference MATLAB 2001.* Praha, 11. říjen 2001. Ed. VŠCHT Praha, 2001, p. 236–241. ISBN 80-7080-446-7.

D. Majerová, J. Kukal, A. Procházka. 2D Image Fuzzy Filters in MATLAB. *Sborník příspěvků 9. ročníku konference MATLAB 2001.* Praha, 11. říjen 2001. Ed. VŠCHT Praha, 2001, p. 231–235. ISBN 80-7080-446-7.

J. Kukal, D. Majerová. Modus Ponens Fuzzy Network in MATLAB. *Sborník příspěvků 8. ročníku konference MATLAB 2000.* Praha, 1. listopad 2000. Ed. VŠCHT Praha, 2000, p. 185–188. ISBN 80-7080-401-7.

D. Majerová, J. Kukal. Fuzzy Technique for Picture Contrasting. *Proceedings of 6th International Conference on Soft Computing MENDEL 2000.* Brno, June 7–9, 2000. Ed. VUT Brno, 2000, p. 273–278. ISBN 80-214-1609-2.

Appendices

Appendix A

Individual FLF Filters and Their Results on Artificial Images

A.1 Individual FLF Filters

Filter	**Mask**	1st **level**	2nd **level**
F_1	$\mathbf{M}_1$	M	
F_2	$\mathbf{M}_2$	M	
F_3	$\mathbf{M}_3$	M	
F_4	$\mathbf{M}_4$	M	
F_5	$\mathbf{M}_5$	M	
F_6	$\mathbf{M}_6$	M	
F_7	$\mathbf{M}_1$	Walsh list	M
F_8	$\mathbf{M}_2$	Walsh list	M
F_9	$\mathbf{M}_3$	Walsh list	M
F_{10}	$\mathbf{M}_4$	Walsh list	M
F_{11}	$\mathbf{M}_5$	Walsh list	M
F_{12}	$\mathbf{M}_6$	Walsh list	M
F_{13}	$\mathbf{M}_1$	Q_1	
F_{14}	$\mathbf{M}_2$	Q_1	
F_{15}	$\mathbf{M}_4$	Q_1	
F_{16}	$\mathbf{M}_5$	Q_1	
F_{17}	$\mathbf{M}_1$	Walsh list	Q_1
F_{18}	$\mathbf{M}_2$	Walsh list	Q_1
F_{19}	$\mathbf{M}_3$	Walsh list	Q_1
F_{20}	$\mathbf{M}_4$	Walsh list	Q_1
F_{21}	$\mathbf{M}_5$	Walsh list	Q_1
F_{22}	$\mathbf{M}_6$	Walsh list	Q_1
F_{23}	$\mathbf{M}_1$	Q_2	
F_{24}	$\mathbf{M}_2$	Q_2	
F_{25}	$\mathbf{M}_4$	Q_2	
F_{26}	$\mathbf{M}_5$	Q_2	
F_{27}	$\mathbf{M}_1$	Walsh list	Q_2

continued on next page...

Table A.1 Individual FLF filters

...continued from previous page

Filter	**Mask**	1st **level**	2nd **level**
F_{28}	$\mathbf{M}_2$	Walsh list	Q_2
F_{29}	$\mathbf{M}_3$	Walsh list	Q_2
F_{30}	$\mathbf{M}_4$	Walsh list	Q_2
F_{31}	$\mathbf{M}_5$	Walsh list	Q_2
F_{32}	$\mathbf{M}_6$	Walsh list	Q_2
F_{33}	$\mathbf{M}_1$	BES	
F_{34}	$\mathbf{M}_2$	BES	
F_{35}	$\mathbf{M}_3$	BES	
F_{36}	$\mathbf{M}_6$	BES	
F_{37}	$\mathbf{M}_1$	Walsh list	BES
F_{38}	$\mathbf{M}_2$	Walsh list	BES
F_{39}	$\mathbf{M}_3$	Walsh list	BES
F_{40}	$\mathbf{M}_4$	Walsh list	BES
F_{41}	$\mathbf{M}_5$	Walsh list	BES
F_{42}	$\mathbf{M}_6$	Walsh list	BES
F_{43}	$\mathbf{M}_2$	DA	
F_{44}	$\mathbf{M}_3$	DA	
F_{45}	$\mathbf{M}_5$	DA	
F_{46}	$\mathbf{M}_6$	DA	
F_{47}	$\mathbf{M}_1$	ERO	
F_{48}	$\mathbf{M}_5$	ERO	
F_{49}	$\mathbf{M}_6$	ERO	
F_{50}	$\mathbf{M}_1$	DIL	
F_{51}	$\mathbf{M}_5$	DIL	
F_{52}	$\mathbf{M}_6$	DIL	
F_{53}	$\mathbf{M}_1$	ERO	
F_{54}	$\mathbf{M}_5$	ERO	
F_{55}	$\mathbf{M}_6$	ERO	
F_{56}	$\mathbf{M}_1$	DIL	ERO
F_{57}	$\mathbf{M}_5$	DIL	ERO
F_{58}	$\mathbf{M}_6$	DIL	ERO

Table A.1: Individual FLF filters

A.2 Artificial Technical Images

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
NOISED	8.8509	0.0842	0.0709	0.0718
F_1	12.2460	0.0570	0.0336	0.0404
F_2	**12.6001**	**0.0547**	0.0373	0.0427
F_3	9.8430	0.0751	0.0625	0.0638
F_4	11.8363	0.0597	0.0418	0.0472
F_5	10.1363	0.0726	0.0404	0.0497
F_6	9.2121	0.0808	0.0694	0.0692
F_7	9.8727	0.0749	**0.0251**	0.0412
F_8	12.1491	0.0576	0.0274	**0.0368**
F_9	11.8369	0.0597	0.0443	0.0489
F_{10}	11.5403	0.0618	0.0312	0.0406
F_{11}	10.3441	0.0709	0.0371	0.0471
F_{12}	11.9605	0.0589	0.0452	0.0485
F_{13}	**13.2110**	**0.0510**	0.0299	**0.0365**
F_{14}	**13.0991**	**0.0516**	0.0350	0.0404
F_{15}	11.2825	0.0636	0.0333	0.0422
F_{16}	8.4739	0.0880	0.0309	0.0499
F_{17}	9.8694	0.0749	**0.0250**	0.0412
F_{18}	12.1592	0.0575	0.0273	**0.0368**
F_{19}	11.9170	0.0591	0.0437	0.0484
F_{20}	11.4805	0.0622	0.0313	0.0406
F_{21}	10.0978	0.0729	0.0351	0.0470
F_{22}	11.9511	0.0589	0.0452	0.0485
F_{23}	9.2974	0.0801	0.0271	0.0443
F_{24}	**13.5415**	**0.0491**	0.0320	0.0376
F_{25}	11.2825	0.0636	0.0333	0.0422
F_{26}	8.4739	0.0880	0.0309	0.0499
F_{27}	11.5737	0.0616	**0.0250**	0.0379
F_{28}	**13.2174**	**0.0509**	0.0274	**0.0355**
F_{29}	12.0672	0.0581	0.0430	0.0475
F_{30}	12.5863	0.0548	0.0327	0.0401
F_{31}	10.0675	0.0732	0.0358	0.0477
F_{32}	11.9654	0.0588	0.0462	0.0489
F_{33}	11.5662	0.0617	0.0292	0.0403
F_{34}	12.0821	0.0581	0.0310	0.0406
F_{35}	11.6749	0.0608	0.0451	0.0495
F_{36}	11.6216	0.0612	0.0497	0.0517
F_{37}	10.8034	0.0673	**0.0255**	0.0394
F_{38}	12.4708	0.0555	0.0281	**0.0373**
F_{39}	12.2042	0.0572	0.0434	0.0472
F_{40}	11.9849	0.0587	0.0313	0.0404
F_{41}	10.3677	0.0707	0.0366	0.0470

continued on next page...

Table A.2 Quality of I_1 filtering

...continued from previous page

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
F_{42}	12.0450	0.0583	0.0443	0.0479
F_{43}	12.0427	0.0583	**0.0264**	0.0373
F_{44}	12.2540	0.0569	0.0384	0.0445
F_{45}	10.0821	0.0731	0.0338	0.0465
F_{46}	12.2030	0.0572	0.0388	0.0447
F_{47}	5.4885	0.1934	0.1305	0.1486
F_{48}	6.0009	0.1683	0.1115	0.1226
F_{49}	5.7174	0.1651	0.1031	0.1163
F_{50}	5.4885	0.1934	0.1305	0.1486
F_{51}	6.0009	0.1683	0.1115	0.1226
F_{52}	5.7174	0.1651	0.1031	0.1163
F_{53}	5.4885	0.1934	0.1305	0.1486
F_{54}	6.0009	0.1683	0.1115	0.1226
F_{55}	5.7174	0.1651	0.1031	0.1163
F_{56}	5.4885	0.1934	0.1305	0.1486
F_{57}	6.0009	0.1683	0.1115	0.1226
F_{58}	5.7174	0.1651	0.1031	0.1163

Table A.2: Quality of I_1 filtering

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
NOISED	7.6815	0.0963	0.0634	0.0759
F_1	12.5867	0.0548	0.0294	**0.0373**
F_2	**12.9901**	**0.0523**	0.0320	0.0394
F_3	9.1698	0.0811	0.0568	0.0651
F_4	12.0640	0.0582	0.0357	0.0439
F_5	9.9601	0.0741	0.0376	0.0494
F_6	8.4087	0.0886	0.0622	0.0712
F_7	9.8504	0.0751	**0.0268**	0.0424
F_8	11.9133	0.0592	0.0294	0.0388
F_9	11.2835	0.0636	0.0426	0.0504
F_{10}	11.2391	0.0639	0.0327	0.0426
F_{11}	10.0693	0.0732	0.0362	0.0483
F_{12}	11.2985	0.0635	0.0434	0.0505
F_{13}	**13.2273**	**0.0509**	0.0274	**0.0353**
F_{14}	**13.3142**	**0.0504**	0.0310	**0.0382**
F_{15}	11.0396	0.0654	0.0335	0.0436
F_{16}	7.8699	0.0943	0.0413	0.0596
F_{17}	9.8469	0.0751	**0.0267**	0.0424
F_{18}	11.9163	0.0592	0.0293	0.0388
F_{19}	11.3674	0.0630	0.0423	0.0499
F_{20}	11.1963	0.0643	0.0326	0.0426
F_{21}	9.6655	0.0767	0.0363	0.0503
F_{22}	11.3154	0.0634	0.0433	0.0504
F_{23}	9.2981	0.0801	0.0281	0.0452
F_{24}	**13.3106**	**0.0504**	0.0305	**0.0377**
F_{25}	11.0396	0.0654	0.0335	0.0436
F_{26}	7.8699	0.0943	0.0413	0.0596
F_{27}	11.2810	0.0637	**0.0265**	0.0395
F_{28}	**12.7513**	**0.0537**	0.0292	**0.0376**
F_{29}	11.4386	0.0625	0.0419	0.0495
F_{30}	12.1784	0.0574	0.0324	0.0411
F_{31}	9.6571	0.0767	0.0366	0.0504
F_{32}	11.3732	0.0630	0.0435	0.0504
F_{33}	11.6650	0.0609	**0.0273**	0.0392
F_{34}	11.9014	0.0593	0.0308	0.0414
F_{35}	10.8771	0.0667	0.0433	0.0520
F_{36}	10.6706	0.0683	0.0456	0.0542
F_{37}	10.7056	0.0681	**0.0264**	0.0406
F_{38}	12.1695	0.0575	0.0291	0.0390
F_{39}	11.3644	0.0630	0.0414	0.0497
F_{40}	11.5629	0.0616	0.0326	0.0426
F_{41}	10.0128	0.0736	0.0361	0.0487
F_{42}	11.1812	0.0644	0.0422	0.0506
F_{43}	11.6092	0.0613	0.0288	0.0404

continued on next page...

Table A.3 Quality of I_2 filtering

...continued from previous page

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
F_{44}	11.3768	0.0629	0.0387	0.0481
F_{45}	9.6081	0.0772	0.0363	0.0504
F_{46}	11.3928	0.0628	0.0376	0.0476
F_{47}	5.1150	0.2123	0.1489	0.1684
F_{48}	5.4975	0.1818	0.1138	0.1344
F_{49}	5.2622	0.1762	0.1021	0.1251
F_{50}	5.1150	0.2123	0.1489	0.1684
F_{51}	5.4975	0.1818	0.1138	0.1344
F_{52}	5.2622	0.1762	0.1021	0.1251
F_{53}	5.1150	0.2123	0.1489	0.1684
F_{54}	5.4975	0.1818	0.1138	0.1344
F_{55}	5.2622	0.1762	0.1021	0.1251
F_{56}	5.1150	0.2123	0.1489	0.1684
F_{57}	5.4975	0.1818	0.1138	0.1344
F_{58}	5.2622	0.1762	0.1021	0.1251

Table A.3: Quality of I_2 filtering

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
NOISED	6.8344	0.1062	0.0528	0.0755
F_1	**13.1778**	**0.0512**	**0.0234**	**0.0324**
F_2	**13.5837**	**0.0488**	0.0256	**0.0343**
F_3	8.7979	0.0847	0.0469	0.0627
F_4	12.6414	0.0544	0.0284	0.0383
F_5	10.1016	0.0729	0.0326	0.0462
F_6	8.0242	0.0926	0.0516	0.0687
F_7	10.0180	0.0737	0.0249	0.0410
F_8	11.8005	0.0600	0.0270	0.0381
F_9	10.9207	0.0663	0.0369	0.0491
F_{10}	11.2031	0.0642	0.0299	0.0415
F_{11}	10.0696	0.0732	0.0325	0.0463
F_{12}	10.9575	0.0661	0.0377	0.0490
F_{13}	**13.4858**	**0.0494**	**0.0236**	**0.0324**
F_{14}	**13.7949**	**0.0477**	0.0258	**0.0342**
F_{15}	10.9743	0.0659	0.0309	0.0427
F_{16}	7.3107	0.1005	0.0450	0.0662
F_{17}	10.0156	0.0737	**0.0249**	0.0410
F_{18}	11.8141	0.0599	0.0270	0.0380
F_{19}	10.9775	0.0659	0.0369	0.0489
F_{20}	11.1600	0.0645	0.0296	0.0415
F_{21}	9.4995	0.0781	0.0356	0.0509
F_{22}	10.9836	0.0659	0.0375	0.0489
F_{23}	9.3679	0.0795	0.0270	0.0445
F_{24}	**13.4457**	**0.0496**	0.0272	**0.0356**
F_{25}	10.9743	0.0659	0.0309	0.0427
F_{26}	7.3107	0.1005	0.0450	0.0662
F_{27}	11.2950	0.0636	**0.0247**	0.0383
F_{28}	12.6323	0.0545	0.0269	0.0367
F_{29}	11.0100	0.0657	0.0367	0.0486
F_{30}	12.1138	0.0578	0.0297	0.0399
F_{31}	9.5030	0.0781	0.0359	0.0510
F_{32}	11.0303	0.0655	0.0373	0.0487
F_{33}	11.9196	0.0592	**0.0242**	0.0366
F_{34}	11.8913	0.0594	0.0286	0.0403
F_{35}	10.3046	0.0712	0.0379	0.0520
F_{36}	10.1265	0.0727	0.0393	0.0535
F_{37}	10.7804	0.0675	0.0257	0.0401
F_{38}	12.0044	0.0586	0.0278	0.0389
F_{39}	10.7935	0.0673	0.0365	0.0494
F_{40}	11.4191	0.0626	0.0315	0.0427
F_{41}	9.9503	0.0742	0.0339	0.0478
F_{42}	10.6550	0.0684	0.0365	0.0500
F_{43}	11.3384	0.0632	0.0294	0.0420

continued on next page...

Table A.4 Quality of I_3 filtering

...continued from previous page

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
F_{44}	10.7349	0.0678	0.0366	0.0495
F_{45}	9.3708	0.0793	0.0364	0.0519
F_{46}	10.8489	0.0669	0.0344	0.0483
F_{47}	4.5321	0.2271	0.1563	0.1800
F_{48}	5.0128	0.1914	0.1138	0.1405
F_{49}	4.8536	0.1842	0.0989	0.1294
F_{50}	4.5321	0.2271	0.1563	0.1800
F_{51}	5.0128	0.1914	0.1138	0.1405
F_{52}	4.8536	0.1842	0.0989	0.1294
F_{53}	4.5321	0.2271	0.1563	0.1800
F_{54}	5.0128	0.1914	0.1138	0.1405
F_{55}	4.8536	0.1842	0.0989	0.1294
F_{56}	4.5321	0.2271	0.1563	0.1800
F_{57}	5.0128	0.1914	0.1138	0.1405
F_{58}	4.8536	0.1842	0.0989	0.1294

Table A.4: Quality of I_3 filtering

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
NOISED	3.6392	0.1535	0.0375	0.0829
F_1	**13.0715**	**0.0518**	**0.0167**	**0.0268**
F_2	**13.4086**	**0.0498**	**0.0186**	**0.0286**
F_3	6.5533	0.1097	0.0338	0.0640
F_4	**12.0591**	**0.0582**	0.0205	**0.0322**
F_5	9.3719	0.0793	0.0249	0.0434
F_6	5.6627	0.1216	0.0367	0.0708
F_7	9.7470	0.0760	0.0202	0.0393
F_8	10.7174	0.0679	0.0219	0.0383
F_9	8.4723	0.0880	0.0280	0.0517
F_{10}	10.1848	0.0722	0.0247	0.0418
F_{11}	9.1679	0.0812	0.0260	0.0457
F_{12}	8.5782	0.0869	0.0285	0.0516
F_{13}	**12.3614**	**0.0563**	**0.0177**	**0.0296**
F_{14}	**13.4357**	**0.0497**	**0.0190**	**0.0293**
F_{15}	9.9678	0.0740	0.0253	0.0428
F_{16}	4.4724	0.1397	0.0527	0.0926
F_{17}	9.7450	0.0760	0.0204	0.0393
F_{18}	10.7446	0.0677	0.0220	0.0382
F_{19}	8.4928	0.0878	0.0279	0.0516
F_{20}	10.1184	0.0728	0.0243	0.0417
F_{21}	7.6838	0.0964	0.0371	0.0621
F_{22}	8.5845	0.0868	0.0284	0.0515
F_{23}	9.2180	0.0808	0.0221	0.0428
F_{24}	11.9378	0.0590	0.0210	0.0343
F_{25}	9.9678	0.0740	0.0253	0.0428
F_{26}	4.4724	0.1397	0.0527	0.0926
F_{27}	10.7384	0.0678	0.0204	0.0370
F_{28}	11.2918	0.0636	0.0219	0.0369
F_{29}	8.4978	0.0877	0.0280	0.0516
F_{30}	10.9836	0.0659	0.0252	0.0402
F_{31}	7.6750	0.0965	0.0376	0.0623
F_{32}	8.5630	0.0871	0.0283	0.0513
F_{33}	11.8120	0.0600	**0.0191**	0.0333
F_{34}	10.6745	0.0682	0.0246	0.0422
F_{35}	7.7606	0.0955	0.0301	0.0566
F_{36}	7.3611	0.1000	0.0292	0.0579
F_{37}	10.2363	0.0718	0.0235	0.0413
F_{38}	10.7407	0.0677	0.0255	0.0420
F_{39}	8.1507	0.0913	0.0289	0.0538
F_{40}	9.8164	0.0754	0.0322	0.0493
F_{41}	8.6723	0.0859	0.0328	0.0530
F_{42}	8.0208	0.0926	0.0280	0.0543
F_{43}	9.4314	0.0788	0.0338	0.0525

continued on next page...

Table A.5 Quality of I_4 filtering

...continued from previous page

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
F_{44}	7.9390	0.0937	0.0360	0.0597
F_{45}	7.4764	0.0987	0.0386	0.0643
F_{46}	8.1228	0.0917	0.0333	0.0572
F_{47}	2.2619	0.2736	0.1614	0.2062
F_{48}	2.9501	0.2260	0.1001	0.1540
F_{49}	3.1232	0.2127	0.0848	0.1387
F_{50}	2.2619	0.2736	0.1614	0.2062
F_{51}	2.9501	0.2260	0.1001	0.1540
F_{52}	3.1232	0.2127	0.0848	0.1387
F_{53}	2.2619	0.2736	0.1614	0.2062
F_{54}	2.9501	0.2260	0.1001	0.1540
F_{55}	3.1232	0.2127	0.0848	0.1387
F_{56}	2.2619	0.2736	0.1614	0.2062
F_{57}	2.9501	0.2260	0.1001	0.1540
F_{58}	3.1232	0.2127	0.0848	0.1387

Table A.5: Quality of I_4 filtering

A.3 Artificial MRI Image

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
NOISED	10.8381	0.0671	0.0225	0.0416
F_1	14.6766	**0.0414**	0.0075	**0.0238**
F_2	**15.0899**	**0.0400**	0.0124	**0.0247**
F_3	11.5335	0.0619	0.0175	0.0387
F_4	14.5742	0.0425	0.0124	**0.0261**
F_5	14.7462	0.0421	0.0175	0.0274
F_6	11.3519	0.0632	0.0196	0.0397
F_7	14.3238	0.0452	0.0237	0.0315
F_8	14.8567	0.0430	0.0225	0.0303
F_9	13.2251	0.0509	0.0184	0.0326
F_{10}	14.6798	0.0439	0.0225	0.0305
F_{11}	14.5942	0.0434	0.0206	0.0291
F_{12}	13.9644	0.0468	0.0187	0.0307
F_{13}	**15.2065**	**0.0394**	0.0138	**0.0248**
F_{14}	**15.2262**	**0.0395**	0.0138	**0.0252**
F_{15}	14.7936	0.0430	0.0216	0.0297
F_{16}	12.4795	0.0612	0.0387	0.0471
F_{17}	14.3308	0.0452	0.0237	0.0315
F_{18}	14.8577	0.0430	0.0225	0.0303
F_{19}	13.2823	0.0506	0.0187	0.0325
F_{20}	14.7158	0.0437	0.0226	0.0305
F_{21}	14.3982	0.0470	0.0287	0.0356
F_{22}	13.9836	0.0467	0.0187	0.0307
F_{23}	14.1368	0.0458	0.0237	0.0315
F_{24}	14.9529	0.0416	0.0187	0.0278
F_{25}	14.7936	0.0430	0.0216	0.0297
F_{26}	12.4795	0.0612	0.0387	0.0471
F_{27}	14.5299	0.0438	0.0206	0.0299
F_{28}	14.8966	0.0426	0.0212	0.0297
F_{29}	13.3458	0.0503	0.0187	0.0325
F_{30}	14.8998	0.0426	0.0212	0.0296
F_{31}	14.5689	0.0455	0.0269	0.0340
F_{32}	14.0081	0.0464	0.0175	0.0303
F_{33}	14.7934	0.0416	0.0169	0.0271
F_{34}	**15.0785**	**0.0414**	0.0204	0.0289
F_{35}	13.2629	0.0514	0.0206	0.0340
F_{36}	13.2623	0.0511	0.0200	0.0336
F_{37}	14.4287	0.0452	0.0249	0.0325
F_{38}	**15.0064**	0.0429	0.0240	0.0314
F_{39}	13.4126	0.0505	0.0205	0.0335
F_{40}	14.8618	0.0440	0.0244	0.0323
F_{41}	14.7690	0.0436	0.0225	0.0313

continued on next page...

Table A.6 Quality of I_5 filtering

...continued from previous page

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
F_{42}	13.8083	0.0485	0.0208	0.0328
F_{43}	14.7205	0.0458	0.0280	0.0351
F_{44}	13.8259	0.0498	0.0249	0.0356
F_{45}	14.3276	0.0475	0.0290	0.0359
F_{46}	14.1702	0.0482	0.0252	0.0354
F_{47}	8.7311	0.0928	**0.0000**	0.0449
F_{48}	10.2439	0.0771	**0.0000**	0.0370
F_{49}	10.2729	0.0763	**0.0000**	0.0362
F_{50}	8.7311	0.0928	**0.0000**	0.0449
F_{51}	10.2439	0.0771	**0.0000**	0.0370
F_{52}	10.2729	0.0763	**0.0000**	0.0362
F_{53}	8.7311	0.0928	**0.0000**	0.0449
F_{54}	10.2439	0.0771	**0.0000**	0.0370
F_{55}	10.2729	0.0763	**0.0000**	0.0362
F_{56}	8.7311	0.0928	**0.0000**	0.0449
F_{57}	10.2439	0.0771	**0.0000**	0.0370
F_{58}	10.2729	0.0763	**0.0000**	0.0362

Table A.6: Quality of I_5 filtering

A.4 Artificial SPECT Images

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
NOISED	11.9460	0.0544	0.0329	0.0408
F_1	18.8417	0.0244	**0.0151**	0.0184
F_2	17.9633	0.0271	0.0167	0.0205
F_3	13.4901	0.0455	0.0302	0.0349
F_4	16.8804	0.0307	0.0189	0.0231
F_5	16.6475	0.0316	0.0199	0.0242
F_6	12.7326	0.0496	0.0322	0.0382
F_7	**19.8476**	**0.0223**	0.0154	0.0179
F_8	19.0561	0.0244	0.0165	0.0193
F_9	15.6935	0.0354	0.0229	0.0273
F_{10}	17.9421	0.0276	0.0181	0.0216
F_{11}	16.9263	0.0308	0.0199	0.0237
F_{12}	15.9762	0.0343	0.0229	0.0267
F_{13}	19.5063	0.0227	**0.0146**	**0.0176**
F_{14}	18.3204	0.0260	0.0165	0.0199
F_{15}	17.7487	0.0281	0.0183	0.0219
F_{16}	16.2255	0.0344	0.0240	0.0277
F_{17}	**19.8730**	**0.0223**	**0.0154**	**0.0178**
F_{18}	19.0622	0.0244	0.0165	0.0193
F_{19}	15.7825	0.0350	0.0227	0.0270
F_{20}	17.9654	0.0275	0.0181	0.0216
F_{21}	17.2020	0.0302	0.0205	0.0241
F_{22}	15.9978	0.0342	0.0230	0.0267
F_{23}	19.5799	0.0229	0.0157	0.0183
F_{24}	18.5224	0.0256	0.0165	0.0198
F_{25}	17.7487	0.0281	0.0183	0.0219
F_{26}	16.2255	0.0344	0.0240	0.0277
F_{27}	**19.9477**	**0.0220**	**0.0148**	**0.0175**
F_{28}	19.1047	0.0242	0.0162	0.0191
F_{29}	15.8841	0.0347	0.0226	0.0268
F_{30}	18.0402	0.0272	0.0178	0.0213
F_{31}	17.1880	0.0301	0.0200	0.0239
F_{32}	15.9701	0.0343	0.0231	0.0267
F_{33}	**19.7647**	**0.0222**	**0.0144**	**0.0174**
F_{34}	18.8497	0.0248	0.0163	0.0196
F_{35}	15.6428	0.0357	0.0226	0.0277
F_{36}	15.1068	0.0379	0.0237	0.0292
F_{37}	**20.0078**	**0.0220**	0.0154	**0.0178**
F_{38}	19.2874	0.0239	0.0164	0.0191
F_{39}	15.9170	0.0347	0.0220	0.0269
F_{40}	18.1652	0.0271	0.0181	0.0215
F_{41}	17.1799	0.0300	0.0195	0.0236

continued on next page...

Table A.7 Quality of I_6 filtering

...continued from previous page

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
F_{42}	16.1003	0.0339	0.0216	0.0264
F_{43}	19.3775	0.0240	0.0171	0.0195
F_{44}	16.6471	0.0322	0.0209	0.0254
F_{45}	17.2340	0.0302	0.0204	0.0240
F_{46}	17.0296	0.0309	0.0198	0.0245
F_{47}	12.6170	0.0817	0.0644	0.0647
F_{48}	13.5732	0.0673	0.0455	0.0512
F_{49}	13.5878	0.0636	0.0373	0.0474
F_{50}	12.6170	0.0817	0.0644	0.0647
F_{51}	13.5732	0.0673	0.0455	0.0512
F_{52}	13.5878	0.0636	0.0373	0.0474
F_{53}	12.6170	0.0817	0.0644	0.0647
F_{54}	13.5732	0.0673	0.0455	0.0512
F_{55}	13.5878	0.0636	0.0373	0.0474
F_{56}	12.6170	0.0817	0.0644	0.0647
F_{57}	13.5732	0.0673	0.0455	0.0512
F_{58}	13.5878	0.0636	0.0373	0.0474

Table A.7: Quality of I_6 filtering

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
NOISED	13.2184	0.0525	0.0302	0.0379
F_1	20.3286	0.0230	**0.0128**	**0.0166**
F_2	19.3497	0.0258	**0.0145**	0.0187
F_3	14.7978	0.0438	0.0266	0.0323
F_4	18.2267	0.0294	0.0165	0.0211
F_5	17.9663	0.0305	0.0182	0.0227
F_6	14.0345	0.0478	0.0295	0.0353
F_7	**21.0479**	0.0222	0.0157	0.0178
F_8	20.1996	0.0243	0.0165	0.0192
F_9	16.9683	0.0343	0.0209	0.0256
F_{10}	19.1235	0.0273	0.0178	0.0211
F_{11}	18.1803	0.0300	0.0189	0.0226
F_{12}	17.2700	0.0331	0.0209	0.0251
F_{13}	20.9386	**0.0216**	**0.0133**	**0.0163**
F_{14}	19.6992	0.0249	**0.0147**	0.0185
F_{15}	18.9881	0.0275	0.0178	0.0212
F_{16}	17.3415	0.0348	0.0250	0.0282
F_{17}	**21.0647**	**0.0222**	0.0157	**0.0178**
F_{18}	20.2052	0.0243	0.0165	0.0192
F_{19}	17.0548	0.0340	0.0208	0.0254
F_{20}	19.1366	0.0272	0.0179	0.0211
F_{21}	18.4005	0.0300	0.0206	0.0240
F_{22}	17.2898	0.0331	0.0210	0.0251
F_{23}	20.8426	0.0225	0.0159	0.0181
F_{24}	19.8087	0.0248	0.0156	0.0190
F_{25}	18.9881	0.0275	0.0178	0.0212
F_{26}	17.3415	0.0348	0.0250	0.0282
F_{27}	**21.1813**	**0.0217**	0.0148	**0.0172**
F_{28}	20.2580	0.0241	0.0161	0.0189
F_{29}	17.1442	0.0337	0.0209	0.0253
F_{30}	19.2418	0.0268	0.0174	0.0207
F_{31}	18.4348	0.0296	0.0198	0.0235
F_{32}	17.2697	0.0331	0.0210	0.0250
F_{33}	**21.1496**	**0.0214**	**0.0136**	**0.0166**
F_{34}	20.0944	0.0243	0.0159	0.0191
F_{35}	16.8672	0.0349	0.0210	0.0265
F_{36}	16.3670	0.0368	0.0216	0.0276
F_{37}	**21.2009**	**0.0220**	0.0160	0.0180
F_{38}	20.4403	0.0239	0.0167	0.0193
F_{39}	17.1379	0.0339	0.0205	0.0258
F_{40}	19.3435	0.0269	0.0181	0.0214
F_{41}	18.4365	0.0294	0.0188	0.0229
F_{42}	17.3155	0.0332	0.0201	0.0253
F_{43}	20.4534	0.0244	0.0180	0.0202

continued on next page...

Table A.8 Quality of I_7 filtering

...continued from previous page

Filter	**SNR** [dB]	**MSE**	**MED**	**MAE**
F_{44}	17.8086	0.0319	0.0205	0.0251
F_{45}	18.4299	0.0299	0.0206	0.0239
F_{46}	18.1868	0.0307	0.0193	0.0243
F_{47}	13.3049	0.0741	0.0505	0.0536
F_{48}	14.4218	0.0614	0.0314	0.0426
F_{49}	14.5877	0.0578	0.0305	0.0393
F_{50}	13.3049	0.0741	0.0505	0.0536
F_{51}	14.4218	0.0614	0.0314	0.0426
F_{52}	14.5877	0.0578	0.0305	0.0393
F_{53}	13.3049	0.0741	0.0505	0.0536
F_{54}	14.4218	0.0614	0.0314	0.0426
F_{55}	14.5877	0.0578	0.0305	0.0393
F_{56}	13.3049	0.0741	0.0505	0.0536
F_{57}	14.4218	0.0614	0.0314	0.0426
F_{58}	14.5877	0.0578	0.0305	0.0393

Table A.8: Quality of I_7 filtering

Appendix B

Results of Filtering of Artificial MRI Image

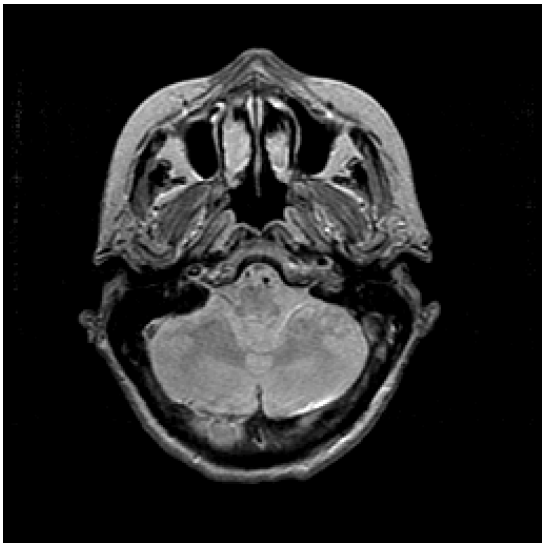


Figure B.1: Ideal MRI image

Figure B.2: Artificial MRI image I_5

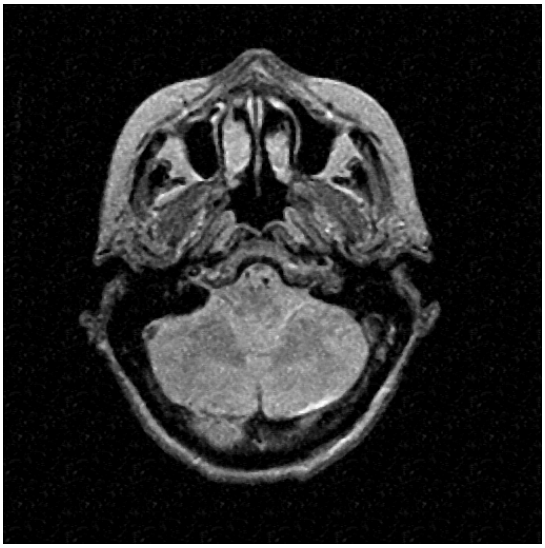


Figure B.3: I_5 filtered by F_{13}

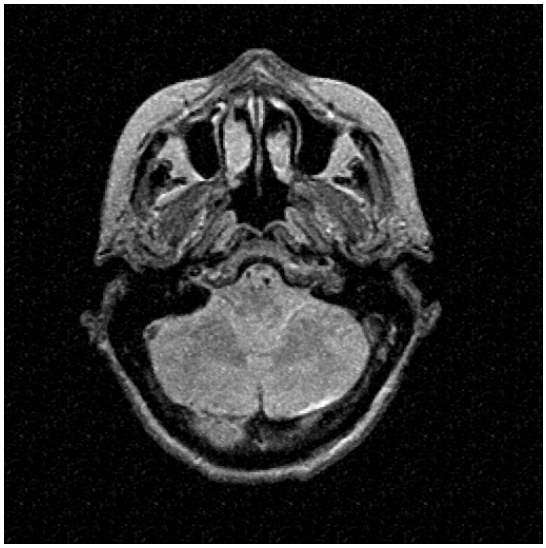


Figure B.4: I_5 filtered by F_{14}

Figure B.5: Artificial MRI image I_5

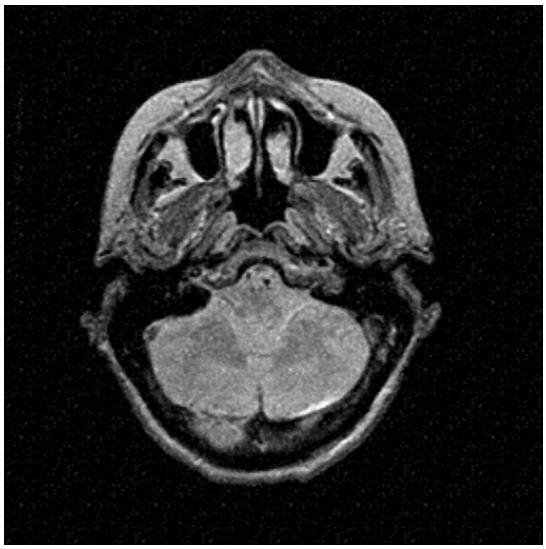


Figure B.6: I_5 filtered by MPFN

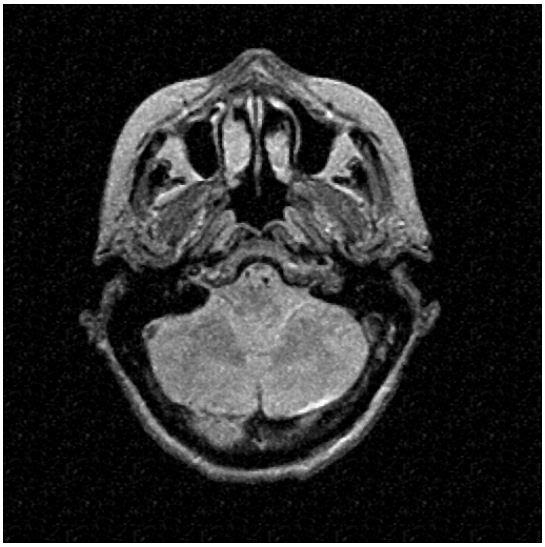


Figure B.7: I_5 filtered by MMFN

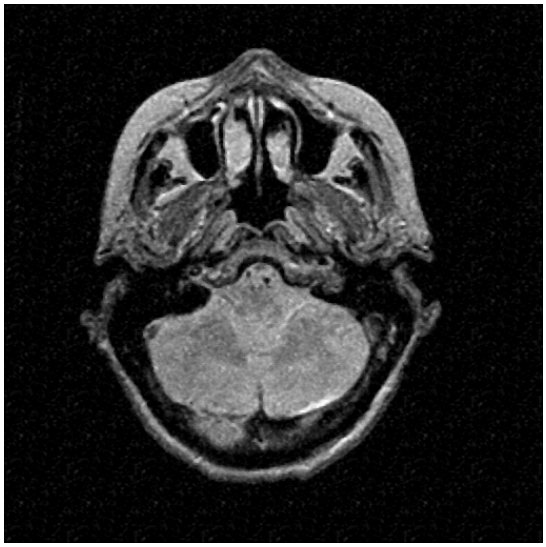


Figure B.8: I_5 filtered by MFLFN

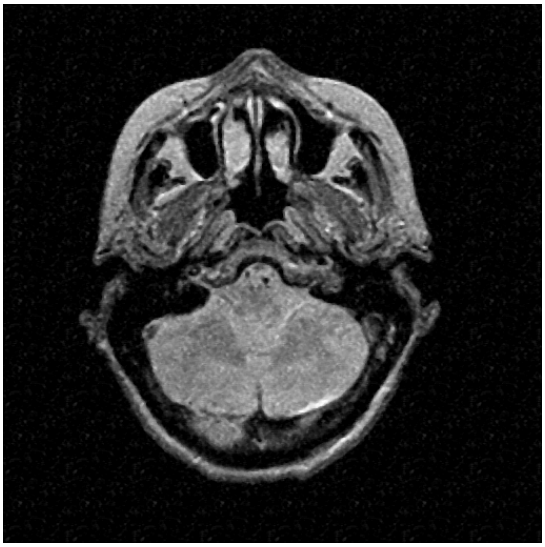


Figure B.9: I_5 filtered by CRNN

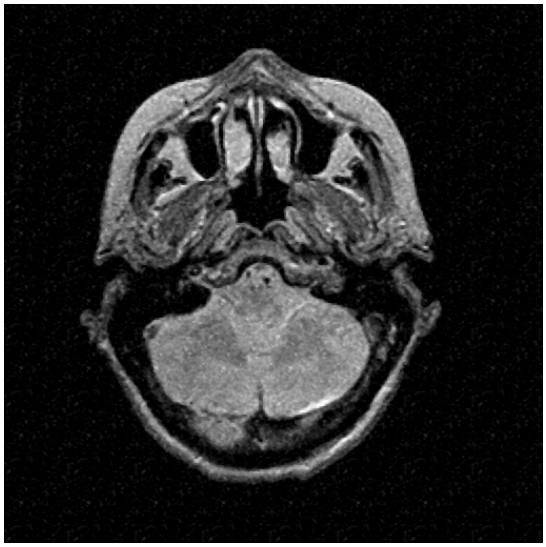


Figure B.10: I_5 filtered by DWNN

Appendix C

Results of Filtering of Real MRI Image (Human Brain)

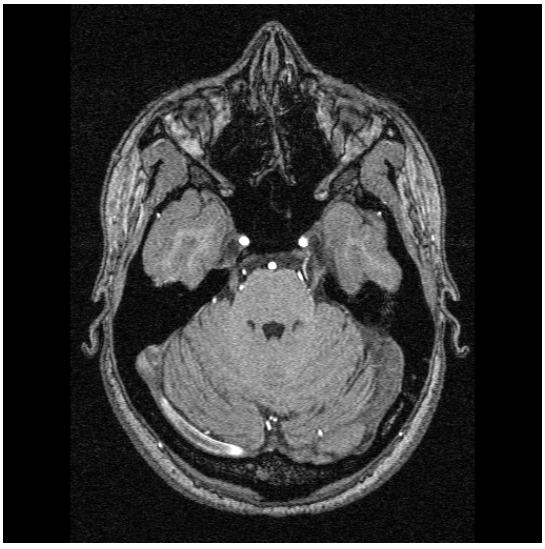


Figure C.1: Real MRI image RI_1

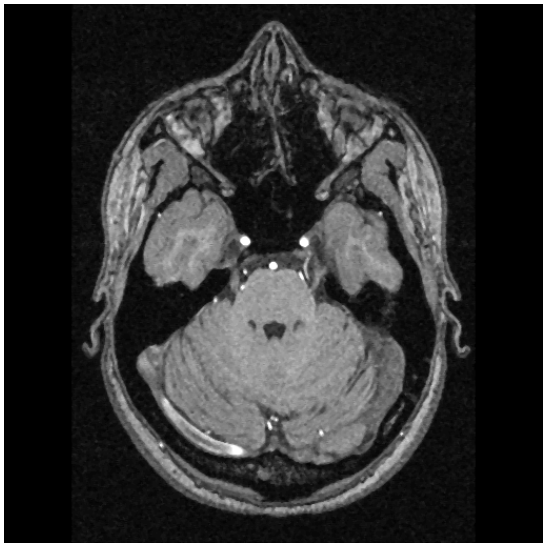


Figure C.2: RI_1 filtered by F_1

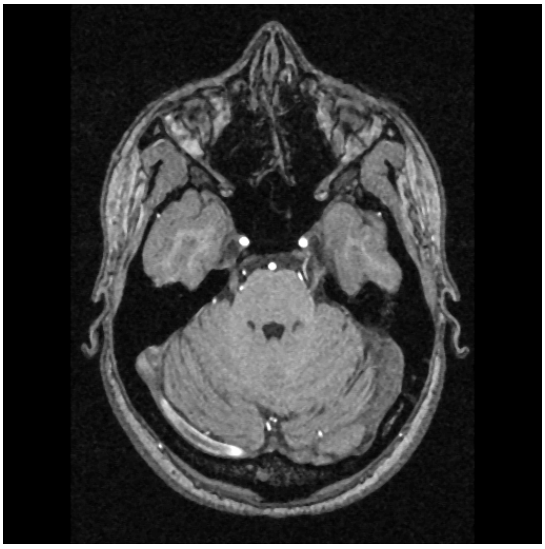


Figure C.3: RI_1 filtered by F_{13}

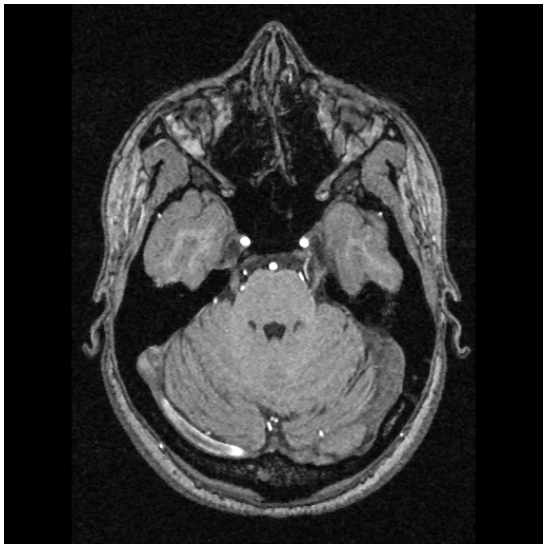


Figure C.4: RI_1 filtered by F_{14}

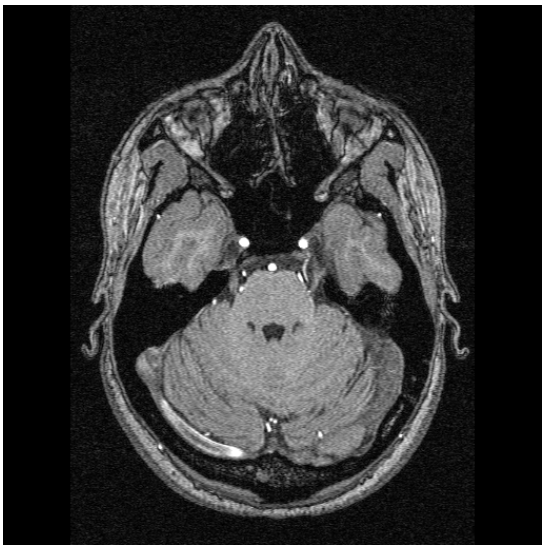


Figure C.5: Real MRI image RI_1

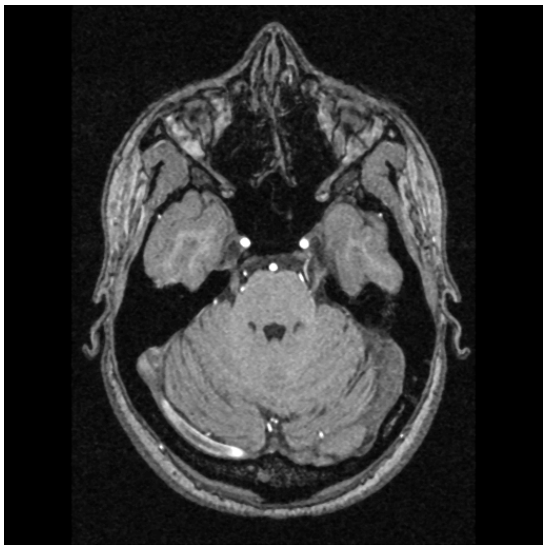


Figure C.6: RI_1 filtered by MPFN

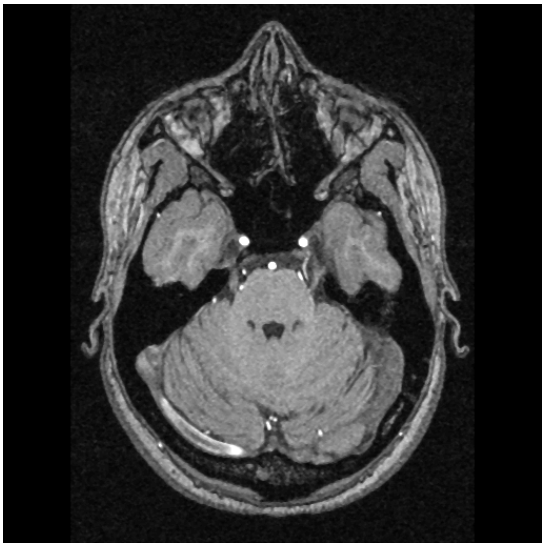


Figure C.7: RI_1 filtered by MMFN

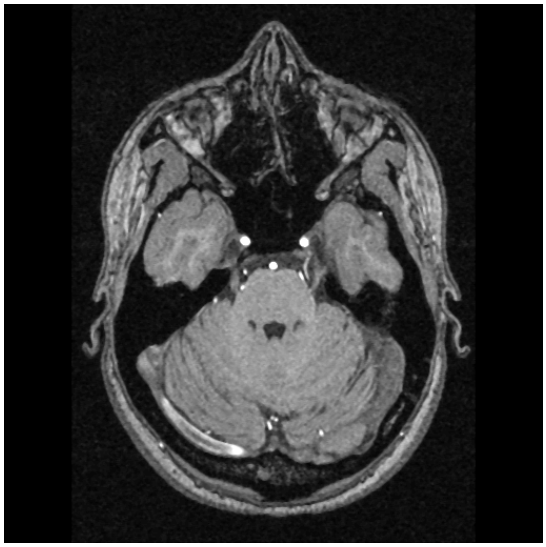


Figure C.8: RI_1 filtered by MFLFN

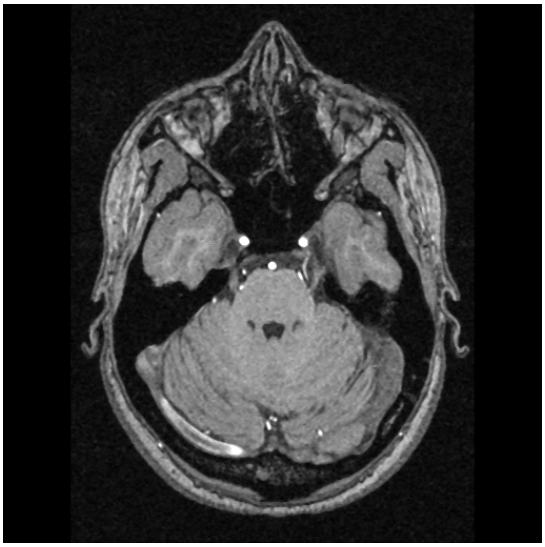


Figure C.9: RI_1 filtered by CRNN

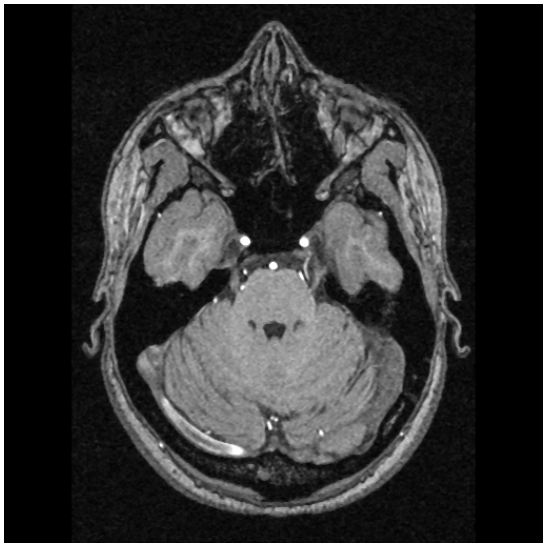


Figure C.10: RI_1 filtered by DWNN

Appendix D

Results of Filtering of Real MRI Image (Human Brain)

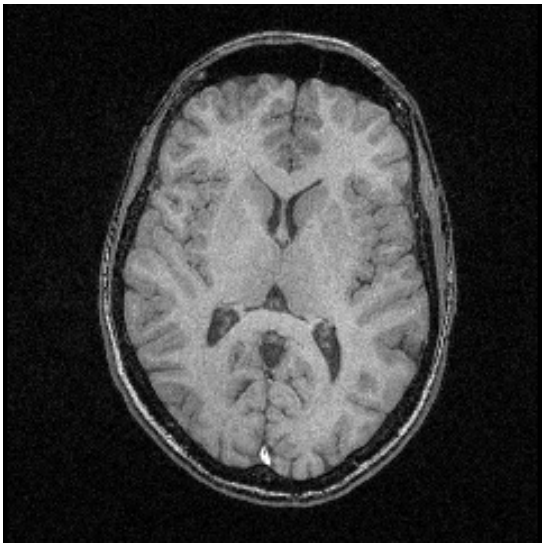


Figure D.1: Real MRI image RI_2

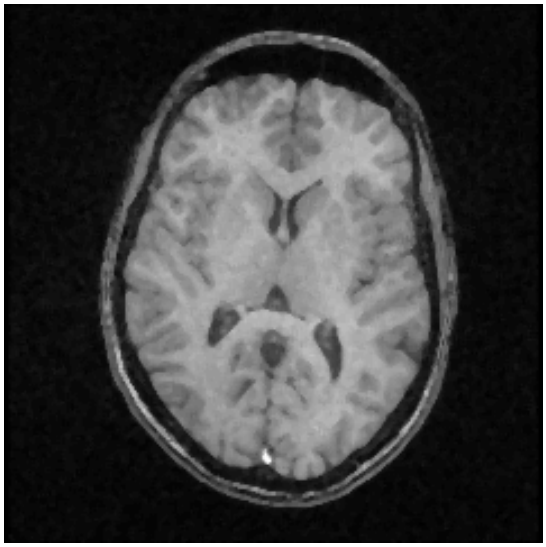


Figure D.2: RI_2 filtered by F_1

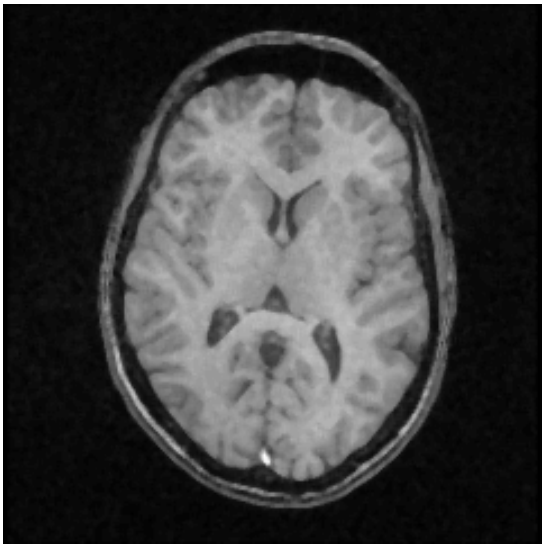


Figure D.3: RI_2 filtered by F_{13}

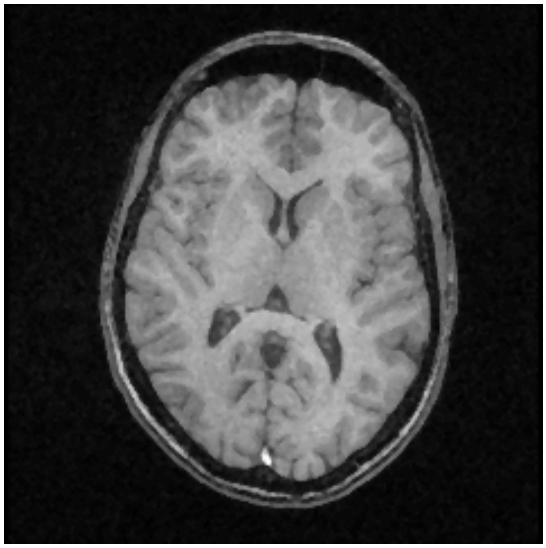


Figure D.4: RI_2 filtered by F_{14}

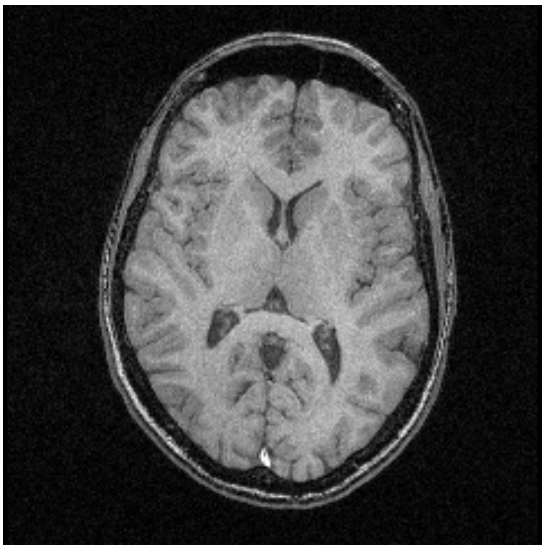


Figure D.5: Real MRI image RI_2

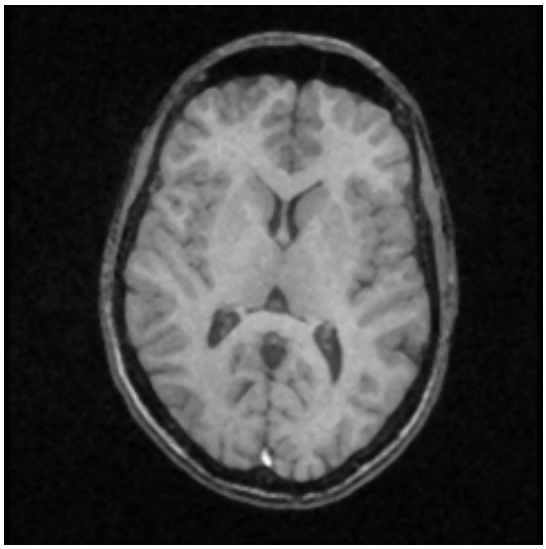


Figure D.6: RI_2 filtered by MPFN

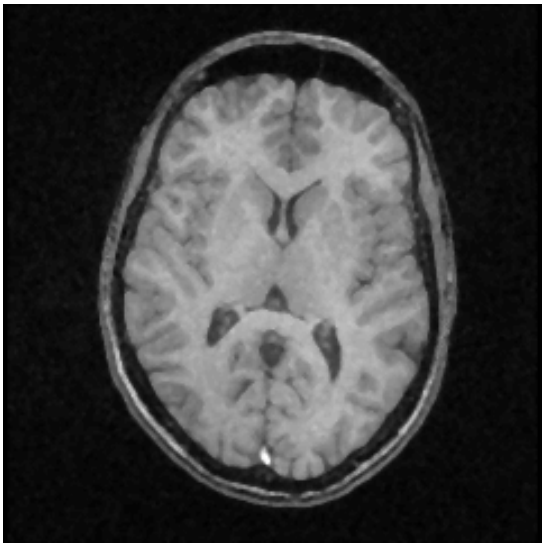


Figure D.7: RI_2 filtered by MMFN

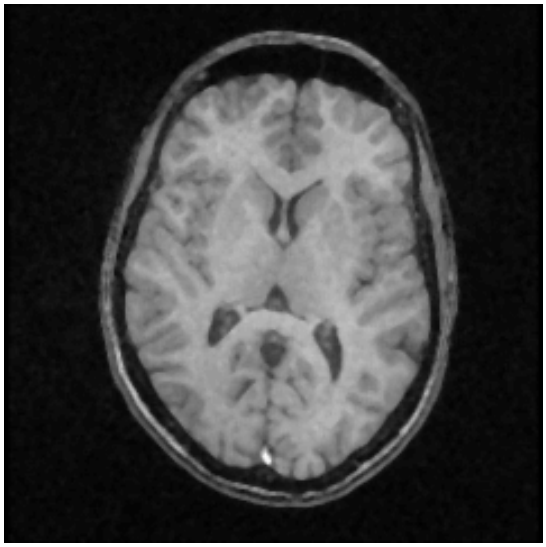


Figure D.8: RI_2 filtered by MFLFN

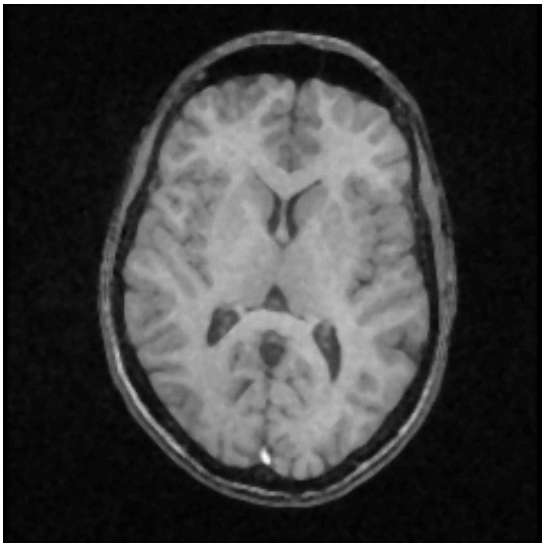


Figure D.9: RI_2 filtered by CRNN

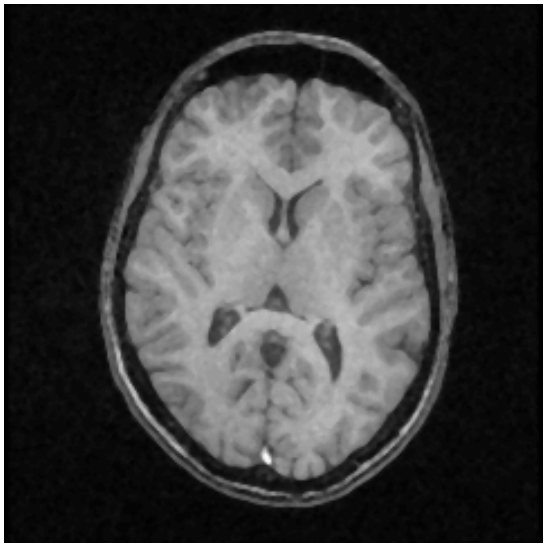


Figure D.10: RI_2 filtered by DWNN

Appendix E

Results of Filtering of Real MRI Image (Human Genu)

Figure E.1: Real MRI image RI_3

Figure E.2: RI_3 filtered by F_1

Figure E.3: RI_3 filtered by F_{13}

Figure E.4: RI_3 filtered by F_{14}

Figure E.5: Real MRI image RI_3

Figure E.6: RI_3 filtered by MPFN

Figure E.7: RI_3 filtered by MMFN

Figure E.8: RI_3 filtered by MFLFN

Figure E.9: RI_3 filtered by CRNN

Figure E.10: RI_3 filtered by DWNN

Appendix F

MATLAB Codes of FLF Networks

F.1 Modus Ponens Fuzzy Network

```
function Y = MPFNresp(F,f,w,wstar)
% MPFNresp - response of MPFN
% Y = MPFNresp(F,f,w,wstar)
% F ......... output filter matrices (m,n,H)
% f ......... selected filter indices (1,d), d<=H
% w,wstar ... weight vectors (1,H)
% Y ... output image

m = size(F,1);
n = size(F,2);
H = size(F,3);

Y = zeros(m,n);
for r = 1:m
    for c = 1:n
        lowerE = 0;
        upperE = 0;
        for i = 1:length(f)
            lowerE = max([lowerE, max([w(i)+F(r,c,f(i))-1,0])]);
            upperE = max([upperE, max([wstar(i)-F(r,c,f(i)),0])]);
        end
        upperE = 1-upperE;
        Y(r,c) = (lowerE+upperE)/2;
    end
end
```

```
function [ssq,w,wstar,ycalc]=MPFNopt(Fpatt,ypatt)
% MPFNopt - optimization of MPFN weights using pattern set
% [ssq,w,wstar,ycalc]=MPFNopt(Fpatt,ypatt)
% Fpatt ... matrix of M patterns (M,H)
% ypatt ... vector of M ideal outputs (M,1)
% ssq ..... sum of squares
% w, wstar ..... MPFN weights (1,H)
% ycalc ........ vector of MPFN output values (M,1)
global FFF yyy H M
FFF = Fpatt;
yyy = ypatt;
[M,H] = size(Fpatt);
lb=zeros(2*H,1);
ub=ones(2*H,1);
w0 = ub;
[w_all,ssq]=fmincon('MPFNobj',w0,[],[],[],[],lb,ub);
w = w_all(1:H);
w = w';
wstar = w_all(H+1:2*H);
wstar = wstar';
ycalc = zeros(M,1);
lowerE = 0;
upperE = 0;
for i = 1:M
    lowerE = max([lowerE, max([w+Fpatt(i,:)-1,0])]);
    upperE = max([upperE, max([wstar-Fpatt(i,:),0])]);
    ycalc(i) = (lowerE+1-upperE)/2;
end
delta=ypatt-ycalc;
ssq=delta'*delta;

function ssq=MPFNobj(w_all)
% MPFNobj
global FFF yyy H M
w = w_all(1:H);
w = w';
wstar = w_all(H+1:2*H);
wstar = wstar';
ycalc = zeros(M,1);
lowerE = 0;
upperE = 0;
for i = 1:M
    lowerE = max([lowerE, max([w+FFF(i,:)-1,0])]);
    upperE = max([upperE, max([wstar-FFF(i,:),0])]);
    ycalc(i) = (lowerE+1-upperE)/2;
end
delta=yyy-ycalc;
ssq=delta'*delta;
```

```
function [w,wstar]=MPFNlearn(Fpatt,ypatt)
% MPFNlearn - MPFN learning from pattern set
% [w,wstar]=MPFNlearn(Fpatt,ypatt)
% Fpatt ... matrix of M patterns (M,H)
% ypatt ... vector of M ideal outputs
% w, wstar ..... MPFN weights (1,H)

[M,H] = size(Fpatt);

w = ones(1,H);
wstar = w;
for i = 1:M
   for j = 1:H
      w(j) = min([w(j), min([1-Fpatt(i,j)+ypatt(i),1])]);
      wstar(j) = min([wstar(j), min([1-Fpatt(i,j)+1-ypatt(i),1])]);
   end
end
```

F.2 MATLAB Realization of MMFN

```
function Y = MMFNresp(F,f,a)
% MMFNresp - output of MMFN
% Y = MMFNresp(F,f,a)
% F ... matrix of filter outputs
% f ... selected filter indices
% a ... selector vector (1x2^n); it contains indices of all subsets
% Y ... output image matrix

m = size(F,1);
n = size(F,2);
Y = zeros(m,n);

d = length(f);
N=2^d; % size of 'a'
X = zeros(d,1);

for p1 = 1:m % each row of 'Y'
    for p2 = 1:n % each column of 'Y'
        for iii = 1:d
            X(iii) = F(p1,p2,f(iii));
        end
        for i=1:N
            if a(i)
                group = 1;
                code = i-1;
                j = 1;
                while code
                    if mod(code,2) & X(j)<group
                        group = X(j);
                    end
                    j = j+1;
                    code = floor(code/2);
                end
                if group>Y(p1,p2)
                    Y(p1,p2) = group;
                end
            end
        end
    end
end
```

```
function [q,aopt]=MMFNopt(Xpatt,ypatt,fmax,fname,hmax,hname,maxne)
% MMFNopt
% [q,aopt]=MMFNopt(Xpatt,ypatt,fmax,fname,hmax,hname,maxne);
% q ... quality
% aopt ... optimum solution (1x2^n)
% Xpatt ... pattern input (mxn)
% ypatt ... pattern output (mx1)
% fmax ... maximum acceptable SSQ
% fname .. name of quality function
% hmax ... maximum value of complexity
% hname .. name of complexity function
% maxne ... maximum number of evaluations

global X y Fname Fmax Hname Hmax

X = Xpatt;
y = ypatt;
Fname = fname;
fopt = 0; % minimum value
Fmax = fmax;
Hname = hname;
Hmax = hmax;

[m,n] = size(X);
N = 2^n;
l = zeros(1,N); u = ones(1,N);
g = ones(1,N)*0.3;
T = 0.3;
H = 100;
r = 6;
fobj = 'MMFNnet';
[aopt,q,ne] = Parallel3R(l,u,fobj,fopt,H,r,T,g,maxne);
```

```
function q = MMFNnet(a)
% MMFNnet
global X y Fmax Fname Hmax Hname
ycalc = y;  [m,n] = size(X);  N=2^n;
for k=1:m
    ycalc(k) = 0;
    for i=1:N
        if a(i)
            group = 1;
            code = i-1;
            j = 1;
            while code
                if mod(code,2) & X(k,j) < group
                    group = X(k,j);
                end
                j = j+1;
                code = floor(code/2);
            end
            if group > ycalc(k)
                ycalc(k) = group;
            end
        end
    end
end
f1 = feval(Fname,y,ycalc); % quality of function
c1 = feval(Hname,a);       % complexity
q = max([0, f1-Fmax]) + max([0, c1-Hmax]); % quality evaluation
% ------------------------------------------------------------
function qual = Quality(ypatt,ycalc)
qual = norm(abs(ypatt-ycalc));
% ------------------------------------------------------------
function s = Complexity(a)
s = sum(a); % number of subsets (nodes of third layer)
% ------------------------------------------------------------
function s = Complexity2(a)
l = length(a);
fl = sum(bitget(l-1,1:52));
s = zeros(1,fl);
for i=1:l
    if a(i)
        code = i-1;
        for j=1:fl
            if mod(code,2)
                s(j) = s(j) + 1;
            end
            code = floor(code/2);
        end
    end
end
s = max(s); % maximum occurrence of any variable in subsets
```

F.3 MATLAB Realization of MFLFN

```
function Y = MFLFNresp(F,f,k)
% MFLFNresp - response of MFLFN
% Y = MFLFNresp(F,f,k)
%% X ..... input image (corrupted)
% F ..... output filter matrices (m,n,H)
% k ..... similarity exponent (k=1,2,...,1000)
% Y ..... output image

m = size(F,1);
n = size(F,2);
H = size(F,3);

Y = zeros(m,n); % prepare

d = length(f);

for i = 1:m
    for j = 1:n
        for iii = 1:d
            fvec(iii) = F(i,j,f(iii));
        end
        r = median(Walsh_list(fvec)); % Hodges-Lehmann median
        g = max(k*(1-abs(fvec-median(fvec)))-k+1,0); % similarity
        g = g + 1 - max(g); % normalization
        m1 = fvec - r;
        m1(m1<0) = 0; % fvec\ominus r
        m1 = g + m1 - 1;
        m1(m1<0) = 0; % g\otimes m1
        m1 = max(m1);

        m2 = r - fvec;
        m2(m2<0) = 0; % fvec\ominus r
        m2 = g + m2 - 1;
        m2(m2<0) = 0; % g\otimes m2
        m2 = max(m2);

        Y(i,j) = max([min([r+m1,1])-m2,0]);
    end
end
```

F.4 MATLAB Realization of CRNN

```
function Y = CRNNresp(F,f,fref,key)
% CRNNresp - response of CRNN
% Y = CRNNresp(F,f,fref,key)
% F ..... output filter matrices (m,n,H)
% f ..... selected filter indices
% fref ... index of referential filter
% key .... order of f_low & f_max filter (1,2,...,length(f)/2))
% Y ..... output image

m = size(F,1);
n = size(F,2);
H = size(F,3);

Y = zeros(m,n);
f_ref=F(:,:,fref); % referential filter

d = length(f);
fs = f; % prepare

if key<1 | key>d
    error('Invalid order of f_low/f_max filter!');
end

for i = 1:m
    for j = 1:n
        for iii = 1:d
            fs(iii) = F(i,j,f(iii));
        end
        fs = sort(fs);
        fl = fs(key);
        fu = fs(end-key+1);
        Y(i,j) = min([fu, max([f_ref(i,j), fl])]);
    end
end
```

F.5 MATLAB Realization of DWNN

```
function Y = DWNNresp(F,f,w)
% DWNNresp - response of DWNN
% Y = DWNNresp(F,w)
% F ... output filter matrices (m,n,H)
% f ... selected filter indices
% w ... dyadic weight vector
% Y ..... output image
m = size(F,1);
n = size(F,2);
H = size(F,3);
Y = zeros(m,n);
for i = 1:length(f)
    Y = Y + F(:,:,f(i))*w(i);
end

function [ssq,m,ycalc,delta]=DWNNlearn(F,y,N)
% DWNNlearn - learning of Dyadic Weight Neural Network
% [ssq,m,ycalc,delta]=DWNNlearn(F,y,N);
% ssq ..... sum of squares of DWNN
% m ....... weight nominator vector of DWNN (n,1)
% ycalc .... output vector of DWNN (m,1)
% delta .... output error vector (m,1)
% F ....... pattern FLF matrix (m,n)
% y ....... vector of given output values (m,1)
% N ....... dyadic exponent as nonnegative integer
global FFF yyy
FFF=F;  yyy=y;  n=size(F,2);
% optimization:
Aeq=ones(1,n); beq=1; lb=zeros(n,1); w0=Aeq'/n;
[w,ssq]=fmincon('DWNNOBJ',w0,[],[],Aeq,beq,lb);
% outputs:
M=2^N;  m=round(w*M);  mdiff=sum(m)-M;
[wht,ind]=max(w);  m(ind)=m(ind)+mdiff;
ycalc=F*m/M;
delta=y-ycalc;
ssq=delta'*delta;

function ssq=DWNNOBJ(w)
% DWNNOBJ objective function for DWNN learning
% ssq=DWNNOBJ(w);
% ssq ..... sum of squares
% w ....... weight vector (n,1)
global FFF yyy
ycalc=FFF*w;
delta=yyy-ycalc;
ssq=delta'*delta;
```

Selected Paper Published