Department of Data Analysis and Mathematical Modelling
Faculty of Bioscience Engineering
Ghent University

Automated Image Analysis Using Gaussian-based Convolutional Kernels and Deep Convolutional Networks

Gang Wang

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor (Ph.D.) of Bioscience Engineering: Mathematical Modelling

Academic year 2019-2020

Supervisor:	Prof. dr. Bernard De Baets Department of Data Analysis and Mathematical Modelling, Ghent University, Belgium
Examination committee:	Prof. dr. ir. Jan Pieters (Chairman) Prof. dr. ir. Jan Verwaeren Prof. dr. Bernard De Baets Prof. dr. Gilbert Van Stappen Prof. dr. ir. Aleksandra Pizurica Prof. dr. Pedro Melo-Pinto
Dean:	Prof. dr. ir. Marc Van Meirvenne
Rector:	Prof. dr. ir. Rik Van de Walle

Gang Wang

Automated Image Analysis Using Gaussian-based Convolutional Kernels and Deep Convolutional Networks

Thesis submitted in partial fulfillment of the requirements for the degree of

Doctor (Ph.D.) of Bioscience Engineering:
Mathematical Modelling

Academic year 2019-2020

Dutch translation of the title:
Geautomatiseerde beeldverwerking met behulp van Gaussiaanse kernels en diepe convolutionele netwerken

Please refer to this work as follows:
Gang Wang (**2019**). Automated Image Analysis Using Gaussian-based Convolutional Kernels and Deep Convolutional Networks, Ph.D. Thesis, Faculty of Bioscience Engineering, Ghent University, Ghent, Belgium.

Acknowledgements

Ghent is still so charming in the evening. At such a tranquil moment, I enjoy the view of this city as if appreciating the painting *Starry Night Over the Rhône*. The wind sounds like a *Nocturne* that recalls numerous memories of the past four years. These memories remind me again that I am arriving at the terminus of my doctoral journey.

Having finalized my doctoral thesis, first and foremost, I would like to express millions of thanks to my supervisor, Prof. dr. Bernard De Baets, the director of the KERMIT research unit and the head of the Dept. of Data Analysis and Mathematical Modelling, Ghent University. My academic experience has been reshaped by Prof. De Baets since the first time we communicated by email. It is his strong support that helps me obtain the doctoral scholarship, finish the research proposal, win the academic awards, output the publications, accomplish the doctoral thesis, etc. Prof. De Baets is very good at supervising, always knowing how to *train* me by adequate *expert knowledge* with appropriate *hyperparameter settings* and effective *tricks*. I can always find comfort and encouragement from his place when suffering frustration, or receive his stimulation when hesitating. Prof. De Baets is a hard-working and self-disciplined professor. I can receive his emails in a very early morning or in a very deep night, on a weekend or on a holiday, from a flying plane or from his bedroom. Of course he never presses us to work too much, but he has performed as an exemplar that will always inspire me in my life, as what he commented on my first annual progress report: *The success will depend on the initiatives you take and your perseverance*. I also believe that his spirit is what contributes to his globally academic reputation. Prof. De Baets is a thoughtful supervisor having very high standards and very kind patience. I can always receive his detailed comments that might cover all the manuscript drafts. I will cherish many of his original comments, which represent numerous efforts he has made on my doctoral research. In daily life, Prof. De Baets is a very good friend full of enthusiasm and humor. He offers me a good introduction to the local cultural characteristics, and always encourages me to broaden my vision. With his smart jokes, our meetings are never boring. His piano show also surprised me. Absolutely, I am proud of being his doctoral student and looking forward to our cooperation in the future.

I would also like to thank Prof. Carlos Lopez-Molina from Universidad Publica de Navarra, Spain, for his insightful comments and continuous en-

couragement. A considerable portion of my doctoral research is finished under his enthusiastic help. I am also grateful to Prof. Gilbert Van Stappen, Christ Mahieu, Ruy Lopes-dos-Santos, Xiaoting Zheng and Biao Han from the Dept. of Animal Sciences and Aquatic Ecology, and Liselotte De Ligne from the Dept. of Environment, as well as my former colleague Guillermo Vidal-Diez de Ulzurrun for their collaborations during my doctoral research. My gratitude also goes to Prof. Irina Perfilieva (University of Ostrava), Prof. Paul Rosin (Cardiff University), Prof. Tony Lindeberg (KTH Royal Institute of Technology), Prof. Weichuan Zhang (Durham University) and Prof. Hui Kong (Nanjing University of Science and Technology) for their constructive suggestions and kind assistance. I am also very appreciative of the time and efforts that the jury members have dedicated to reading and evaluating my doctoral thesis. Their thoughtful comments and constructive suggestions have helped me improve this thesis significantly.

My thanks also go to my great office mates: Michiel, Peter, Marc, Hilde, Laura and Maxime. There is a very good atmosphere in our office. Your words can always comfort me when I have a difficult time. You have also offered me a lot of helpful tips for living in Ghent. Thank you for sharing so many nice working days with me.

Also, I want to thank my great colleagues in our department: Mengzi, Jim, Ruth, Timpe, Yuanyuan, Jose, Raul, Aisling, Wouter, Bac, Wenwen, Tim, Christina, Shuyun, Zengyuan, Steffie, Hang, Yexing, Alejandra, Bram, David, Gisele, Thomas, Jan Baetens, Willem, Jan Verwaeren, Daile, Lander, Dimitri, Jan Roels, Margot, Dimitrios, Ingmar, Min, Dorien, Tinne, Sophie, Juan, Michael, Yohannis, Daan, Jenna, Alemu, Chen Wang, Karel, Tiago, Sina, Chaim, Andreia, Niels, Giacomo, Stijn, Youri, Timothy, Gurmeet, Attilio, et al. I am really lucky to have you all as colleagues. We have shared lots of great moments together, such as team building days, social beer activities, dinner parties, watching football matches, indoor football matches, table game nights, sports days, new year banquets, academic conferences, and so on. I will also cherish some of our funny but private stories for a lifetime.

During my doctoral research, I have also made many other friends in Ghent: Guoxiang, Fan Li, Dan Sun, Jiawei, Shiyu, Shusheng, Longhui, Ting Liu, Lidong, Junfeng, Xia Zhao, Jiahao, Yunting, Lu Wang, Yubing, Changyuan, Honglin, Shaoguang, Meizhu, Mingsheng, Lingshan, Chunlian, Chen Yang, Feifan, Sanwang, Xiaojie, Lipeng, Lijuan, Jinquan, Limin, Xinyu, Haidong, Luoluo, Xiang Wang, Pengliang, Hao Xiang, Huanyu, Tao Fang, Haichao, Junwei, Xiaolin, Dongdong Wang, Dongdong Zhang, Lin Ouyang, Guoliang, De Wei, Wenxin, Wenlei, Kun Guo, Alfredo, Orly, Haitao, Radisras, Feixuan, Olimi, Ildephonse, Bart, et al. You all have painted my life in Ghent more

colorful. Your company is helpful in relieving my homesickness. I have also learned a lot from your suggestions, stories and experiences.

In addition, I would like to thank the China Scholarship Council, Embassy of the P. R. China in the Kingdom of Belgium, and Ghent University for the help and support during my doctoral study. Long live China-Belgium friendship!

Furthermore, I express my gratitude to my parents and relatives. This doctoral thesis is also for you.

Gang Wang
Ghent, Belgium
November, 2019

Contents

Acknowledgements v

Summary xv

Nederlandstalige samenvatting xvii

List of acronyms xix

List of symbols xxi

I Introduction and preliminaries 1

1 Introduction 3
1.1 A general overview . . . 3
1.2 Scope of the thesis . . . 7

2 Preliminaries 11
2.1 Several fundamental concepts in image processing . . . 11
2.1.1 About a digital image . . . 11
2.1.2 Spatial filtering and convolutional kernels . . . 12
2.1.3 Gaussian kernel and scale-space representation . . . 13
2.2 Several typical image analysis tasks . . . 14
2.2.1 Image denoising . . . 14
2.2.2 Low-level feature extraction . . . 15
2.2.3 Image segmentation and superpixel segmentation . . . 17
2.2.4 Object classification and object detection . . . 17
2.3 Several fundamental concepts in deep learning . . . 19
2.3.1 Supervised learning . . . 19
2.3.2 Artificial neural networks . . . 19
2.3.3 Deep convolutional neural networks . . . 21

II Exploration of Gaussian-based kernels for image analysis 25

3 First-order derivative of anisotropic Gaussian kernel with its applications 27
3.1 Motivation . . . 28

3.2 Related work . . . 32
3.2.1 Normalized scale-space representation and its derivative 32
3.2.2 The non-normalized first-order derivative of an-isotropic Gaussian kernel . . . 33
3.2.3 Graph-based superpixel segmentation . . . 34
3.3 Normalized first-order derivative of anisotropic Gaussian kernel 35
3.3.1 Modelling the scaled edges . . . 35
3.3.2 Normalization in scale-space . . . 37
3.3.3 Alleviation of the anisotropy stretch effect . . . 39
3.3.4 Discrete filter bank . . . 42
3.3.5 Compensated anisotropic edge strength . . . 43
3.4 Application to contour detection . . . 45
3.4.1 Hierarchical superpixel maps . . . 45
3.4.2 Hierarchical superpixel contrast maps . . . 48
3.4.3 Contour strength map . . . 49
3.4.4 Binarization . . . 49
3.4.5 Experimental validation . . . 50
3.5 Application to superpixel segmentation . . . 56
3.5.1 Superpixel segmentation incorporating anisotropic edge strength . . . 56
3.5.2 Experimental validation . . . 57
3.6 Conclusions . . . 68

4 Second-order anisotropic Gaussian kernel with application to line detection 71
4.1 Motivation . . . 71
4.2 Related work . . . 75
4.3 Normalized and adaptive second-order anisotropic Gaussian kernel . . . 76
4.3.1 Modelling a line segment . . . 76
4.3.2 Rebuilding the conventional second-order anisotropic Gaussian kernel . . . 76
4.3.3 Scale-invariant normalization . . . 78
4.3.4 Adaptive anisotropy factor . . . 79
4.3.5 Discrete kernels . . . 82
4.3.6 Postprocessing on lineness map . . . 84
4.4 Experimental validation . . . 85
4.4.1 Application to fungus detection . . . 85
4.4.2 Other applications . . . 91
4.5 Conclusions . . . 92

5 Unilateral second-order Gaussian kernel with application to

image denoising **93**
5.1 Motivation . . . 93
5.2 Related work . . . 97
5.2.1 Real-world image denoising . . . 97
5.2.2 Blob detection . . . 99
5.3 The unilateral second-order Gaussian kernel . . . 100
5.3.1 Modelling a blob structure . . . 100
5.3.2 Scale-invariant normalized second-order Gaussian kernel 101
5.3.3 The unilateral second-order Gaussian kernel . . . 104
5.3.4 Topographical measurement of blob characteristics . . . 106
5.4 High-ISO long-exposure image denoising . . . 108
5.4.1 Spatially modelling blob noise . . . 108
5.4.2 Denoising methods incorporating blob reduction . . . 110
5.5 Experimental validation . . . 111
5.5.1 Experiments on removing synthetic blobs and noise . . . 112
5.5.2 Experiments on removing real noise . . . 115
5.6 Conclusions . . . 125

6 Iterative Laplacian-of-Gaussian filtering with application to blob detection **127**
6.1 Motivation . . . 127
6.2 Related work . . . 130
6.3 Method for detecting overlapping blobs . . . 131
6.3.1 Reducing the degree of overlap by iterative Laplacian of Gaussian filtering . . . 131
6.3.2 Non-blob structure suppression using unilateral second-order Gaussian kernels . . . 137
6.4 Experimental validation . . . 141
6.4.1 Evaluation on an example image . . . 141
6.4.2 Evaluation of robustness to noise . . . 143
6.4.3 Evaluation on fluorescence microscopy cell images . . . 143
6.4.4 Evaluation on nanoparticle images . . . 149
6.5 Conclusions . . . 151

III Exploration of deep convolutional networks for image analysis **153**

7 Automated *Artemia* detection and counting **155**
7.1 Motivation . . . 155
7.2 Preliminaries on convolutional neural networks . . . 159
7.2.1 Deep convolutional neural networks . . . 159

7.2.2 U-shaped fully convolutional networks 160
7.3 Automated *Artemia* detection and counting method 160
7.3.1 The *Artemia* detection and counting dataset 162
7.3.2 The marker proposal network 162
7.3.3 The target classifier 166
7.4 Experiments and results 168
7.4.1 Training procedure 168
7.4.2 The watershed-based method for comparison 170
7.4.3 Performance evaluation 171
7.5 Conclusions 174

8 Automated *Artemia* length measurement 175
8.1 Motivation 175
8.2 Materials and methods 178
8.2.1 The *Artemia* length measurement dataset 178
8.2.2 Automated *Artemia* length measurement using U-shaped fully convolutional networks 180
8.2.3 A method using mathematical morphology and polynomial curve fitting 184
8.3 Experiments and results 187
8.3.1 Comparison between two models trained by different types of label maps 187
8.3.2 Length measurement evaluation 188
8.4 Conclusions 192

IV Epilogue 193

9 Conclusions and future work 195
9.1 Conclusions 195
9.2 Potential research directions 197
9.2.1 Texture suppression using superpixels for contour detection 197
9.2.2 Line detection based on image segmentation and line thinning 198
9.2.3 Noise-aware and content-aware image denoising 199
9.2.4 Extensions of the automated *Artemia* analysis methods 200
9.2.5 Transfer use of the developed *Artemia* image analysis methods 201
9.2.6 Open-source software and image material 201

Appendices 203

A Appendix 205

A.1 Proof of Eq. (3.13) 205
A.2 Proof of Eq. (3.15) 207
A.3 Proof of Eq. (4.11) 208
A.4 Proof of Eq. (5.9) 210

Bibliography **211**

Curriculum Vitae **237**

Summary

With the wide use of imaging systems in scientific studies, automated image analysis has attracted increasing attention over the past decades. Automated image analysis can extract meaningful information from raw images and generate novel insight, having superiority in processing efficiency and assessment objectivity over manual image analysis. Since many image analysis tasks can be essentially viewed as problems of feature mapping, feature extraction plays a pivotal role in automated image analysis. Feature extraction methods can be generally divided into hand-crafted methods and deep learning methods. Hand-crafted methods have comparatively better interpretability, and their processing procedures can be well controlled by explicitly defined parameters. Representative hand-crafted features include edges, lines, and blobs. Deep learning methods have a powerful ability to learn and represent features, and thus, they can accomplish comparatively more complex tasks. Hand-crafted methods and deep learning methods are not mutually exclusive. Many deep learning methods have greatly benefited from hand-crafted features. Therefore, it is still necessary to further explore the potential of hand-crafted features to have more explicit toolkits, and to provide more explanations for deep learning techniques. Moreover, it would be worth developing methods that jointly exploit the advantages of hand-crafted methods and deep learning methods.

In bioscience engineering, although a large amount of image analysis methods have been developed, there are still many particular tasks that remain unsolved. In aquaculture, the brine shrimp *Artemia* is a widely used cost-effective diet for fish and crustaceans, and recently, the number of *Artemia* studies is increasing. Since *Artemia* objects are very small in size, they are usually observed by a stereo-microscope, which can acquire a large amount of *Artemia* images containing many *Artemia* objects. Conventionally, most of the *Artemia* image analysis tasks are carried out manually, which is rather time-consuming and labor-intensive. Hence, it is quite necessary to design tailor-made methods for analyzing *Artemia* images.

In this thesis, on the one hand, we investigate several Gaussian-based convolutional kernels to extract hand-crafted features. We present the normalized first-order derivative of anisotropic Gaussian kernel that can detect multiscale edges with a good noise-robustness. This kernel is applied to contour detection and superpixel segmentation. Also, we develop a line detection method using the normalized and adaptive second-order anisotropic Gaussian kernel. This

method can effectively detect multiscale lines in a noisy environment. For blob characterization, we propose the unilateral second-order Gaussian kernel that can quantitatively measure the blob prominence, scale, and position, while yielding little response for non-blob structures. The favorable properties of this kernel are confirmed in image denoising. Moreover, we propose a blob detection method using iterative Laplacian-of-Gaussian filtering and the unilateral second-order Gaussian kernel. This method can handle overlapping blobs effectively. On the other hand, aiming at automated *Artemia* image analysis, we propose an *Artemia* detection and counting method using U-shaped fully convolutional networks and deep convolutional networks. Besides, by jointly using deep learning techniques and hand-crafted features, we develop an automated method that can measure the *Artemia* length accurately in images. All the proposed methods are validated on either widely adopted datasets or in-house datasets.

Nederlandstalige samenvatting

Geautomatiseerde beeldverwerking is de laatste decennia een sterk opkomend studiegebied. Dit komt deels door het wijdverspreid gebruik van beeldverwerking voor wetenschappelijke studies. Geautomatiseerde beeldverwerking kan relevante informatie extraheren van ruw beeldmateriaal, vaak met een betere efficiëntie en objectiviteit vergeleken met een handmatige beeldverwerking. Gezien vele beeldverwerkingstaken in essentie gezien kunnen worden als zogenaamde feature mapping problemen is het extraheren van features een kernaspect van geautomatiseerde beeldverwerking. Feature extractie kan men ruwweg onderverdelen in handmatige methoden en deep learning methoden. De handmatige methoden zijn relatief gezien makkelijker te interpreteren en de procedure wordt gestuurd door expliciet gedefinieerde parameters. Typische voorbeelden van handmatige features zijn randen (edges), lijnen en blobs. De methoden vanuit deep learning daarentegen hebben het vermogen om features te leren en voor te stellen. Dusdoende kunnen deze gebruikt worden voor meer complexe taken. Handmatige en deep learning features kunnen elkaar aanvullen, vele deep learning features zijn gebaseerd op handmatige features. Daardoor is het nog steeds nodig om het potentieel van handmatige features verder te onderzoeken, dit leidt ook tot nieuwe inzichten in deep learning methoden. Verder is het nuttig om methoden te ontwikkelen die de voordelen van zowel handmatige als deep learning methoden samen exploiteren.

In het domein van de bio-ingenieurswetenschappen, hoewel er reeds een enorme verscheidenheden aan beeldverwerkingsmethoden beschikbaar is, zijn er nog altijd vele specifieke taken die onopgelost blijven. In aquacultuur, wordt het pekelkreeftje *Artemia* vaak gebruikt als goedkoop voeder voor vissen en schaaldieren, wat zich uit in een recente stijging in het aantal *Artemia* studies. Gezien *Artemia* objecten zeer klein zijn, worden ze gewoonlijk geobserveerd met een stereomicroscoop, welke een grote hoeveelheid aan *Artemia* beelden kunnen verzamelen. Typisch worden deze beelden handmatig verwerkt, wat vrij tijds- en arbeidsintensief is. Vandaar de noodzaak om domeinspecifieke methoden te ontwikkelen om *Artemia* beelden te analyseren.

In dit doctoraat onderzoeken we twee pistes. Eerst bespreken we verschillende Gaussiaanse convolutionele kernels om handmatige features te extraheren. We presenteren de genormaliseerde eerste-orde afgeleide van de anisotropische Gaussiaanse kernel die op meerdere schalen randen kan detecteren met een goede robuustheid voor ruis. Deze kernel wordt toegepast voor contourdetectie en superpixel segmentatie. We ontwikkelen ook een lijndetectie methode ge-

bruik makende van de genormaliseerde en adaptieve tweede-orde Gaussiaanse kernel. Deze methode kan efficiënt lijnen detecteren op meerdere schalen in ruizige afbeeldingen. Voor blob karakterisatie stellen we de unilaterale tweede-orde Gaussiaanse kernel voor die kwantitatief de prominentie, schaal en positie van een blob kan meten terwijl die weinig respons vertoont voor niet-blob objecten. De goede eigenschappen van deze kernel werden bevestigd voor het ontruizen van afbeeldingen. Verder stellen we een blobdetectie methode voor gebaseerd op iteratieve Laplaciaan-van-Gaussiaan filtering en de unilaterale tweede-orde Gaussiaanse kernel. Deze methode kan vlot omgaan met overlappende blobs. In het tweede deel van dit doctoraat, met als doel geautomatiseerde *Artemia* analyse, stellen we een *Artemia* detectiemethode voor via een volledig geconnecteerd convolutioneel netwerk met een U-vorm en diepe convolutionele netwerken. Verder, door deep learning features met handmatige features to combineren ontwikkelden we een automatische methode om de lengte van *Artemia* accuraat te meten in afbeeldingen. Alle methoden zijn gevalideerd op ofwel gekende datasets of datasets verzameld voor dit project.

List of acronyms

iLoG	iterative Laplacian-of-Gaussian
ArtDeCo	*Artemia* Detection and Counting
AD	Anisotropic Diffusion
ANN	Artificial Neural Network
ASA	Achievable Segmentation Accuracy
BF	Bilateral Filtering
BR	Blob Reduction
CNN	deep Convolutional Neural Network
CBM3D	Color version of the Block-Matching and 3-D filtering
FAG	First-order derivative of Anisotropic Gaussian
FDG	First-order derivative of Gaussian
FCN	Fully Convolutional Network
FP	False Positive
FPN	Fixed Pattern Noise
FN	False Negative
GT	Ground Truth
ISO	International Organization for Standardization
LoG	Laplacian-of-Gaussian
MAE	Mean Absolute Error
Marker-CNN	Marker proposal network and deep Convolutional Neural Network
MCWS	Marker-Controlled WaterShed
MLP	Multi-Layer Perceptron
MMPCF	Mathematical Morphology and Polynomial Curve Fitting
MWNNM	Multi-channel Weighted Nuclear Norm Minimization
NASAG	Normalized and Adaptive Second-order Anisotropic Gaussian
NFAG	Normalized First-order derivative of Anisotropic Gaussian
NLM	Non-local means
NMS	Nonmaxima suppression
ODS	Optimal Dataset Scale
OIS	Optimal Image Scale

SAG	Second-order Anisotropic Gaussian
SNR	Signal-to-Noise Ratio
SOG	Second-Order Gaussian
TP	True Positive
TN	True Negative
TWSC	Trilateral Weighted Sparse Coding
UE	Undersegmentation Error
UNet	U-shaped fully convolutional network
USG	Unilateral Second-order Gaussian

List of symbols

b	Baseline of a blob or a line
c	True strength of an edge
e	The base of the natural logarithm
f	Continuous signal
g	Gaussian kernel
h	Histogram
i	Element index
j	Element index
k	Iteration number of filtering
l	Index of hierarchical levels or neural network layers
m	Image coordinate
p	Prominence of a blob or a line
q	Image channel index
s	Bias in an artificial neuron model
t	Ground truth category label
u	Alternative image coordinate
w	Weights
x	Continuous planar coordinate
z	Predicted category label
$\mathbf{m}$	Discrete image coordinates
$\mathbf{r}$	Fully connected layer
$\mathbf{s}$	Bias vector
$\mathbf{t}$	Ground truth category vector
$\mathbf{w}$	Weight vector for the input of an artificial neuron model
$\mathbf{x}$	Continuous planar coordinates
$\mathbf{z}$	Predicted probability vector
$\mathcal{D}(\cdot)$	Degradation function
$\mathcal{F}(\cdot)$	Spatial filtering
$\mathcal{H}(\cdot)$	Heaviside function
$\mathcal{N}(\cdot)$	Non-linear activation function
$\mathcal{T}(\cdot)$	Distance function between neighboring trees

B Maximum response value for a blob model
D Distance in terms of color or textures
E Maximum response value for an edge model
F F-measure
H H-dome value
J The total number of elements
K The total number of iterations of filtering
L The maximum response value for a line
N Number
R Radius
X Input of an artificial neuron model
Y Interim result of an artificial neuron model
Z Output of an artificial neuron model

B Blob strength map
C Contour strength map
E Edge strength map
G Local patch generated by a 2D Gaussian function
I Image
J Response of Laplacian-of-Gaussian filtering
K Black and white (binary) image
L Line strength map
M Image moment
P Map of length measuring line structures
R Rotation matrix
S Scale map
T Ground truth map
U Response of unilateral second-order Gaussian kernels
V Covariance matrix
W Weights matrix
X Input of an artificial neuron model
Y Convolutional response of a kernel in convolutional neural networks
Z Output of a marker proposal network

$\mathbb{A}$	Set of arc weights in an undirected graph
$\mathbb{B}$	Set of the pixels in a blob model
$\mathbb{C}$	Set of the pixels in a convex region
$\mathbb{D}$	Set of possible directions of kernels
$\mathbb{I}$	Set of images
$\mathbb{P}$	Set of pixels in a superpixel
$\mathbb{Q}$	Set of pixels in a ground truth superpixel
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of positive real numbers
$\mathbb{S}$	Set of possible scales of kernels
$\mathbb{T}$	Spanning tree
$\mathbb{V}$	Set of vertices in an undirected graph
$\mathbb{U}$	Undirected graph
$\mathbb{Z}_+$	Set of positive integers

α	Order of differentiation
β	Factor controlling the extent of the filtering response
γ	Scale normalization factor
ε	Noise level
ζ	Standard deviation of a binary image
η	Parameter allowing the kernel to be applicable to bright or dark lines
θ	Kernel direction
ϑ	Angle range
ι	Parameter to adjust the constraint of the shared superpixel border
κ	Convolutional kernel
ξ	Additive noise
ϖ	The grid cell size of the initially latticed image in the SLIC method
ρ	Conventional anisotropy factor
ϱ	Output value of the last fully connected layer for a category
σ	Kernel scale
ς	Scale decay factor
τ	Threshold for binarization
φ	Anisotropy factor
ω	The scale of an edge, a line or a blob

Λ	Length of a border between two neighboring superpixels
Υ	Mean color of a superpixel
Φ	Merging cost of two neighboring superpixels
$\mathbf{\Gamma}$	Modelled line
$\mathbf{\Theta}$	Direction map
$\mathbf{\Lambda}$	Modelled blob
$\mathbf{\Xi}$	Modelled edge
$\mathbf{\Omega}$	Superpixel contrast map
∇^2	Laplace operator
$*$	Convolution operation

List of Figures

1.1 The flowchart of a typical image analysis method based on hand-crafted features (Dewan et al., 2014) (a), the flowchart of a typical image analysis method based on deep learning (Wang et al., 2017a) (b) and their potential collaborations. 5
1.2 Example images to be analyzed in this thesis. (a) An example fungus image; (b) An example cell image; (c) An example *Artemia* image. 8
1.3 The structure of the thesis. 9

2.1 Illustration of the mechanics of a convolutional operation. . . . 13
2.2 Illustration of the scale-space representation for an example image. (a) Original image; (b) Scale-space representation at the scale $\sigma = 0.5$; (c) Scale-space representation at the scale $\sigma = 2$; (d) Scale-space representation at the scale $\sigma = 4$. 14
2.3 A noisy image (a) and a denoising result (b). 15
2.4 An example image (a) and an example contour map (b). 16
2.5 An example image (a) and a line detection result (b). 16
2.6 An example image (a) and a blob detection result (b). 18
2.7 Illustration of the tasks of image segmentation, superpixel segmentation and object detection. (a) Original image; (b) An image segmentation result; (c) A superpixel segmentation result; (d) A result of *bear* detection. 18
2.8 Illustration of an artificial neuron. 20
2.9 Illustration of an example MLP architecture. 21
2.10 Illustration of an architecture using deep convolutional networks. 22
2.11 Illustration of an architecture using fully convolutional networks. 23

3.1 Examples of edges with different widths. Their scales are $\omega_0 \approx 0$ (a), $\omega_0 = 1$ (b), $\omega_0 = 3$ (c) and $\omega_0 = 5$ (d), respectively. 36
3.2 The horizontal intensity profile through the center of the image shown in Fig. 3.1(a), 3.1(b), 3.1(c) and 3.1(d), respectively. . . 37
3.3 Examples of discrete NFAG kernels. The control scale is set as $\sigma_{\text{con}} = 2$. Top row: Kernels with $\sigma = 1$ and $\varphi = 4$. Second row: Kernels with $\sigma = 1.5$ and $\varphi = 1.78$. Third row: Kernels with $\sigma = 2$ and $\varphi = 1$. Fourth row: Kernels with $\sigma = 3$ and $\varphi = 1$. Bottom row: Kernels with $\sigma = 4$ and $\varphi = 1$. The intensity range of each patch has been adjusted for better display. 43

3.4 Illustration of the monoscale response map and multiscale response map. (a) The original image (courtesy of Dave Johnson); (b) Response map obtained at a single scale; (c) Response map obtained by multiscale NFAG kernels. 44
3.5 Comparison of the NMS result and the compensated NMS result. (a) The NMS result of Fig. 3.4(c); (b) The compensated NMS result. 44
3.6 The flowchart of our framework for obtaining anisotropic edge strength. 45
3.7 Illustration of hierarchical superpixel maps (left column), the corresponding superpixel mean color maps (middle column) and the superpixel contrast maps (right column). In the images in the left column, the number of superpixels is 3152, 394, 50, 7 and 4, respectively. 47
3.8 Sample images and the ground truth contour maps from the BSDS500 dataset. Left column: Original images. Second to sixth columns: Ground truth contour maps labelled by different annotators for each original image. Right column: Aggregated ground truth for each original image. 50
3.9 PR curves of different methods obtained on the BSDS500 dataset. 54
3.10 Detection results of different methods on 10 sample images from the BSDS500 dataset. Green pixels represent true positive detection pixels, blue ones stand for false positive detection pixels and red ones denote unmatched ground truth pixels. Note that the matched ground truth pixels are also colored in green. Contours are thickened to two pixels wide for better illustration. Please zoom electronically for a better view. 55
3.11 Sample images as well as their multiple GT segmentation maps taken from the BSDS500 dataset. Left column: Original images. Second to sixth columns: GT segmentation maps labelled by different annotators for each original image. 60
3.12 Evaluation results in terms of the average ASA and average UE. (a) and (b): Results obtained on the BSDS500 dataset. (c) and (d): Results obtained on the SBAIS dataset. (e) and (f): Results obtained on the NSEMS dataset. 61
3.13 Superpixel maps yielded by different methods on sample images taken from the BSDS500 dataset. The number of superpixels in each full superpixel map is set to 400. For a better visualization, zoomed-in versions are displayed. 62

3.14 Sample images as well as their multiple GT segmentation maps taken from the SBAIS dataset. Left column: Original images. Second to fifth columns: GT segmentation maps labelled by different annotators for each original image. 63
3.15 Superpixel maps yielded by different methods on sample images taken from the SBAIS dataset. The number of superpixels in each full superpixel map is set as 500. For a better visualization, zoomed-in versions are displayed. 65
3.16 Sample images as well as their GT segmentation maps taken from the NSEMS dataset. The first and the third columns: Original images. The second and fourth columns: Corresponding GT segmentation maps. 66
3.17 Superpixel maps yielded by different methods on sample images taken from the NSEMS dataset. The number of superpixels in each full superpixel map is set as 2000. For a better visualization, zoomed-in versions are displayed. 67
3.18 Sample results of saliency detection obtained by the SCA+SLIC and SCA+SH+NFAG methods. 68

4.1 Illustration of the modelled local line segment and two measurable characteristics: prominence and base level. 77
4.2 Illustration of the discrete version of NASAG kernels. The control scale is set as $\sigma_{\text{con}} = 4$. Top Row: Kernels with $\sigma = 2$ and $\varphi = 4$. Second Row: Kernels with $\sigma = 3$ and $\varphi = 1.78$. Third Row: Kernels with $\sigma = 4$ and $\varphi = 1$. Bottom Row: Kernels with $\sigma = 5$ and $\varphi = 1$. The intensity range of each patch has been adjusted for better display. 83
4.3 Illustration of the line detection process. (a) Original image *circuit*; (b) Lineness map; (c) NMS result; (d)-(f) Hysteresis segmentation results using different thresholds. 85
4.4 Detection results on Images 010, 020, 030, 040 and 050. Please note that the original images have been reversed (dark lines with a bright background) for better display. 88
4.5 Detection results on the noisy version of Images 010, 020, 030, 040 and 050 corrupted by white Gaussian noise with an intensity of $\varepsilon_0 = 10$. Please note that the original images have been reversed (dark lines with a bright background) for better display. 89
4.6 Illustration of detection results of vessels, roads and rivers using the proposed method. (a)-(c) are original images while (d)-(f) are their corresponding line detection results. 91

5.1 Image patch containing blob noise (a) and the 3D visualization of its red channel (b). 96
5.2 Visual representation of a blob structure and its measurable characteristics. 100
5.3 Illustration of a modelled blob and the obtained responses at the blob center. (a) A modelled blob ($\omega_0 = 4, p_0 = 0.8$ and $b_0 = 0.1$); (b) The responses yielded by the normalized SOG kernels in scale-space and the analytical values of B obtained by varying σ in scale-space ($\gamma = 1$, $\beta = 4$). 104
5.4 Three-dimensional and planar representations of a normalized SOG kernel. 105
5.5 Three-dimensional illustration of a USG kernel. 106
5.6 A bank of kernels that is used to generate USG kernels at a specific scale. The top row shows $(\kappa_c + 2\kappa_l)$, the middle row shows $(\kappa_c + 2\kappa_r)$ and the bottom row shows $(-\kappa_l + \kappa_r)$. The intensity range of each patch has been adjusted for a better display. 108
5.7 Illustration of the real high-ISO long-exposure noise (left) and the modelled noise (right). Please zoom electronically for a better view. 110
5.8 Illustration of the process of the proposed denoising scheme. (a) A noisy image; (b) The red channel of (a); (c) Result of a blob reduction on the red-channel image; (d) The denoising result of the proposed BR-CBM3D method. 111
5.9 Results of blob reconstruction (b) and blob reduction (c) obtained by the USG method on a synthetic image (a) as well as the denoising result obtained by the BR-NLM method (d). For a better visualization, the images are displayed using the heat maps of intensity. 113
5.10 Responses obtained by the USG method (a), the TH method (b), the MLoG method (c) and the gLoG method (d). For a better visualization, the images are displayed using the heat maps of intensity. 114
5.11 Noisy standard images corrupted by real noise along with the ground truth, and the denoising results obtained by the oAD, sAD, and DnCNN methods. Please zoom electronically for a better view. 119
5.12 Denoising results on the noisy standard images obtained by the BF, NLM, NLMC methods and the proposed methods incorporating blob reduction. Please zoom electronically for a better view. 120

5.13 Denoising results on the noisy standard images obtained by the CBM3D, MWNNM, TWSC methods and the corresponding methods incorporating blob reduction. Please zoom electronically for a better view. 121
5.14 Five real-world noisy images (top row) (Courtesy: Peter K. Burian, Dave Johnson and Ziwei Liu (Liu et al., 2014)) and the processing results of the sAD, DnCNN and LECARM methods. Please zoom electronically for a better view. 122
5.15 Denoising results on five real-world noisy images obtained by the BF, NLM, NLMC methods and the corresponding methods incorporating blob reduction. Please zoom electronically for a better view. 123
5.16 Denoising results on five real-world noisy images obtained by the CBM3D, MWNNM, TWSC methods and the corresponding methods incorporating blob reduction. Please zoom electronically for a better view. 124
5.17 Illustration of the removed noise on the real-world noisy Image #5 (red channel) obtained by the (a) BF, (b) BR-BF, (c) NLM, (d) BR-NLM, (e) NLMC, (f) BR-NLMC, (g) CMB3D, (h) BR-CBM3D, (i) MWNNM, (j) BR-MWNNM, (k) TWSC and (l) BR-TWSC methods, respectively. For a better visualization, the images are displayed using the heat maps of intensity. . . . 125

6.1 Illustration of a conventional Laplacian of Gaussian kernel. (a) Three-dimensional visualization; (b) Planar visualization. . . . 131
6.2 Comparison of the protrusion region and overlapping region in terms of average local contrast. (a) The protrusion region has positive and significant local contrast over the orientation range ϑ_1; (b) The overlapping region has positive and significant local contrast over the orientation range $\vartheta_2 + \vartheta_3$. 132
6.3 A modelled Gaussian blob ($b_0 = 0$, $p_0 = 0.8$ and $\omega_0 = 15$) as well as its corresponding response yielded by an LoG kernel at the scale $\sigma = 15$. 133
6.4 The horizontal intensity profiles through the center of Figs. 6.3(a) and 6.3(b). 136
6.5 Illustration of the responses of iLoG filtering. (a)-(d): The original image, the response of the first LoG filtering, the response of the second LoG filtering and the response of the third LoG filtering. 137
6.6 Illustration of a USG kernel. (a) Three-dimensional visualization; (b) Planar visualization. 138

6.7 Illustration of the responses of USG kernels as well as the blob detection results. (a) The response of the USG kernels obtained on the image shown in Fig. 6.5(d); (b) The Otsu thresholding result of (a); (c) The Rosin thresholding result of (a); (d) The extracted blob markers (in red) superimposed on the original image. 140
6.8 Example image containing a collection of blobs on a heterogeneous background (a) and the blob detection results obtained by the MLoG method (b), the MPP method (c) and the iLoG-USG method (d). The red, green and blue crosses denote the correctly detected, falsely rejected and falsely detected results, respectively. 142
6.9 Detection results on images corrupted by Gaussian noise. (a-b) Image corrupted by zero-mean Gaussian noise with a variance of 0.7^2 and the detection result; (c-d) Image corrupted by zero-mean Gaussian noise with a variance of 0.8^2 and the detection result. The red and blue crosses denote the correct and incorrect detections, respectively. 144
6.10 Detection results on images corrupted by speckle noise. (a-b) Image corrupted by speckle noise with a variance of 1.0^2 and the detection result; (c-d) Image corrupted by speckle noise with a variance of 1.1^2 and the detection result. The red and blue crosses denote the correct and incorrect detections, respectively. 145
6.11 Detection result (b) on an image corrupted by Poisson noise (a). 145
6.12 Cell detection results obtained by the proposed method on the Images PoC 0%-1 (a), PoC 0%-2 (b), PoC 15%-1 (c) and PoC 15%-2 (d). The red, green and blue crosses denote the TP, FN and FP results, respectively. For a better illustration, the images in the first row are the blue channel of the original fluorescence microscopic images. In each column, the image in the second row shows the zoomed-in patches corresponding to the green windows in the first row. 147
6.13 Cell detection results obtained by the proposed method on the Images PoC 30%-1 (a), PoC 30%-2 (b), PoC 45%-1 (c) and PoC 45%-2 (d). The red, green and blue crosses denote the TP, FN and FP results, respectively. For a better illustration, the images in the first row are the blue channel of the original fluorescence microscopic images. In each column, the image in the second row shows the zoomed-in patches corresponding to the green windows in the first row. 149

6.14 Nanoparticle detection results of the proposed method. The red, green and blue crosses denote the TP, FN and FP results, respectively. 150

7.1 Illustration of the *Artemia* growth stages. 160
7.2 Illustration of the proposed Marker-CNN method for *Artemia* detection and counting. 161
7.3 Sample *Artemia* images and the corresponding labelled marker maps for training the marker proposal network. 163
7.4 Sample images for training the CNN-based target classifier. Top row: Samples of cysts; Second row: Samples of nauplii; Bottom row: Samples of non-targets. 163
7.5 Architecture of the UNet-based marker proposal network. . . . 164
7.6 Architecture of our CNN-based target classifier. 167
7.7 Illustration of the data augmentation results. (a) Original training image and the corresponding ground truth marker map; (b)-(e) The augmented training images and the corresponding ground truth marker maps. 169
7.8 Evolution of the prediction maps yielded by the UNet as the number of epochs increases. (a) Original images; (b)-(e) The prediction maps that are obtained when the number of iterations is 5, 10, 50 and 200, respectively. 170
7.9 Training accuracy and validation accuracy of the target classifier with respect to the number of training epochs. 171
7.10 Illustration of the marker maps and detection results yielded by our method, and the detection results obtained by the MCWS method. (a) Original images; (b) Marker maps yielded by the marker proposal network; (c) Detection results obtained by the Marker-CNN method; (d) Detection results obtained by the MCWS method. The blue and green bounding boxes indicate the detection results of cysts and nauplii, respectively, while the red rectangles indicate the incorrect detection results. . . . 172

8.1 Illustration of an adult female *Artemia* (a) and an adult male *Artemia* (b). 178
8.2 Sample label maps of length measuring lines (in green) superimposed on the original images. First row: Label maps of thin measuring lines. Second row: Label maps of thick length measuring lines. 179
8.3 Architecture of the UNet for centerline area segmentation. . . . 181
8.4 Curves of train losses obtained by two different settings of batch size. 182

8.5 Evolution of the prediction map as the number of epochs increases. (a) Original image; (b)-(f) The prediction map that is obtained when the number of iterations is 5, 20, 50, 100 and 200, respectively. 182
8.6 The results of measuring line extraction (red lines superimposed on the original image) obtained by the MS method (a) and the NASAG method (b) on the image shown in Fig. 8.5(f). For a better demonstration, a zoom-in view of the regions of interest (indicated by windows) is displayed in the second row. 184
8.7 Demonstration of the MMPCF method. (a) The original image; (b) The grayscale version; (c) The foreground silhouette; (d) The horizontally aligned silhouette; (e) The result of skeletonization, the starting point and the endpoint (in green); (f) The result of geodesic distance transform, with the starting point as the seed location; (g) The result of geodesic distance transform, with the endpoint as the seed location; (h) The principal morphological skeleton (in red); (i) The fitted polynomial curve overlaid on the aligned silhouette; (j) The length measuring line overlaid on the original image. 185
8.8 Sample images and the prediction maps yielded by the UNet^{tn}. First column: The label maps of thin measuring lines superimposed on the original image. Second column to fifth column: The prediction maps obtained on the sample images when the number of epochs is 5, 10, 50 and 200, respectively. 188
8.9 Sample images and the prediction maps yielded by the UNet^{tk}. First column: The label maps of thick measuring lines (in green) superimposed on the original image. Second column to fifth column: The prediction maps obtained on the sample images when the number of epochs is 5, 10, 50 and 200, respectively. . 189
8.10 Comparison of different methods for measuring line extraction. First row: The label maps of measuring lines (in green) superimposed on the sample images. Second row to bottom row: The results of length measuring line extraction (in red) obtained by different methods. 191

List of Tables

3.1 Evaluation results obtained by different methods on the BSDS500 dataset. 54
3.2 Evaluation results in terms of runtime. 56
3.3 Saliency detection evaluation results in terms of mean absolute error obtained on the sample images. 69

4.1 Evaluation results in terms of F-measure. 87
4.2 Evaluation results in terms of runtime (s). 90

5.1 PSNR (dB) of denoising results. 117
5.2 Execution time (s) of the different methods on each image. . . 118

6.1 Evaluation results obtained by each method on each fluorescence microscopy cell image. 148
6.2 The average runtime (s) of each method for processing the fluorescence microscopy cell images. The symbol † denotes the execution time on the ImageJ platform. 149
6.3 The number of the correctly detected particles by different methods on each nanoparticle image. 151

7.1 Evaluation results in terms of detection accuracy. 173
7.2 Evaluation results in terms of precision, recall and F-measure. 174
7.3 Evaluation results in terms of average runtime (s). 174

8.1 Quantitative evaluation results in terms of RMSE, MAE and MAPE obtained by different methods. 190

PART I

INTRODUCTION AND PRELIMINARIES

1 Introduction

1.1. A general overview

Image analysis plays an indispensable role in many scientific studies, allowing to extract meaningful information from raw images and generate novel insights (Meijering et al., 2016). For instance, in bioscience engineering, microscopy image analysis provides quantitative support for characterizing organisms and assessing their activities (Xing et al., 2018). However, in many scenarios, the labor-intensive manual analysis can hardly handle the increasing amount of imagery data. It is highly desired to have access to automated image analysis techniques that are more accurate, reliable and time-efficient.

Automated image analysis is an interdisciplinary task that involves subjects of image processing, computer vision, machine learning, etc. One of its pivotal procedures is feature extraction, since many image analysis tasks can be essentially viewed as problems of feature mapping (LeCun et al., 2015). For instance, image segmentation is a process of pixel-wise classification based on both the local features (e.g., color and texture) and global features (e.g., spatial structure and semantic category) (Pont-Tuset et al., 2017).

Generally, existing feature extraction methods can be divided into two categories: hand-crafted methods and deep learning methods (Litjens et al., 2017). Most of the hand-crafted features are mathematically formulable and have solid theoretical foundations. Therefore, they have comparatively better interpretability, and their processing procedures can be well controlled by explicitly defined parameters (Lopez-Molina et al., 2015). Representative hand-crated features include edges (Canny, 1986), lines (Obara et al., 2012a) and blobs (Ruusuvuori et al., 2010).

Edges are usually defined as sets of pixels at which the image intensity or color changes abruptly. Edges are a kind of visual feature bridging the gap between image pixels and many image analysis tasks (Li et al., 2015a), such as image segmentation (Yu et al., 2012) and object detection (Zitnick and Dollár, 2014). In literature, numerous edge detection methods have been developed. Nevertheless, many existing methods are sensitive to noise and have limitations in detecting multiscale edges. Lines (a.k.a. ridges or curvilinear structures) are generally defined as elongated regions with dissimilar intensities compared to their neighboring pixels. Such structures hold key information for some image analysis tasks, such as road detection (Ferraz

et al., 2016), fungus extraction (Vidal-Diez de Ulzurrun et al., 2015), blood vessel detection (Yang and Shi, 2014), and so on. Despite the significant efforts carried out in the past decades, it is difficult to accurately detect lines of which the prominence and widths are highly heterogeneous, especially in a noisy environment. Blobs (a.k.a. particles or dots) are usually defined as small structures of which the visual properties are different from those in their surrounding regions. Many objects in images show a blob appearance, and as such, blob detection has found applications in a wide variety of fields, such as cell counting (Ruusuvuori et al., 2010), vanishing point detection (Kong et al., 2013b), quantum dot recognition (Xu and Lu, 2013), and so on. Although a number of blob detection methods have been developed, most of them have limitations in dealing with adjacent blobs. In addition, existing methods can hardly quantitatively measure the blob characteristics. Moreover, conventional methods yield significant responses for not only blobs, but also non-blob structures like lines and edges. Thus far, blob detection still remains a challenging task.

In image analysis, hand-crafted methods are usually adopted when shallow features can accomplish the task well. But when shallow features are insufficient for solving the problems, it is preferred to use deep learning methods to exploit high-level features (Krizhevsky et al., 2012). Deep learning methods can automatically learn and represent the features from the raw data, and have a powerful ability in feature learning and representation. Therefore, compared with hand-crafted methods, deep learning methods require less human interventions and usually obtain better performances for specific tasks. Representative applications of deep learning methods include nucleus detection using deep convolutional networks (Xu et al., 2016) and organ segmentation using fully convolutional networks (Li et al., 2018). Despite their powerful learning abilities, deep learning methods yet have limitations (Xing et al., 2018). Firstly, many deep learning methods are supervised learning methods and usually contain a large volume of parameters, and as such, they require a large amount of training data and a great many of computing resources to train the models. In particular, many end-to-end learning methods train the models using the raw data and the ultimately desired output, which would make the models more complex and less interpretable. Secondly, deep learning methods sometimes are quite dependent on the training dataset. It is inconvenient to adjust the trained model to adapt to another kind of input (Schmidhuber, 2015).

Hand-crafted methods and deep learning methods are not mutually exclusive. On the contrary, in many applications, hand-crafted features have been used for pre-processing input images or post-processing output results,

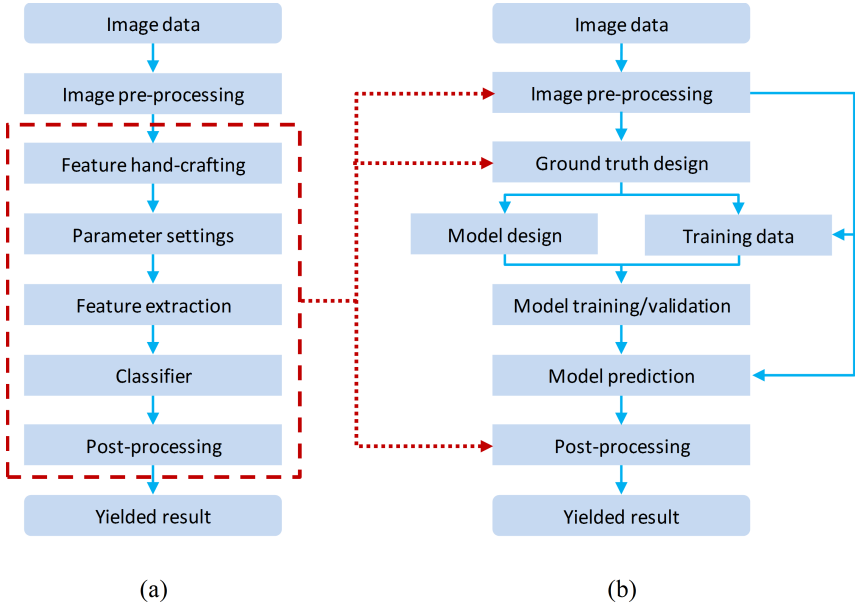


Figure 1.1: The flowchart of a typical image analysis method based on hand-crafted features (Dewan et al., 2014) (a), the flowchart of a typical image analysis method based on deep learning (Wang et al., 2017a) (b) and their potential collaborations.

thereby easing the difficulties of model design (Girshick et al., 2016). Besides, hand-crafted methods can be used for labelling the ground truth (Vajda et al., 2015), making data annotation easier to accomplish. Hence, in image analysis, it is still necessary to further explore the potentials of hand-crafted features to have more explicit toolkits, and to provide more explanations for deep learning techniques. Moreover, it would be worth developing methods that jointly exploit the advantages of hand-crafted methods and deep learning methods. We show several potential collaborations between the two kinds of methods in Fig. 1.1.

As mentioned earlier, with the advances in image processing, computer vision, machine learning, etc., many automated image analysis methods have been developed. Nonetheless, there are still many particular image analysis tasks that have been seldom addressed. In aquaculture, the brine shrimp *Artemia* is a widely used cost-effective diet for fish and crustacean, and recently, the number of studies on *Artemia* is increasing. Since *Artemia* cysts and nauplii/metanauplii are very small in size, they are usually observed by a stereo-microscope, which can acquire a large amount of *Artemia* images containing many *Artemia* objects. Conventionally, most of the *Artemia* image analysis tasks are carried out manually. For example, when assessing

the hatching rate in *Artemia* incubation, the researcher should manually count the number of cysts and nauplii separately. This process is quite time-consuming and labor-intensive. Although many automated methods have been developed for detecting moths (Ding and Taylor, 2016), fish (French et al., 2015), shrimps (Kesvarakul et al., 2017; Kaewchote et al., 2018), etc., they cannot be readily transferred to detect and count *Artemia*. This is because the *Artemia* objects in images are usually seriously adjacent. Moreover, the appearance of *Artemia* varies significantly over the different growth stages. Therefore, it is quite necessary to design tailor-made methods for analyzing *Artemia* images.

This thesis will be presented based on the considerations above. Specifically, the main goals as well as the subtasks of this thesis are listed as follows:

1. Investigating an existing Gaussian-based convolutional kernel named the first-order derivative of anisotropic Gaussian (FAG) kernel for edge detection. Although the first-order derivative of isotropic Gaussian kernel has been intensively studied for extracting edges (McIlhagga, 2011), recent biological findings reveal that kernels for edge detection should be anisotropic (Shapley and Hawken, 2011). Some preliminary studies have confirmed the superiority of anisotropic kernels in noise-robustness over isotropic kernels (Shui and Zhang, 2012; Zhang et al., 2017b). It is meaningful to expand the conventional FAG kernel into scale-space for better detecting multiscale edges. Besides, the problem of anisotropy stretch effect incurred by anisotropic kernels should be addressed. The applications of the modified FAG kernels should also be studied.

2. Modifying an existing Gaussian-based convolutional kernel called the second-order anisotropic Gaussian kernel for line detection. A preliminary work validates that the second-order anisotropic Gaussian kernel is appropriate for detecting lines (Lopez-Molina et al., 2015). Nevertheless, the preliminary work leaves several problems unsolved, including how to normalize the kernels in scale-space and how to alleviate the anisotropy stretch effect. The efficacy of the modified kernel can be tested on a fungus extraction task. For illustration, an example fungus image is displayed in Fig. 1.2(a).

3. Proposing novel Gaussian-based convolutional kernels for blob detection. Among the existing kernels for blob detection, the second-order Gaussian kernel and the Laplacian-of-Gaussian kernel have been widely used (Kong et al., 2013a). However, both kinds of kernels have limitations. They yield significant responses for not only blobs, but also non-blob structures such as lines and edges. Besides, they can hardly quantitatively measure

the blob characteristics. Moreover, they usually underperform when the blobs to be detected are seriously adjacent. These problems should be addressed. The developed methods can be validated by performing tasks that are highly related to blob detection, e.g., cell counting. For illustration, we show an example cell image in Fig. 1.2(b), which contains adjacent blobs.

4. Developing a method built on the deep convolutional network (CNN) (Krizhevsky et al., 2012) to accomplish a particular image analysis task: automated *Artemia* detection and counting. In many *Artemia* studies, the numbers of *Artemia* objects in images are needed. As discussed earlier, this problem has been seldom addressed so far (Kim and Cho, 2013). Moreover, existing object detection methods cannot readily be transferred to detect and count *Artemia*, since the *Artemia* objects in images are usually seriously adjacent, and the appearances of *Artemia* objects vary significantly over the different growth stages, as shown in Fig. 1.2(c). Therefore, it is highly desired to develop an automated method to accomplish this task. And to this end, a dataset should also be collected to train and test the proposed method.

5. Jointly using hand-crafted kernels and deep learning techniques to solve a particular problem: automated *Artemia* length measurement. In many *Artemia* studies, the length measures of the *Artemia* objects in images are desired. Conventional manual length measurement is too labor-intensive to process a large amount of images. Thus, an automated *Artemia* length measurement method would greatly benefit a lot of *Artemia* studies. To train and test the developed method, collecting a dataset is also necessary.

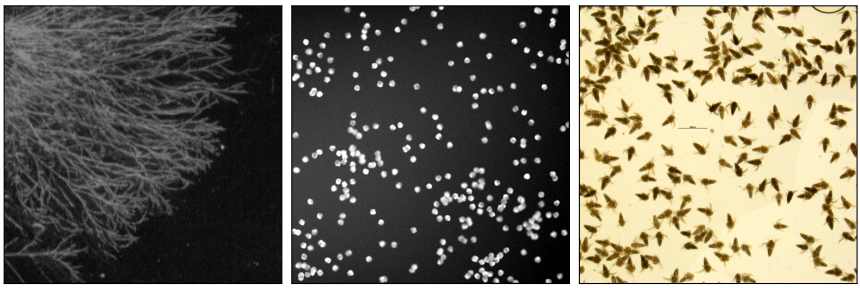
In the next section, we provide an outline of how we present our contributions to the main goals and subtasks of this PhD thesis.

1.2. Scope of the thesis

This thesis is composed of four parts: one introductory part (I), two central parts (II and III), and one concluding part (IV), as displayed in Fig. 1.3.

Firstly, for the reader's convenience, we include two chapters (1-2) in Part I. Besides the outline of this thesis, some concepts that are quite relevant to our work are presented, including the basics of image processing, computer vision and deep learning.

Part II consists of four chapters (3-6) and elaborates our contributions to the investigations on three kinds of Gaussian-based convolutional kernels (the



(a) (b) (c)

Figure 1.2: Example images to be analyzed in this thesis. (a) An example fungus image; (b) An example cell image; (c) An example *Artemia* image.

normalized first-order derivative of anisotropic Gaussian (NFAG) kernel, the normalized and adaptive second-order anisotropic Gaussian (NASAG) kernel and the unilateral second-order Gaussian (USG) kernel) and a novel filtering method (iterative Laplacian-of-Gaussian (iLoG) filtering). By proposing novel normalization methods, we enable the NFAG kernel and NASAG kernel to be used in scale-space. These kernels can exploit the multiscale information in feature extraction, while obtaining a good robustness to noise. In addition, a scale-adaptive anisotropy factor is designed to attenuate the anisotropy stretch effect. The favorable properties of the NFAG kernel are validated on the tasks of contour detection and superpixel segmentation, while the effectiveness of the normalized NASAG kernel is confirmed on the task of fungus extraction. Capitalizing on the analysis of second-order Gaussian kernels, we also propose a novel kernel named USG kernel. The USG kernel can quantitatively characterize the blobs, while yielding little response for non-blob structures. With its good properties, the USG kernel is successfully applied to high-ISO long-exposure image denoising. Moreover, building on the existing Laplacian-of-Gaussian kernel, we propose the iLoG filtering that can be used for separating seriously adjacent blobs. Combining the iLoG filtering and USG kernels, we propose an automated cell counting method whose efficacy is validated by experiments.

Part III presents how we explore deep learning techniques for automated *Artemia* image analysis. In Chapter 7, we propose an automated *Artemia* detection and counting method using U-shaped fully convolutional networks (Ronneberger et al., 2015) and deep convolutional networks (Simonyan and Zisserman, 2015). Different from conventional methods, our method employs a marker proposal network to yield target candidates. The target candidates are subsequently classified into the categories or labelled as a non-target by

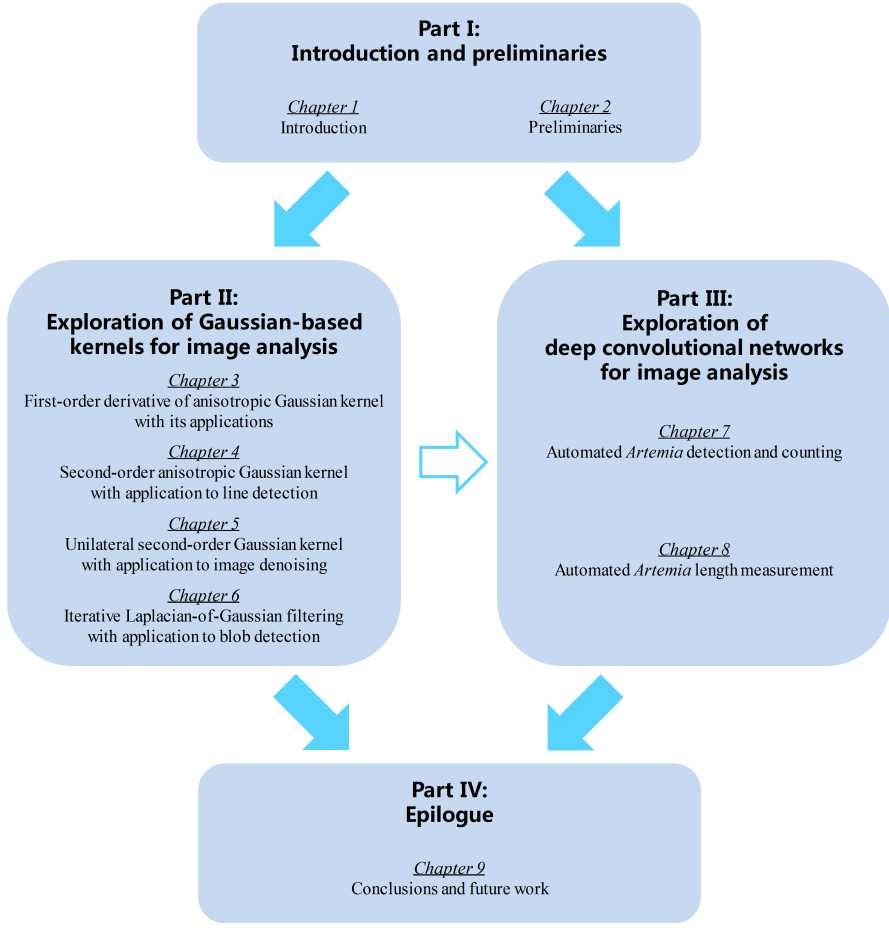


Figure 1.3: The structure of the thesis.

the designed CNN-based network. We also collect a dataset to train and test the proposed method. Moreover, in Chapter 8, by employing both the U-shaped fully convolutional network (UNet) and NASAG kernels, we develop an automated *Artemia* length measurement method. The designed UNet module is used for extracting the length measuring structure, while NASAG kernels are employed to transform the length measuring structure into a thin length measuring line with minimal loss of the length measurement. The performance of this method is evaluated on a collected dataset.

Finally, Chapter 9 in Part IV lists the conclusions and potential research directions.

2 Preliminaries

In this chapter, to assist the reader in understanding the contributions of this dissertation, we introduce some relevant concepts that are widely used in image analysis. We explain the basics of image processing, and, subsequently, present several typical image analysis tasks. Example images are also displayed for a better explanation. We also introduce several fundamental concepts in deep learning. We refer the reader to the books *Digital Image Processing* (González and Woods, 2010), *Pattern Recognition and Machine Learning* (Bishop, 2006) and *Deep Learning* (Goodfellow et al., 2016) for more detailed descriptions.

2.1. Several fundamental concepts in image processing

2.1.1. About a digital image

When talking about a digital intensity (a.k.a. grayscale or brightness) image, we refer to a 2-D array that is denoted by $\mathbf{I}(\mathbf{m})$ in this dissertation, where $\mathbf{m} = [m_x, m_y]^{\mathbf{T}}$ denotes the discrete image coordinates. An intensity image is the result of sampling and quantization on a continuous signal f. In an image, the location $\mathbf{m}$ and the corresponding intensity value is termed a pixel, while the plane that is spanned by the image coordinate is termed the spatial domain.

For a pixel with coordinates $\mathbf{m}$, its 8-neighbors include four horizontal and vertical neighbors and four diagonal neighbors. In an intensity image, a connected component is defined as a subset of pixels, in which each pixel can find a path to any other pixels within the subset.

For an intensity image with 8-bit unsigned integers, its values are taken from the range $[0, 255]$, while for an intensity image with double precision, the value of each pixel is usually taken from the range $[0, 1]$. For a binary image, the values are either 0 or 1. The size of the image (a.k.a. resolution) is represented as the number of columns by the number of rows in $\mathbf{I}$.

A color image is composed of multiple channels, each of which is represented by an intensity image map. There are quite a few color spaces for representing the color information in the image field. One of the most commonly used color space in practice is the RGB space, which consists of the red, green and

blue channels. A color image can be denoted as $\mathbf{I} = (\mathbf{I}^{\mathrm{r}}, \mathbf{I}^{\mathrm{g}}, \mathbf{I}^{\mathrm{b}})$, in which $\mathbf{I}^{\mathrm{r}}$, $\mathbf{I}^{\mathrm{g}}$ and $\mathbf{I}^{\mathrm{b}}$ are the color channels.

2.1.2. Spatial filtering and convolutional kernels

Spatial filtering is one of the principle image processing techniques. It operates directly in the spatial domain, as opposed to the frequency domain, in which the operations are performed on, for example, the Fourier-transformed result.

The spatial filtering can be denoted by the expression

$$\tilde{\mathbf{I}}(\mathbf{m}) = \mathcal{F}\left(\mathbf{I}(\mathbf{m})\right) , \tag{2.1}$$

where $\mathbf{I}(\mathbf{m})$ and $\tilde{\mathbf{I}}(\mathbf{m})$ are the input image and the output image, respectively, and $\mathcal{F}$ stands for an operator defined over a window (a.k.a. mask, patch or neighborhood region) centered at the coordinate $\mathbf{m}$. At each location $\mathbf{m}$, the spatial filtering operator yields a new pixel of which the value is the result of the filtering operation. Accordingly, a full filtering map is obtained as the center of the operator visits all the pixels of the input image. If the operation performed on the image pixels is linear, the spatial filtering is termed a linear spatial filtering operator. There are two kinds of approaches when performing linear spatial filtering: correlation operations and convolution operations. A correlation operation is a process of moving a filtering operator over the image and computing the sum of products at each location, while the process of convolution operation is the same except that the operator is first rotated by 180°. A convolution operation is implemented by sliding a convolutional kernel over the image. Let κ be a convolutional kernel with size $(2 \cdot r_{\mathrm{w}} + 1) \times (2 \cdot r_{\mathrm{h}} + 1)$, where r_{w} and r_{h} are positive integers. When convolving κ with an image $\mathbf{I}$, we obtain the convolutional response at the location $\mathbf{m}$ as follows:

$$\tilde{\mathbf{I}}(\mathbf{m}) = \sum_{u_x=-r_{\mathrm{w}}}^{r_{\mathrm{w}}} \sum_{u_y=-r_{\mathrm{h}}}^{r_{\mathrm{h}}} \kappa(u_x, u_y) \cdot \mathbf{I}(m_x - u_x, m_y - u_y) , \tag{2.2}$$

where $[u_x, u_y]^{\mathrm{T}}$ denotes the kernel coordinates. Figure 2.1 displays how a convolutional kernel yields a response value at a specific location. By varying $\mathbf{m}$, we make the convolutional kernel κ visit all the pixels in $\mathbf{I}$, thereby obtaining the convolutional result

$$\tilde{\mathbf{I}} = \kappa * \mathbf{I} , \tag{2.3}$$

where $*$ denotes the convolutional operation.

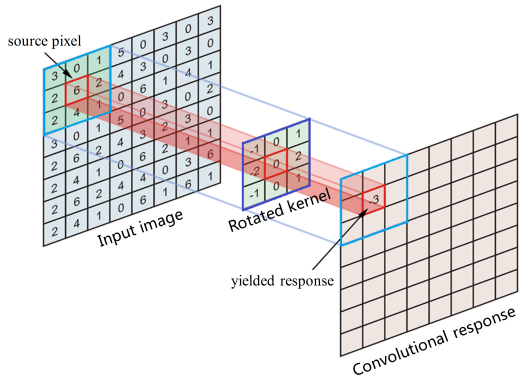


Figure 2.1: Illustration of the mechanics of a convolutional operation.

2.1.3. Gaussian kernel and scale-space representation

A fundamental convolutional kernel is the Gaussian kernel that can be used for smoothing images or enhancing image features at different scales. A Gaussian kernel is defined as

$$g(\mathbf{m};\sigma) = \frac{1}{2\pi\sigma}\exp\left(-\frac{\mathbf{m}^T\mathbf{m}}{2\sigma^2}\right), \tag{2.4}$$

where σ is referred to as the scale parameter controlling the spatial extent of the kernel. By convolving the Gaussian kernel with an image $\mathbf{I}$, we obtain scale-space representation of the image as follows (Lindeberg, 1998b):

$$\mathbf{I}_{\text{ss}}(\mathbf{m};\sigma) = g(\mathbf{m};\sigma) * \mathbf{I}(\mathbf{m}) \ . \tag{2.5}$$

In Fig. 2.2, we show the scale-space representation of an example image at several scales. It can be seen that the noise points in the original image are smoothed at the scale $\sigma = 0.5$. As the scale value increases, the image details in the scale-space representation are reduced. Only the main contents of the image are retained at a large scale.

Using the derivatives of the Gaussian kernel, we can also obtain the derivatives of the scale-space representation, which is formulated as

$$\mathbf{I}_{\text{ss}}^{(\alpha)}(\mathbf{m};\sigma) = g^{(\alpha)}(\mathbf{m};\sigma) * \mathbf{I}(\mathbf{m}) \, , \tag{2.6}$$

where $\alpha \in \mathbb{N}$ denotes the order of differentiation.

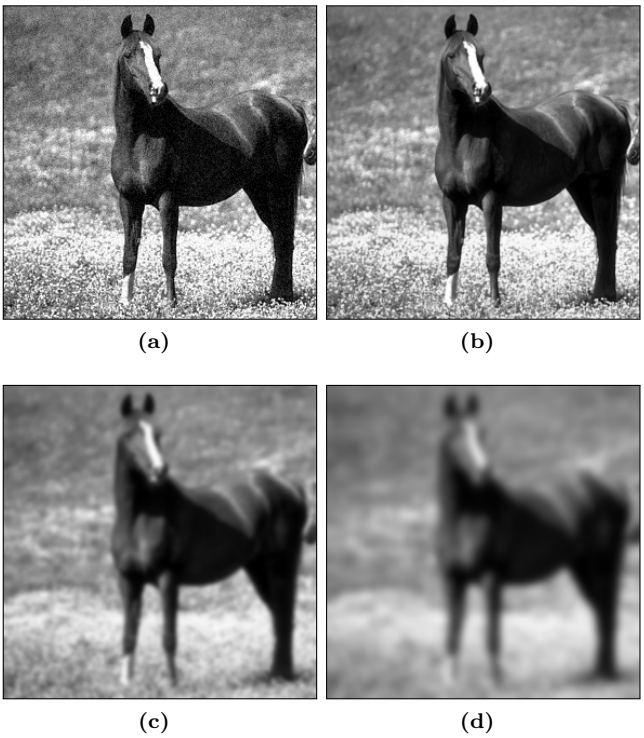


Figure 2.2: Illustration of the scale-space representation for an example image. (a) Original image; (b) Scale-space representation at the scale $\sigma = 0.5$; (c) Scale-space representation at the scale $\sigma = 2$; (d) Scale-space representation at the scale $\sigma = 4$.

2.2. Several typical image analysis tasks

2.2.1. Image denoising

Image denoising is usually a requisite procedure in image analysis, since digital images can hardly get rid of the corruption of noise (Chang et al., 2000a). Noise usually comes from the image acquisition procedure, since the performance of imaging systems can be affected by numerous factors, e.g., the environmental condition and the sensor temperature. Besides, the image can be degraded during the transmission procedure. For instance, when the image is transmitted through radio channels, it might be degraded by the interference of atmospheric disturbance (Dabov et al., 2007).

Usually, the image degradation process caused by noise can be modelled as a degradation function, which is used to represent the impact of non-additive noise, together with an additive noise term. Let $\mathcal{D}$ be a degradation function

(a) (b)

Figure 2.3: A noisy image (a) and a denoising result (b).

and η be the additive noise term, the noisy image $\mathbf{I}_{\mathrm{n}}$ can be formulated by

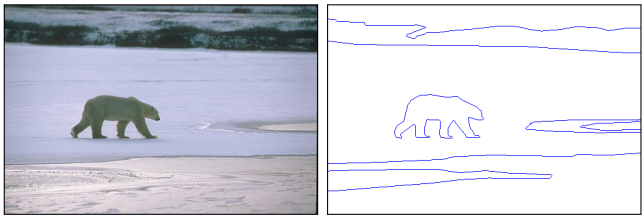
$$\mathbf{I}_{\mathrm{n}}(\mathbf{m}) = \mathcal{D}\left[\mathbf{I}_{\mathrm{c}}(\mathbf{m})\right] + \eta(\mathbf{m}), \tag{2.7}$$

where $\mathbf{I}_{\mathrm{c}}$ represents the clean image. Typical noise types include Gaussian noise, impulse noise and high-ISO noise. The goal of image denoising is to restore the clean image $\mathbf{I}_{\mathrm{c}}$ from the noisy image $\mathbf{I}_{\mathrm{n}}$ as much as possible. For illustration, Fig. 2.3(a) displays a noisy image, and Fig. 2.3(b) shows a denoising result. It can be seen that the image quality has been significantly improved by an image denoising procedure.

2.2.2. Low-level feature extraction

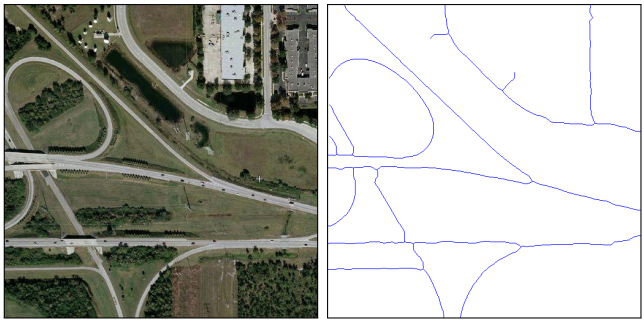
In the image processing and computer vision fields, low-level features are usually locally defined image patterns (Sochen et al., 1998). These image patterns mainly describe the discontinuity and similarity in terms of intensity values or color information. Low-level features reflect the preliminary processing results of the human visual system (Akbarinia and Parraga, 2018). In some scenarios, low-level features can be directly used to solve practical problems (Smith and Brady, 1997). They can also be employed for further extracting the features at higher levels. Typical low-level features include edges, lines and blobs.

Edges are usually defined as sets of pixels at which the image intensity or image color changes abruptly (Canny, 1986). It is believed that edges embody key visual information of an image. Edge detection is among the earliest studied image processing techniques, and has been widely used in a lot of image analysis systems (Sobel, 1970). A typical application of edges is contour detection. Contours (a.k.a. boundaries) are highly related to edges, which are defined as the borders between the objects and the background or



(a) (b)

Figure 2.4: An example image (a) and an example contour map (b).



(a) (b)

Figure 2.5: An example image (a) and a line detection result (b).

between different objects (Liu et al., 2017). For illustration, Fig. 2.4 displays an example image and one manually annotated contour map. It can be seen that not all the locations that have color discontinuities are reflected in the contour map. Only the edges that are on the borders between different semantic object regions form the contours (Arbelaez et al., 2011).

Lines are generally defined as elongated regions with dissimilar intensities or color information compared to their neighboring pixels (Chaudhuri et al., 1989). If we visualize an intensity image in three dimensions, the lines can be viewed as either the geographical ridges (i.e., bright lines) or geographical valleys (i.e., dark lines). Line detection has been employed for detecting blood vessels (Obara et al., 2012a), fungal networks (Lopez-Molina et al., 2015), power lines (Candamo et al., 2009), roads (Bae et al., 2015), and so on. For illustration, we display a remote sensing image in Fig. 2.5(a) and a road detection result in Fig. 2.5(b).

Blobs can be defined as small structures of which the visual properties are different from those in their surrounding region (Koenderink, 1984). Many

objects in images show a blob appearance, and as such, blob detection has found applications in a wide variety of fields, such as cell counting (Ruusuvuori et al., 2010), vanishing point detection (Kong et al., 2013b), quantum dot recognition (Xu and Lu, 2013), and so on. For illustration, Fig. 2.6(a) shows an image containing nanoparticles, and Fig. 2.6(b) displays a blob detection result. From the detection result, some desired information, e.g., the number of the nanoparticles, can be obtained.

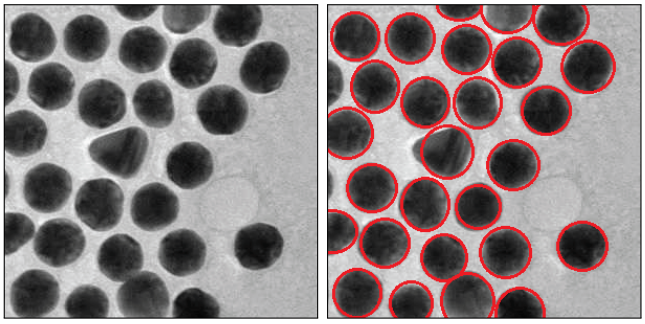
2.2.3. Image segmentation and superpixel segmentation

Image segmentation is the process of partitioning an image into multiple regions according to the visual contents (Martin, 2003), as shown in Fig. 2.7(b). It can make the representation of an image more meaningful to analyze, and therefore, image segmentation has played an indispensable role in numerous applications (Arbelaez et al., 2011), such as instance detection, object tracking, and so on. Image segmentation is more a semantic task than a low-level image processing task, and thus, it usually suffers from either over-segmentation or under-segmentation (Zhang et al., 2008). Comparatively, superpixel segmentation is a more practical technique for pixel grouping.

A superpixel denotes a set of pixels that have similar visual properties, as displayed in Fig. 2.7(c). Compared with pixels, superpixels embody higher-level features, and can therefore reduce the complexity of subsequent image analysis tasks. With a great many of successful applications in object detection, saliency detection, stereo matching, etc., superpixel segmentation has been a fundamental task in image analysis (Stutz et al., 2018).

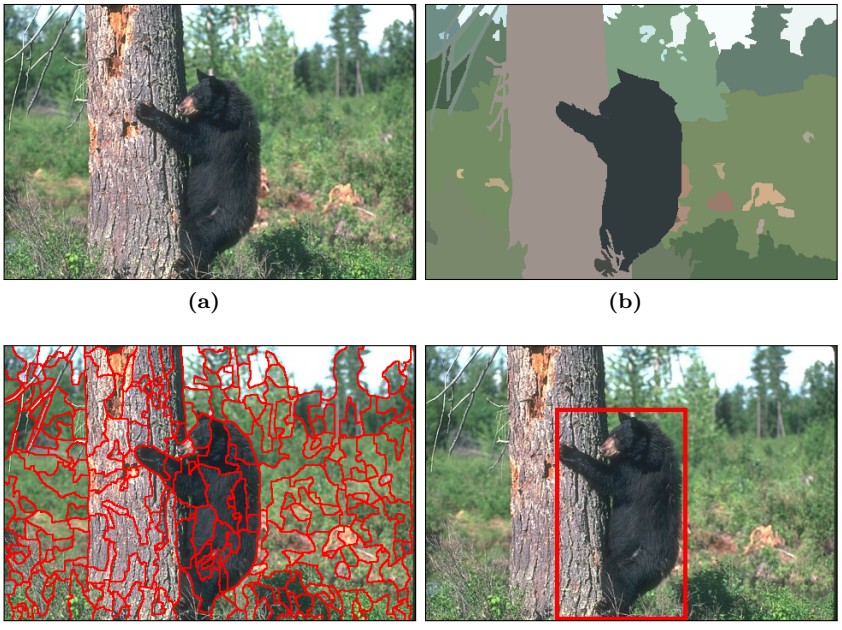
2.2.4. Object classification and object detection

Object classification and object detection are fundamental, sometimes highly related, image analysis tasks. Object classification (a.k.a. object/target recognition, image classification) is a process of classifying the content of the image into a category (Haralick et al., 1973), while object detection is a process of localizing and identifying the instances of semantic objects in an image (Viola et al., 2001). For instance, the object classification result for Fig. 2.7(a) can be the category of *bear*, while the object detection results should also indicate the location of the object belonging to each category. As an illustration, Fig. 2.7(d) shows a *bear* detection result. Object classification and object detection are both pivotal procedures to accomplish a complete image understanding. Representative applications of object classification include image retrieval (Karakasis et al., 2015), scene recognition (Farabet et al., 2013) and image captioning (You et al., 2016), while example applications of object detection include face detection (Sun et al., 2018), pedestrian localization (Rasouli



(a) (b)

Figure 2.6: An example image (a) and a blob detection result (b).



(a) (b)

(c) (d)

Figure 2.7: Illustration of the tasks of image segmentation, superpixel segmentation and object detection. (a) Original image; (b) An image segmentation result; (c) A superpixel segmentation result; (d) A result of *bear* detection.

et al., 2017) and traffic surveillance (Moranduzzo and Melgani, 2013).

2.3. Several fundamental concepts in deep learning

Deep learning techniques have a powerful ability in feature learning, representation and mapping, and have therefore achieved a remarkable success in image analysis (LeCun et al., 2015). A brief introduction to several basic concepts related to deep learning is presented as follows.

2.3.1. Supervised learning

Machine learning is a technique that provides systems with the ability to automatically learn knowledge from experience without being explicitly programmed (Bishop, 2006). From the observations or examples (i.e., training data), a machine learning technique explores their inherent patterns with a strategy, thereby producing an inferred function (i.e., optimized model) to make predictions or decisions for new data (i.e., test data).

The types of machine learning algorithms differ in the training strategies, the types of training data, and the tasks that they are intended to solve. Generally, machine learning techniques can be categorized into unsupervised learning, semi-supervised learning, supervised learning and reinforcement learning (Bishop, 2006). This thesis is mainly related to supervised learning, in which the training data contain both the input and the desired output (i.e., ground truth labels).

Intuitively, the learning strategy should aim at a minimization of the difference between the predictions and the desired output, which is also known as empirical risk minimization. However, this strategy might lead to the problem of overfitting, which degrades the generalization ability. Therefore, the structural risk minimization, which minimizes the summation of the empirical risk and a regularizer, is usually adopted in practice.

2.3.2. Artificial neural networks

Among the existing machine learning techniques, artificial neural networks (ANN) are a kind of model that is inspired by biological neural networks in the human brain (Hopfield, 1988). Artificial neural networks are composed of a collection of connected units called artificial neurons.

An artificial neuron is essentially a mathematical function (Bishop, 2006). It distributes a weight to each input (including a bias), and then computes

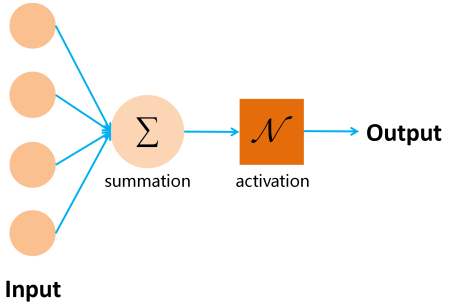


Figure 2.8: Illustration of an artificial neuron.

the sum of the weighted inputs. Subsequently, it passes the sum through a non-linear activation function to obtain the output, as illustrated in Fig. 2.8. Let $\mathbf{X} = [X_1, X_2, \ldots, X_J]^{\mathrm{T}}$ be an input consisting of J elements, s a bias, $\mathbf{w} = [w_1, \ldots, w_J]^{\mathrm{T}}$ the weight vector and $\mathcal{N}(\cdot)$ the activation function. In most cases, the activation function is configured to be a non-linear function. The output of an artificial neuron for the input is given by

$$Z = \mathcal{N}\left(\sum_{j=1}^{J} w_j X_j + s\right) . \tag{2.8}$$

A representative activation function is the sigmoid function, which is given by

$$\mathcal{N}_{\mathrm{s}}(Y) = \frac{1}{1 + e^{-Y}} , \tag{2.9}$$

where Y denotes the input of the function. Another widely used activation function is the rectified linear unit (Hinton, 2010), which is given by

$$\mathcal{N}_{\mathrm{r}}(Y) = \max(0, Y) . \tag{2.10}$$

Artificial neural networks are designed by combining the artificial neurons into layers and combining the layers into networks (Hopfield, 1988). A typical architecture of artificial neural networks is the multi-layer perceptron (MLP), which is a class of feedforward networks wherein connections between the neurons do not form a cycle (Ruck et al., 1990). An MLP consists of at least three kinds of layers: the input layer, hidden layer and output layer. For illustration, we display an example MLP architecture in Fig. 2.9, which has one input layer, three hidden layers and one output layer. Except for the input layer, the input of each layer is composed of the output of the previous layer and a bias. During the training procedure, all the weights in the MLP are adjusted by the so-called backpropagation technique (LeCun et al., 1989),

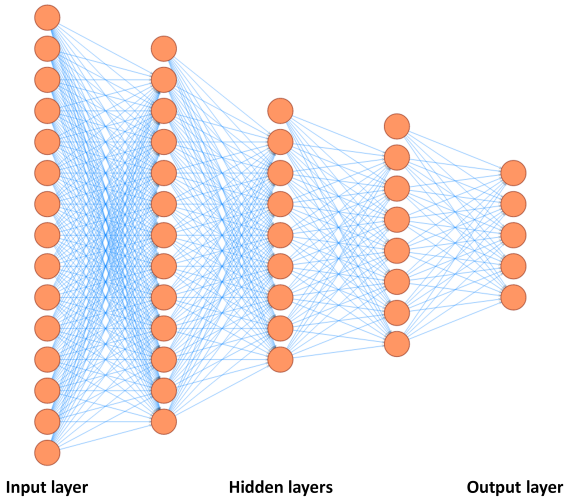


Figure 2.9: Illustration of an example MLP architecture.

which is built on gradient descent optimization.

2.3.3. Deep convolutional neural networks

Capitalizing on artificial neural networks and the developments of computing power, many deep learning techniques have been proposed for image analysis (Krizhevsky et al., 2012). Compared with conventional artificial neural networks, deep learning techniques increase the number of layers through which the data is transformed, and use more complex network architectures (LeCun et al., 2015), thereby obtaining a more powerful ability in feature learning and representation.

Among the deep learning techniques, deep convolutional neural networks (CNN) have become one of the most popular architectures in the computer vision field. As illustrated in Fig. 2.10, a typical CNN architecture consists of a sequence of convolutional layers, pooling layers and fully connected layers (Krizhevsky et al., 2012).

A convolutional layer is composed of convolutional neurons, each of which is represented by a trainable convolutional kernel (LeCun et al., 2015). Each kernel performs a convolutional operation on the input, thereby yielding a response map. Subsequently, the output feature map is obtained by passing the response map through a non-linear activation unit.

A pooling layer is usually used for selecting the most promising features in each feature map (LeCun et al., 2015). It slides a window over the input map.

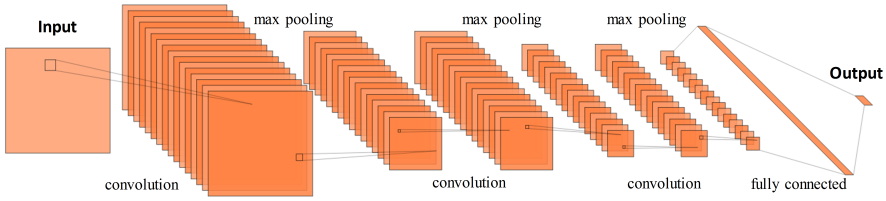


Figure 2.10: Illustration of an architecture using deep convolutional networks.

At each location, the values in the sliding window are aggregated into a single value. As a result, the dimension of the feature map is significantly reduced. A widely adopted pooling scheme is the so-called max pooling. For each input map, a max pooling layer partitions the input map into non-overlapping regions and retains the maximum value in each region. A max pooling layer is fixed and non-trainable.

The initial fully connected layer transforms the ultimate outputs of the convolutional layers and the pooling layers into a vector. Then, each of the following fully connected layer multiplies the layer input by a weight matrix and then adds a bias vector.

For multi-category classification, the predicted probability of each category is usually obtained by a *softmax* activation function. Using the prediction results and the ground truth labels, all the parameters in the CNN are iteratively updated by the backpropagation mechanism (LeCun et al., 1989).

In view of the great success of CNN in object classification, many works have tried to apply CNN for image segmentation. A straightforward scheme is to classify all the image patches cropped by a sliding window (Ciresan et al., 2012), but this scheme inevitably leads to a serious redundancy and a heavy computation. To address such shortcomings, the architecture of fully convolutional networks (FCN) (Long et al., 2015) was proposed for image segmentation. We display a typical FCN architecture in Fig. 2.11. Compared with a full CNN model, an FCN model replaces the fully connected layers in the CNN model with the so-called deconvolution layers. In order to make the sizes of the output map and input image identical, the FCN model upsamples the output of the last convolutional layer by a deconvolution operation, while exploiting the pooling results that are yielded by the intermediate layers of the CNN. In the last layer, the network predicts the category for each pixel using a non-linear activation function, e.g., the sigmoid function. Accordingly, the training loss is an aggregation of the pixel-wise dissimilarity between the prediction and the ground truth.

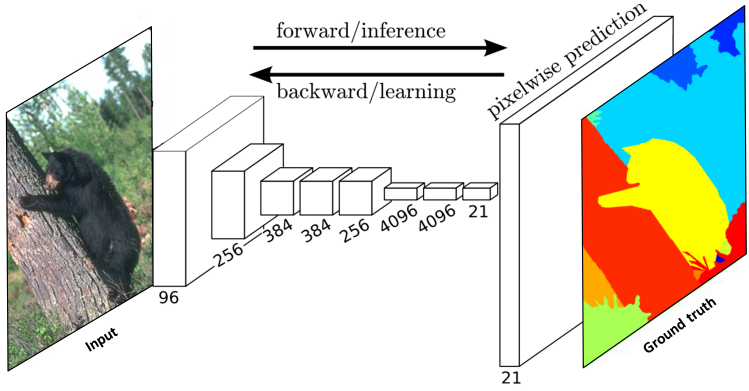


Figure 2.11: Illustration of an architecture using fully convolutional networks.

PART II

EXPLORATION OF GAUSSIAN-BASED KERNELS FOR IMAGE ANALYSIS

3 First-order derivative of anisotropic Gaussian kernel with its applications

In this section, we present a multiscale version of the normalized first-order derivative of anisotropic Gaussian (NFAG) kernel. By proposing a multiscale normalization method, we solve the problem that conventional anisotropic Gaussian kernels (Shui and Zhang, 2012; Zhang et al., 2017b) cannot be applied in scale-space directly. An adaptive anisotropy factor is also designed to alleviate the anisotropy stretch effect. Using NFAG kernels, we obtain the anisotropic edge strength that can benefit two subsequent methods. Firstly, by employing the anisotropic edge strength and hierarchical superpixel contrast, we propose a contour detection method that can exploit both edge features and object shape information. Secondly, we develop a superpixel segmentation method by incorporating the anisotropic edge strength into an existing superpixel segmentation method, in which the distance between neighboring superpixels is determined by both the color information and the edge strength. Experiments performed on publicly available datasets validate the efficacy of the proposed methods.

The material of this chapter is based on the following publications:

- Wang, G. and De Baets, B. (2017). Edge detection based on the fusion of multiscale anisotropic edge strength measurements. In *Proceedings of the Conference of the European Society for Fuzzy Logic and Technology*, volume 3, pages 530–536
- Wang, G., Lopez-Molina, C., and De Baets, B. (2019b). Multiscale edge detection using first-order derivative of anisotropic Gaussian kernels. *Journal of Mathematical Imaging and Vision*, 61(8):1096–1111
- Wang, G. and De Baets, B. (2019a). Contour detection based on anisotropic edge strength and hierarchical superpixel contrast. *Signal, Image and Video Processing*, 13(8):1657–1665
- Wang, G. and De Baets, B. (2019b). Superpixel segmentation based on anisotropic edge strength. *Journal of Imaging*, 5(6):57

3.1. Motivation

Edges are fundamental visual features bridging the gap between image pixels and many image analysis tasks (Li et al., 2015a), like superpixel segmentation (Wang and De Baets, 2019b) and object detection (Zitnick and Dollár, 2014). In general, a good edge detector is expected to reliably detect the locations that reveal critical structural information of the image contents (Coleman et al., 2010), which facilitates subsequent practical applications. Over the past several decades, numerous edge detectors have been developed. Most of them focus on two aspects: (1) computing the edge strength, and (2) binarizing the edge strength into the binary edge map. In this section, we intend to address the former one. The edge strength (a.k.a. edge saliency (Lindeberg, 1998a), edginess (Li and Chen, 1994) or edge response (Bao et al., 2005)) is a measure of the local intensity variation and is widely considered the most relevant representation of edges (Lopez-Molina et al., 2018). For the sake of efficiency and interpretability, the edge strength is usually obtained by differentiation-based methods. These methods can be subdivided into two categories: monoscale methods and multiscale methods.

Early monoscale methods, such as the Sobel method (Sobel, 1970) and the Prewitt method (Prewitt, 1970), employ very simple differential kernels to compute the gradients, and accordingly compute the edge strength by the magnitude of the gradients. These kernels are computationally cheap, yet are of fixed size and sensitive to noise (Zhang et al., 2017b). To overcome these shortcomings, some monoscale approaches employ kernels of which the size is alterable. Marr and Hildreth (Marr and Hildreth, 1980) use the Laplacian of Gaussian as convolution kernel. By adjusting the kernel size through the scale of the Gaussian function, the Laplacian of Gaussian kernels are applicable to either thin edges or wide edges. Nevertheless, it is the Canny detector (Canny, 1986) that has gained the most prominent position in the literature. For detecting isolated step edges corrupted by additive white Gaussian noise, Canny derived the theoretically optimal operator based on three criteria, i.e., good detection, good localization and low spurious response. To make the detector computationally efficient, he uses the first-order derivative of (isotropic) Gaussian (FDG) kernel as an approximation of the optimal operator. However, the scale choice in the Canny method is a dilemma. Although obtaining a better localization, a smaller scale is more sensitive to noise compared with a coarser scale. It is worth noting that Canny also agrees with the need for kernels with multiple scales and multiple orientations (Canny, 1986). Employing the Canny method as a standard, many edge detectors have been developed to revise the Canny method (Ding and Goshtasby, 2001; Jacob and Unser, 2004; Xu et al., 2014b). Zhang et

al. (Zhang et al., 2017b) proposed a method using a directional version of the first-order derivative of anisotropic Gaussian (FAG) kernel, ensuring a good noise-robustness while maintaining a good detection for adjacent edges. The anisotropy of this method makes an indispensable contribution to the improvement of the signal-to-noise ratio (Zhang et al., 2017b). However, due to the fact that the blurring extents of an anisotropic kernel along two orthogonal directions are different, the use of the anisotropic kernel can incur an anisotropy stretch effect. As a consequence, in the resulting edge strength map, there are significant but factitious responses in the vicinity of blobs and corners, and subsequently, these factitious responses will lead to spurious results when a low threshold is used to detect weak edges (Zhang et al., 2017b).

The monoscale edge detectors mentioned above are mostly designed to detect step edges, but edges usually extend over different widths (Perona and Malik, 1990). Due to the focal properties of sensors, the penumbral phenomenon of light, etc., in natural images, one can generally find spatially scaled edges of which the intensity discontinuities vary over different widths (Elder and Zucker, 1998). Therefore, the assumptions of most monoscale edge detectors do not conform to actual situations, and as such, those monoscale detectors are weak at detecting edges with heterogeneous widths (Torre and Poggio, 1986; Elder and Zucker, 1998).

Multiscale edge detection, which is inspired by the mechanism of the human visual system (Atick and Redlich, 1992), has gained increasing attention (Lopez-Molina et al., 2018). Following an early proposal (Rosenfeld and Thurston, 1971), authors have developed different strategies to implement the multiscale detection. One popular strategy is to fuse multiple edge strength maps that are obtained at different scales. Bao et al. (2005) proposed such a method using scale multiplication. Exploiting the fact that the product is large only when all of the factors are large, this method computes the product of the gradients obtained at two different scales as the edge strength. Adopting a similar strategy, Shui and Zhang (2012) developed a method based on both isotropic and anisotropic Gaussian kernels. This method obtains the edge strength by computing the geometric mean of two gradient maps. One gradient map is obtained by isotropic kernels, while the other one is obtained by anisotropic kernels. In this way, the anisotropy stretch effect can also be mitigated. Nonetheless, the employed isotropic kernels are sensitive to noise (Shui and Zhang, 2012), which would lead to factitious responses in the vicinity of true edges (Zhang et al., 2017b). Moreover, both the methods in (Bao et al., 2005) and (Shui and Zhang, 2012) only consider gradient maps at two user-specified scales, and therefore still have limitations in detecting edges with heterogeneous widths.

There are also multiscale methods that find the optimal scale for each edge. Elaborating the need for a multiscale framework, Elder and Zucker (1998) proposed a scale estimation method based on the second-order derivative of the image intensity, but prior knowledge of sensor noise is required. Building on the scale-space theory, Lindeberg (1998a) developed an automatic scale selection method using the normalized and directional isotropic Gaussian kernels. A family of kernels covering all possible scales and orientations is employed to filter the image, and for each pixel, the maximum response among all the kernels is retained to form the edge strength map. We refer to this scheme as *maximum aggregation*. Note that Lindeberg's method is rotationally invariant because a set of kernels with all possible orientations is employed. However, as Lindeberg stated, there is a systematic bias in the edge strength map produced by his method (Lindeberg, 1998a). For a spatially scaled edge, the bias renders the obtained edge strength smaller than the true edge strength. Moreover, like other methods based on isotropic kernels, Lindeberg's method is not robust to noise.

Besides the spatial-domain-based methods mentioned above, there are quite a few wavelet-based methods (Mallat and Hwang, 1992) for edge detection. Mallat and Zhong (1992) proposed an edge detection method using a constructed quadratic spline dyadic wavelet that essentially approximates the first derivative of Gaussian kernel. Therefore, this method is equivalent to the Canny method (Canny, 1986). Zhang and Bao (2002) developed an edge detection method using scale multiplication in the wavelet domain, which can obtain a better noise-robustness than the Canny method. Nevertheless, wavelets usually have a limited capability in extracting directional information. To overcome this shortcoming, some authors employ contourlets (Po and Do, 2006), complex wavelets (Selesnick et al., 2005), curvelets (Gebäck and Koumoutsakos, 2009) and shearlets (Yi et al., 2009) for edge detection. It is claimed that the shearlet-based methods have more desirable properties among the wavelet-based/wavelet-like methods (Duval-Poo et al., 2015). However, the evaluation results reported in (Duval-Poo et al., 2015) show that the shearlet-based methods have limited superiority over the spatial-domain-based methods, e.g., the Canny method.

Contour detection is an image analysis task that is highly related to edge detection. Compared to edge detection, contour detection requires not only low-level features but also mid-level and even high-level cues. In particular, for detecting contours, edges that appear in the textural regions should be suppressed (Martin et al., 2004). To this end, Grigorescu et al. (2003) proposed a contour detection method based on the non-classical receptive field inhibition, which can suppress the values of edge strength in textural regions. However,

this method is based on the surround modulation and mainly emphasizes the orientation tuning property of the non-classical receptive fields (Yang et al., 2014). Adopting a surround inhibition framework, Yang et al. (2014) proposed a method using multiple features, including orientation, luminance and luminance contrast. Although this method can suppress textures, it is time-consuming to execute.

Another typical application of edge detection is superpixel segmentation. A superpixel is a group of pixels that have similar image properties. Compared with pixels, superpixels embody higher-level features and can sometimes reduce the complexity of subsequent image analysis tasks. Quite a few superpixel segmentation methods have been developed (Stutz et al., 2018), most of which can be categorized into partition-based and graph-based methods. Partition-based superpixel segmentation methods initially partition pixels into different segments that look like grid cells, and then iteratively refine the segments until some convergence criteria are satisfied (Wang et al., 2017c). A typical example is the method based on simple linear iterative clustering (SLIC method) (Achanta et al., 2012). However, the superpixels yielded by partition-based methods might not adhere to object contours well, especially when coarse superpixels are desired (Wei et al., 2018). Graph-based superpixel segmentation methods group pixels or superpixels into larger superpixels according to graph-based criteria (Liu et al., 2011). These methods treat each pixel (or superpixel) as a vertex in a graph and use the distance (i.e., dissimilarity) between two neighboring superpixels as arc weight (Stutz et al., 2018). Then, the superpixels are obtained by minimizing a cost function defined over the weighted graph. Recently, Wei et al. (2018) proposed a graph-based method to obtain the so-called superpixel hierarchy (SH method). This method constructs the minimum spanning tree using the Borůvka algorithm, which considers a graph as a forest in which each vertex is initially a tree. These trees grow as the iteration proceeds. For each tree, its aggregate feature value is obtained by aggregating the features (e.g., color) embodied by all the internal vertices. Compared with methods that only use pixel-level features, this method is more robust for segmentation. The SH method benefits greatly from the edge strength, and moreover, it has been confirmed that the segmentation accuracy is dependent on the edge strength measurement methods (Wei et al., 2018).

As reviewed above, edge detection, contour detection and superpixel segmentation are three tasks that can benefit each other. In this section, firstly, we present an edge detection method using NFAG kernels. By proposing a multiscale normalization method, we solve the problem that conventional anisotropic Gaussian kernels (Shui and Zhang, 2012; Zhang et al., 2017b) can-

not be applied in scale-space directly. We also propose an adaptive anisotropy factor whose value decreases as the kernel scale increases to alleviate the anisotropy stretch effect. Secondly, we propose a contour detection method using hierarchical superpixel contrast and the anisotropic edge strength yielded by NFAG kernels. The hierarchical superpixel contrast embodies shape information and can facilitate a textural suppression. The anisotropic edge strength can well reflect the extent of local discontinuities. Therefore, we compute the contour strength map by multiplying the anisotropic edge strength map by the average of the hierarchical superpixel contrast maps. Thirdly, we develop a superpixel segmentation method by incorporate the anisotropic edge strength into an existing superpixel segmentation method, in which the distance between neighboring superpixels is determined by both the color information and the edge strength.

This chapter is organized as follows. Section 3.2 elaborates some preliminary concepts that are of interest to our work. In Section 3.3, we present the NFAG kernel, and subsequently, we use NFAG kernels to compute the anisotropic edge strength. Besides, an adaptive anisotropy factor is also designed to address the anisotropy stretch effect. Section 3.4 and Section 3.5 present the applications of NFAG kernels in contour detection and superpixel segmentation, respectively. We list the conclusions in Section 3.6.

3.2. Related work

This section presents several concepts related to this work: the scale-space representation of images, the FAG kernel and graph-based superpixel segmentation.

3.2.1. Normalized scale-space representation and its derivative

In image analysis, the scale-space framework has been applied to a large variety of topics, such as image matching (Lindeberg, 2015), line detection (Wang et al., 2019c) and blob reconstruction (Kong et al., 2013a). The scale-space representation of a signal is obtained by convolving the signal with Gaussian kernels (Lindeberg, 2013). Considering the fact that the magnitude of a non-normalized Gaussian kernel decreases over scales, Lindeberg introduced a γ-parameterized normalized Gaussian kernel as follows (Lindeberg, 1998a):

$$\hat{g}(\mathbf{x};\sigma) = \sigma^{2\gamma} \cdot g(\mathbf{x};\sigma)\,, \tag{3.1}$$

where $\mathbf{x} = [x, y]^{\mathrm{T}}$ denotes the planar coordinates, $\sigma \in \mathbb{R}_+$ stands for a scale in scale-space, $\gamma \in \mathbb{R}_+$ is referred to as the scale normalization factor and $g(\mathbf{x}; \sigma)$ represents the non-normalized isotropic Gaussian kernel:

$$g(\mathbf{x}; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\sigma^2}\right). \tag{3.2}$$

From Eq. (3.1), it is easy to get the γ-parameterized normalized version of the FDG kernel:

$$\hat{g}'(\mathbf{x}; \sigma) = \sigma^{2\gamma} \cdot g'(\mathbf{x}; \sigma), \tag{3.3}$$

where $g'(\mathbf{x}; \sigma)$ is the first-order derivative of $g(\mathbf{x}; \sigma)$ w.r.t. the argument x in $\mathbf{x}$:

$$g'(\mathbf{x}; \sigma) = -\frac{x}{\sigma^2} \cdot g(\mathbf{x}; \sigma). \tag{3.4}$$

According to Lindeberg's scale-space framework (Lindeberg, 1998a), given a two-dimensional signal $\mathbf{I}(\mathbf{x})$, its normalized scale-space representation $\mathbf{I}_{\mathrm{ss}}(\mathbf{x}; \sigma)$ is computed by filtering the signal with the γ-parameterized normalized Gaussian kernel:

$$\begin{aligned} \mathbf{I}_{\mathrm{ss}}(\mathbf{x}; \sigma) &= \hat{g}(\mathbf{x}; \sigma) * \mathbf{I}(\mathbf{x}) \\ &= \big(\sigma^{2\gamma} \cdot g(\mathbf{x}; \sigma)\big) * \mathbf{I}(\mathbf{x}), \end{aligned} \tag{3.5}$$

where $*$ denotes the convolution operation. Likewise, using Eq. (3.3), the normalized first-order derivative of the scale-space representation is given by:

$$\begin{aligned} \mathbf{I}'_{\mathrm{ss}}(\mathbf{x}; \sigma) &= |\hat{g}'(\mathbf{x}; \sigma) * \mathbf{I}(\mathbf{x})| \\ &= \big|\big(\sigma^{2\gamma} \cdot g'(\mathbf{x}; \sigma)\big) * \mathbf{I}(\mathbf{x})\big|. \end{aligned} \tag{3.6}$$

Lindeberg's method (Lindeberg, 1998a) computes the edge strength at all the scales using Eq. (3.6). Subsequently, for each edge, the matched scale is identified as the scale that makes the edge strength locally maximal in scale-space.

3.2.2. The non-normalized first-order derivative of anisotropic Gaussian kernel

A key development to the conventional Canny kernel is the use of anisotropic Gaussian kernels (Shui and Zhang, 2012; Zhang et al., 2017b; Shui and Wang, 2017). Anisotropic Gaussian kernels have also found applications in corner

detection (Shui and Zhang, 2013) and ridge detection (Lopez-Molina et al., 2015).

The directional version of an anisotropic Gaussian kernel is defined by (Shui and Zhang, 2012; Wang and De Baets, 2017):

$$g(\mathbf{x};\sigma,\varphi,\theta) = \frac{1}{2\pi\varphi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\mathbf{x}^{\mathrm{T}}\mathbf{R}_\theta^{\mathrm{T}}\begin{bmatrix}1 & 0\\ 0 & \varphi^{-2}\end{bmatrix}\mathbf{R}_\theta\mathbf{x}\right), \tag{3.7}$$

where $\varphi \geq 1$ represents the anisotropy factor,

$$\mathbf{R}_\theta = \begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix} \tag{3.8}$$

denotes the rotation matrix, and $\theta \in [0,\, 2\pi[$ stands for the orientation.

Note that the anisotropic Gaussian kernel in Eq. (3.7) reduces to an isotropic version when $\varphi = 1$. From Eq. (3.7), the FAG kernel is given by:

$$g'(\mathbf{x};\sigma,\varphi,\theta) = -\frac{[\cos\theta,\sin\theta]\mathbf{x}}{\sigma^2}g(\mathbf{x};\sigma,\varphi,\theta)\,. \tag{3.9}$$

In the methods that use first-order derivative of Gaussian kernels (Shui and Zhang, 2012; Zhang et al., 2017b), for a specified scale, a family of first-order derivative of Gaussian kernels covering all possible orientations is used to filter the image signal, and, subsequently, the maximum response is selected as the edge strength. Correspondingly, the direction of each edge is indicated by the kernel that produces the maximum response.

It has been verified that the use of an anisotropy factor helps to improve the robustness to noise (Shui and Zhang, 2012; Zhang et al., 2017b). Nonetheless, the conventional methods only address the edge detection at one or two scales, and as such, they still have limitations in detecting edges with heterogeneous widths.

3.2.3. Graph-based superpixel segmentation

As introduced in Chapter 2, image segmentation and superpixel segmentation are two highly related tasks. Image segmentation refers to the procedure of partitioning an image into segments that have coherent semantic meanings. Therefore, image segmentation methods require either interactions with users or sufficient data to train *ad hoc* models (Arbelaez et al., 2011). Superpixel segmentation groups pixels that have coherent color or other low-level properties, and as such, superpixel segmentation usually yields an oversegmented

result.

Both image segmentation and superpixel segmentation are supposed to be hierarchical tasks, because even for a human observer, it is difficult to determine a unique meaningful segmentation of a given image (Xu et al., 2017b). Illustrations can be found in Section 3.5.2. In superpixel segmentation, the hierarchical level is usually determined by the number of superpixels. In most cases, the number of superpixels is manually specified (Achanta et al., 2012).

Among existing superpixel segmentation methods, graph-based methods are more appropriate to generate hierarchical superpixels, since they are usually built on graph theory, which intrinsically provides a hierarchical structure. Moreover, graph-based methods are also efficient to implement.

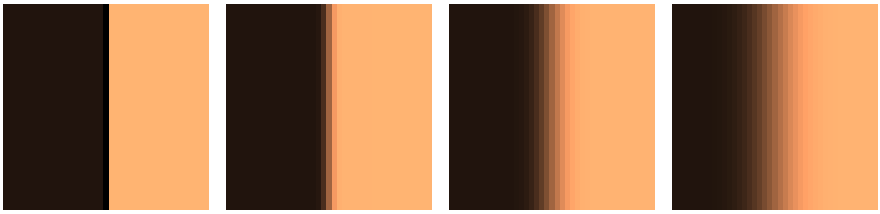
In mathematics, a weighted graph is a structure consisting of a set of vertices, in which some pairs of vertices are connected by weighted arcs. Particularly, an undirected graph is a graph whose arcs have no orientation. In graph-based superpixel segmentation methods, each pixel in a given image is represented by a vertex, while the distance between two neighboring pixels is represented by an arc weight (Felzenszwalb and Huttenlocher, 2004). Then, the superpixels are obtained by finding the minimum spanning tree of the graph. A spanning tree is an undirected, weighted, and acyclic subgraph. The minimum spanning tree is the spanning tree that has the lowest total arc weight among all the possible spanning trees. Since superpixel segmentation is a procedure of grouping pixels that are meaningfully similar, many graph-based methods segment the image by constructing a minimum spanning tree (Saglam and Baykan, 2017). In the literature, some superpixel segmentation methods use the Kruskal algorithm (Calderero and Marques, 2010) and the Borůvka algorithm (Wei et al., 2018) to construct the minimum spanning tree.

3.3. Normalized first-order derivative of anisotropic Gaussian kernel

In this section, we propose a multiscale version of the NFAG kernel to identify the edge scales while yielding the multiscale edge strength maps. We also design an adaptive anisotropy factor to address the anisotropy stretch effect.

3.3.1. Modelling the scaled edges

Many existing edge detectors are designed with the assumption that the local variation of image intensity shows a step appearance. In fact, besides step



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

Figure 3.1: Examples of edges with different widths. Their scales are $\omega_0 \approx 0$ (a), $\omega_0 = 1$ (b), $\omega_0 = 3$ (c) and $\omega_0 = 5$ (d), respectively.

edges, there are also spatially scaled edges in real images (McIlhagga, 2011). In this work, we model the variation of image intensity using scaled edges, which can be regarded as the convolutional result of a step edge and a Gaussian kernel:

$$\boldsymbol{\Xi}_0(\mathbf{x}) = g(\mathbf{x}; \omega_0) * \big(c_0 \ \mathcal{H}\big([\cos\theta_0, \sin\theta_0]\mathbf{x}\big) + b_0\big) \ , \tag{3.10}$$

where

$$\mathcal{H}(\tilde{x}) = \begin{cases} 0, & \text{if } \tilde{x} < 0 \\ 1, & \text{if } \tilde{x} \geq 0 \end{cases} \tag{3.11}$$

is the Heaviside step function, and

$$g(\mathbf{x}; \omega_0) = \frac{1}{2\pi\omega_0^2} \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\omega_0^2}\right) \tag{3.12}$$

is a Gaussian kernel, in which $\theta_0 \in [0, \pi[$ denotes the normal direction of the edge, $\omega_0 \in \mathbb{R}_+$ stands for the edge scale which determines the spatial width of intensity variation, $c_0 \in [0, 1]$ is a constant that controls the true edge strength (i.e., extent of intensity change) and b_0 ($b_0 + c_0 \leq 1$) denotes the base level reflecting the background intensity. Note that a scaled edge modelled by Eq. (3.10) reduces to a step edge in case $\omega_0 \approx 0$. As a matter of fact, with respect to digital images, we always have $\omega_0 > 0$, since digital images are discrete signals. In addition, practical edge detectors usually impose an intrinsic smoothing on the image to get rid of the interference from noise, and as such, we have $\omega_0 \geq 1$ in most actual situations.

As an illustration, Fig. 3.1 displays several edges with different widths, and Fig. 3.2 shows their horizontal profiles through the image centers.

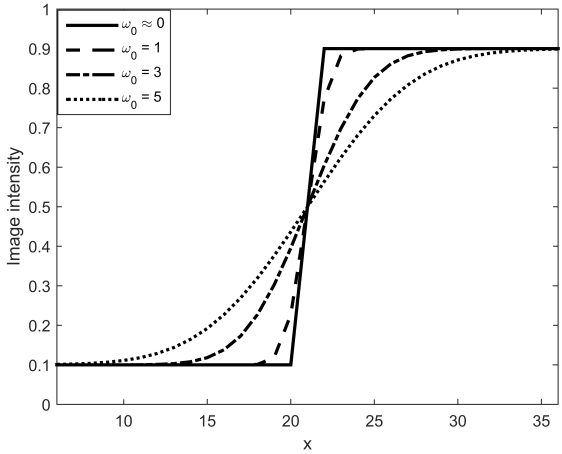


Figure 3.2: The horizontal intensity profile through the center of the image shown in Fig. 3.1(a), 3.1(b), 3.1(c) and 3.1(d), respectively.

3.3.2. Normalization in scale-space

Conventional non-normalized FAG kernels have been applied in edge detection (Shui and Zhang, 2012; Zhang et al., 2017b). To study their performance in scale-space, from Eqs. (3.9) and (3.10), we compute the maximum edge strength obtained by non-normalized FAG kernels at the center location of the modelled edge. It has been proved that among all the directions, a kernel for edge detection achieves its maximum response on the direction $\theta = \theta_0$ (Shui and Zhang, 2012). Then, the maximum response of non-normalized FAG kernels is given by:

$$\begin{aligned} E_{\text{non}} &= \max_{\theta} |g'(\mathbf{x}; \sigma, \varphi, \theta) * \boldsymbol{\Xi}_0(\mathbf{x})|_{\mathbf{x}=\mathbf{0}} \\ &= |g'(\mathbf{x}; \sigma, \varphi, \theta_0) * ((c_0 H(\tilde{x}) + b_0) * g(\tilde{\mathbf{x}}; \omega_0))|_{\mathbf{x}=\mathbf{0}} \\ &= \frac{c_0}{\sqrt{2\pi(\omega_0^2 + \sigma^2)}}. \end{aligned} \tag{3.13}$$

The detailed proof is presented in Appendix A.1. Note that the edge strength in Eq. (3.13) is independent of the anisotropy factor φ. Unfortunately, it can be learned that E_{non} is a monotonically decreasing function of σ. This means the obtained edge strength E_{non} decreases as the kernel scale increases. Therefore, we conclude that the non-normalized FAG kernels cannot be applied in scale-space directly.

According to the analysis above, we learn that non-normalized FAG kernels cannot identify the scale of the observed edge. To solve this problem, a

novel normalization method is needed. In literature, Lindeberg proposed a γ-parameterized normalization method for Gaussian kernels (Lindeberg, 1998a), as formulated in Eq. (3.3). Inspired by this method, we obtain the NFAG for edge detection by normalizing the FAG kernel as follows:

$$g'_{\mathrm{E}}(\mathbf{x};\sigma,\varphi,\theta) = \beta\sigma^{2\gamma}\cdot g'(\mathbf{x};\sigma,\varphi,\theta)\,, \tag{3.14}$$

where β is a factor that controls the extent of the filtering response. We determine the parameters as follows. As mentioned earlier, the maximum response of directional FAG kernels occurs when $\theta=\theta_0$. In scale-space, we compute the edge strength E_{norm} obtained by the NFAG kernel at the center location of an edge model as follows:

$$\begin{aligned} E_{\mathrm{norm}} &= \max_{\theta} |g'_{\mathrm{E}}(\mathbf{x};\sigma,\varphi,\theta) * \boldsymbol{\Xi}_0(\mathbf{x})|_{\mathbf{x}=\mathbf{0}} \\ &= \big|\big(\beta\sigma^{2\gamma}\cdot g'(\mathbf{x};\sigma,\varphi,\theta_0)\big) * ((c_0 H(\tilde{x})+b_0) * g(\tilde{\mathbf{x}};\omega_0))\big|_{\mathbf{x}=0} \\ &= \beta c_0 \frac{\sigma^{2\gamma}}{\sqrt{2\pi(\omega_0^2+\sigma^2)}}\,. \end{aligned} \tag{3.15}$$

The detailed proof is presented in Appendix A.2. To study the monotonicity of E_{norm} in scale-space, we compute the derivative of E_{norm} w.r.t. σ as follows:

$$\frac{\partial E_{\mathrm{norm}}}{\partial\sigma} = \frac{\sqrt{2\pi}}{\pi}\beta c_0\gamma\sigma^{2\gamma-1}(\omega_0^2+\sigma^2)^{-\frac{1}{2}} - \frac{\sqrt{2\pi}}{2\pi}\beta c_0\sigma^{2\gamma+1}(\omega_0^2+\sigma^2)^{-\frac{3}{2}}. \tag{3.16}$$

For $\sigma\in\mathbb{R}_+$, it is easy to verify that the second-order derivative of E_{norm} w.r.t. σ is negative. Therefore, by setting the partial derivative in Eq. (3.16) to zero, we find that E_{norm} reaches its maximum value at the scale

$$\sigma^* = \sqrt{\frac{2\gamma}{1-2\gamma}}\omega_0\,. \tag{3.17}$$

Since $\sigma\in\mathbb{R}_+$, we have $\gamma\in]0,0.5[$. To make E_{norm} reach its maximum value at the original edge scale, i.e., $\sigma^*=\omega_0$, we set $\gamma=0.25$. Accordingly, the edge strength in Eq. (3.15) becomes

$$E^*_{\mathrm{norm}} = \frac{\beta c_0}{2\sqrt{\pi\omega_0}}\,. \tag{3.18}$$

Based on Eq. (3.18), β should be set as $\beta=2\sqrt{\pi\omega_0}$ to make E^*_{norm} equal c_0, because in this case, the obtained edge strength would precisely reflect the true edge strength. However, the parameter ω_0, i.e., the scale of the observed edge, is

unknown beforehand. Fortunately, from Eqs. (3.16) and (3.17), the maximum edge strength value in scale-space and the scale identification procedure are both independent of the argument β. Here, we set $\beta = 2\sqrt{\pi}$ at first, and accordingly, the edge strength in Eq. (3.18) becomes

$$E^*_{\text{norm}} = \frac{c_0}{\sqrt{\omega_0}}\,. \tag{3.19}$$

Once the edge scale ω_0 is identified, the edge strength will be compensated by a multiplication factor $\sqrt{\omega_0}$ to make the obtained edge strength equal the true edge strength c_0. The compensation procedure will be elaborated later. Note that in Eq. (3.19), E^*_{norm} does not depend on the anisotropy factor φ, which means that the choice of φ will not affect the measure of edge strength in a noise-free case.

Hence, having $\beta = 2\sqrt{\pi}$ and $\gamma = 0.25$, from Eq. (3.14), the NFAG kernel is given by:

$$g'_{\text{E}}(\mathbf{x}; \sigma, \varphi, \theta) = 2\sqrt{\pi}\sigma^{\frac{1}{2}} \cdot g'(\mathbf{x}; \sigma, \varphi, \theta)\,. \tag{3.20}$$

Consequently, using a family of NFAG kernels covering all possible scales and orientations, we are able to identify the scales of the observed edges while obtaining the edge strength maps in scale-space.

3.3.3. Alleviation of the anisotropy stretch effect

We have presented the multiscale normalization method for FAG kernels, but the problem of the anisotropy stretch effect, as mentioned in the previous sections, still remains unsolved. In order to obtain a good noise-robustness and to alleviate the anisotropy stretch effect, based on an analysis in terms of signal-to-noise ratio (SNR), we propose an adaptive anisotropy factor for the NFAG kernel in this section.

It has been validated that the non-normalized FAG kernel obtains higher robustness to noise compared with isotropic Gaussian kernels (Shui and Zhang, 2012). Next, we theoretically analyze the noise-robustness of the NFAG kernels. To this end, we adopt the SNR, which is defined as the quotient of the maximum signal response and the standard deviation of the filtered noise (McIlhagga, 2011), to represent the capability of a kernel to suppress noise (Li et al., 2015b).

Suppose that the scale set of the bank of NFAG kernels is $\mathbb{S} = \{\sigma \mid \sigma_{\min} \leq \sigma \leq \sigma_{\max}\}$ and the image to be processed is corrupted by zero-mean white Gaussian noise $\xi(\mathbf{x})$ with variance ε_0^2. According to (Canny, 1986; Shui and

Zhang, 2012; McIlhagga, 2011), the intensity of filtered noise is represented by the root-mean-squared response. Let

$$\begin{aligned}\tilde{\mathbf{x}} &= [\tilde{x}, \tilde{y}]^{\mathrm{T}} \\ &= \begin{bmatrix} \cos\theta_0 & \sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 \end{bmatrix} \mathbf{x} \end{aligned} \tag{3.21}$$

be the rotated plane coordinates. Based on the computing method presented in (Canny, 1986), we obtain the intensity of noise filtered by an NFAG kernel at a given scale σ as follows:

$$\begin{aligned}\varepsilon_{\mathrm{norm}} &= \varepsilon_0 \sqrt{\iint_{\mathbb{R}^2} \left(g'_{\mathrm{E}}(\mathbf{x}; \sigma, \varphi, \theta)\right)^2 \mathrm{d}\mathbf{x}}\Bigg|_{\theta=\theta_0} \\ &= 2\sqrt{\pi}\sigma^{\frac{1}{2}}\varepsilon_0 \sqrt{\iint_{\mathbb{R}^2} \frac{\tilde{x}^2}{4\pi^2\varphi^2\sigma^8} \exp\left(-\frac{\tilde{x}^2 + \varphi^{-2}\tilde{y}^2}{\sigma^2}\right) \mathrm{d}\tilde{\mathbf{x}}} \\ &= 2\sqrt{\pi}\sigma^{\frac{1}{2}}\varepsilon_0 \sqrt{\frac{1}{4\pi^2\varphi^2\sigma^8} \int_{-\infty}^{+\infty} \tilde{x}^2 \exp\left(-\frac{\tilde{x}^2}{\sigma^2}\right) \mathrm{d}\tilde{x} \cdot \int_{-\infty}^{+\infty} \left(-\frac{\tilde{y}^2}{(\varphi\sigma)^2}\right) \mathrm{d}\tilde{y}} \\ &= \frac{\varepsilon_0}{\sigma\sqrt{2\varphi\sigma}}\,. \end{aligned} \tag{3.22}$$

It can be seen that $\varepsilon_{\mathrm{norm}}$ has an inverse relationship with the kernel scale σ as well as the anisotropy factor φ. Thus, for a given scale set $\mathbb{S} = \{\sigma \mid \sigma_{\min} \leq \sigma \leq \sigma_{\max}\}$, kernels that have smaller scales tend to produce larger noise responses, which therefore are more likely to be selected as the final edge strength. Thus, the intensity of noise in the edge strength is mainly determined by the small-scale kernels.

When ω_0 is within the interval $[\sigma_{\min}, \sigma_{\max}]$, multiscale NFAG kernels yield the maximum response at the scale $\sigma^* = \omega_0$. According to Equations (3.18) and (3.22), the obtained SNR is computed as:

$$\begin{aligned}\mathrm{SNR}^*_{\mathrm{norm}} &= \frac{E^*_{\mathrm{norm}}}{\varepsilon_{\mathrm{norm}}}\Bigg|_{\sigma=\sigma_{\min}} \\ &= \frac{2c_0\sigma^2_{\min}\sqrt{\varphi}}{\varepsilon_0\sqrt{2\omega_0}\sigma_{\min}}\,. \end{aligned} \tag{3.23}$$

Therefore, the proposed NFAG kernel ($\varphi > 1$) obtains a higher SNR than the isotropic kernel ($\varphi = 1$). Also, when we use NFAG kernels to detect an edge in a noisy image, the SNR is mainly determined by φ and $\sigma_{\min}$. Thus, we learn that it is not necessary to apply anisotropy factors in kernels at all the scales, especially in kernels at large scales, since the latter can ensure a

good robustness to noise even if the anisotropy factor is absent.

It has been reported that the use of first-order derivative of anisotropic Gaussian kernels incurs an anisotropy stretch effect (Shui and Zhang, 2012), because the blurring extent w.r.t. x is determined by σ, while the blurring extent w.r.t. y is determined by $\varphi\sigma$. To address this problem, we embed an adaptive anisotropy factor in NFAG kernels.

Given a scale set $\mathbb{S} = \{\sigma \mid \sigma_{\min} \leq \sigma \leq \sigma_{\max}\}$, we obtain an adaptive anisotropy factor as follows:

$$\varphi(\sigma) = \begin{cases} \frac{\sigma_{\text{con}}^2}{\sigma^2} & \text{, if } \sigma_{\min} \leq \sigma \leq \sigma_{\text{con}} \\ 1 & \text{, otherwise} \end{cases}, \tag{3.24}$$

where $\sigma_{\text{con}} \in [\sigma_{\min}, \sigma_{\max}]$ stands for the robustness control scale. On the one hand, when we use kernels at $\sigma \leq \sigma_{\text{con}}$, we introduce a large anisotropy factor to improve the robustness to noise. On the other hand, when we use a kernel at $\sigma > \sigma_{\text{con}}$, the anisotropy factor is set as $\varphi = 1$ because a large σ is already able to guarantee a good robustness to noise.

By substituting Eq. (3.24) into Eq. (3.23), we have

$$\text{SNR}_{\text{norm}}^{*} = \frac{2c_0\sigma_{\min}\sigma_{\text{con}}}{\varepsilon_0\sqrt{2\omega_0\sigma_{\min}}}. \tag{3.25}$$

It can be seen that once the scale set $\mathbb{S}$ is given, the SNR of the edge strength can be controlled by σ_{con}.

As for the setting of the control scale, we generally recommend to set σ_{con} as the geometric mean of $\sigma_{\min}$ and $\sigma_{\max}$, because in this case the blurring extent at each scale is determined by $\sigma\varphi$, which is given by

$$\sigma\varphi = \sigma \cdot \frac{\sigma_{\text{con}}^2}{\sigma^2} = \frac{\sigma_{\min}\sigma_{\max}}{\sigma}. \tag{3.26}$$

Since $\sigma_{\min} \leq \sigma \leq \sigma_{\max}$, it holds that $\sigma\varphi \leq \sigma_{\max}$, which means that in the resulting multiscale edge strength, the blurring extent over all the directions would be mainly determined by $\sigma_{\max}$. Consequently, the anisotropy stretch effect can be alleviated by using an adaptive anisotropy factor.

Eventually, by substituting Eq. (3.24) into Eqs. (3.9) and (3.20), we get NFAG kernels with an adaptive anisotropy factor.

3.3.4. Discrete filter bank

In order to accommodate the proposed NFAG kernels to digital image processing, discrete versions of these kernels are needed. We obtain both the discrete Gaussian kernel and the discrete NFAG kernel by sampling the formulae in Eqs. (3.7), (3.9) and (3.20) in the 2D integer coordinate $\mathbb{Z}^2$:

$$g(\mathbf{m};\sigma_i,\varphi_i,\theta_j) = \frac{1}{2\pi\varphi_i\sigma_i^2}\exp\left(-\frac{1}{2\sigma_i^2}\mathbf{m}^{\mathrm{T}}\mathbf{R}_{\theta_j}^{\mathrm{T}}\begin{bmatrix}1 & 0\\ 0 & \varphi_i^{-2}\end{bmatrix}\mathbf{R}_{\theta_j}\mathbf{m}\right)$$
$$g'_{\mathrm{E}}(\mathbf{m};\sigma_i,\varphi_i,\theta_j) = -2\sqrt{\pi}\sigma_i^{\frac{1}{2}}\cdot\frac{[\cos\theta_j,\sin\theta_j]\mathbf{m}}{\sigma_i^2}g(\mathbf{m};\sigma_i,\varphi_i,\theta_j)\,, \tag{3.27}$$

where

$$\varphi_i = \max\left(\frac{\sigma_{\mathrm{con}}^2}{\sigma_i^2},\ 1\right),$$
$$\mathbf{R}_{\theta_j} = \begin{bmatrix}\cos\theta_j & \sin\theta_j\\ -\sin\theta_j & \cos\theta_j\end{bmatrix}, \tag{3.28}$$

in which $\mathbf{m} = [m_x, m_y]^{\mathrm{T}}$ stands for the image coordinates, $\sigma_i \in \mathbb{S}$, $\theta_j \in \mathbb{D}$ and φ_i denote the scale, orientation and adaptive anisotropy factor, respectively, while $\mathbb{S}$ and $\mathbb{D}$ denote the scale set and orientation set, respectively. Examples of discrete NFAG kernels are illustrated in Fig. 3.3.

Natural images have either one channel (grayscale images) or three channels (color images). Let a 2D image signal $\mathbf{I}^{(q)}(\mathbf{m})$ denote the q-th channel of a given image $\mathbf{I}(\mathbf{m})$. A filter bank of discrete NFAG kernels covering all the possible scales and orientations is employed to convolve each channel of the image:

$$\mathbf{E}^{(q)}(\mathbf{m};\sigma_i,\varphi_i,\theta_j) = \left|g'_{\mathrm{E}}(\mathbf{m};\sigma_i,\varphi_i,\theta_j) * \mathbf{I}^{(q)}(\mathbf{m})\right|\,. \tag{3.29}$$

Note that $q = 1$ for grayscale images while $q \in \{1,2,3\}$ for color images.

At each location, by employing the maximum aggregation scheme in scale-space for each channel and, subsequently, by selecting the maximum value of the edge strength over all the channels (Koschan and Abidi, 2005; Wang and Shui, 2016), we obtain the maximum response map by

$$\mathbf{E}(\mathbf{m}) = \max_{q}\max_{\sigma_i\in\mathbb{S}}\max_{\theta_j\in\mathbb{D}}\mathbf{E}^{(q)}(\mathbf{m};\sigma_i,\varphi_i,\theta_j)\,. \tag{3.30}$$

As an illustration, Fig. 3.4(c) shows the resulting maximum response map obtained on Fig. 3.4(a). In contrast, Fig. 3.4(b) shows an edge strength map obtained at a single scale. Obviously, Fig. 3.4(c) has a better representation

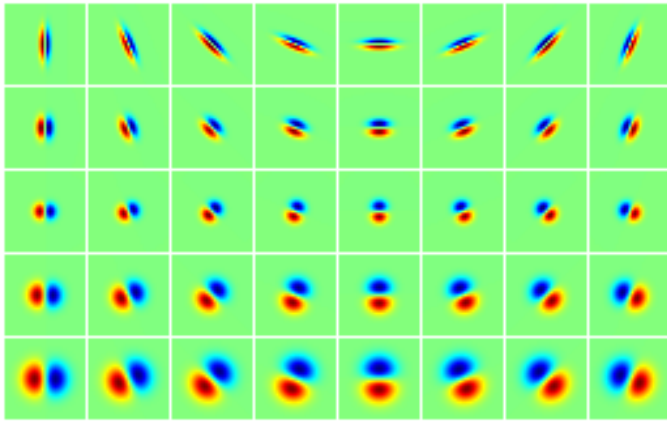


Figure 3.3: Examples of discrete NFAG kernels. The control scale is set as $\sigma_{\mathrm{con}} = 2$. Top row: Kernels with $\sigma = 1$ and $\varphi = 4$. Second row: Kernels with $\sigma = 1.5$ and $\varphi = 1.78$. Third row: Kernels with $\sigma = 2$ and $\varphi = 1$. Fourth row: Kernels with $\sigma = 3$ and $\varphi = 1$. Bottom row: Kernels with $\sigma = 4$ and $\varphi = 1$. The intensity range of each patch has been adjusted for better display.

for the spatially scaled edges than Fig. 3.4(b).

From Eqs. (3.16), (3.17) and (3.18), we learn that the response reaches its maximum value at the scale of the original edge scale, and as such, at each pixel position, the scale map is obtained by maximizing $\mathbf{E}^{(q)}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)$ in scale-space. More specifically, we obtain the scale map $\mathbf{S}_{\mathrm{E}}(\mathbf{m})$ by

$$\mathbf{S}_{\mathrm{E}}(\mathbf{m}) = \underset{\sigma_i \in \mathbb{S}}{\operatorname{argmax}} \max_{q} \max_{\theta_j \in \mathbb{D}} \mathbf{E}^{(q)}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)\,. \tag{3.31}$$

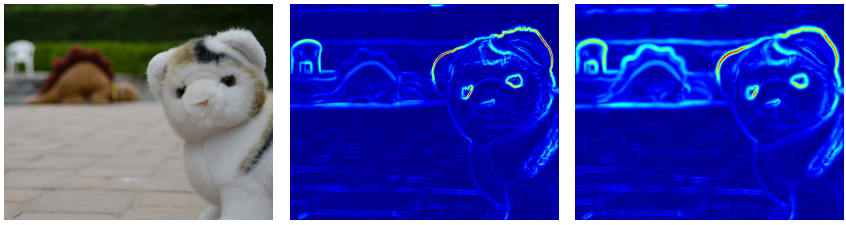
In addition, the edge direction map $\mathbf{\Theta}_{\mathrm{E}}(\mathbf{m})$ is obtained by maximizing the edge strength as follows:

$$\mathbf{\Theta}_{\mathrm{E}}(\mathbf{m}) = \underset{\theta_j \in \mathbb{D}}{\operatorname{argmax}} \max_{q} \max_{\sigma_i \in \mathbb{S}} \mathbf{E}^{(q)}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)\,. \tag{3.32}$$

3.3.5. Compensated anisotropic edge strength

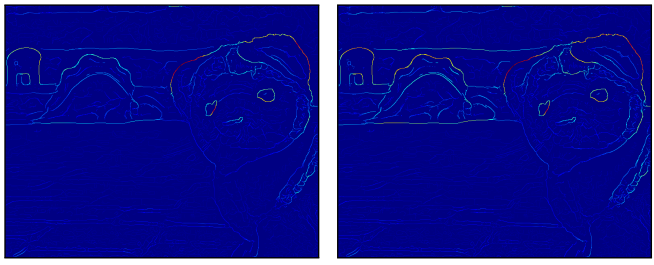
As mentioned earlier, there is a bias in the edge strength obtained by Eq. (3.30). Fortunately, this bias can be compensated once the edge scale is identified. We recommend a scale-based compensation procedure for the edge strength map, especially in the scenario that many spatially scaled edges occur in the image.

In practice, an edge detection result is usually represented by an edge map



(a) (b) (c)

Figure 3.4: Illustration of the monoscale response map and multiscale response map. (a) The original image (courtesy of Dave Johnson); (b) Response map obtained at a single scale; (c) Response map obtained by multiscale NFAG kernels.



(a) (b)

Figure 3.5: Comparison of the NMS result and the compensated NMS result. (a) The NMS result of Fig. 3.4(c); (b) The compensated NMS result.

consisting of curved centerlines (one pixel wide) of edges. The candidate pixels of edge centerlines can be identified by the technique of nonmaxima suppression (NMS) (Rosenfeld et al., 1972). Thus, for the sake of computational efficiency, there is no need to compensate the edge strength at all the positions. In other words, it suffices to only compensate the edge strength at positions of candidates of edge centerlines. According to Eq. (3.19), the compensation factor is determined by the square root of the identified scale. Denoting the result of the NMS procedure as $\mathbf{E}_{\mathrm{nms}}(\mathbf{m})$, we obtain the compensated edge strength map as

$$\mathbf{E}_{\mathrm{ae}}(\mathbf{m}) = \mathbf{E}_{\mathrm{nms}}(\mathbf{m}) \cdot \sqrt{\mathbf{S}_{\mathrm{E}}(\mathbf{m})}\,. \tag{3.33}$$

As an illustration, the NMS result of Fig. 3.4(c) is shown in Fig. 3.5(a), while the compensated edge strength map is shown in Fig. 3.5(b). Comparing Fig. 3.5(b) to Fig. 3.5(a), it can be seen that the edge strength of some spatially scaled edges has been enhanced by the compensation procedure.

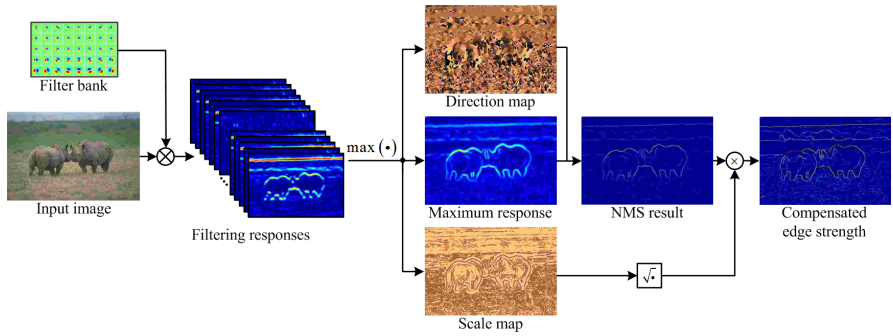


Figure 3.6: The flowchart of our framework for obtaining anisotropic edge strength.

In Fig. 3.6, we show the flowchart of our framework for extracting anisotropic edge strength. Moreover, we summarize the proposed method to compute the anisotropic edge strength in Algorithm 1.

3.4. Application to contour detection

Contour detection is an image analysis task that is highly related to edge detection. In this section, we present a contour detection method using the anisotropic edge strength and hierarchical superpixel contrast. We also evaluate the proposed method on a widely adopted dataset.

3.4.1. Hierarchical superpixel maps

In literature, many superpixel segmentation methods have been developed (Wei et al., 2018). We adopt the scheme of region merging, obtaining a set of superpixel maps by hierarchically merging fine superpixels at different hierarchy levels (Hu et al., 2013).

Firstly, the superpixel segmentation method based on simple linear iterative clustering (SLIC method) (Achanta et al., 2012) is employed to generate a superpixel map at a fine level. Denoting the number of pixels in the image by $|\mathbf{I}|$, we set the desired number of SLIC superpixels to be

$$N_{\mathrm{sp}}^{(0)} = \left\lceil \frac{|\mathbf{I}|}{\varpi \times \varpi} \right\rceil , \tag{3.34}$$

where the symbol $\lceil \cdot \rceil$ denotes the ceiling function and ϖ denotes the grid cell size of the initially latticed image. The local binary pattern map as well as its local contrast are computed simultaneously (Ojala et al., 2002).

Secondly, the obtained superpixels are organized as a region adjacency

Algorithm 1 The proposed method for computing anisotropic edge strength

Require: Image $\mathbf{I}$, scale set $\mathbb{S}$, orientation set $\mathbb{D}$, control scale σ_{con}
Ensure: Anisotropic edge strength $\mathbf{E}_{\text{ae}}$

for each $\mathbf{I}^{(q)} \in \mathbf{I}$ **do**
  for each $\sigma_i \in \mathbb{S}$ **do**
    $\varphi_i \leftarrow \max\left(\frac{\sigma_{\text{con}}^2}{\sigma_i^2},\ 1\right)$
    for each $\theta_j \in \mathbb{D}$ **do**
      $\mathbf{E}^{(q)}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j) \leftarrow \left|g'_{\text{E}}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j) * \mathbf{I}^{(q)}(\mathbf{m})\right|$
    end for
  end for
end for
$\mathbf{E}(\mathbf{m}) \leftarrow \max\limits_{q} \max\limits_{\sigma_i \in \mathbb{S}} \max\limits_{\theta_j \in \mathbb{D}} \mathbf{E}^{(q)}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)$
$\mathbf{S}_{\text{E}}(\mathbf{m}) \leftarrow \operatorname{argmax}\limits_{\sigma_i \in \mathbb{S}} \max\limits_{q} \max\limits_{\theta_j \in \mathbb{D}} \mathbf{E}^{(q)}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)$
$\mathbf{\Theta}_{\text{E}}(\mathbf{m}) \leftarrow \operatorname{argmax}\limits_{\theta_j \in \mathbb{D}} \max\limits_{q} \max\limits_{\sigma_i \in \mathbb{S}} \mathbf{E}^{(q)}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)$
$\mathbf{E}_{\text{nms}}(\mathbf{m}) \leftarrow$ NMS using $\mathbf{E}(\mathbf{m})$ and $\mathbf{\Theta}_{\text{E}}(\mathbf{m})$
$\mathbf{E}_{\text{ae}}(\mathbf{m}) \leftarrow \mathbf{E}_{\text{nms}}(\mathbf{m}) \cdot \sqrt{\mathbf{S}_{\text{E}}(\mathbf{m})}$

graph. In this graph, each arc weight represents the merging cost $\Psi_{i,j}$ of two superpixels:

$$\Psi_{i,j} = \frac{|\mathbb{P}_i| \cdot |\mathbb{P}_j|}{|\mathbb{P}_i| + |\mathbb{P}_j|} \cdot \frac{1}{\Phi_{i,j}^{\iota}} \cdot \left(w_{\text{c}} D_{\text{c}} + w_{\text{t}} D_{\text{t}}\right), \tag{3.35}$$

where $|\mathbb{P}_i|$ and $|\mathbb{P}_j|$ are the numbers of pixels in two neighbouring superpixels $\mathbb{P}_i$ and $\mathbb{P}_j$, $\Phi_{i,j}$ is the length of the shared border, $\iota \in [0, 1]$ is a parameter to adjust the constraint of the shared border, D_{c} and D_{t} denote the distances in terms of color and texture, while w_{c} and w_{t} are the corresponding weights of D_{c} and D_{t}. The use of ι makes the neighboring superpixels with a larger $\Phi_{i,j}$ have a lower merging cost. We empirically set ι as 0.5. To determine w_{c} and w_{t}, we compute the histograms of each superpixel on each image channel. Then, we obtain w_{c} and w_{t} as

$$w_{\text{c}} = \sqrt{\min\left(\bar{h}_i,\ \bar{h}_j\right)} \quad \text{and} \quad w_{\text{t}} = 1 - w_{\text{c}}\,, \tag{3.36}$$

respectively, where $\bar{h}_i = \frac{1}{3}\sum_{q=1}^{3} h_i^{(q)}$, in which $h_i^{(q)}$ denotes the maximum frequency value in the histogram obtained on the q-th channel of $\mathbb{P}_i$, while $\bar{h}_j$ is obtained likewise.

Then, the adjacent superpixels with the lowest merging cost are grouped into a single region. This step is carried out iteratively until the number of

Figure 3.7: Illustration of hierarchical superpixel maps (left column), the corresponding superpixel mean color maps (middle column) and the superpixel contrast maps (right column). In the images in the left column, the number of superpixels is 3152, 394, 50, 7 and 4, respectively.

superpixels reduces to the desired number, thereby resulting in hierarchical superpixel maps. The desired number of superpixels at each level is:

$$N_{\mathrm{sp}}^{(l)} = \lceil N_{\mathrm{sp}}^{(0)} \cdot 2^{-l} \rceil \,, \tag{3.37}$$

where $l \in \{1, 2, 3, \ldots\}$ denotes the hierarchy level.

As an illustration, the left column of Fig. 3.7 displays several results of superpixel merging at different levels.

3.4.2. Hierarchical superpixel contrast maps

Due to the regular spatial shapes, kernels with very large sizes will blur the image seriously and result in gradient maps that show limited visual information (Lindeberg, 1998b). By contrast, superpixels have spatial shapes that are adapted to the image contents, and therefore, superpixels can better capture the image contents at coarse levels. Contours are the borders between different objects (Yang et al., 2014). Thus, in a superpixel map, the border between two adjacent superpixels that have a larger difference is more likely to belong to the contours. For an efficient implementation, we measure the difference between two neighbouring superpixels using the color contrast. Given a superpixel map at the level l, the mean color of each superpixel on each channel is computed as:

$$\Upsilon_i^{(l,\,q)} = \frac{1}{|\mathbb{P}_i^{(l)}|} \sum_{\mathbf{m}\in\mathbb{P}_i^{(l)}} \mathbf{I}^{(q)}(\mathbf{m})\,, \tag{3.38}$$

where $\mathbb{P}_i^{(l)}$ denotes the i-th superpixel at the l-th level and $|\mathbb{P}_i^{(l)}|$ represents the number of pixels in $\mathbb{P}_i^{(l)}$. As an illustration, the middle column of Fig. 3.7 displays the superpixel mean color maps at different levels. The superpixel mean color maps retain the image contents hierarchically. The textures have been suppressed well in the maps at coarse levels. This facilitates the recognition of contours that are situated in textures.

In a color image, the color contrast between two neighbouring superpixels, which is referred to as the superpixel contrast, is obtained by:

$$\Omega_{i,j}^{(l)} = \max_{q\in\{1,2,3\}} \Upsilon_i^{(l,\,q)} - \Upsilon_j^{(l,\,q)}\,. \tag{3.39}$$

Then, we obtain the superpixel contrast as follows:

$$\mathbf{\Omega}^{(l)}(\mathbf{m}) = \begin{cases} \Omega_{i,j}^{(l)} & \text{, if } \mathbf{m} \text{ is on the border of } \mathbb{P}_i^{(l)} \text{ and } \mathbb{P}_j^{(l)} \\ 0 & \text{, otherwise} \end{cases}\,. \tag{3.40}$$

The right column of Fig. 3.7 displays the superpixel contrast maps at different levels. It can be seen that positions situated on contours are more likely to show significant values in the superpixel contrast maps. Nevertheless, since the borders of superpixels are artificially depicted, in the superpixel contrast maps, some positions where the contours do not exist also have significant values. This problem will be addressed in the next step.

3.4.3. Contour strength map

As discussed earlier, the anisotropic edge strength is a measurement of the local discontinuity of the image intensity, but textures also have significant local discontinuities. The superpixel contrast maps suppress textures well while embodying some mid-level information, such as the shapes of objects and the layout of the image contents, but the borders of superpixels are artificially depicted, and as such, some positions at which the contours do not exist may also show significant values.

To exploit the advantages of the anisotropic edge strength and the hierarchical superpixel contrast, we compute the contour strength at each hierarchy level:

$$\mathbf{C}^{(l)}(\mathbf{m}) = \mathbf{\Omega}^{(l)}(\mathbf{m}) \cdot \mathbf{E}_{\mathrm{ae}}(\mathbf{m}) \,, \tag{3.41}$$

where $\mathbf{E}_{\mathrm{ae}}$ is the anisotropic edge strength obtained in Eq. (3.33).

In this way, only positions with both significant edge strength and significant superpixel contrast show significant contour strength. To aggregate the contour strength at different levels, we compute the average of all the contour strength maps as the final contour strength:

$$\mathbf{C}(\mathbf{m}) = \frac{1}{|\mathbb{L}|} \sum_{l \in \mathbb{L}} \mathbf{C}^{(l)}(\mathbf{m}) \,, \tag{3.42}$$

where $\mathbb{L}$ denotes the set of hierarchy levels and $|\mathbb{L}|$ stands for the number of hierarchy levels.

3.4.4. Binarization

Having obtained the contour strength map, we next require a binarization procedure to segment and thin the contour strength map into a binary contour map. As briefly mentioned in Section 3.1, quite a few techniques have been proposed (Ray, 2013) to implement the binarization procedure. We straightforwardly adopt the widely used hysteresis segmentation (Canny, 1986; Shui and Zhang, 2013; Wang and Shui, 2016; Zhang et al., 2017b) to produce the binary contour map.

Hysteresis segmentation is a double-thresholding process (Canny, 1986). It employs an upper threshold τ_1 and a lower threshold τ_2. Here, we empirically set $\tau_2 = \tau_1/2$. Mostly, hysteresis segmentation is realized in two steps. Given a contour strength map processed by the NMS procedure, all the locations at which the contour strength exceeds the upper threshold are first labelled as contour pixels. The locations at which the contour strength is between

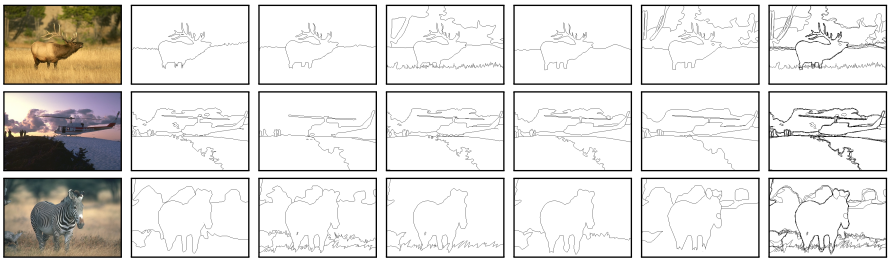


Figure 3.8: Sample images and the ground truth contour maps from the BSDS500 dataset. Left column: Original images. Second to sixth columns: Ground truth contour maps labelled by different annotators for each original image. Right column: Aggregated ground truth for each original image.

the upper threshold and the lower threshold are labelled as potential contour pixels. For each of these potential contour pixels, if there is a path to connect it with a contour pixel, it is also marked as a contour pixel. Subsequently, all these contour pixels form the binary detection result $\mathbf{C}_{\mathrm{d}}$.

3.4.5. Experimental validation

We have proposed a contour detection method based on the anisotropic edge strength and the hierarchical superpixel contrast. To evaluate its performance, like many contour detection studies (Pan et al., 2014; Yang et al., 2015a), we test the proposed method on the Berkeley Segmentation Data Set and benchmarks 500 (BSDS500) (Martin et al., 2004). In the BSDS500 dataset [1], there are 200 test images as well as their ground truth contour maps, which are labelled by different annotators (Martin et al., 2004). We illustrate three sample images as well as their multiple ground truth contour maps in Fig. 3.8. Despite the deviations between annotations depicted by different annotators, the overall ground truth contour maps reflect the contours in the original images very well. Note that there are a lot of spatially scaled edges in these natural images. To make the evaluation more convincing, our method will also be compared with a well-known contour detection method and several recently proposed methods.

Evaluation metrics

Given a contour detection result and its corresponding ground truth contour map(s), an evaluation procedure is needed to determine how well the detection result approximates the ground truth. Since contour detection can be considered as a binary classification procedure discriminating contour pixels

[1] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources

form non-contour pixels, the comparison between the detection result and the ground truth can be formulated in terms of success and failure. In this respect, the performance can be evaluated by the precision-recall framework. Precision is the probability that a detected contour pixel is a true contour pixel, while recall is the probability that a true contour pixel is successfully detected. In practice, we carry out the performance evaluation using the paradigm proposed by Martin (Martin, 2003), which is also widely adopted in literature (Arbelaez et al., 2011; Dollár and Zitnick, 2015).

Let $\mathbf{C}_{\mathrm{d}}$ be a contour detection result and let $\mathbb{I}_{\mathrm{gt}} = \left(\mathbf{I}_{\mathrm{gt}}^{(1)}, \mathbf{I}_{\mathrm{gt}}^{(2)}, \ldots, \mathbf{I}_{\mathrm{gt}}^{(N_{\mathrm{gt}})}\right)$ be its corresponding ground truth contour map(s), where $N_{\mathrm{gt}} \in \mathbb{N}_{+}$ denotes the total number of ground truth contour map(s). The Cost Scaling Assignment (CSA) algorithm (Goldberg and Kennedy, 1995) is employed to perform a pixel-to-pixel matching between $\mathbf{C}_{\mathrm{d}}$ and each of the ground truth contour maps in $\mathbb{I}_{\mathrm{gt}}$. On the one hand, for each detected contour pixel in $\mathbf{C}_{\mathrm{d}}$, if it matches any of the ground truth contour pixels in $\mathbb{I}_{\mathrm{gt}}$ within a spatial tolerance distance, this detected contour pixel is considered as a true positive detection pixel; otherwise, it is considered as a false positive detection pixel. On the other hand, for each true contour pixel in each ground truth contour map, if it is matched by a detected contour pixel in $\mathbf{C}_{\mathrm{d}}$ within a spatial tolerance distance, this ground truth contour pixel is counted as a matched ground truth pixel; otherwise, it is counted as an unmatched ground truth pixel. These matched ground truth pixels and unmatched ground truth pixels in all the ground truth map(s) compose the aggregate matched ground truth pixels and aggregate unmatched ground truth pixels, respectively. In this way, the precision and recall are computed by:

$$\textsc{Prec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}\,, \tag{3.43}$$

and

$$\textsc{Rec} = \frac{\mathrm{MT}}{\mathrm{MT} + \mathrm{UM}}\,, \tag{3.44}$$

where TP, FP, MT and UM are the numbers of true positive detection pixels, false positive detection pixels, aggregate matched ground truth pixels and aggregate unmatched ground truth pixels, respectively. Different thresholds lead to different detection results, thus resulting different Prec-Rec pairs, which can subsequently be interpolated to form a Precision-Recall (PR) curve. For two PR curves, the curve that is farther from the origin is preferred. That is, a curve that has larger area under the PR curve indicates a better detection performance (Arbelaez et al., 2011). The area under the PR curve can also be measured by the *Average Precision* (AP) (Dollár and Zitnick, 2015). A

deeper insight in evaluation paradigms can be found in (Lopez-Molina et al., 2013, 2016).

To determine the optimal threshold for a contour detector, the F-measure, which computes the harmonic mean of precision and recall as expressed in Eq. (3.45), is used as an evaluation measure:

$$F = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}} . \tag{3.45}$$

There are two ways to determine the optimal threshold in terms of the F-measure. The first way is to apply an identical threshold to all the contour strength maps, and accordingly the threshold yielding the largest F-measure (F_{ODS}) is named as the optimal dataset scale (ODS) threshold. The second way is to apply a separate threshold to the contour strength maps one by one, and for each contour strength map, the threshold yielding the largest F-measure is selected to form the optimal image scale (OIS) thresholds. Accordingly, we can obtain the OIS F-measure (F_{OIS}) using the OIS thresholds.

We also adopt the evaluation measure *Recall at* 50% *Precision* (R50), which reflects the detection accuracy when the recall is high (Dollár and Zitnick, 2015).

Experimental setup

To make the evaluation more convincing, we compare our method with several competing methods, including the Canny method (Canny, 1986), the method based on isotropic and anisotropic Gaussian kernels (IAGK method) (Shui and Zhang, 2012), the method based on a gradient matrix with anisotropic Gaussian direction derivatives (GM-AGDD method) (Wang and Shui, 2016) and the method based on automatic anisotropic Gaussian kernels (AAGK method) (Zhang et al., 2017b). All the competing methods are differentiation-based and Gaussian-based methods, representing different strategies of measuring the contour strength. To highlight the impact of different strategies on contour strength measurement, the binarization procedure is kept identical for all the methods.

The source codes of the IAGK and GM-AGDD methods have been obtained from the authors[2], while the codes of the Canny and AAGK methods are reproduced according to the original papers. In this experimental validation section, each method has been configured as follows:

- Canny: The scale is set to be $\sqrt{2}$. This setting is widely adopted in

[2] http://see.xidian.edu.cn/faculty/plshui/resources.htm/

literature (Shui and Zhang, 2012; Zhang et al., 2017b; Shui and Wang, 2017).

- IAGK^2: According to the original setting (Shui and Zhang, 2012), the scale is specified as 4; the anisotropy factor is set to be $2\sqrt{2}$ and the number of kernel orientations is set to be 16.
- GM-AGDD^2: The scale is set to be 6; the anisotropy factor is specified as 6 and the number of kernel orientations is set to be 16. These parameter settings are originally recommended in (Wang and Shui, 2016).
- AAGK: The number of kernel orientations is originally set to be 8 (Zhang et al., 2017b), and accordingly, the scale and the anisotropy factor are computed as $\sqrt{10}$ and $\sqrt{5}$, respectively.
- *Proposed*: For the bank of NFAG kernels, the scale set is configured as $\mathbb{S} = \{2 + 0.2 \cdot (i-1) \,|\, i = 1, 2, \ldots, 11\}$, the control scale is set as 2, and the number of kernel orientations is set as 8. The grid cell size of the initially latticed image in the SLIC method is set as $\varpi = 13$, and accordingly, the number of superpixels in each superpixel map is computed by Eq. (3.34) or Eq. (3.37).

Furthermore, in the binarization procedure, the upper threshold is specified with values from 0.01 to 0.99 with a step of 0.01. The matching tolerance distance in the evaluation procedure is set to be 0.75% of the image diagonal. This setting is widely adopted in literature (Arbelaez et al., 2011; Liu et al., 2017; Xie and Tu, 2017).

Experimental results

The PR curves obtained by all the methods are illustrated in Fig. 3.9. It is easy to see that the proposed method outperforms all the other competing methods, since its PR curve is the farthest from the origin among all the methods. We also report the quantitative evaluation results in Tab. 3.1. As we can see, the proposed method achieves a better performance than the competing methods in terms of F_{ODS}, F_{OIS}, AP and R50.

We illustrate samples of contour detection results in Fig. 3.10. Compared with the competing methods, our method detects more true contours while yielding less false contours. For instance, in the results illustrated in the first row of Fig. 3.10, our method depicts a full outline of the foreground flower, while all the competing methods fail at depicting the full outline. Moreover, due to the interference from textures, the competing methods yield many false contours within the area of leaves in the detection results. In contrast,

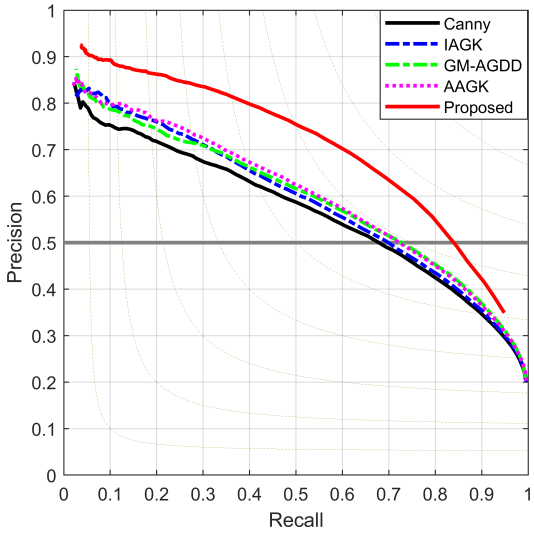


Figure 3.9: PR curves of different methods obtained on the BSDS500 dataset.

Table 3.1: Evaluation results obtained by different methods on the BSDS500 dataset.

Methods	F_{ODS}	F_{OIS}	AP	R50
Canny	0.58	0.61	0.57	0.68
IAGK	0.58	0.63	0.59	0.70
GM-AGDD	0.59	0.63	0.59	0.73
AAGK	0.59	0.64	0.60	0.72
Proposed	**0.67**	**0.70**	**0.76**	**0.84**

the result yielded by our method shows few false contours in the textural areas, which demonstrates that our method can suppress textures in contour detection.

With respect to the time efficiency, we test each method on a PC configured with Intel Core i7-3770 CPU (3.40GHz) with 16-GB RAM, running MATLAB R2018a. The runtime of each method is reported in Tab. 3.2. As can be seen, our method is more efficient than the GM-AGDD method. Although consuming more runtime than the Canny, IAGK and AAGK methods, our method obtains a better detection performance at a cost of acceptable runtime.

Figure 3.10: Detection results of different methods on 10 sample images from the BSDS500 dataset. Green pixels represent true positive detection pixels, blue ones stand for false positive detection pixels and red ones denote unmatched ground truth pixels. Note that the matched ground truth pixels are also colored in green. Contours are thickened to two pixels wide for better illustration. Please zoom electronically for a better view.

Table 3.2: Evaluation results in terms of runtime.

Methods	Canny	IAGK	GM-AGDD	AAGK	Proposed
Runtime (s)	0.05	0.52	4.80	0.27	2.14

3.5. Application to superpixel segmentation

In this section, we introduce the graph-based SH method (Wei et al., 2018) for superpixel segmentation. The SH method uses a graph to represent the image, and constructs a minimum spanning tree using the Borůvka algorithm to obtain the superpixel map. In order to improve its performance in segmentation accuracy, we incorporate the anisotropic edge strength, which has been obtained in Section 3.3, into the distance measure between neighboring superpixels.

3.5.1. Superpixel segmentation incorporating anisotropic edge strength

In graph-based image processing, given an image $\mathbf{I}$, each pixel $\mathbf{I}(\mathbf{m})$ is represented by a vertex $V_i \in \mathbb{V}$ and the distance between two neighboring pixels, V_i and V_j, is represented by an arc weight $\mathcal{A}(V_i, V_j)$. All the arc weight values form the arc weight set $\mathbb{A}$. Then $\mathbb{V}$ and $\mathbb{A}$ form an undirected graph $\mathbb{U} = (\mathbb{V}, \mathbb{A})$. To construct a minimum spanning tree, initially, for each vertex, the Borůvka algorithm finds its nearest neighbor in terms of the Euclidean distance in color space, and then groups them into a single tree. Subsequently, for each tree, the Borůvka algorithm finds its nearest neighbor. Denoting by $\mathbb{T}_1$ and $\mathbb{T}_2$ a pair of neighboring trees, we compute the distance between them as follows:

$$\mathcal{T}(\mathbb{T}_1, \mathbb{T}_2) = \min_{V_i \in \mathbb{T}_1,\, V_j \in \mathbb{T}_2} \mathcal{A}(V_i, V_j)\,. \tag{3.46}$$

Essentially, $\mathcal{T}(\mathbb{T}_1, \mathbb{T}_2)$ is a pixel-level distance measure, and as such, it is reasonable when the number of vertices in each tree is small. However, when the trees grow rather large, $\mathcal{T}(\mathbb{T}_1, \mathbb{T}_2)$ could hardly reflect the dissimilarity between a pair of neighboring trees. Therefore, the pixel grouping procedure is carried out for 4 iterations (Wei et al., 2018).

We have obtained the preliminary superpixel map by a pixel grouping procedure, in which each superpixel is represented by a tree in a graph. Nevertheless, as mentioned earlier, the distance measure defined in Eq. (3.46) is inappropriate to be used for segmenting superpixels at higher hierarchy

levels, since it only makes use of pixel-level features. As the superpixel grows, the pixel-level features are sensitive to outliers, e.g., noisy pixels. Therefore, in the following iterations, the distance between a pair of neighboring trees is defined as follows:

$$\mathcal{T}(\mathbb{T}_1, \mathbb{T}_2) = \mathcal{T}_{\mathrm{e}}(\mathbb{T}_1, \mathbb{T}_2) \cdot \mathcal{T}_{\mathrm{c}}(\mathbb{T}_1, \mathbb{T}_2)\,, \tag{3.47}$$

where $\mathcal{T}_{\mathrm{e}}$ represents the mean edge strength of the pixels that are situated on the shared border between $\mathbb{T}_1$ and $\mathbb{T}_2$. In the superpixel segmentation method presented in (Wei et al., 2018), the edge strength is obtained by the learning-based structured forest method (Dollár and Zitnick, 2015). In this section, we use the NFAG-based anisotropic edge strength obtained in Eq. (3.33) to compute $\mathcal{T}_{\mathrm{e}}(\mathbb{T}_1, \mathbb{T}_2)$, thereby applying the anisotropic edge strength to superpixel segmentation.

In addition, in Eq. (3.47), $\mathcal{T}_{\mathrm{c}}$ denotes the Chi-Squared histogram distance (Pele and Werman, 2010) between $\mathbb{T}_1$ and $\mathbb{T}_2$, which is computed as (Arbelaez et al., 2011):

$$\begin{aligned}\mathcal{T}_{\mathrm{c}}(\mathbb{T}_1, \mathbb{T}_2) &= \chi^2(h_{\mathbb{T}_1}, h_{\mathbb{T}_2}) \\ &= \frac{1}{2}\sum_{i}^{N_{\mathrm{bin}}} \frac{\left(h_{\mathbb{T}_1}(i) - h_{\mathbb{T}_2}(i)\right)^2}{h_{\mathbb{T}_1}(i) + h_{\mathbb{T}_2}(i)}\,,\end{aligned} \tag{3.48}$$

where $h_{\mathbb{T}_1}$ and $h_{\mathbb{T}_2}$ are the color histograms of $\mathbb{T}_1$ and $\mathbb{T}_2$, respectively, i denotes the bin index and N_{bin} stands for the number of bins in each histogram. According to the parameter settings in (Wei et al., 2018), N_{bin} is set to 20.

According to the distance measure in Eq. (3.47), for each tree, the Borůvka algorithm finds its nearest tree and groups them into a single tree. As the iteration continues, superpixel maps at higher hierarchy levels can be obtained.

3.5.2. Experimental validation

We have presented the SH-based superpixel segmentation method which incorporates the NFAG-based edge strength (SH+NFAG method). In order to test whether or not the SH+NFAG method works well, we use our method to obtain superpixels on three publicly available datasets, including the BSDS500 dataset[3] (Arbelaez et al., 2011), the systematic benchmarking for aerial image segmentation (SBAIS) dataset[4] (Yuan et al., 2013) and the

[3] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping
[4] http://jiangyeyuan.com/ASD

neuronal structures in electron microscopy stacks dataset (NSEMS) dataset[5]. In addition, to explore the impact of different kinds of edge strength on the segmentation accuracy, we also test the SH methods that incorporate edge strength obtained by the structured forest edge (SH+SFE) method (Dollár and Zitnick, 2015), the sparseness-constrained color-opponency (SH+SCO) method (Yang et al., 2015a), the automated anisotropic Gaussian kernel (SH+AAGK) method (Zhang et al., 2017b) and the surrounded-modulation edge detection (SH+SED) method (Akbarinia and Parraga, 2018). Furthermore, we also select the widely used SLIC method (Achanta et al., 2012) for comparison.

Evaluation metrics

In order to evaluate the segmentation accuracy, we adopt two widely used evaluation measures: the Achievable Segmentation Accuracy (ASA) and the Undersegmentation Error (UE).

The ASA reflects the fraction of ground truth (GT) segments that are correctly labelled by superpixels. When compared with the GT, a correctly segmented superpixel is supposed to be totally contained in a GT segment. Otherwise, the superpixel overlaps with more than one GT segment. Then, we use the label value of the GT segment that has the largest overlapping region with this superpixel to label all the pixels within this superpixel. Consequently, a map $\mathbf{I}_{\text{lb}}$ consisting of such labels is obtained. Comparing this map with the GT, we compute the fraction of the correct labels in $\mathbf{I}_{\text{lb}}$ as the ASA. That is, given a superpixel map $\mathbf{I}_{\text{sp}}$ and the corresponding ground truth $\mathbf{T}_{\text{sp}}$, the ASA is computed as (Wei et al., 2018):

$$\text{ASA} = \frac{1}{\sum\limits_{\mathbb{Q}_j \in \mathbf{T}_{\text{sp}}} |\mathbb{Q}_j|} \sum_{\mathbb{P}_i \in \mathbf{I}_{\text{sp}}} \max_{\mathbb{Q}_j \in \mathbf{T}_{\text{sp}}} |\mathbb{P}_i \cap \mathbb{Q}_j| \,, \tag{3.49}$$

where $\mathbb{P}_i$ and $\mathbb{Q}_j$ denote the segments in $\mathbf{I}_{\text{sp}}$ and $\mathbf{T}_{\text{sp}}$, respectively, and $|\cdot|$ stands for the number of pixels. For a superpixel segmentation method, a higher ASA value is preferred.

The UE reflects the *leakage* of superpixels with respect to the corresponding GT (Stutz et al., 2018). Comparing the superpixels with the GT, we affiliate each superpixel with the GT segment that has the largest mutually overlapping region. Then, within each superpixel, the pixels that incorrectly match the GT segment are considered as *leakage*. Given a superpixel map $\mathbf{I}_{\text{sp}}$ and the corresponding ground truth $\mathbf{T}_{\text{sp}}$, the UE is computed as (Stutz et al., 2018):

[5] http://brainiac2.mit.edu/isbi_challenge/home

$$\mathrm{UE} = \frac{1}{\sum\limits_{\mathbb{Q}_j \in \mathbf{T}_{\mathrm{sp}}} |\mathbb{Q}_j|} \sum_{\mathbb{Q}_j \in \mathbf{T}_{\mathrm{sp}}} \sum_{\mathbb{P}_i \cap \mathbb{Q}_j \neq \emptyset} \min \left\{ |\mathbb{P}_i \cap \mathbb{Q}_j| \, , \, |\mathbb{P}_i \setminus \mathbb{Q}_j| \right\} , \tag{3.50}$$

where $\mathbb{P}_i$ and $\mathbb{Q}_j$ denote the segments in $\mathbf{I}_{\mathrm{sp}}$ and $\mathbf{T}_{\mathrm{sp}}$, respectively, and $\setminus$ stands for the set difference operation. For superpixel segmentation, a lower UE is preferred.

Parameter settings

As mentioned earlier, we have selected several methods, including the SLIC, SH+SFE, SH+SCO, SH+AAGK and SH+SED methods, as the competing methods. To make the evaluation results reproducible, we list the parameter settings of each method:

- SLIC: According to the original implementation, the compactness is set to be 10 (Achanta et al., 2012).
- SH+SFE: The compactness parameter is set as 0.53. The SFE method, in which the number of trees is set as 4, is trained on the BSDS500 dataset (Dollár and Zitnick, 2015).
- SH+SCO: The compactness parameter is set as 0.53. According to the parameter setting reported in (Yang et al., 2015a), the size of the receptive field is set as 1.1. The number of orientations is set as 8. The connection weights from cones to retinal ganglion cells are set as 0.7 and -0.7, respectively. The size of the local window for sparseness measure is set as 11.
- SH+AAGK: The compactness parameter is set as 0.53. According to (Zhang et al., 2017b), the number of kernel orientations is set as 8, and accordingly, the scale and the anisotropy factor are computed as $\sqrt{10}$ and $\sqrt{5}$, respectively.
- SH+SED: The compactness parameter is set as 0.53. Adopting the implementation of SED in (Akbarinia and Parraga, 2018), the size of the receptive field in the lateral geniculate nucleus layer and in the primary visual cortex is set as 0.5 and 1.5, respectively. The number of directions is set as 12.
- SH+NFAG: The compactness parameter is set as 0.53. The number of directions is set as 8. The scale set is configured as $\mathbb{S} = \{1.0, 1.1, 1.2, \ldots, 1.5\}$, while the control scale is set as 1.2.

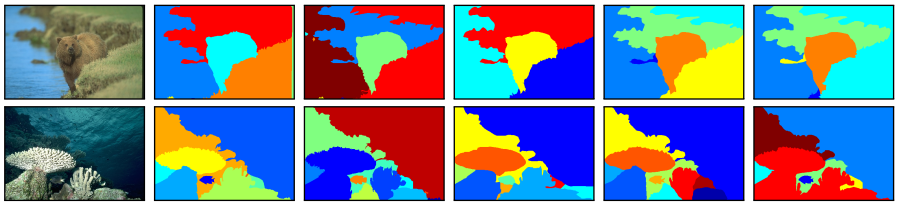


Figure 3.11: Sample images as well as their multiple GT segmentation maps taken from the BSDS500 dataset. Left column: Original images. Second to sixth columns: GT segmentation maps labelled by different annotators for each original image.

Evaluation on the BSDS500 dataset

The BSDS500 dataset contains 200 test images and represents the diversity of natural images. The resolution of each image is either $481 \times 321 \times 3$ or $321 \times 481 \times 3$. The corresponding GT segmentation maps of each test image are labelled by different annotators (Arbelaez et al., 2011). We illustrate two sample images as well as their multiple GT segmentation maps in Figure 3.11. For each image, when comparing a superpixel map with the multiple GT segmentation maps, we obtain multiple ASA and UE values. The best values are retained as the evaluation results.

For each method, the curves of the average ASA and average UE values over all 200 images with respect to different numbers of superpixels are displayed in Figures 3.12(a) and 3.12(b). Among the SH-based methods, the SH+NFAG and SH+SFE method obtain a competitive performance, achieving higher average ASA values and lower average UE values than other methods. This indicates that the edge strength maps obtained by the NFAG and SFE method are more appropriate for superpixel segmentation. By comparison, the SH+SCO and SH+AAGK methods obtain slightly worse performances, while the SH+SED method obtains a limited performance. The main reason is that the SED method mainly focuses on texture suppression. The edge strength map yielded by the SED method is usually fragmented. Such edge strength will inevitably lead to errors when used in the SH method for superpixel segmentation. Moreover, compared with SH-based methods, the SLIC method obtains a competitive performance when the number of superpixels is large, but underperforms when coarse superpixels are desired. This is mainly because the SLIC method is a partition-based method. It takes the edge strength into consideration only in the initialization step.

Besides the quantitative evaluation results, we also illustrate sample superpixel maps yielded by different methods in Figure 3.13. It can be seen that, compared with the SLIC method, the SH-based methods, especially the SH+NFAG and SH+SFE methods, adhere better to the object contours.

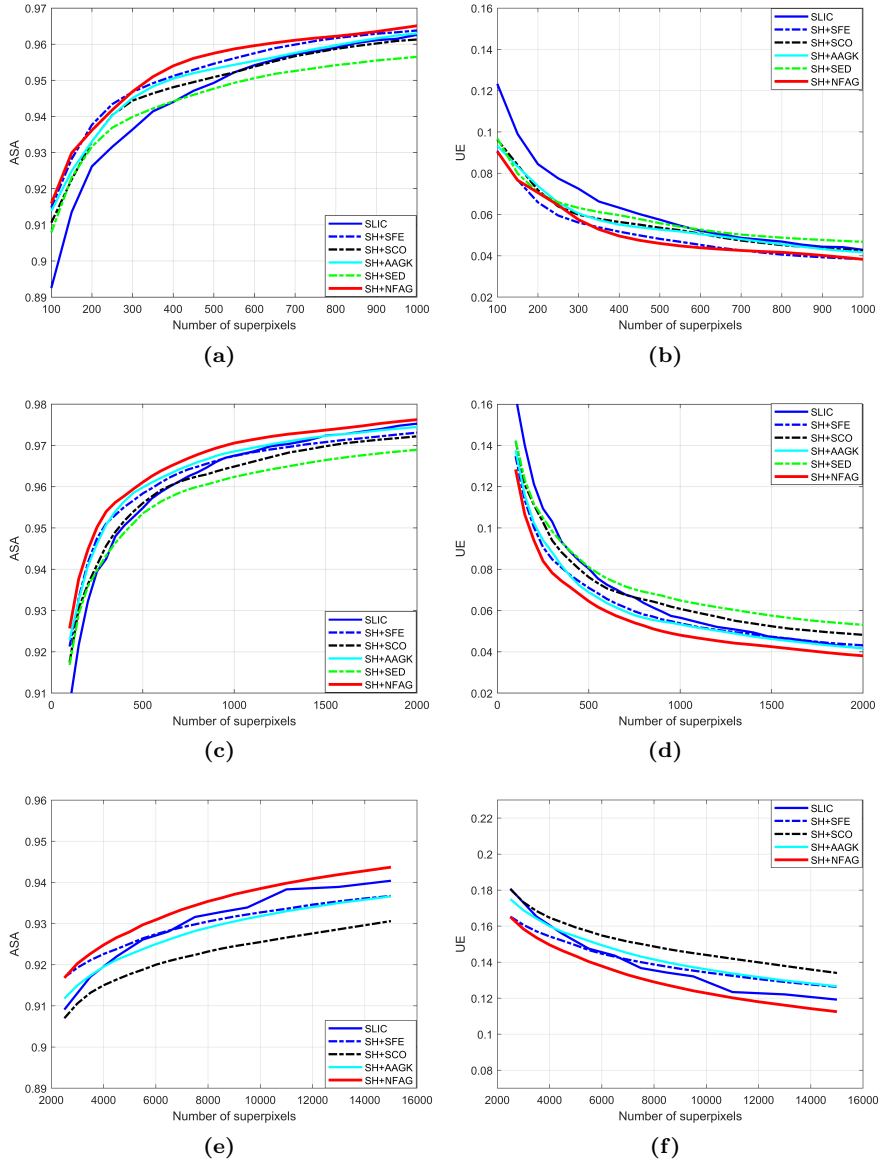


Figure 3.12: Evaluation results in terms of the average ASA and average UE. (a) and (b): Results obtained on the BSDS500 dataset. (c) and (d): Results obtained on the SBAIS dataset. (e) and (f): Results obtained on the NSEMS dataset.

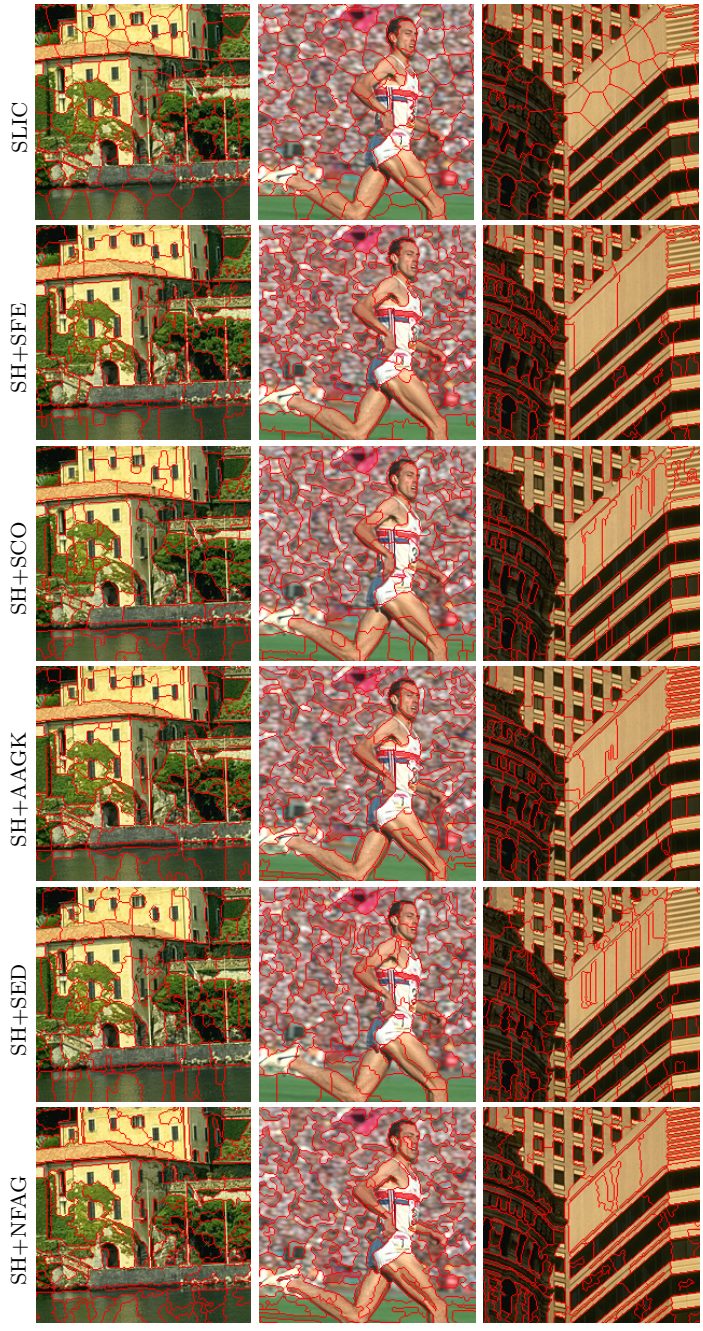


Figure 3.13: Superpixel maps yielded by different methods on sample images taken from the BSDS500 dataset. The number of superpixels in each full superpixel map is set to 400. For a better visualization, zoomed-in versions are displayed.

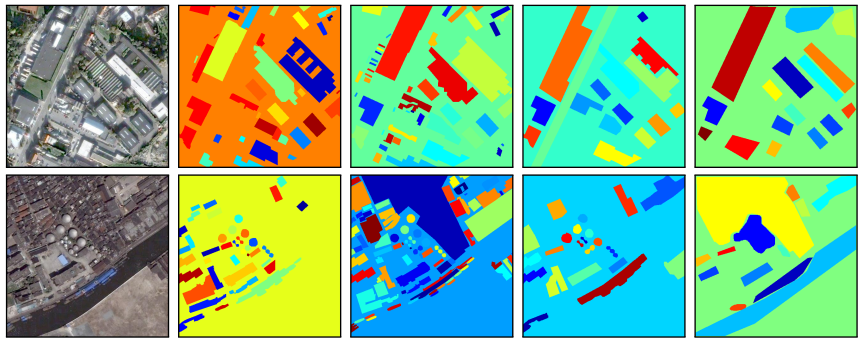


Figure 3.14: Sample images as well as their multiple GT segmentation maps taken from the SBAIS dataset. Left column: Original images. Second to fifth columns: GT segmentation maps labelled by different annotators for each original image.

Evaluation on the SBAIS dataset

In the SBAIS dataset, there are 80 aerial images, each of which has a resolution of $512 \times 512 \times 3$. For each image, there are four manually labelled GT segmentation maps. Two sample images as well as the GT segmentation maps are displayed in Figure 3.14. For each aerial image, we obtain four ASA values and four UE values when comparing a superpixel map with the multiple GT segmentation maps. The best ASA and UE values for each image are retained as the evaluation results.

For each method, the curves of average ASA and average UE values over all the 80 images with respect to different numbers of superpixels are shown in Figures 3.12(c) and 3.12(d). It can be seen that our method performs the best among all the methods, obtaining higher average ASA values and lower average UE values than the competing methods. Note that the SH+SFE method is not the method obtaining the best performance in terms of the average ASA and average UE, which is different from the results obtained on the BSDS500 dataset. This is mainly because the edge detection model in the learning-based SFE method is trained on the BSDS500 dataset which consists of natural images. As a result, the SH+SFE method underperforms on the aerial images from the SBAIS dataset. In addition, the SH+SED method, the edge strength maps of which are not smooth, still performs the worst among all the SH-based methods. Compared with the SH-based methods, the SLIC method obtains a competitive performance when the number of superpixels is large. That is, when the hierarchy level is low, but underperforms when the number of superpixels is small. This is consistent with the experimental results obtained on the BSDS500 dataset. As mentioned earlier, the main reason is that the SLIC method is a partition-based method. It does not consider the

edge strength during the process of pixel clustering.

The superpixel maps yielded by different methods are illustrated in Figure 3.15. One can see that the SH-based methods adhere better to the contours of regions than the SLIC method. In particular, compared with others, the SH+NFAG and SH+AAGK methods retain more contours of regions in the superpixel segmentation. It is believed that some subsequent tasks, e.g., remote sensing imagery segmentation and classification (Zhang et al., 2015a; Csillik, 2017), will benefit from the accurately segmented superpixels.

Evaluation on the NSEMS dataset

In the NSEMS dataset, there are 30 grayscale electron microscopy images, each of which has a resolution of 512×512, as well as the GT segmentation maps. Two sample images and the corresponding GT segmentation maps are illustrated in Figure 3.16. Note that there is one GT segmentation map for each microscopy image. In this experiment, we do not test the SH+SED method since it is not applicable to grayscale images (Akbarinia and Parraga, 2018).

For each method, the curves of average ASA and average UE values over all the 30 images with respect to different numbers of superpixels are shown in Figures 3.12(e) and 3.12(f). It can be seen that our method performs better than all the other methods, achieving higher average ASA values and lower average UE values than the competing methods in most cases. Note that the SH+SFE method is not the method obtaining the best performance in terms of the average ASA and average UE, which is similar to the experimental results obtained on the SBAIS dataset. This is because the edge detection model in the SFE method is trained on the BSDS500 dataset, and as such, it underperforms on electron microscopy images. Among all the methods, it is the SH+SCO method that performs the worst. In addition, the SLIC method performs better for fine superpixel segmentation than for coarse superpixel segmentation, which is consistent with the experimental results reported earlier. For illustration, Figure 3.17 displays the superpixel maps yielded by different methods. It can be observed that our method adheres to the contours better than the competing methods.

Superpixels for saliency detection

In previous experiments, we have confirmed that in superpixel segmentation, our method has an overall advantage compared with the competing methods. We next show how the superpixels yielded by our method facilitate subsequent image processing with an example task of saliency detection.

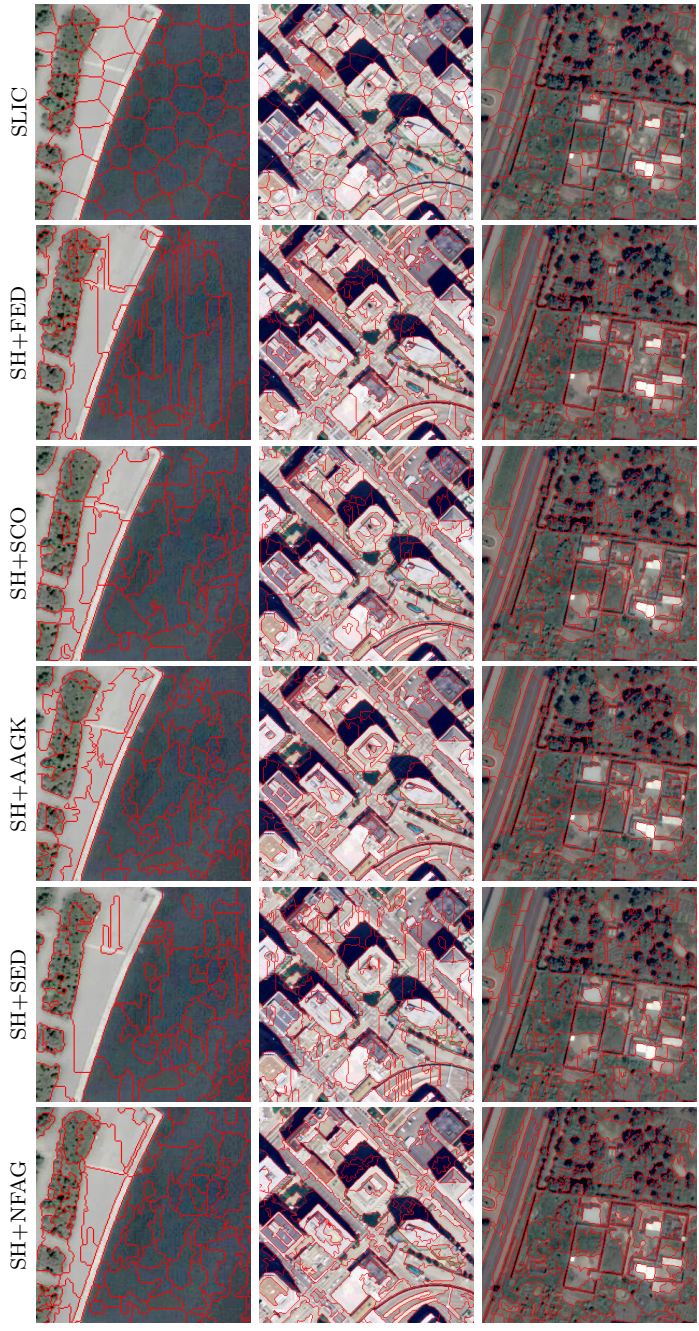


Figure 3.15: Superpixel maps yielded by different methods on sample images taken from the SBAIS dataset. The number of superpixels in each full superpixel map is set as 500. For a better visualization, zoomed-in versions are displayed.

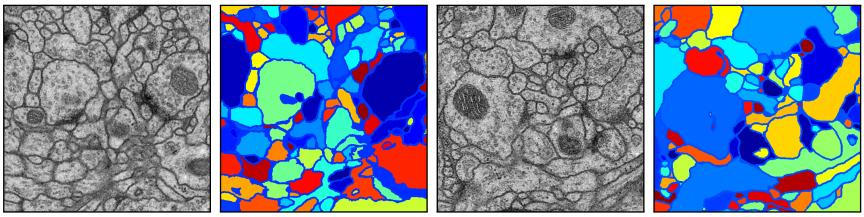


Figure 3.16: Sample images as well as their GT segmentation maps taken from the NSEMS dataset. The first and the third columns: Original images. The second and fourth columns: Corresponding GT segmentation maps.

In computer vision, saliency detection is aimed at finding the pixels that belong to the most salient region attracting the attention of human visual system. In literature, there are quite a few saliency detection methods that are built on superpixel maps (Qin et al., 2018). A prominent method was proposed by Qin et al. (Qin et al., 2015), in which the superpixels evolve into a saliency detection result based on a cellular automaton mechanism (SCA method). In the SCA method, the superpixels in the vicinity of the image frame are grouped into several background clusters. Subsequently, a propagation mechanism based on a cellular automaton is designed to exploit the intrinsic dissimilarity between each superpixel and the background clusters in terms of color difference and spatial distance. The saliency map is obtained after a number of iterations. In the original method, the superpixels used are generated by the SLIC method (SCA+SLIC) (Qin et al., 2015).

In this experiment, we use the SCA method based on superpixels generated by our method (SCA+SH+NFAG) to obtain the saliency detection results on four sample images that are taken from the extended complex scene saliency dataset[6] (Shi et al., 2016). We also compare our method with the original SCA+SLIC method. The parameters are configured according to the original implementation (Qin et al., 2015). In particular, the number of superpixels in each method is set to be 300.

Figure 3.18 illustrates the saliency detection results. Compared with the SCA+SLIC method, the SCA+SH+NFAG method yields saliency maps with a higher accuracy. For instance, in the second row of Figure 3.18, the salient boat is well separated from the non-salient regions by the SCA+SH+NFAG method, while the SCA+SLIC method also assigns significant saliency values to some non-salient pixels.

Complementary to the visual comparison, we also quantitatively evaluate the saliency detection performance of the SCA+SLIC and SCA+SH+NFAG

[6] http://www.cse.cuhk.edu.hk/~leojia/projects/hsaliency/dataset.html

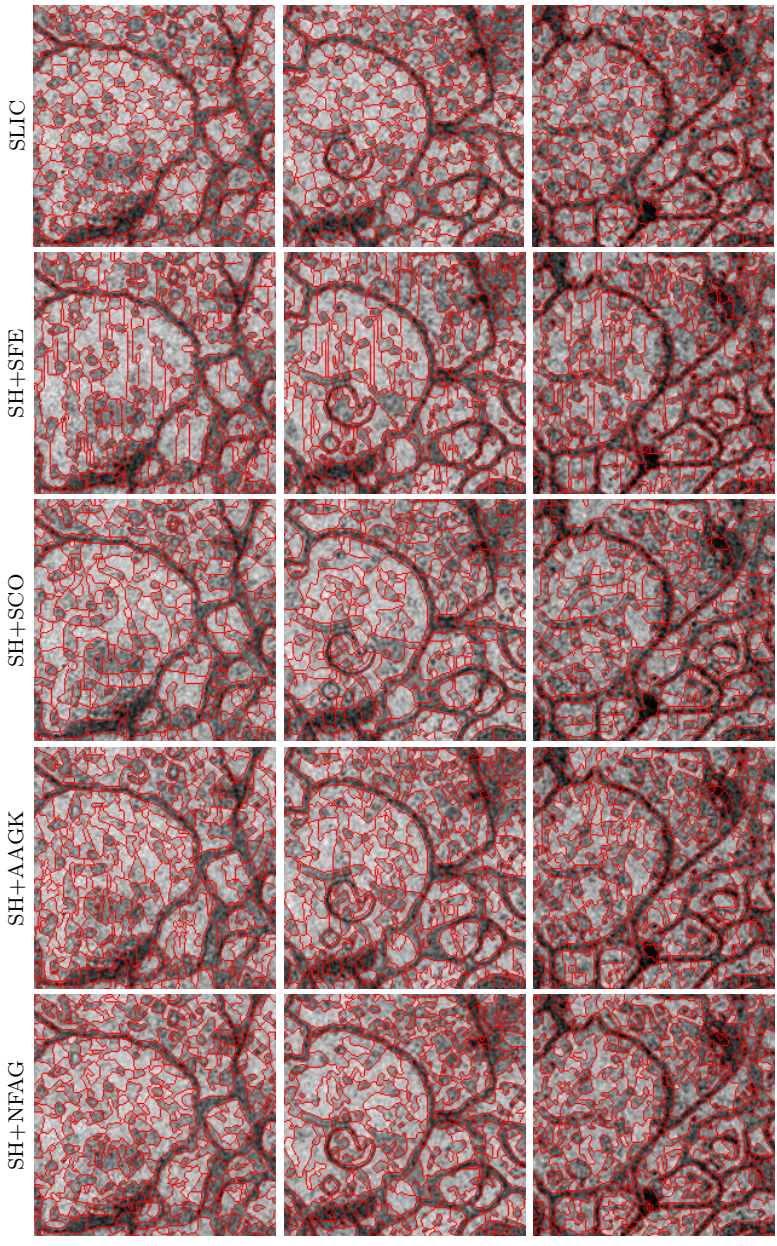


Figure 3.17: Superpixel maps yielded by different methods on sample images taken from the NSEMS dataset. The number of superpixels in each full superpixel map is set as 2000. For a better visualization, zoomed-in versions are displayed.

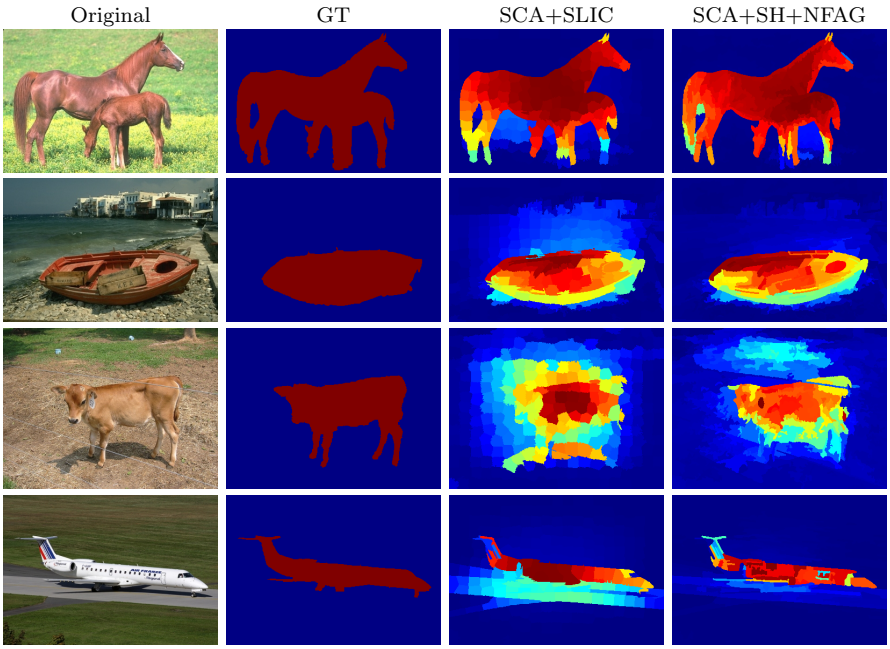


Figure 3.18: Sample results of saliency detection obtained by the SCA+SLIC and SCA+SH+NFAG methods.

methods in terms of the mean absolute error (MAE) representing the difference between the obtained saliency map and the GT saliency map (Qin et al., 2018):

$$\mathrm{MAE} = \frac{1}{N_{\mathrm{sl}}} \sum_{i=1}^{N_{\mathrm{sl}}} \left| \mathbf{I}_{\mathrm{sl}}(\mathbf{m}_i) - \mathbf{T}_{\mathrm{sl}}(\mathbf{m}_i) \right| , \tag{3.51}$$

where $\mathbf{m}_i$ denotes the pixel location, $\mathbf{I}_{\mathrm{sl}}$ and $\mathbf{T}_{\mathrm{sl}}$ represent the obtained saliency map and the GT saliency map, respectively, N_{sl} stands for the number of pixels in the saliency map, and $|\cdot|$ denotes the absolute value of a real number. For saliency detection, a lower MAE value is preferred. The evaluation results obtained on the sample images are presented in Table 3.3. As can be seen, our method obtains lower MAE values on all the sample images, outperforming the SCA+SLIC method.

3.6. Conclusions

In this chapter, firstly, we have presented a multiscale version of NFAG kernels, and accordingly, have developed a method to obtain the anisotropic edge strength using the multiscale NFAG kernels. A scale-adaptive anisotropy

Table 3.3: Saliency detection evaluation results in terms of mean absolute error obtained on the sample images.

Methods	**Index of the Sample Images**			
	1	**2**	**3**	**4**
SCA+SLIC	0.0710	0.1094	0.1921	0.0789
SCA+SH+NFAG	0.0585	0.0846	0.1561	0.0519

factor has also been designed to address the problem of anisotropy stretch effect. This method is able to quantitatively measure the edge strength, edge direction and edge scale simultaneously, while reducing the impact of noise. It is quite reliable in detecting edges with heterogeneous widths. Secondly, we have developed a contour detection method using the NFAG-based anisotropic edge strength and the hierarchical superpixel contrast. The experimental results on a widely adopted dataset have validated the superiority of our contour detection method over the competing method. Thirdly, we have presented an SH-based superpixel segmentation method that incorporates the anisotropic edge strength. Experimental results have confirmed that, in the SH-based superpixel segmentation, the method incorporating the anisotropic edge strength is less dataset-dependent. We have also illustrated that the superpixels yielded by our method can facilitate a subsequent saliency detection task.

4 Second-order anisotropic Gaussian kernel with application to line detection

In this section, we investigate an existing convolutional kernel named the second-order anisotropic Gaussian (SAG) kernel (Lopez-Molina et al., 2015). After rebuilding this kernel, we study a normalization method that enables SAG kernels to quantitatively measure the line prominence and identify the scales with scale-invariance in scale-space. Also, we theoretically prove that the anisotropy of SAG kernels can improve the SNR of lineness. Based on the SNR analysis, we design an adaptive anisotropy factor to enhance the noise-robustness of small-scale kernels, thereby proposing the normalized and adaptive second-order anisotropic Gaussian (NASAG) kernel. Using NASAG kernels, we propose a full-fledged line detection method. Experiments performed on a publicly available dataset validate the efficacy of the proposed method.

The material of this chapter is based on the following publication:

- Wang, G., Lopez-Molina, C., Vidal-Diez de Ulzurrun, G., and De Baets, B. (2019c). Noise-robust line detection using normalized and adaptive second-order anisotropic Gaussian kernels. *Signal Processing*, 160:252–262

4.1. Motivation

Lines, a.k.a. curvilinear structures or ridges, are generally defined as elongated regions with dissimilar intensities compared to their neighboring pixels. Such structures hold key information for some computer vision tasks, and as such, line detection has been extensively studied in literature. In the photogrammetry and remote sensing field, line detection is used to extract roads (Ferraz et al., 2016) and rivers (Yang et al., 2015b) from different types of images. In many biological or medical imagery analysis systems, line detection is widely employed to extract graph-like structures, including fungal networks (Vidal-Diez de Ulzurrun et al., 2015), blood vessels (Yang and Shi, 2014), neurons (Meijering et al., 2004), fibrous tissues (Krylov and Nelson, 2014) and leaf veins (Bühler et al., 2015). Moreover, line detection is also applied to extract pavement cracks (Oliveira and Correia, 2013), painting

cracks (Cornelis et al., 2013), facial wrinkles (Batool and Chellappa, 2015), palmprints (Krylov and Nelson, 2014) and power lines (Candamo et al., 2009). Nevertheless, line detection still faces problems other than the specificities of each type of imagery. For example, the prominence, direction and scales (a.k.a. sizes or widths) of line structures are usually heterogeneous in different images. Also, images are usually corrupted by noise. These facts make line detection remain a challenging task, despite the significant efforts carried out in the past several decades.

Conventional methods can be divided into two categories, i.e., interactive methods and automated methods. Interactive (a.k.a. semi-automated) methods are implemented under manual supervision. For instance, Benmansour and Cohen (2011) proposed a vessel extraction method using a minimal path formulation and anisotropic enhancement. However, the user has to click the source points and the endpoints of the line structures and provide the minimum and maximum radius values. Compared with interactive approaches, automated methods can be run without human interventions, and as such are more efficient. The side benefits of this fact include the possibility of processing large datasets with little labor, and the possibility of the algorithm to be run by non-experts.

Concerning automated methods, one can adopt very different taxonomies. A common taxonomy refers to the mathematical tool that the detector uses. Adopting this taxonomy, we can identify methods based on transformation, watershed, Hessian matrix, matched filters, etc. Transformation-based methods rely on the manifestations of lines in transformed spaces, paramount examples being those using the Hough transform (Xu et al., 2015) or the Radon transform (Krylov and Nelson, 2014). These methods are computationally expensive and not applicable to the detection of lines with large curvature. Watershed-based methods (Obara et al., 2012b) do not suffer from this limitation, but at the same time are only appropriate to extract dense line structure networks in clean images.

Hessian-based methods have been more comprehensively studied compared with others. A pioneering work is proposed by Haralick (1983), who defined a bright line as positions at which the main eigenvalue has a maximal absolute value. Lindeberg extended Haralick's ideas to a multiscale context and proposed a framework of scale-space representation for different visual structures (Lindeberg, 1998a). Multiscale and normalized Hessian matrices are calculated to get the corresponding eigenvalues that reflect the local change of intensity, and to get the eigenvectors that denote the primary direction of the local structure. Since visual structures suffer from decreasing visibility when projected to large scales in scale-space, Lindeberg proposed a normalization

method to compensate the magnitude decrease over scales, and accordingly, developed a line detector using the square of the normalized eigenvalue difference. However, Lindeberg's method usually produces multiple peak responses for one line segment, and as a result, outputs spurious detections. Meanwhile, other works were also elaborated on the concepts of linear scale-space theory (Koenderink, 1984; Florack et al., 1992). The multiscale vessel enhancement filter proposed by Frangi et al. (1998) enhances the prominence of vessels by combining the ratio of two eigenvalues with the local contrast, which is represented by the root-sum square of the eigenvalues. Obara et al. (2012a) further developed Lindeberg's work and proposed a contrast-independent approach named the phase congruency tensor vesselness, employing the oriented phase congruency. The relationship between eigenvalues was also used by Meijering et al. (2004), who successfully applied their proposal to neurite detection. Steger (1998, 2013) proposed an approach based on differential geometric properties of the image intensity. Based on both the first and second derivatives of Gaussian kernels, the Taylor polynomial is computed to decide whether or not the current pixel belongs to a line structure. This method mainly addresses the bias problem of line position, but its performance strongly depends on the profile of the lines. In addition, assuming that the profile of a line structure is bar-shaped, Bae et al. (2015) discussed the normalization of the Hessian matrix and measured the prominence of roads in remote sensing images using its maximum negative eigenvalue.

Hessian-based methods have a good computational efficiency because they identify the direction of a line segment using the eigenvectors of the Hessian matrix. However, the Hessian matrix usually provides only a qualitative estimation instead of a quantitative measurement for a line segment. Besides, all the above-mentioned Hessian-based methods employ isotropic Gaussian kernels as well as their first and second derivatives (Shui and Zhang, 2012) to compute the Hessian matrix. Because of the sensitivity of isotropic Gaussian kernels to noise, these methods are not robust to noise in detecting narrow lines.

Methods based on matched filters have received significant attention as well. This might be due to their ability to make up for the weaknesses of Hessian-based methods. In order to handle lines with various scales and directions, these methods usually make use of a family of kernels that covers a set of possible scales and directions. The maximum response among all kernels is pixelwise selected as the *lineness* (a.k.a. line strength). We term this scheme as *maximum aggregation*. Note that this kind of method also preserves rotational invariance under the assumption that the set of kernels considers all possible directions.

A very relevant proposal based on matched filters was proposed by Chaudhuri et al. (1989), using oriented Gaussian-shaped curve matching filters. With the aim of improving the line detection resolution, Xiao et al. (2013) designed the bi-Gaussian kernel by merging two Gaussians with different parameters. Although this kind of kernel reduces adjacent disturbances, it is not noise-robust enough for lines that have low SNR (Xiao et al., 2013). In order to suppress the noise and deal with lines with different scales, multiscale filters based on second-order Gaussian kernels have been studied as well. Based on Lindeberg's work (Lindeberg, 1998b), Sofka and Stewart (2006) discussed the normalization problem of second-order Gaussian kernels, and employed the same normalization parameter as proposed by Lindeberg. This method has solid theoretical foundations and produces acceptable results in practical applications. The obtained lineness usually shows a single peak response for a given line segment, which helps to ease the problem of locating the centerline. However, this method also has a critical drawback, since it shows scale-variance in scale-space. In practical terms, for two line segments with identical prominence but different scales, the lineness values obtained by this method are different.

Recently, kernels based on the derivative of anisotropic Gaussian functions have been successfully applied in edge detection (Shui and Zhang, 2012; Wang and Shui, 2016; Zhang et al., 2017b; Shui and Wang, 2017; Wang and De Baets, 2017) and corner classification (Shui and Zhang, 2013), showing certain advantages compared with the isotropic Gaussian derivatives. Lopez-Molina et al. (2015) introduced SAG kernels to line detection and validated their performance on a dataset of fungal imagery. On line detection tasks, the method based on SAG kernels outperforms methods based on isotropic kernels in detecting bifurcations, junctions, crossings and adjacent lines. However, essential theoretical explanations of efficacy are still absent in the work of Lopez-Molina et al. (2015). In addition, they did not provide the normalization of the kernels in scale-space. Moreover, the method is sensitive to noise and easily produces spurious results on noisy images. Furthermore, because of the stretch effect of anisotropic kernels, this approach produces spurious results on endpoints of lines, as well as on blob structures.

In this chapter, we intend to expand the work in (Lopez-Molina et al., 2015). We first present a rebuilt version of the SAG kernel. Subsequently, we study a normalization method that enables SAG kernels to quantitatively measure the line prominence and identify the scales with scale-invariance in scale-space. We also prove that the anisotropy of the SAG kernel can improve the SNR of lineness. Accordingly, we propose an adaptive anisotropy factor to enhance the noise-robustness of small-scale kernels. Moreover, we present

that the proposed NASAG kernels help to reduce the number of employed kernels and alleviate the anisotropy stretch effect. Using the proposed NASAG kernels, we develop a novel line detection method.

The remainder of this chapter is organized as follows. Section 4.2 recalls some preliminary concepts that are of interest to this work. In Section 4.3, we model line structures using directional Gaussian functions. Also, we present a rebuilt version of SAG kernels together with the corresponding scale-invariant normalization method. Based on a noise-robustness analysis in terms of the SNR, we propose a novel adaptive anisotropy factor. Subsequently, the proposed line detection method is elaborated. Experimental results are reported in Section 4.4, while Section 4.5 lists our conclusions.

4.2. Related work

Anisotropic Gaussian kernels are a class of kernels based on the elongation of regular (isotropic) Gaussian kernels, usually with the aim of improving their adjustment to non-isotropic visual artifacts. These kernels, as well as their derivatives, have been successfully applied in edge detection (Shui and Zhang, 2012; Wang and Shui, 2016; Zhang et al., 2017b; Shui and Wang, 2017; Wang and De Baets, 2017) and corner detection (Shui and Zhang, 2013). Conventionally, a two-dimensional anisotropic Gaussian kernel is defined as

$$g(\mathbf{x};\sigma,\rho,\theta) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\mathbf{x}^{\mathrm{T}}\mathbf{R}_\theta^{\mathrm{T}}\begin{bmatrix}\rho^2 & 0\\ 0 & \rho^{-2}\end{bmatrix}\mathbf{R}_\theta\mathbf{x}\right), \tag{4.1}$$

where all the arguments are defined the same as those in Eq. (3.7) except that ρ is referred to as the anisotropy index. Note that this kernel reduces to an isotropic Gaussian kernel when $\rho = 1$. Based on Eq. (4.1), an SAG kernel is derived as follows:

$$g''(\mathbf{x};\sigma,\rho,\theta) = \left(\frac{(x\cos\theta + y\sin\theta)^2}{\rho^{-4}\sigma^4} - \frac{\rho^2}{\sigma^2}\right)\cdot g(\mathbf{x};\sigma,\rho,\theta)\,. \tag{4.2}$$

Lopez-Molina et al. (2015) employed a family of discrete SAG kernels combining all possible scales, directions and anisotropy indices to filter the image signal. Using the maximum aggregation, for each pixel, the maximum response among all kernels is then selected as the lineness. Correspondingly, the direction and scale of each line structure are inferred by the kernel that produces the maximum response.

4.3. Normalized and adaptive second-order anisotropic Gaussian kernel

In this section, we firstly model a line structure using a direction Gaussian function. Secondly, we present the rebuilt vision of the SAG kernel as well as the corresponding normalization method. Thirdly, an adaptive anisotropy factor is proposed. Based on filtering schemes and postprocessing techniques, a novel line detection method using NASAG kernels is developed.

4.3.1. Modelling a line segment

Inspired by the work of Steger (2013), we model a line segment as a Gaussian profile. In a small-enough local region, a line segment can be considered straight and therefore is modelled by

$$\mathbf{\Gamma}_0(\mathbf{x}) = p_0 \cdot \exp\left(-\frac{(\tilde{\mathbf{x}} - \mathbf{x_0})^{\mathrm{T}}(\tilde{\mathbf{x}} - \mathbf{x_0})}{2\omega_0^2}\right) + b_0\,, \tag{4.3}$$

where

$$\tilde{\mathbf{x}} = [\cos\theta_0, \sin\theta_0]\mathbf{x}\,, \tag{4.4}$$

$\mathbf{x_0} = [x_0, y_0]^{\mathrm{T}}$ stands for the center location of the local line segment, and the parameters b_0, p_0, ω_0 and θ_0 represent the base level, prominence, scale and normal direction, respectively.

Figure 4.1 illustrates a local part of the line segment modelled by Eq. (4.3). The base level reflects the intensity of the background, while the prominence implies the local contrast between the background and the center of the line. As the signal intensity $\mathbf{\Gamma}_0(\mathbf{x})$ at position $\mathbf{x_0}$ is known, the base level b_0 can be calculated once the prominence p_0 is obtained. Therefore, the task of quantitative line detection can be considered as finding the parameters of prominence p_0, scale ω_0 and direction θ_0.

4.3.2. Rebuilding the conventional second-order anisotropic Gaussian kernel

We rebuild the conventional anisotropic Gaussian kernel formulated in Eq. (4.1) by introducing a new anisotropy factor φ instead of the original ρ:

$$g(\mathbf{x}; \sigma, \varphi, \theta) = \frac{1}{2\pi\varphi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\mathbf{x}^{\mathrm{T}}\mathbf{R}_\theta^{\mathrm{T}}\begin{bmatrix}1 & 0\\ 0 & \varphi^{-2}\end{bmatrix}\mathbf{R}_\theta\mathbf{x}\right), \tag{4.5}$$

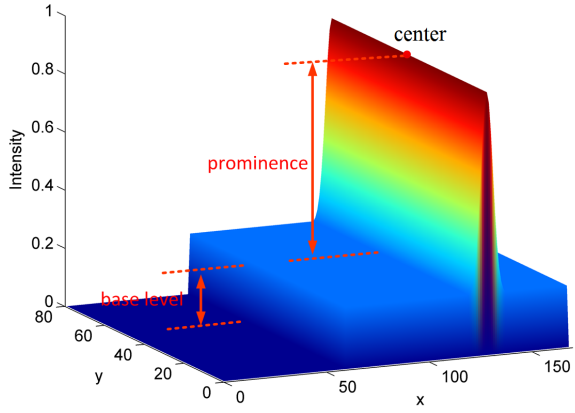


Figure 4.1: Illustration of the modelled local line segment and two measurable characteristics: prominence and base level.

where $\varphi \geq 1$ and the other arguments remain the same as those in Eq. (4.1). The function in Eq. (4.5) is different from the conventional anisotropic Gaussian kernel in (Shui and Zhang, 2012; Lopez-Molina et al., 2015; Zhang et al., 2017b). Compared to Eq. (4.1), Eq. (4.5) can better show the relationship between the scales in the x direction and the y direction, since it avoids binding the two scales together with the parameter ρ. Note that the kernel in Eq. (4.5) becomes isotropic when $\varphi = 1$.

Elaborating on Eq. (4.5), we obtain the rebuilt version of the SAG kernel as follows:

$$g''(\mathbf{x}; \sigma, \varphi, \theta) = \left(\frac{(x \cos\theta + y \sin\theta)^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \cdot g(\mathbf{x}; \sigma, \varphi, \theta) \,. \tag{4.6}$$

As introduced in previous chapters, the scale-space representation of a two-dimensional signal $\mathbf{I}(\mathbf{x})$ is conventionally obtained by isotropic Gaussian kernels. Here, by using the rebuilt anisotropic Gaussian kernel in Eq. (4.5), we define the anisotropic and directional scale-space representation as follows:

$$\mathbf{I}_{\mathrm{ss}}(\mathbf{x}; \sigma, \varphi, \theta) = g(\mathbf{x}; \sigma, \varphi, \theta) * \mathbf{I}(\mathbf{x}) \,. \tag{4.7}$$

Accordingly, the derivatives of the anisotropic and directional scale-space representation are given by

$$\mathbf{I}^{(\alpha)}_{\mathrm{ss}}(\mathbf{x}; \sigma, \varphi, \theta) = g^{(\alpha)}(\mathbf{x}; \sigma, \varphi, \theta) * \mathbf{I}(\mathbf{x}) \,. \tag{4.8}$$

4.3.3. Scale-invariant normalization

In order to compensate the magnitude decrease in scale-space, inspired by Lindeberg's normalization method (Lindeberg, 1998a), we define a general expression of the normalized SAG kernel, which would be used for line detection, as

$$g''_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta) = (-1)^{\eta}\cdot\beta\cdot\sigma^{2\gamma}\cdot g''(\mathbf{x};\sigma,\varphi,\theta)\,, \tag{4.9}$$

where β is a constant ensuring that the obtained lineness precisely reflects the prominence and γ denotes the scale normalization factor. Besides, the parameter $\eta \in \{0,1\}$ allows the kernel to be applicable to both bright and dark line structures. Optionally, we set $\eta = 1$ for bright line detection and $\eta = 0$ for dark line detection. Without loss of generality, η is set as 1 in the following elaboration within this section. Correspondingly, the normalized Gaussian kernel in this chapter is denoted by $g_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta)$.

Based on Eq. (4.8), when we filter the line segment modelled in Eq. (4.3) using the normalized SAG kernel in Eq. (4.9), we obtain the response as follows:

$$\mathbf{L}(\mathbf{x};\sigma,\varphi,\theta) = g''_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta) * \mathbf{\Gamma}_0(\mathbf{x})\,. \tag{4.10}$$

For a given scale, the normalized SAG kernel that has the same direction as the line segment produces the peak response, i.e., $\theta = \theta_0$. Also, without loss of generality, we consider the case $\mathbf{x}_0 = \mathbf{0}$ and $\theta_0 = 0$. Then, the peak response L at the center location of the modelled line is given by

$$\begin{aligned} L &= \mathbf{L}(\mathbf{x};\sigma,\varphi,\theta)|_{\mathbf{x}=\mathbf{0},\theta=\theta_0=0} \\ &= \beta p_0\omega_0\sigma^{2\gamma}(\omega_0^2+\sigma^2)^{-\frac{3}{2}}\,. \end{aligned} \tag{4.11}$$

The detailed proof is presented in Appendix A.3. Obviously, L is dependent on the kernel scale σ. Thus, in scale-space, the derivative of L with respect to σ is calculated as

$$\frac{\partial L}{\partial\sigma} = \beta p_0\omega_0\sigma^{2\gamma-1}(\omega_0^2+\sigma^2)^{-\frac{5}{2}}\left(2\gamma\omega_0^2+(2\gamma-3)\sigma^2\right)\,. \tag{4.12}$$

In order to identify the scale of the line structure, we need to find the scale σ that produces the largest response in scale-space, that is, we should find the scale σ that makes the derivative of L with respect to σ equal zero. This leads to

$$\sigma^* = \sqrt{2}\gamma^{\frac{1}{2}}(3-2\gamma)^{-\frac{1}{2}}\omega_0\,, \tag{4.13}$$

where σ^* denotes the scale at which the kernel $g''_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta)$ produces the maximum response in scale-space. Equation (4.13) indicates that σ^* depends on the parameter γ. In addition, we also impose $0<\gamma<1.5$ to ensure σ^* to be positive. By substituting Eq. (4.13) into Eq. (4.11), we get the maximum response of the modelled line structure in scale-space:

$$L=\frac{\sqrt{3}}{9}\beta p_0(2\gamma)^{\gamma}(3-2\gamma)^{\frac{3}{2}-\gamma}\omega_0^{2\gamma-2}\,. \tag{4.14}$$

Note that the maximum response in Eq. (4.14) is independent of the anisotropy factor φ. Also, to ensure that the response does not rely on ω_0, i.e., the normalized SAG kernel is scale-invariant in scale-space, we should set $\gamma=1$. Consequently, L achieves its maximum value at the scale $\sigma^*=\sqrt{2}\omega_0$. In addition, Eq. (4.14) then becomes

$$L^*=\frac{2\sqrt{3}}{9}\beta p_0\,. \tag{4.15}$$

Consequently, we set $\beta=3\sqrt{3}/2$, so that L^*_{p} equals p_0 while achieving its maximum response at the scale $\sigma^*=\sqrt{2}\omega_0$.

Eventually, taking Eq. (4.9) and the discussion above into consideration, we get the scale-invariant normalized SAG kernel as follows:

$$g''_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta)=-\frac{3\sqrt{3}}{2}\sigma^2\cdot g''(\mathbf{x};\sigma,\varphi,\theta)\,. \tag{4.16}$$

We are now in a position to summarize that the proposed normalized SAG kernel yields the maximum response value p_0 at the position $\mathbf{x}_0$ and at the scale $\sqrt{2}\omega_0$.

4.3.4. Adaptive anisotropy factor

According to the differentiation properties of the convolution operation, from Eqs. (4.8) and (4.10), we get the result of filtering a two-dimensional signal $\mathbf{I}$ with a normalized SAG kernel

$$\begin{aligned}\mathbf{L}(\mathbf{x};\sigma,\varphi,\theta)&=g''_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta)*\mathbf{I}(\mathbf{x})\\&=g_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta)*\mathbf{I}''(\mathbf{x})\,,\end{aligned} \tag{4.17}$$

where $\mathbf{I}''$ is the second-order derivative of $\mathbf{I}$ with respect to x along the direction θ. Therefore, the response $\mathbf{L}(\mathbf{x};\sigma,\varphi,\theta)$ can also be considered as a smoothing operation on $\mathbf{I}''$ by $g_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta)$. It is inevitable that this Gaussian

smoothing leads to blurring. For an anisotropic Gaussian kernel along the direction θ, the blurring extent with respect to x is determined by σ, while the blurring extent with respect to y is determined by $\varphi\sigma$. As a result, the SAG kernel incurs a stretch effect on the endpoints of the lines and blob structures, especially for those kernels with large anisotropy factors. A similar phenomenon also appears in edge detection (Shui and Zhang, 2012). This phenomenon is called *anisotropy stretch effect*, which as a result leads to spurious detection results. As presented in Chapter 3, the anisotropy stretch effect also occurs in edge detection methods using first-order derivative of Gaussian kernels. For this reason, large values of φ should be avoided. On the contrary, the stretch effect of the isotropic Gaussian kernel is dim because it blurs the signal isotropically.

In Chapter 3, we have confirmed that the NFAG kernel has a good noise-robustness and have designed an adaptive anisotropy factor. Next, we analyze the noise-robustness of the normalized SAG kernel. As presented in Chapter 3, the capability of filters to suppress white Gaussian noise can be measured by the standard deviation of the filtered noise (Canny, 1986; Shui and Zhang, 2012). Assuming that the two-dimensional signal $\mathbf{I}(\mathbf{x})$ is corrupted by additive white Gaussian noise $\xi(\mathbf{x})$ with zero mean and variance ε_0^2, we obtain the standard deviation ε of the noise that is smoothed by the kernel in Eq. (4.9) as follows:

$$\begin{aligned}\varepsilon &= \sqrt{\mathcal{E}\Big\{\big(g''_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta) * \xi(\mathbf{x})\big)^2\Big\}} \\ &= \varepsilon_0 \sqrt{\iint_{\mathbb{R}^2} \big(g''_{\mathrm{L}}(\mathbf{x};\sigma,\varphi,\theta)\big)^2 \,\mathrm{d}\mathbf{x}} \\ &= \frac{\sqrt{3}\beta}{4\sqrt{\pi}} \sigma^{2\gamma-3} \varphi^{-\frac{1}{2}} \varepsilon_0 \,, \end{aligned} \tag{4.18}$$

where $\mathcal{E}(\cdot)$ denotes the mathematical expectation.

Since $0 < \gamma < 1.5$ and β is a constant, one can see that ε is proportional to the noise intensity ε_0, but has a negative correlation with the kernel scale σ as well as the anisotropy factor φ. Thus, for a given kernel scale set $\mathbb{S} = \{\sigma \mid \sigma_{\min} \leq \sigma \leq \sigma_{\max}\}$, if a maximum aggregation is employed, kernels with smaller scales tend to produce larger noise responses, which therefore are more likely to be retained in the final lineness map.

The SNR of the normalized SAG kernel is defined as the quotient of the maximum signal response and the standard deviation of the smoothed noise in the final lineness map. Assuming that we have identified the maximum response of a line segment in scale-space, the SNR can be calculated by

combining Eqs. (4.14) and (4.18):

$$\begin{aligned}\mathrm{SNR} &= \frac{L}{\varepsilon} \\ &= \frac{4\sqrt{\pi}}{9\varepsilon_0} p_0 (2\gamma)^{\gamma} (3-2\gamma)^{\frac{3}{2}-\gamma} \omega_0^{2\gamma-2} \sigma^{3-2\gamma} \varphi^{\frac{1}{2}} \,. \end{aligned} \tag{4.19}$$

From Eq. (4.19), we learn that the SNR is positively correlated with the line prominence p_0, the line scale ω_0, the anisotropy factor φ and the scale-normalization parameter γ. Note that the SNR is independent of the parameter β. Thus, for a given line structure with specific p_0 and ω_0, the SNR is mainly determined by σ, φ and γ.

In the works of Lindeberg (1998b) and Sofka and Stewart (2006), isotropic kernels ($\varphi = 1$) are employed while setting the scale normalization factor $\gamma = 0.75$. Note that $\sigma_{\max}$ occurs at ω_0 in their works and therefore the value of $\sigma_{\min}$ in the scale set $\mathbb{S}$ should be selected to satisfy $\sigma_{\min} \leq \omega_0$. Based on Eq. (4.18), we can infer that the noise achieves its maximum response at the scale $\sigma_{\min}$. Hence, these methods obtain the following SNR:

$$\begin{aligned}\mathrm{SNR}_1 &= \frac{L}{\varepsilon}\bigg|_{\gamma=\frac{3}{4},\, \varphi=1,\, \sigma=\sigma_{\min}} \\ &= \frac{\sqrt{6\pi}}{3\varepsilon_0} p_0 \omega_0^{-\frac{1}{2}} \sigma_{\min}^{\frac{3}{2}} \,. \end{aligned} \tag{4.20}$$

In addition, Bae et al. (2015) also employed isotropic Gaussian kernels ($\varphi = 1$) while setting $\gamma = 1$. The SNR of this method is given by

$$\begin{aligned}\mathrm{SNR}_2 &= \frac{L}{\varepsilon}\bigg|_{\gamma=1,\, \varphi=1,\, \sigma=\sigma_{\min}} \\ &= \frac{8\sqrt{\pi}}{9\varepsilon_0} p_0 \sigma_{\min} \,. \end{aligned} \tag{4.21}$$

Furthermore, having $\varphi \geq 1$ and $\gamma = 1$, the normalized SAG kernel obtains the following SNR:

$$\begin{aligned}\mathrm{SNR}_3 &= \frac{L}{\varepsilon}\bigg|_{\gamma=1} \\ &= \frac{8\sqrt{\pi}}{9\varepsilon_0} p_0 \sigma \varphi^{\frac{1}{2}} \,. \end{aligned} \tag{4.22}$$

From Eq. (4.22), we learn that the SNR performance of the normalized SAG

kernel is mainly determined by $\sigma\varphi^{\frac{1}{2}}$. Hence, on the one hand, for small-scale kernels, it is necessary to employ an anisotropy factor to obtain a higher SNR. On the other hand, for large-scale kernels, we do not have to use an anisotropy factor since large scales can guarantee a high SNR even if the anisotropy factor is absent.

In light of SNR enhancement and anisotropy stretch reduction, we propose the adaptive anisotropy factor as follows:

$$\varphi = \begin{cases} \frac{\sigma_{\text{con}}^2}{\sigma^2} & \text{, if } \sigma_{\min} \leq \sigma \leq \sigma_{\text{con}} \\ 1 & \text{, otherwise} \end{cases} , \tag{4.23}$$

where $\sigma_{\text{con}} \in [\sigma_{\min}, \sigma_{\max}]$ stands for the robustness control scale. Larger values of σ_{con} yield higher SNR, but exacerbate the stretch effect. Thus, the setting of σ_{con} should partly depend on the level of noise. Empirically, σ_{con} is specified as the geometric mean of $\sigma_{\min}$ and $\sigma_{\max}$, since such a σ_{con} always ensures $\sigma\varphi \leq \sigma_{\max}$.

In this way, the adaptive anisotropy factor has large values for small scales and therefore the noise response will be suppressed. Also, the anisotropy stretch effect can be alleviated, since φ is close to 1 for larger scales. Besides, Eq. (4.11) indicates that the lineness map is independent of the anisotropy factor φ, so the adaptive anisotropy factor will not hinder us from getting the expected lineness.

Based on Eqs. (4.22) and (4.23), for all scales not larger than the control scale σ_{con}, the proposed NASAG method obtains the following SNR:

$$\text{SNR}_3 = \frac{8\sqrt{\pi}}{9\varepsilon_0} p_0 \sigma_{\text{con}} . \tag{4.24}$$

Thus, for a given noise-corrupted line structure, the SNR of the proposed NASAG method is determined by σ_{con}.

Comparing Eqs. (4.20), (4.21) and (4.24), we learn that $\text{SNR}_1 < \text{SNR}_2 \leq \text{SNR}_3$, which means the proposed NASAG kernel obtains the highest SNR among the three kernels. Hence, the proposed NASAG kernel helps both to reduce the anisotropy stretch effect and to improve the noise-robustness.

4.3.5. Discrete kernels

In order to accommodate the proposed method to digital image filtering, the discrete version of the NASAG kernel should be given. We get both the discrete anisotropic Gaussian kernel and the NASAG kernel by sampling the formulae in Eqs. (4.5), (4.6) and (4.16) in the two-dimensional integer

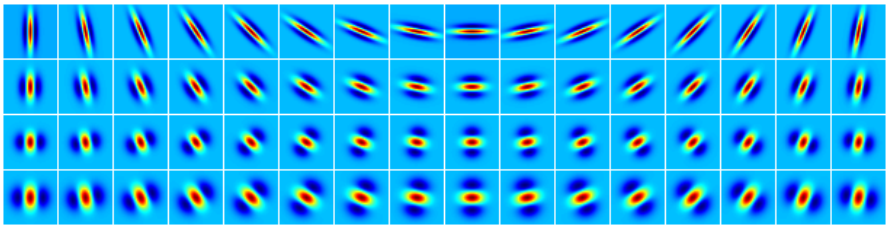


Figure 4.2: Illustration of the discrete version of NASAG kernels. The control scale is set as $\sigma_{\text{con}} = 4$. Top Row: Kernels with $\sigma = 2$ and $\varphi = 4$. Second Row: Kernels with $\sigma = 3$ and $\varphi = 1.78$. Third Row: Kernels with $\sigma = 4$ and $\varphi = 1$. Bottom Row: Kernels with $\sigma = 5$ and $\varphi = 1$. The intensity range of each patch has been adjusted for better display.

coordinate $\mathbb{Z}^2$:

$$
\begin{aligned}
g(\mathbf{m}; \sigma_i, \varphi_i, \theta_j) &= \frac{1}{2\pi\varphi_i\sigma_i^2}\exp\left(-\frac{1}{2\sigma_i^2}\mathbf{m}^{\mathrm{T}}\mathbf{R}_{\theta_j}^{\mathrm{T}}\begin{bmatrix}1 & 0\\ 0 & \varphi_i^{-2}\end{bmatrix}\mathbf{R}_{\theta_j}\mathbf{m}\right)\\
g''_{\mathrm{L}}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j) &= -\frac{3\sqrt{3}}{2}\sigma_i^2 \cdot \left(\frac{([\cos\theta_j, \sin\theta_j]\mathbf{m})^2}{\sigma_i^4} - \frac{1}{\sigma_i^2}\right) \cdot g(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)\,,
\end{aligned}
\tag{4.25}
$$

where

$$
\begin{aligned}
\varphi_i &= \max\left(\frac{\sigma_{\text{con}}^2}{\sigma_i^2}\ ,\ 1\right),\\
\mathbf{R}_{\theta_j} &= \begin{bmatrix}\cos\theta_j & \sin\theta_j\\ -\sin\theta_j & \cos\theta_j\end{bmatrix},
\end{aligned}
$$

in which the arguments remain the same as those in Eq. (3.27). Note that the arguments to determine an NASAG kernel are the scale, direction and robustness control scale. Examples of discrete NASAG kernels are illustrated in Fig. 4.2.

Hence, the result of filtering an image $\mathbf{I}$ with a discrete NASAG kernel is given by:

$$
\mathbf{L}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j) = g''_{\mathrm{L}}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j) * \mathbf{I}(\mathbf{m})\,. \tag{4.26}
$$

Using the maximum aggregation, for each pixel, the largest value among the responses of all kernels is retained as the value of the lineness map:

$$
\mathbf{L}(\mathbf{m}) = \max_{\sigma_i \in \mathbb{S}} \max_{\theta_j \in \mathbb{D}} \mathbf{L}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)\,. \tag{4.27}
$$

As an illustration, Fig. 4.3(b) shows the resulting lineness map of the original image in Fig. 4.3(a).

Usually, the line direction map is also required in postprocessing procedures. As stated earlier, the kernel that has the same direction as the line produces the peak response. Thus, for each pixel, the direction information is indicated by the kernel that produces the largest response. That is, the line direction map $\boldsymbol{\Theta}_{\mathrm{L}}(\mathbf{m})$ is obtained by:

$$\boldsymbol{\Theta}_{\mathrm{L}}(\mathbf{m}) = \underset{\theta_j \in \mathbb{D}}{\operatorname{argmax}} \max_{\sigma_i \in \mathbb{S}} \mathbf{L}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)\,. \tag{4.28}$$

4.3.6. Postprocessing on lineness map

On many line detection tasks in which the centerline is desired, the lineness map needs further processing to output a centerline map (Steger, 2013; Lopez-Molina et al., 2015; Sironi et al., 2016). To this end, after obtaining the lineness map, the techniques of NMS (Rosenfeld et al., 1972), double-thresholding and hysteresis segmentation (Canny, 1986) are employed to produce the centerline map.

The NMS technique was originally used for thinning edgeness in edge detection (Canny, 1986). Here, we use the NMS technique to thin the lineness map. For each pixel, if the lineness value $\mathbf{L}(\mathbf{m})$ at the current pixel location $\mathbf{m}$ is the largest value compared to the other pixels along the direction $\boldsymbol{\Theta}_{\mathrm{L}}(\mathbf{m})$, the value will be retained. Otherwise, it will be nullified. The result of applying the NMS technique to Fig. 4.3(b) is shown in Fig. 4.3(c).

The hysteresis segmentation requires an upper threshold τ_1 and a lower threshold τ_2. In this chapter, we compute the two thresholds by the double-thresholding technique that is presented in (Liu et al., 2013). All the pixels of which the lineness exceeds the upper threshold are first labelled as strong line candidates. The pixels whose lineness value is between the lower threshold and the upper threshold are labelled as weak line candidates. Subsequently, for each of those weak line candidates, if it is connected to a strong line candidate, it is also marked as a strong line candidate. The hysteresis segmentation results of Fig. 4.3(c) with different thresholds are illustrated in Figs. 4.3(d), 4.3(e) and 4.3(f).

Finally, the strong line candidates are processed to form the detection result $\mathbf{L}_{\mathrm{d}}$ using morphological closing and length threshold deleting (Lopez-Molina et al., 2015).

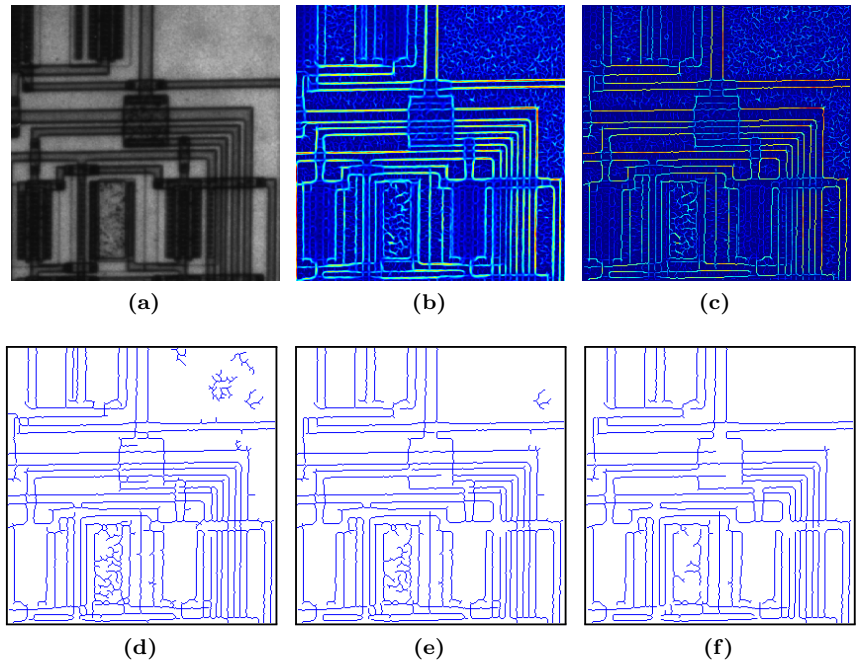


Figure 4.3: Illustration of the line detection process. (a) Original image *circuit*; (b) Lineness map; (c) NMS result; (d)-(f) Hysteresis segmentation results using different thresholds.

4.4. Experimental validation

In this section, we first quantitatively evaluate our method on a fungus image dataset as well as a noisy version thereof. The proposed method is compared with five selected competing methods. Furthermore, we present some potential applications.

4.4.1. Application to fungus detection

We apply the proposed NASAG method, which is summarized in Algorithm 2, to the Ghent University Fungal Images 1 [1] (GUFI-1) dataset (Lopez-Molina et al., 2015) as well as a noisy version thereof that is obtained by adding white Gaussian noise with the level of $\varepsilon_0 = 10$. The GUFI-1 dataset contains 100 fungal images and 100 corresponding ground truth images, each of which has an identical resolution of 300×300. Five example images as well as their ground truth line map are illustrated in Fig. 4.4. One can see that these

[1] http://www.kermit.ugent.be

Algorithm 2 Proposed line detection method

Require: Intensity image $\mathbf{I}$, parameter η, scale set $\mathbb{S}$, direction set $\mathbb{D}$, control scale σ_{con}
Ensure: Line detection result $\mathbf{L}_{\mathrm{d}}$
1: **for** each $\sigma_i \in \mathbb{S}$ **do**
2: $\quad\varphi_i \leftarrow \max\left(\frac{\sigma_{\mathrm{con}}^2}{\sigma_i^2}, 1\right)$
3: $\quad$**for** each $\theta_j \in \mathbb{D}$ **do**
4: $\quad\quad\mathbf{L}(\mathbf{m};\sigma_i,\varphi_i,\theta_j) \leftarrow \mathbf{I}(\mathbf{m}) * g''_{\mathrm{L}}(\mathbf{m};\sigma_i,\varphi_i,\theta_j)$
5: $\quad$**end for**
6: **end for**
7: $\mathbf{L}(\mathbf{m}) \leftarrow \max_{\sigma_i\in\mathbb{S}} \max_{\theta_j\in\mathbb{D}} \mathbf{L}(\mathbf{m};\sigma_i,\varphi_i,\theta_j)$
8: $\mathbf{\Theta}_{\mathrm{L}}(\mathbf{m}) \leftarrow \operatorname{argmax}_{\theta_j\in\mathbb{D}} \max_{\sigma_i\in\mathbb{S}} \mathbf{L}(\mathbf{m};\sigma_i,\varphi_i,\theta_j)$
9: $\mathbf{L}_{\mathrm{nms}}(\mathbf{m}) \leftarrow$ NMS using $\mathbf{L}(\mathbf{m})$ and $\mathbf{\Theta}_{\mathrm{L}}(\mathbf{m})$
10: $\mathbf{L}_{\mathrm{d}}(\mathbf{m}) \leftarrow$ double-thresholding, hysteresis segmentation and line length thresholding on $\mathbf{L}_{\mathrm{nms}}(\mathbf{m})$

images contain a variety of line structures with various scales, directions and prominence (Lopez-Molina et al., 2015).

We compare our method with several conventional line detectors, including the methods based on the multiscale vessel enhancement filter (MVEF) (Frangi et al., 1998; Jerman et al., 2015), the optimally oriented flux (OOF) (Law et al., 2012), the phase congruency tensor vesselness (PCTV) (Obara et al., 2012a), the normalized second derivative map (NSDM) (Bae et al., 2015) and the conventional SAG kernels (Lopez-Molina et al., 2015). For a fair comparison, the same post-processing procedures are applied to the lineness map obtained by each method.

Since line detection can be considered as a binary classification procedure of discriminating line pixels from non-line pixels, the comparison between the output of a line detector and the manually specified ground truth can be formulated in terms of success and failure. Hence, the precision-recall framework is adopted as evaluation method. Precision is the quotient of the number of true positive pixels and the number of all positive pixels in output. It represents the probability that the detection result is valid. Recall is the quotient of the number of true positive pixels and the number of all positive pixels in GT. It represents the probability that the true line pixels have been detected.

In this experiment, we also use F-measure to evaluate the performance of each method in fungus detection. Different from the contour detection dataset adopted in Chapter 3, the GUFI-1 dataset has only one ground truth map

for each fungus image. Therefore, the precision and recall are computed as follows:

$$\textsc{Prec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}\,, \tag{4.29}$$

and

$$\textsc{Rec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}\,, \tag{4.30}$$

where TP, FP and FN are the numbers of true positive pixels, false positive pixels and false negative pixels, respectively. Then, we obtain the F-measure by computing the harmonic mean of the precision and the recall.

Considering that the detection result of a line detector might be slightly different from the ground truth, when the detection result is compared with the ground truth, the detected result which is slightly displaced from the true position within a tolerance is also considered as correct detection. In this experiment, the tolerance is set as 2% of the length of the image diagonal (Lopez-Molina et al., 2015).

With respect to the parameter configuration, the scale set in all methods is set as $\mathbb{S} = \{1 + 0.2 \cdot (i-1) \mid i \in \{1, 2, 3, ..., 11\}\}$. For reproducibility, other parameters are listed as follows:

- MVEF: the thresholds controlling the *blobness measure* and the *second order structureness* are set as 0.5 and 15, respectively;
- OOF: the largest eigenvalue of the OOF tensor is used;
- PCTV: the thresholds controlling the *blobness measure* and the *second order structureness* are set as 0.5 and 15, respectively; the factor controlling the sharpness of the *cutoff* is specified as 2;
- NSDM: the scale normalization factor is set as 1;
- SAG: the anisotropy factors set is configured with $\{1.0, 1.1, ..., 1.5\}$; the number of filtering directions is set as 16;
- NASAG: the indicator for bright line detection, the control scale σ_{con} and the number of filtering directions are specified as 1, 2 and 8, respectively.

Table 4.1: Evaluation results in terms of F-measure.

Noise	Methods					
	MVEF	OOF	PCTV	NSDM	SAG	NASAG
$\varepsilon_0 = 0$	**0.89**	**0.89**	**0.89**	0.86	0.88	**0.89**
$\varepsilon_0 = 10$	0.80	0.70	0.71	0.78	0.75	**0.87**

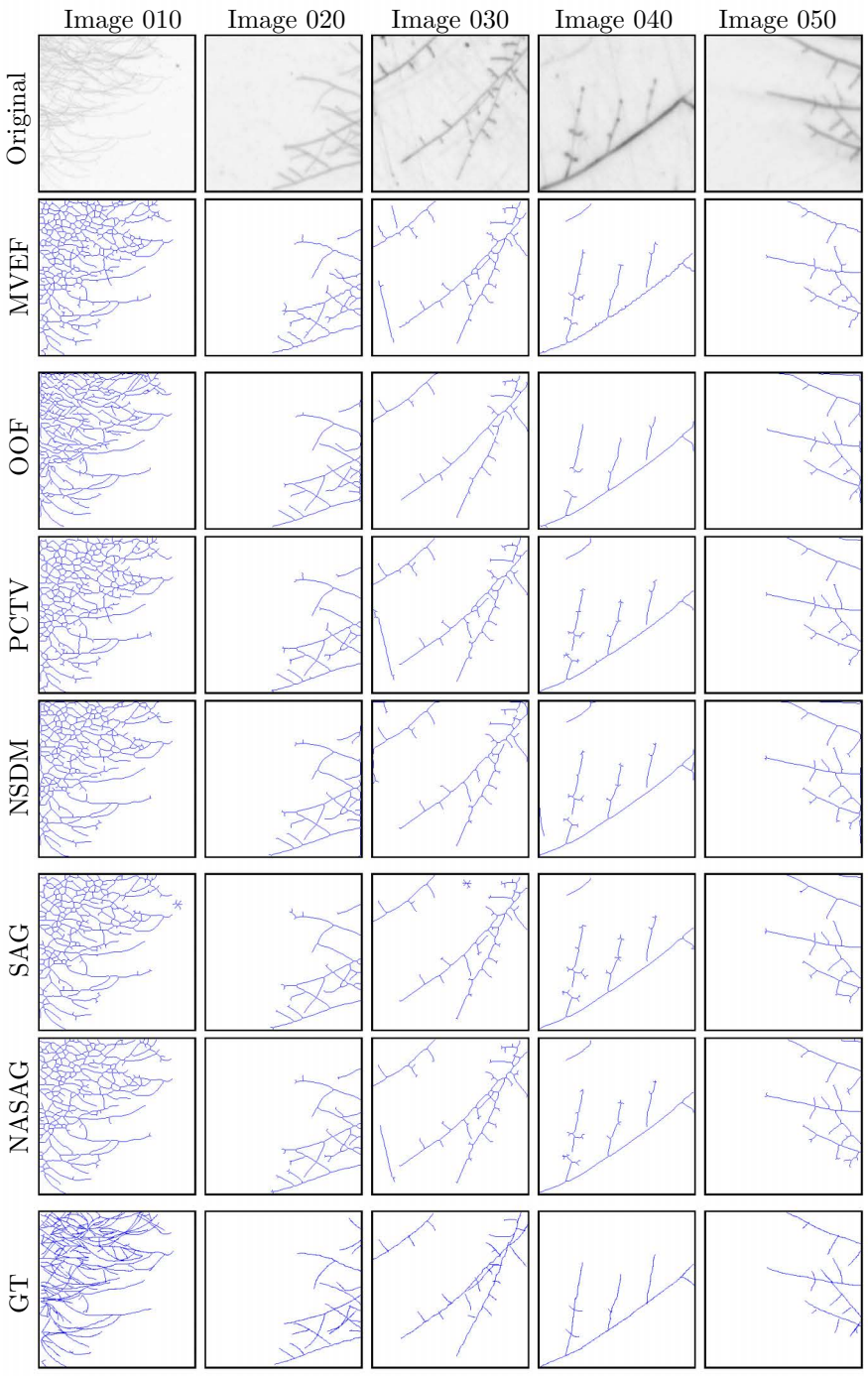


Figure 4.4: Detection results on Images 010, 020, 030, 040 and 050. Please note that the original images have been reversed (dark lines with a bright background) for better display.

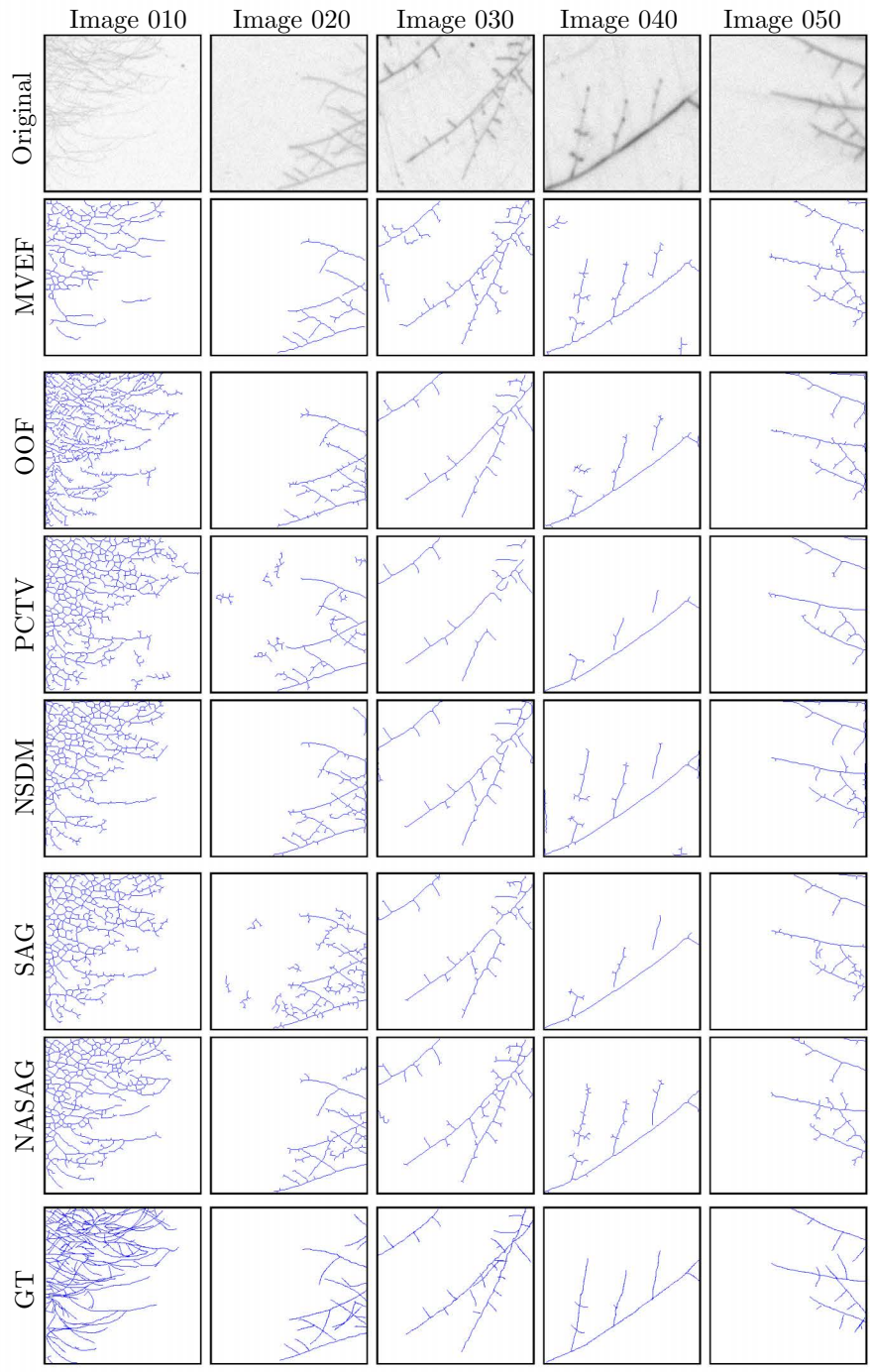


Figure 4.5: Detection results on the noisy version of Images 010, 020, 030, 040 and 050 corrupted by white Gaussian noise with an intensity of $\varepsilon_0 = 10$. Please note that the original images have been reversed (dark lines with a bright background) for better display.

Table 4.2: Evaluation results in terms of runtime (s).

Methods	MVEF	OOF	PCTV	NSDM	SAG	NASAG
Runtime	0.17	1.31	5.19	0.18	3.59	0.91

In the noise-free case, the quantitative evaluation results are reported in Tab. 4.1. It is observed that all the methods yield acceptable results in terms of F-measure. Specifically speaking, the MVEF method, the OOF method, the PCTV method as well as the proposed method obtain the best performance, while the NSDM method provides slightly lower F-measure values compared with the other methods. We also illustrate the detection results of each method on five sample images in Fig. 4.4. As we can see, most of the fungus centerlines have been successfully extracted by each method respectively. Nevertheless, the centerlines yielded by the MVEF method are not smooth enough, which may cause subsequent errors for some applications, e.g. fungus length measurement. In addition, the conventional SAG method sometimes yields false detection results at positions of blob structures. This is mainly caused by the intrinsic defect of the anisotropy stretch effect.

In the noisy case, as shown in Tab. 4.1, the F-measure values of the competing methods decrease sharply as a consequence of noise corruption. In addition, the detection results of each method on five sample noisy images are illustrated in Fig. 4.5. In the results yielded by the competing method, many true lines are missing, while some spurious results occur alongside the true lines and in the background. In contrast, the NASAG method still obtains an acceptable performance in terms of F-measure, and yields the best visual results as shown in Fig. 4.5. This is mainly because we employ adaptive anisotropy factors, thereby enhancing the noise-robustness. According to these experimental results, we learn that the NASAG method is more robust to noise than the competing methods.

With respect to the time efficiency, the runtime of each method has been tested on a personal computer configured with Intel Core i7-3770 CPU (3.40GHz) with 16-GB RAM, running MATLAB R2014b. For each method, the average runtime for processing a single image is reported in Tab. 4.2. One can see that the MVEF method and the NSDM method are among the most efficient methods. The major reason is that they use Hessian-based methods instead of matched filters, avoiding filtering the image in all directions. Moreover, the runtime of our method is acceptable compared with the OOF method, the PCTV method and the conventional SAG method. It is noteworthy that the runtime of our method is much lower than the conventional SAG method,

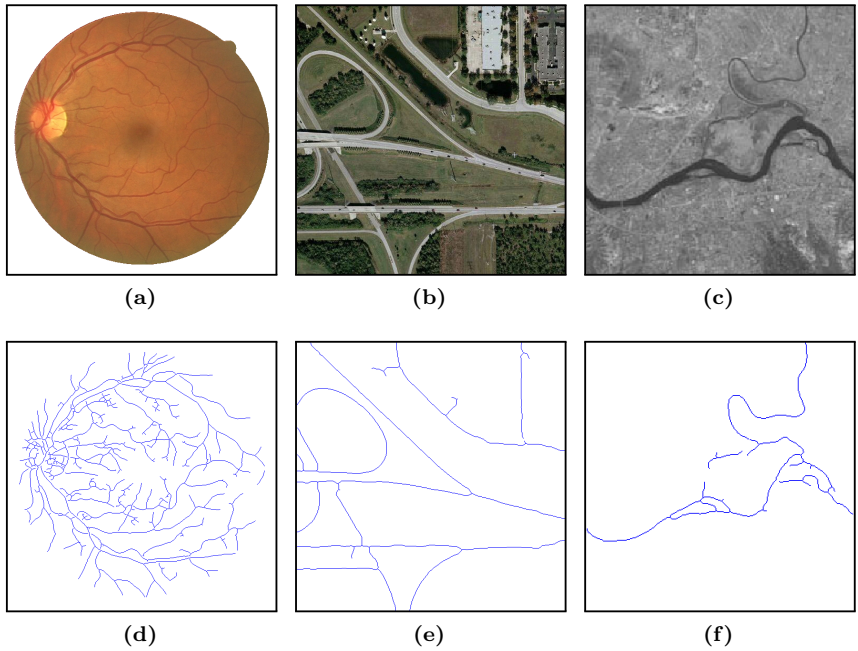


Figure 4.6: Illustration of detection results of vessels, roads and rivers using the proposed method. (a)-(c) are original images while (d)-(f) are their corresponding line detection results.

although both methods employ anisotropic Gaussian kernels. This is because the adaptive anisotropy factor used in the NASAG method reduces a great many unnecessary filtering steps.

In summary, on the one hand, in terms of the quantitative evaluation and the visual illustration, the NASAG method is more noise-robust than the competing methods, achieving a satisfactory detection performance in both the noise-free and the noisy cases. On the other hand, the NASAG method consumes an acceptable execution time and is more efficient than the OOF method, the PCTV method and the SAG method. Although the MVEF method runs the fastest among all the methods, it is vulnerable to noise, and moreover, its detection results for straight lines are wavy rather than smooth.

4.4.2. Other applications

While we have evaluated our method on a task of fungus detection, our proposed method also shows considerable potential for some other applications

in which line features play the leading role. As illustrated in Fig. 4.6, our method yields acceptable results on tasks of vessel detection in retina images, road detection in satellite images and river detection in synthetic-aperture radar images. Further evaluation for other applications is the subject of future research.

4.5. Conclusions

We have studied and presented a filtering framework for line detection based on novel anisotropic Gaussian kernels. This framework can be easily adapted to noisy environments, and is also reliable to detect lines that are heterogeneous in terms of width, prominence, etc. We have also developed a full-fledged algorithm for line detection using the proposed NASAG kernels. Experimental results on a biological image dataset as well as a noisy version thereof demonstrate that compared with the selected competing methods, the proposed NASAG method improves the noise-robustness while consuming an acceptable execution time.

5 Unilateral second-order Gaussian kernel with application to image denoising

In this chapter, we address the problem of blob characterization. We view blobs as prominent and isolated visual artifacts, which are rather close to the notion of mountains in topography. Therefore, we characterize a blob using characteristics of spatial location, spatial scale, and intensity prominence. To formalize this proposal, we study the properties of the normalized second-order Gaussian kernel. Accordingly, we propose a novel kernel named the unilateral second-order Gaussian kernel to obtain a quantitative measurement of blob characteristics. This method not only identifies the blob position, prominence and scale, but also suppresses non-blob structures well, and as such, this method can facilitate the implementation of blob reconstruction and blob reduction. Besides, to tackle the blob-like noise that occurs in high-ISO long-exposure images, we develop a denoising scheme by employing a blob reduction procedure for each of the selected conventional denoising methods. The experimental results demonstrate that in high-ISO long-exposure image denoising, the methods incorporating blob reduction outperform the original conventional methods.

The material of this chapter is based on the following publications:

- Wang, G., Lopez-Molina, C., and De Baets, B. (2017b). Blob reconstruction using unilateral second order Gaussian kernels with application to high-ISO long-exposure image denoising. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4817–4825
- Wang, G., Lopez-Molina, C., and De Baets, B. (2019a). High-ISO long-exposure image denoising based on quantitative blob characterization. *IEEE Transactions on Image Processing*, Under revision

5.1. Motivation

Noise is intrinsic to imaging systems. When generating raw data, an image sensor incorporates certain temporal noise, correlated random noise (Maggioni et al., 2014) and fixed pattern noise (FPN) (Xu et al., 2018a; Tsin et al., 2001). When transforming the raw data into digital images, the in-camera

image signal processor performs a series of operations, e.g. demosaicing, value clipping, white balance, color adjustment, gamma correction, tone mapping, JPEG compression, etc (Nam et al., 2016). These procedures inevitably add or increase image noise (Tsin et al., 2001). Over the past several decades, numerous methods have been proposed for image denoising (Shao et al., 2014). Most of them assume that the occurred noise can be approximately synthesized by Poisson or/and Gaussian noise (Foi et al., 2008; Luisier et al., 2011).

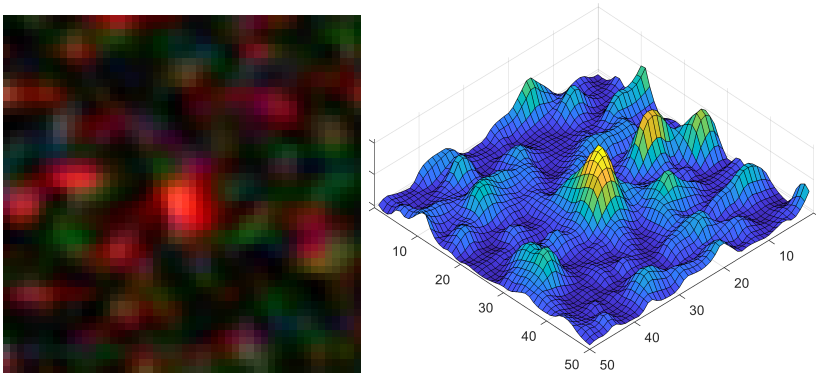
Generally, image denoising methods can be categorized into spatial-domain-based methods, transform-domain-based methods and learning-based methods. Spatial-domain-based methods mainly utilize local image correlations (e.g., the anisotropic diffusion (AD) method (Black et al., 1998) and the bilateral filtering (BF) method (Tomasi and Manduchi, 1998)) or non-local image self-similarities (e.g., the non-local means (NLM) method (Buades et al., 2005)). Transform-domain-based methods usually represent image patches by the orthonormal basis (e.g., wavelets (Chang et al., 2000b), curvelets (Starck et al., 2002), contourlets (Do and Vetterli, 2005) and bandelets (Le Pennec and Mallat, 2005)) with a series of coefficients. The smaller coefficients are the high-frequency part of the processed image that are related to the image details and noise. Therefore, noise can be suppressed by adjusting the smaller coefficients. Representative methods include a method based on wavelet soft-thresholding (Chang et al., 2000b) and a method based on Bayes least squares-Gaussian scale mixtures (Portilla et al., 2003). In particular, Dabov et al. (2007) proposed a method by exploiting the self-similarities of image patches and the correlation of wavelet coefficients. Besides, there are also methods based on dictionary learning (Elad and Aharon, 2006) or deep neural networks (Zhang et al., 2017a; Remez et al., 2018).

The denoising methods mentioned above generally work well when the image is captured with appropriate camera settings in a good light condition, or when noise adapts to well-known parameterizations. Nevertheless, real-world noise is more complex than synthetic noise, varying with different image sensors, camera settings (e.g. ISO sensitivity and exposure value) and even imaging environments. Typically, in images taken in low-light conditions, there is noise incurred by high-ISO or/and long-exposure settings (Plotz and Roth, 2017), especially when using suboptimal settings. This type of noise is composed of signal-independent temporal noise, signal-dependent temporal noise (Liu et al., 2006), spatially correlated random noise and spatially correlated FPN (Maggioni et al., 2014). It has been reported that many representative methods, including state-of-the-art ones, have limitations in removing such noise. Although there are methods that can relieve low-light imaging problems at time of capture, such as flash/no-flash pairs (Petschnigg et al., 2004), raw data

denoising (Chen et al., 2018a), low-light image enhancement (Guo et al., 2017) or multiple frame denoising (Godard et al., 2018), methods for removing real-world noise in existing images are still highly desired. In the specific case of imaging in a low-light environment, alternative denoising methods in literature include non-linear filtering (Rabie, 2004), multi-resolution denoising (Zhang and Gunturk, 2008; Pyo et al., 2011) and sparse-coding-based processing (Xu et al., 2018b). Nevertheless, few methods have addressed the noise incurred by high-ISO and long-exposure settings.

In digital photography, high-ISO and long-exposure settings are necessary for taking photos of objects in a low-light environment (Petschnigg et al., 2004), especially when a large depth-of-field should be guaranteed. However, such settings will compound FPN, thereby entailing heavy noise. The noise will become more complex after passing through the in-camera image signal processor. Visually, the resulting noise has a significant spatial heterogeneity. An example patch of a black background affected by high-ISO and long-exposure noise is displayed in Fig. 5.1. It can be seen that some noisy spots are spatially non-uniform and locally isolated. In image processing, such structures differing from their surroundings in visual properties, e.g. brightness or color, are also referred to as blobs (Koenderink, 1984). Therefore, we refer to the spatially non-uniform and locally isolated noise appearing in high-ISO long-exposure images as blob noise. It can be seen that some noisy spots differ from their surroundings in color. We refer to the spatially non-uniform but locally isolated noise as blob noise. Different from other types of noise, blob noise has a certain shape while occupying a certain area, and by its very nature, blob noise seriously damages the image self-similarity (Buades et al., 2005). Conventional denoising algorithms can hardly recognize whether a structure is a noise blob or a visual image element (Chen et al., 2018a). Moreover, it is difficult to tackle blob noise in the frequency domain, since blob noise has a large number of low frequency components that are completely mixed with true image contents (Pyo et al., 2011). These reasons make blob noise reduction a very challenging task.

In order to tackle blob noise, we propose a denoising scheme that incorporates three sequential steps: blob characterization, blob reconstruction and blob reduction. With respect to blob characterization, quite a few blob detection methods have been developed over the past several decades, several representative ones being the Laplacian of Gaussian (LoG) method (Lindeberg, 1998b), the Top-Hat (TH) method (Breen et al., 1991) and the Hessian-based LoG (HLoG) method (Zhang et al., 2015b). Despite their popularity, these methods are essentially used for blob enhancement, and thus, they can hardly yield a quantitative measurement of blob characteristics. Therefore, they



(a) **(b)**

Figure 5.1: Image patch containing blob noise (a) and the 3D visualization of its red channel (b).

cannot be applied to blob reconstruction and blob reduction. Moreover, they also yield significant responses for some non-blob structures like edges and lines. Hence, we are still in need of a method that can inherently and quantitatively capture the physical characteristics of blobs, which can subsequently facilitate blob reconstruction and blob reduction.

In this paper, capitalizing on multiscale computer vision, we explore a topographical approach to characterize blobs, interpreting the image as a geographical surface. This interpretation is not new and has produced interesting methods in literature. In line detection, for example, authors use jargon like valleys or ridges to describe curvilinear structures (Lindeberg, 1998a). The topographical counterpart of a blob measurement is simple: mountain measurement (Helman, 2005). As displayed in Fig. 5.1(b), blobs can be thought of as prominent and isolated visual artifacts, which are rather close to the notion of mountain in topography. Hence, in this work, we characterize a blob using characteristics of spatial location, spatial scale and intensity prominence.

To formalize our proposal, we study the properties of the normalized second-order Gaussian kernel, and accordingly, propose a novel kernel, namely the unilateral second-order Gaussian (USG) kernel, to obtain a quantitative measurement of blob characteristics. In the scale-space framework, the USG kernels are normalized to yield a maximum response, which exactly reflects the blob prominence, at the scale of the observed blob. In addition, the proposed USG kernels topographically retain the minimum response among all directions

and therefore suppress non-blob structures effectively. In this way, we are able to obtain a quantitative measurement of the blob position, the scale as well as the prominence, and accordingly, we can reconstruct a blob map. Subsequently, we design a scheme to tackle blob noise in high-ISO long-exposure images, employing a blob reduction procedure as a cheap preprocessing step for conventional denoising methods. The main part of the blob-like noise is reduced by a reconstructed blob map, while the residual noise is further removed by each of the selected conventional denoising methods, i.e., the BF method (Tomasi and Manduchi, 1998), the NLM method (Buades et al., 2005) and the color version of block-matching and 3-D filtering (CBM3D) method (Dabov et al., 2007), the multi-channel weighted nuclear norm minimization (MWNNM) method (Xu et al., 2017a) and the trilateral weighted sparse coding (TWSC) method (Xu et al., 2018b).

This chapter is organized as follows. Section 5.2 recalls related work on real-world image denoising and blob detection. In Section 5.3, we elaborate a method to quantify blob characteristics. Subsequently, Section 5.4 presents a denoising scheme targeting high-ISO long-exposure noise. The experimental results as well as discussions are included in Section 5.5, while Section 5.6 concludes this chapter.

5.2. Related work

In this section, we recall related work on real-world image denoising and blob detection, respectively.

5.2.1. Real-world image denoising

Generally, image noise can be divided into categories of temporal noise, correlated noise and FPN (Azzari et al., 2018). Temporal noise includes thermal noise, readout noise, quantization noise, shot noise, dark current noise, etc (Xu et al., 2018a; Tsin et al., 2001). The specific causes for each type of noise vary considerably. For example, thermal noise is generated by the load resistor, and renders into additive white Gaussian noise (Hui and O'Sullivan, 2009). Readout noise, alternatively, is generated during the process of charge-to-voltage conversion, which is inherently inaccurate (Brooks et al., 2019). More detailed inspections for shot noise and temporal noise can be found in (Hui and O'Sullivan, 2009) and (Akyüz and Reinhard, 2007), respectively. Generally, temporal noise is usually assumed to obey Poisson or/and Gaussian distributions. However, noise in real-world images also contains correlated noise and FPN, which displays non-uniform spatial characteristics (Goossens et al., 2012; Azzari et al., 2018). The correlated noise might be incurred by the

physics of the acquisition system, the readout process, the cross-talk between neighboring pixels, or the processing performed on the raw data (Azzari et al., 2018). The major sources of FPN include dark signal non-uniformity and photon-response non-uniformity (Xu et al., 2018a). FPN will arise with the increase of ISO sensitivity or/and exposure time. Thus, FPN usually appears in images taken in a low-light environment. After passing through an image signal processor, the noise will become very difficult to address (Tsin et al., 2001). Due to the mismatch between the actual noise characteristics and the simplified noise models, many conventional methods might underperform in removing real-world noise (Pižurica et al., 2013).

In literature, there are already some proposals for denoising images taken in low-light environments. A pioneering method is the one proposed by Rabie (2004), which uses adaptive hybrid mean and median filters to attenuate stuck-pixel noise, blue-channel noise and JPEG artifacts. But this method still lacks an extensive and quantitative evaluation. The method presented in (Zhang and Gunturk, 2008) classifies noise into low-frequency and high-frequency noise, and then reduces low-frequency noise by multiresolution bilateral filters. This proposal is further developed in (Pyo et al., 2011), which uses the BM3D method (Dabov et al., 2007) to process the down-sampled chrominance channel. Sadly, such down-sampling also leads to a loss of certain image details. Chatterjee et al. (2011) proved that demosaicing is the main cause of random mid-frequency splotch noise, which led to a joint proposal for denoising and demosaicing. Although promising, this method is not applicable to noisy images, of which the raw data is not available. There are also methods developed for addressing correlated noise (a.k.a. colored noise) that is caused by the physics of the acquisition system, the procedure of demosaicking, etc. (Azzari et al., 2018). Goossens et al. (2008) modified the NLM method by further employing a post-processing procedure and a correlated noise estimator. Compared with the methods designed for Gaussian noise, the improved NLM method for correlated noise removal (NLMC method) (Goossens et al., 2008) can better suppress the correlated noise. In addition, building on the estimation of the probability of signal presence, *hidden Markov tree* models and Gaussian scale mixture models, Goossens et al. (2009) modelled the signal of interest in the wavelet domain for correlated noise removal. Other example methods for correlated noise removal can be found in (Matrecano et al., 2012), (Mäkinen et al., 2019) and (Tiirola, 2019). For removing noise specifically brought by high-ISO settings, Xu et al. (2017a) proposed the MWNNM method, but this method is rather time-consuming to execute. A very recent method uses the trilateral weighted sparse coding scheme (TWSC method) (Xu et al., 2018b) and obtains a promising performance, but it still has limitations to address noisy patches that are quite spatially correlated. Besides, benefiting from the

modelling capability of deep neural networks, some methods introduce deep learning techniques to image denoising, and have achieved state-of-the-art performance on some datasets. However, such methods might be over-fitted to the noise in the training datasets, and thus, might generalize poorly to noisy images with more complex noise (Guo et al., 2019). Despite the works mentioned above, high-ISO long-exposure image denoising still remains a challenging task.

5.2.2. Blob detection

Blob detection is a long-standing task in image processing. Quite a few blob detection methods have been developed, and most of them rely on a signal-based interpretation. Two pioneering approaches are the LoG method and Difference of Gaussian method (Marr and Hildreth, 1980), both of which are inspired by the early computational models of the human visual system. The two methods perform an intrinsic Gaussian smoothing that makes them more noise-robust than morphology-based methods, like the TH method (Breen et al., 1991) or the H-dome method (Vincent, 1993). For detecting blobs with heterogeneous sizes, Lindeberg (1998b) proposed the multiscale LoG (MLoG) method in the context of the then-new scale-space theory. The MLoG method convolves the image with a bank of kernels that covers all possible scales of blobs, yielding a maximum convolutional response at the matched scale (a.k.a. characteristic scale) for each blob (Diciotti et al., 2010). Lindeberg (1998b) also used the determinant of the Hessian (DoH) to detect blob structures, embodying the idea that positions in the image with large and positive DoH values are likely to belong to blobs. Compared to the MLoG method, the DoH method only produces responses for regions that contain significant variations along two orthogonal directions, and therefore implies a more restrictive condition for detecting blobs than the MLoG method (Lindeberg, 2015). Elaborating on Lindeberg's work and the theory of multiscale computer vision, several other authors have presented more advanced proposals. For instance, Kong et al. (2013a) presented a method based on generalized LoG kernels (gLoG method), which estimates scales (a.k.a. sizes), shapes and orientations of the observed blobs using a bank of multiscale and anisotropic convolutional kernels, at the cost of heavy computation. Also, Zhang et al. (2015b) proposed the HLoG method that smooths the image using LoG kernels and subsequently identifies the (overall) optimal scale using a Hessian analysis. However, the HLoG method tends to underperform when the blobs in the image are heterogeneous in size.

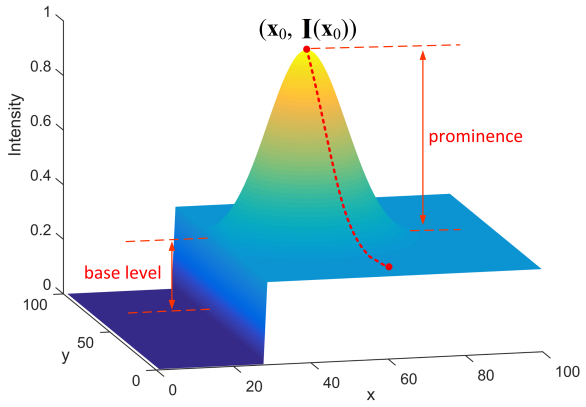


Figure 5.2: Visual representation of a blob structure and its measurable characteristics.

5.3. The unilateral second-order Gaussian kernel

As stated earlier, existing blob detection methods are not able to yield a quantitative measurement of blob characteristics. Moreover, these methods sometimes produce significant responses for non-blob structures. To overcome these shortcomings, in this section, we study the properties of the normalized second-order Gaussian kernels in scale-space, thereby proposing the USG kernel, which can quantitatively measure the blob characteristics.

5.3.1. Modelling a blob structure

Blobs have been modelled in different ways in literature. In this paper, we adopt the definition presented by Lindeberg (1993), which mathematically describes a blob based on the idea that a blob would extend until it merges with another blob. Let a two-dimensional continuous signal $\mathbf{I} : \mathbb{R}^2 \to \mathbb{R}$ obtain a local maximum $\mathbf{I}(\mathbf{x}_0)$ at the location $\mathbf{x}_0 = [x_0, y_0]^{\mathrm{T}}$, and let $\mathbb{C}(\mathbf{x}_0)$ be a convex region that contains $\mathbf{x}_0$. For any value of the signal intensity $b \in [0, \mathbf{I}(\mathbf{x}_0)[$, topographically, the protuberance surface $\mathbb{B}(\mathbf{x}; \mathbf{x}_0, b)$ is defined as follows:

$$\mathbb{B}(\mathbf{x}; \mathbf{x}_0, b) = \big\{ (\mathbf{x}, \mathbf{I}(\mathbf{x})) \mid \mathbf{x} \in \mathbb{C}(\mathbf{x}_0),\ b \leq \mathbf{I}(\mathbf{x}) \leq \mathbf{I}(\mathbf{x}_0) \big\}\,. \tag{5.1}$$

Obviously, $(\mathbf{x}_0, \mathbf{I}(\mathbf{x}_0))$ is the peak point of the protuberance surface.

Since $(\mathbf{x}_0, \mathbf{I}(\mathbf{x}_0))$ is a local maximum, in general, we can find a $b < \mathbf{I}(\mathbf{x}_0)$ that results in an $\mathbb{B}(\mathbf{x}; \mathbf{x}_0, b)$ such that there exists a monotonically increasing

path from any point within the set $\mathbb{B}(\mathbf{x};\mathbf{x}_0,b)$ to the peak point $(\mathbf{x}_0,\mathbf{I}(\mathbf{x}_0))$. Subsequently, we reduce b until we reach a b_0 such that within the resultant $\mathbb{B}(\mathbf{x};\mathbf{x}_0,b_0)$ not all the points have a monotonically increasing path to the peak point $(\mathbf{x}_0,\mathbf{I}(\mathbf{x}_0))$. Here, the resultant $\mathbb{B}(\mathbf{x};\mathbf{x}_0,b_0)$ is referred to as a blob structure, while the b_0 is defined as its base level. Accordingly, the blob prominence is defined as the intensity difference between the local maximum and the base level, i.e.,

$$p_0 = \mathbf{I}(\mathbf{x}_0) - b_0\,. \tag{5.2}$$

Mathematically, we assume that a blob structure has a Gaussian shape (Smal et al., 2010). Based on the aforementioned definition of a blob structure as well as its characteristics, we describe a blob structure using four parameters: the center position, the scale, the prominence and the base level. Specifically, we formulate a blob structure as follows:

$$\mathbf{\Lambda}_0(\mathbf{x}) = p_0 \cdot \exp\left(-\frac{(\mathbf{x}-\mathbf{x}_0)^{\mathrm{T}}(\mathbf{x}-\mathbf{x}_0)}{2\omega_0^2}\right) + b_0\,, \tag{5.3}$$

where $\mathbf{x}_0$ represents the center position, $p_0 \in]0,\,1]$ the prominence, $b_0 \in [0,\,1[$ the base level and ω_0 the spatial scale. As an illustration, Fig. 5.2 displays a blob structure as well as its characteristics.

In fact, the image intensity at each location is known once the image is given. Thus, according to Eq. (5.2), we can have b_0 once p_0 is obtained. That is, the center position, the prominence and the blob scale are sufficient to describe a modelled blob. Therefore, the blob can be delimited and reconstructed once these parameters are determined.

5.3.2. Scale-invariant normalized second-order Gaussian kernel

The second-order Gaussian (SOG) kernel and its variants have been widely used for blob detection (Kong et al., 2013a). The expression of an SOG kernel is given by (Wang et al., 2019c):

$$g''(\mathbf{x};\sigma) = \frac{x^2-\sigma^2}{2\pi\sigma^6}\exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\sigma^2}\right)\,. \tag{5.4}$$

We easily obtain the directional version of an SOG kernel by rotating the kernel with an orientation θ (Lopez-Molina et al., 2015):

$$g''(\mathbf{x};\sigma,\theta) = \frac{([\cos\theta,\sin\theta]\mathbf{x})^2-\sigma^2}{2\pi\sigma^6}\exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{R}_\theta^{\mathrm{T}}\mathbf{R}_\theta\mathbf{x}}{2\sigma^2}\right)\,, \tag{5.5}$$

where

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{5.6}$$

Since the magnitude of a non-normalized Gaussian kernel (and that of its derivatives) decreases as σ increases, conventionally, the SOG kernel is normalized as follows (Lindeberg, 1998a):

$$\hat{g}''(\mathbf{x};\sigma) = \sigma^{2\gamma} \cdot g''(\mathbf{x};\sigma)\,. \tag{5.7}$$

where $\gamma \in \mathbb{R}_+$ is referred to as the scale normalization factor.

Although having gained popularity, conventionally normalized SOG kernel also produces significant responses for some non-blob structures. Moreover, the relationship between the obtained response and the true blob prominence is implicit, thereby leading to failures of blob reconstruction and blob reduction. For this reason, we intend to design a kernel that can yield a quantitative measurement of blob characteristics. Inspired by the normalization method in Eq. (5.7), we normalize the SOG kernel, which would be used for blob characterization, as follows:

$$g''_{\mathrm{B}}(\mathbf{x};\sigma) = (-1)^\eta \cdot \beta \cdot \sigma^{2\gamma} \cdot g''(\mathbf{x};\sigma)\,, \tag{5.8}$$

where β is a constant ensuring that the obtained prominence measurement precisely reflects the original blob prominence, while $\eta \in \{0, 1\}$ allows the kernel to be applicable to both bright ($\eta = 1$) and dark ($\eta = 0$) blob detection. In this work, we address the detection of bright blobs, thus setting $\eta = 1$.

To obtain the response of the SOG kernel to a blob, we convolve $g''_{\mathrm{B}}(\mathbf{x};\sigma)$ with the blob structure in Eq. (5.3). Without loss of generality, we consider the case $\mathbf{x}_0 = \mathbf{0}$. The resulting response at the center location of the blob is given by:

$$\begin{aligned} B &= g''_{\mathrm{B}}(\mathbf{x};\sigma) * \mathbf{\Lambda}_0(\mathbf{x}) \,|_{\mathbf{x}=\mathbf{x}_0} \\ &= -\beta\sigma^{2\gamma} \cdot g''(\mathbf{x};\sigma) * \left(p_0 \cdot \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\omega_0^2} \right) + b_0 \right) \Bigg|_{\mathbf{x}=\mathbf{x}_0} \\ &= \beta p_0 \sigma^{2\gamma-2} \omega_0^2 (\omega_0^2 + \sigma^2)^{-1} \left(1 - \omega_0^2 (\omega_0^2 + \sigma^2)^{-1} \right). \end{aligned} \tag{5.9}$$

The detailed proof is presented in Appendix A.4.

In order to identify the scale of the blob, we intend to find the σ that yields a maximum response in scale-space. For $\sigma \in \mathbb{R}_+$, it is easy to verify that the second-order derivative of B w.r.t. σ is negative. Hence, we determine the

maximum value of B in scale-space by computing the first derivative of B w.r.t. σ and setting it to zero. After a few algebraic manipulations, we find that B reaches its maximum value at the scale:

$$\sigma^* = \gamma^{\frac{1}{2}}(2-\gamma)^{-\frac{1}{2}}\omega_0\,. \tag{5.10}$$

Substituting the σ in Eq. (5.9) by that in Eq. (5.10), we obtain the maximum response in scale-space:

$$B^* = \frac{\beta p_0 \gamma}{4(2-\gamma)^{\gamma-2}}\omega_0^{2\gamma-2}\,. \tag{5.11}$$

In order to make the normalized SOG kernel scale-invariant, i.e., the response B^* is independent from ω_0, we set $\gamma = 1$. Accordingly, Eq. (5.10) becomes $\sigma^* = \omega_0$, while Eq. (5.11) becomes

$$B^* = \frac{1}{4}\beta p_0\,. \tag{5.12}$$

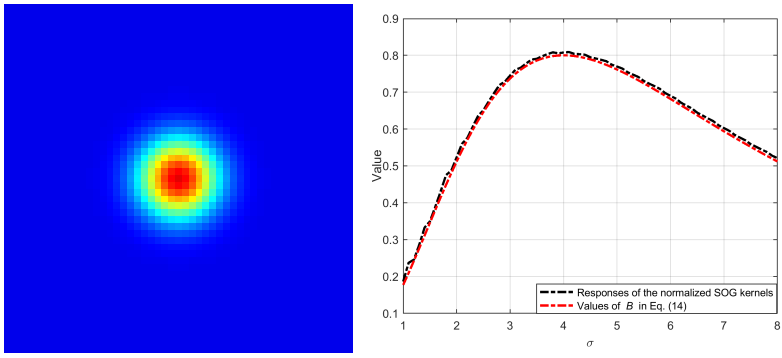
To make the obtained response precisely reflect the blob prominence, i.e., $B^* = p_0$, we set $\beta = 4$.

Eventually, taking Eq. (5.8) and the discussion above into consideration, we get the scale-invariant normalized SOG kernel:

$$g''_{\mathrm{B}}(\mathbf{x};\sigma) = -4\sigma^2 \cdot g''(\mathbf{x};\sigma)\,. \tag{5.13}$$

To illustrate the effectiveness of the normalized SOG kernel in blob characterization, we model a blob based on Eq. (5.3), setting $\omega_0 = 4$, $p_0 = 0.8$ and $b_0 = 0.1$, as displayed in Fig. 5.3(a). Subsequently, we convolve $g''_{\mathrm{B}}(\mathbf{x};\sigma)$ with the modelled blob. The black curve in Fig. 5.3(b) illustrates the responses obtained at the blob center in scale-space. According to Eq. (5.9), the theoretical responses obtained at the blob center are supposed to follow the red curve in Fig. 5.3(b). As can be seen, the obtained responses agree with the theoretical values very well. Moreover, in scale-space, the normalized SOG kernel obtains its maximum value at the scale of 4, which is identical to the scale of the modelled blob. Furthermore, the maximum response of $g''_{\mathrm{B}}(\mathbf{x};\sigma)$ is 0.8, which equals the prominence of the modelled blob.

We are now in a position to summarize that the scale-invariant normalized SOG kernels yield the maximum response at the location $\mathbf{x}_0$ and at the scale ω_0. Moreover, the maximum response B^* precisely reflects the blob prominence p_0, regardless of the blob scale.



(a) **(b)**

Figure 5.3: Illustration of a modelled blob and the obtained responses at the blob center. (a) A modelled blob ($\omega_0 = 4$, $p_0 = 0.8$ and $b_0 = 0.1$); (b) The responses yielded by the normalized SOG kernels in scale-space and the analytical values of B obtained by varying σ in scale-space ($\gamma = 1$, $\beta = 4$).

5.3.3. The unilateral second-order Gaussian kernel

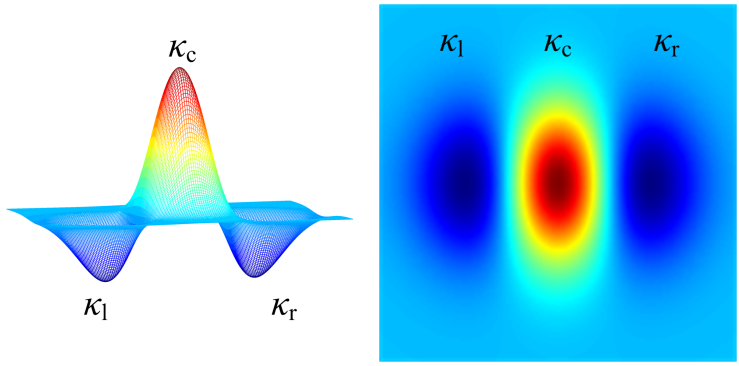
We premise the discussion above on the assumption that the blob structure has a flat surrounding background. In fact, blobs are usually situated adjacent to other image structures. That is, the intensity differences in different directions are not always identical. According to the definition of blob prominence in Section 5.2, we need to find the prominence as the minimum intensity difference among all directions. For this purpose, we propose to use the USG kernel, which inherits the aforementioned merit of the scale-invariant normalized SOG kernel, to obtain the intensity difference along each direction.

According to Eqs. (5.5) and (5.13), the directional version of the scale-invariant normalized SOG kernel is given by:

$$g''_{\mathrm{B}}(\mathbf{x};\sigma,\theta) = 2\,\frac{\sigma^2 - \left([\cos\theta,\sin\theta]\mathbf{x}\right)^2}{\pi\sigma^4}\exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{R}_{-\theta}\mathbf{R}_{\theta}\mathbf{x}}{2\sigma^2}\right). \tag{5.14}$$

We illustrate such a kernel in Fig. 5.4.

From Eq. (5.14) and Fig. 5.4, we see that a scale-invariant normalized SOG kernel spatially consists of three parts, i.e., a central part κ_{c} as well as two symmetrical side parts κ_{l} and κ_{r}. Each of them can be expressed as an



(a) **(b)**

Figure 5.4: Three-dimensional and planar representations of a normalized SOG kernel.

individual kernel, leading to:

$$\kappa_c(\mathbf{x};\sigma,\theta) = \begin{cases} g''_B(\mathbf{x};\sigma,\theta) & \text{, if } g''_B(\mathbf{x};\sigma,\theta) > 0 \\ 0 & \text{, otherwise} \end{cases}$$
$$\kappa_l(\mathbf{x};\sigma,\theta) = \begin{cases} g''_B(\mathbf{x};\sigma,\theta) & \text{, if } g''_B(\mathbf{x};\sigma,\theta) < 0 \text{ and } x < -y\tan\theta \\ 0 & \text{, otherwise} \end{cases} . \quad (5.15)$$
$$\kappa_r(\mathbf{x};\sigma,\theta) = \begin{cases} g''_B(\mathbf{x};\sigma,\theta) & \text{, if } g''_B(\mathbf{x};\sigma,\theta) < 0 \text{ and } x > -y\tan\theta \\ 0 & \text{, otherwise} \end{cases}$$

Note that $\kappa_l(\mathbf{x};\sigma,\theta)$ and $\kappa_r(\mathbf{x};\sigma,\theta)$ have the same sign, while $\kappa_c(\mathbf{x};\sigma,\theta)$ has the opposite one.

As a matter of fact, the kernel $g''_B(\mathbf{x};\sigma,\theta)$ measures the intensity difference between the central part and its two side parts, i.e., the average of the intensity difference on both sides along the direction θ. As a result, traditional SOG kernels usually lead to a great loss of directional information, since they always bind the two side parts together. In order to measure the blob prominence defined in Section 2, we propose the unilateral second-order Gaussian kernel as follows.

For a given scale σ, the USG kernel along the direction θ is defined by

$$\kappa_u(\mathbf{x};\sigma,\theta) = \kappa_c(\mathbf{x};\sigma,\theta) + 2\lambda(\mathbf{x})\kappa_l(\mathbf{x};\sigma,\theta) + 2(1-\lambda(\mathbf{x}))\kappa_r(\mathbf{x};\sigma,\theta)\,, \quad (5.16)$$

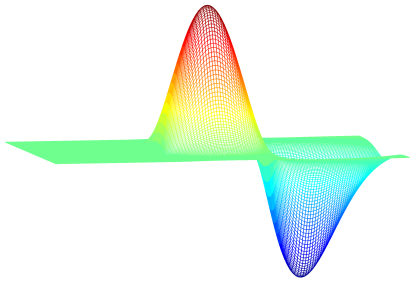


Figure 5.5: Three-dimensional illustration of a USG kernel.

where

$$\lambda(\mathbf{x}) = \begin{cases} 1 & \text{, if } |\kappa_{\mathrm{l}}(\mathbf{x};\sigma,\theta) * \mathbf{I}_{\mathrm{p}}(\mathbf{x})| > |\kappa_{\mathrm{r}}(\mathbf{x};\sigma,\theta) * \mathbf{I}_{\mathrm{p}}(\mathbf{x})| \\ 0 & \text{, otherwise} \end{cases}, \tag{5.17}$$

where $\mathbf{I}_{\mathrm{p}}(\mathbf{x})$ is the image patch centered on $\mathbf{x}$ with the same size of $\kappa_{\mathrm{u}}(\mathbf{x};\sigma,\theta)$. As an illustration, Fig. 5.5 shows a three-dimensional appearance of an USG kernel.

To make the filtering more efficient, we develop an equivalent implementation, convolving the SOG kernel with the signal as follows:

$$\begin{aligned} \mathbf{B}_{\mathrm{u}}(\mathbf{x};\sigma,\theta) &= \kappa_{\mathrm{u}}(\mathbf{x};\sigma,\theta) * \mathbf{I}(\mathbf{x}) \\ &= \lambda(\mathbf{x})\big[(\kappa_{\mathrm{c}}(\mathbf{x};\sigma,\theta) + 2\kappa_{\mathrm{l}}(\mathbf{x};\sigma,\theta)) * \mathbf{I}(\mathbf{x})\big] + \\ &\quad (1-\lambda(\mathbf{x}))\left[(\kappa_{\mathrm{c}}(\mathbf{x};\sigma,\theta) + 2\kappa_{\mathrm{r}}(\mathbf{x};\sigma,\theta)) * \mathbf{I}(\mathbf{x})\right], \end{aligned} \tag{5.18}$$

where

$$\lambda(\mathbf{x}) = \begin{cases} 1 & \text{, if } (-\kappa_{\mathrm{l}}(\mathbf{x};\sigma,\theta) + \kappa_{\mathrm{r}}(\mathbf{x};\sigma,\theta)) * \mathbf{I}(\mathbf{x}) > 0 \\ 0 & \text{, otherwise} \end{cases}. \tag{5.19}$$

Obviously, in this way, the directions uniformly selected from the range of $[0,\pi[$ are able to cover all possible directions.

5.3.4. Topographical measurement of blob characteristics

In order to accommodate the proposed USG kernels to digital image processing, discrete versions of these kernels are needed. We get the discrete SOG kernel

by sampling the formula in Eq. (5.14) in the 2D integer coordinates:

$$g''_{\mathrm{B}}(\mathbf{m};\sigma_i,\theta_j) = 2\frac{\sigma_i^2 - ([\cos\theta_j, \sin\theta_j]\mathbf{m})^2}{\pi\sigma_i^4} \exp\left(-\frac{\mathbf{m}^{\mathrm{T}}\mathbf{R}_{\theta_j}^{\mathrm{T}}\mathbf{R}_{\theta_j}\mathbf{m}}{2\sigma_i^2}\right) \tag{5.20}$$

with

$$\mathbf{R}_{\theta_j} = \begin{bmatrix} \cos\theta_j & \sin\theta_j \\ -\sin\theta_j & \cos\theta_j \end{bmatrix}, \tag{5.21}$$

where σ_i denotes the discrete scale taken from a scale set $\mathbb{S}$, and θ_j stands for the discrete direction taken from a direction set $\mathbb{D}$.

According to the expression of an SOG kernel in Eq. (5.20), we can easily obtain the discrete USG kernel $\kappa_{\mathrm{u}}(\mathbf{m};\sigma_i,\theta_j)$ by Eqs. (5.15), (5.16) and (5.17). Figure 5.6 displays the kernels that are used to generate USG kernels at a specific scale.

We obtain the response of the discrete USG kernel as follows:

$$\mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i,\theta_j) = \kappa_{\mathrm{u}}(\mathbf{m};\sigma_i,\theta_j) * \mathbf{I}(\mathbf{m})\,. \tag{5.22}$$

According to the definition of the blob prominence in Section 2, at each scale, the minimum response among all the directions is selected as the scale blob prominence. Therefore, at a give scale σ_i, the blob prominence is represented by

$$\mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i) = \min_{\theta_j\in\mathbb{D}} \mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i,\theta_j)\,. \tag{5.23}$$

Subsequently, the maximum blob prominence in scale-space is retained as the final blob prominence:

$$\mathbf{B}_{\mathrm{u}}(\mathbf{m}) = \max_{\sigma_i\in\mathbb{S}} \mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i)\,. \tag{5.24}$$

We then identify the positions of the blobs as the local maximizers of $\mathbf{B}_{\mathrm{u}}(\mathbf{m})$, and thereby obtaining the map of blob prominence based on the responses at these positions.

With respect to the scale information, as stated earlier, the response in scale-space reaches its maximum value at the scale of the original blob scale. As such, the scale of each blob is identified by

$$\mathbf{S}_{\mathrm{u}}(\mathbf{m}) = \operatorname*{argmax}_{\sigma_i\in\mathbb{S}} \mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i)\,. \tag{5.25}$$

Consequently, we obtain the blob characterization method based on the

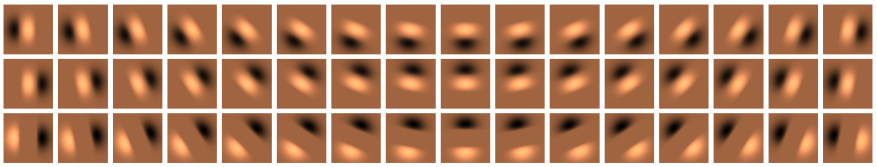


Figure 5.6: A bank of kernels that is used to generate USG kernels at a specific scale. The top row shows $(\kappa_c + 2\kappa_l)$, the middle row shows $(\kappa_c + 2\kappa_r)$ and the bottom row shows $(-\kappa_l + \kappa_r)$. The intensity range of each patch has been adjusted for a better display.

unilateral second-order Gaussian kernel (USG method) to quantitatively measure the blob characteristics, including the position, the prominence and the scale.

5.4. High-ISO long-exposure image denoising

Having proposed a method to quantitatively measure blob characteristics, in this section, we intend to tackle the problem of high-ISO long-exposure image denoising, which is rarely addressed in literature. We firstly present an approach to model high-ISO long-exposure noise. Subsequently, we propose a denoising scheme by employing a step of blob reduction as a preprocessing step for six selected conventional denoising methods, i.e. the BF (Tomasi and Manduchi, 1998), NLM (Buades et al., 2005), NLMC (Goossens et al., 2008), CBM3D (Dabov et al., 2007), MWNNM (Xu et al., 2017a) and TWSC (Xu et al., 2018b) methods.

5.4.1. Spatially modelling blob noise

Denoising is the process of restoring the original image by reducing the undesirable noise from a noisy image (Huang et al., 2014). As argued earlier, real-world noise is very complex, and might combine signal-dependent noise, signal-independent noise, FPN, etc (Xu et al., 2018a), apart from that induced by the own image processor. Approximately, real-world noise ψ_{rw} can be modelled by

$$\psi_{\text{rw}}(\mathbf{m}) = \mathcal{D}_{\text{isp}}\big(\mathcal{D}_{\text{clip}}\big(\psi_{\text{sd}}\left(\mathbf{I}_{\text{c}}(\mathbf{m})\right) + \psi_{\text{si}} + \psi_{\text{scr}}(\mathbf{I}_{\text{c}}(\mathbf{m}), \mathbf{m}) + \psi_{\text{fpn}}(\mathbf{I}_{\text{c}}(\mathbf{m}), \mathbf{m})\big)\big)\,, \tag{5.26}$$

where $\mathcal{D}_{\text{isp}}$ denotes the degradation in the image signal processor (Guo et al., 2019), $\mathcal{D}_{\text{clip}}$ represents the degradation brought by the clipping operation (Plotz and Roth, 2017), ψ_{sd} stands for spatially uncorrelated signal-dependent noise, ψ_{si} denotes spatially uncorrelated signal-independent noise, ψ_{scr}

stands for spatially correlated random noise (Maggioni et al., 2014), ψ_{fpn} represents spatially correlated FPN and $\mathbf{I}_{\text{c}}$ is the true (i.e. clean) image signal. From Eq. (5.26), it can be learned that modelling ψ_{rw} in the spatial space is very difficult. We simplify this problem by assuming that the blob-like artifacts in ψ_{rw} can be represented as additive blob noise. Therefore, we spatially model the degradation process brought by high-ISO long-exposure settings as a degradation function together with additive blob noise.

Accordingly, we model the noisy image in the spatial domain as follows:

$$\mathbf{I}_{\text{noi}}^{(q)}(\mathbf{m}) = \mathcal{D}^{(q)}\left(\mathbf{I}_{\text{c}}^{(q)}(\mathbf{m})\right) + \xi_{\text{b}}^{(q)}(\mathbf{m})\,, \tag{5.27}$$

where $\mathbf{I}_{\text{noi}}^{(q)}$ ($q \in \{1,2,3\}$) denote the q-th channel of a noisy color image, $\mathbf{I}_{\text{c}}^{(q)}$ denotes the q-th channel of the true (i.e., clean) image, $\xi_{\text{b}}^{(q)}$ stands for the additive blob-like noise, while $\mathcal{D}^{(q)}$ represents a degradation process on the image imposed by other types of noise, e.g., white Gaussian noise, Poisson noise, etc. As for $\xi_{\text{b}}^{(q)}$, we model it using spatially mixed Gaussian functions as follows:

$$\xi_{\text{b}}^{(q)} = \sum_{\mathbf{m}_i \in \mathbf{I}_{\text{c}}^{(q)}} \mathcal{H}^{(q)}(\epsilon - \epsilon_0) \cdot \mathbf{G}^{(q)}(\mathbf{m}; \mathbf{m}_i, \hat{\sigma}_i, \hat{p}_i)\,, \tag{5.28}$$

where $\mathcal{H}^{(q)}$ represents the Heaviside step function:

$$\mathcal{H}^{(q)}(\epsilon - \epsilon_0) = \begin{cases} 0 & \text{, if } \epsilon - \epsilon_0 < 0 \\ 1 & \text{, otherwise} \end{cases}\,, \tag{5.29}$$

in which ϵ denotes an argument that follows the continuous uniform distribution on the unit interval $[0,1]$ and $\epsilon_0 \in [0,1]$ is a constant. The term $H^{(q)}$ reflects the probability of FPN occuring at a location. If FPN occurs at $\mathbf{m}_i$, the blob noise is approximated by a local blob structure $G^{(q)}$:

$$\mathbf{G}^{(q)}(\mathbf{m}; \mathbf{m}_i, \hat{\sigma}_i, \hat{p}_i) = \hat{p}_i \cdot \exp\left(-\frac{(\mathbf{m} - \mathbf{m}_i)^{\mathrm{T}}(\mathbf{m} - \mathbf{m}_i)}{2\hat{\sigma}_i^2}\right)\,, \tag{5.30}$$

where $\hat{\sigma}_i$ and $\hat{p}_i$, which are both normally distributed, denote the prominence and scale, respectively, and $\mathbf{m}_i$ represents the center position of a blob-like structure. In Fig. 5.7, we display a blob noise map modelled by Eq. (5.28). Although not perfect, the modelled noise can reflect the appearance of real high-ISO long-exposure noise.

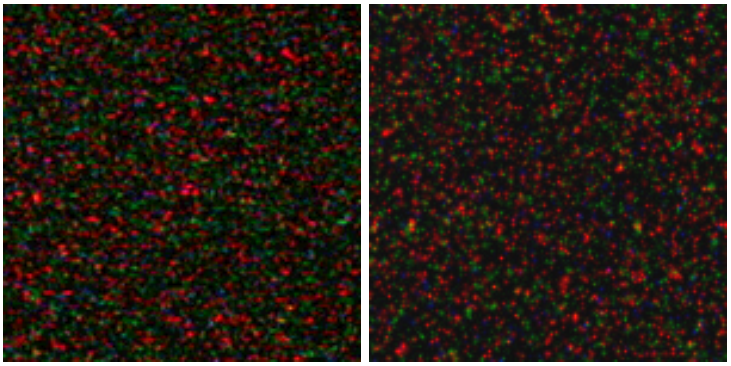


Figure 5.7: Illustration of the real high-ISO long-exposure noise (left) and the modelled noise (right). Please zoom electronically for a better view.

5.4.2. Denoising methods incorporating blob reduction

As elaborated in the previous section, we use the USG method to obtain a quantitative measurement of blob characteristics, including the position, the prominence and the scale. Subsequently, according to the blob model in Eq. (5.3), we are able to reconstruct the blobs using the quantitative measurements. Specifically, we obtain a binary blob center map $\mathbf{B}_{\mathrm{lm}}$ using the local maximizers of $\mathbf{B}_{\mathrm{u}}$. For a blob centered at the location $\mathbf{m}_i$, we can get its blob prominence $p_i = \mathbf{B}_{\mathrm{u}}(\mathbf{m}_i)$ and blob scale $\sigma_i = \mathbf{S}_{\mathrm{u}}(\mathbf{m}_i)$. Then, similar to Eq. (5.30), we reconstruct this blob as follows:

$$\mathbf{\Lambda}^{(i)}(\mathbf{m}) = p_i \cdot \exp\left(-\frac{(\mathbf{m}-\mathbf{m}_i)^{\mathrm{T}}(\mathbf{m}-\mathbf{m}_i)}{2\sigma_i^2}\right). \tag{5.31}$$

Subsequently, the aggregate of all the reconstructed $\mathbf{\Lambda}^{(i)}$ is computed as the blob reconstruction map. In this way, for a given channel of the high-ISO long-exposure image, we can also obtain a map of reconstructed blob noise $\hat{\mathbf{I}}_{\mathrm{b}}^{(q)}$. Then, according to Eqs. (5.28) and (5.30), for each channel of the noisy image, we use $\hat{\mathbf{I}}_{\mathrm{b}}^{(q)}$ as an approximation of $\xi_{\mathrm{b}}^{(q)}$. Therefore, using $\hat{\mathbf{I}}_{\mathrm{b}}^{(q)}$, we reduce the main part of the blob-like noise as follows:

$$\mathbf{I}_{\mathrm{br}}^{(q)} = \mathbf{I}_{\mathrm{noi}}^{(q)} - \xi_{\mathrm{b}}^{(q)} \approx \mathbf{I}_{\mathrm{noi}}^{(q)} - \hat{\mathbf{I}}_{\mathrm{b}}^{(q)}. \tag{5.32}$$

In a real-world noisy image case, Fig. 5.8(b) displays the red channel of the noisy image shown in Fig. 5.8(a). Figure 5.8(c) displays the blob reduction result based on Eq. (5.32). One can see that, although the result is not perfect, most of the blob-like noise has been reduced.

To further restore the image, we select several conventional denoising

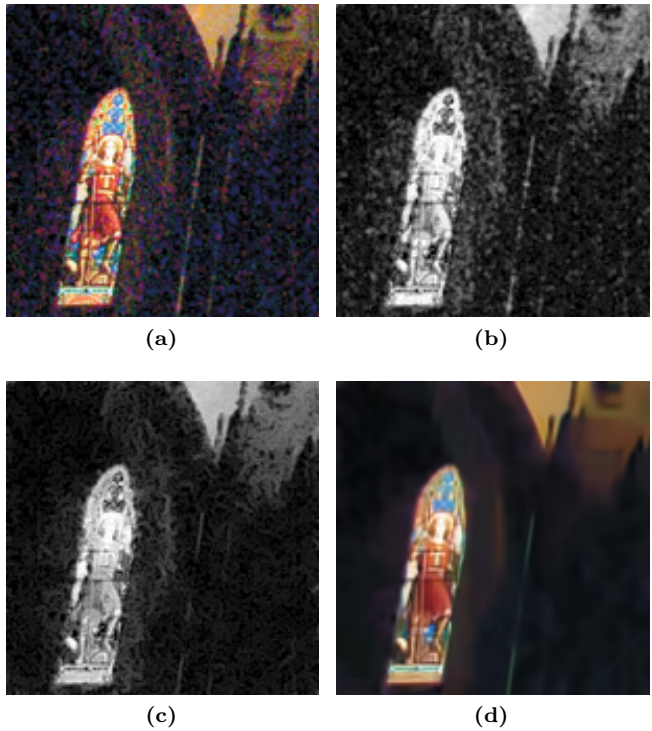


Figure 5.8: Illustration of the process of the proposed denoising scheme. (a) A noisy image; (b) The red channel of (a); (c) Result of a blob reduction on the red-channel image; (d) The denoising result of the proposed BR-CBM3D method.

methods to reduce both the residual noise and the errors caused by the blob reduction procedure, respectively. In this section, we adopt three widely used methods, i.e., the BF (Tomasi and Manduchi, 1998), NLM (Buades et al., 2005), NLMC (Goossens et al., 2008) and CBM3D (Dabov et al., 2007) methods, and two state-of-the-art methods, i.e., the MWNNM (Xu et al., 2017a) and TWSC (Xu et al., 2018b) methods. In this way, we get six denoising methods that incorporate a blob reduction (BR) procedure, and accordingly, we refer to the six proposed denoising methods as the BR-BF, BR-NLM, BR-NLMC, BR-CBM3D, BR-MWNNM and BR-TWSC methods. As an example, Fig. 5.8(d) shows a denoising result obtained by the BR-CBM3D method.

5.5. Experimental validation

We have presented the USG method to quantitatively measure the blob characteristics. Building on this method, we have also developed a denoising

Algorithm 3 The USG method for blob reconstruction and reduction

Require: Original image $\mathbf{I}$, scale set $\mathbb{S}$, direction set $\mathbb{D}$
Ensure: Blob reduction result $\mathbf{I}_{\mathrm{br}}$
for each $\sigma_i \in \mathbb{S}$ **do**
  for each $\theta_j \in \mathbb{D}$ **do**
    $\mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i,\theta_j) \leftarrow \kappa_{\mathrm{u}}(\mathbf{m};\sigma_i,\theta_j) * \mathbf{I}(\mathbf{m})$
  end for
  $\mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i) \leftarrow \min_{\theta_j \in \mathbb{D}} \mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i,\theta_j)$
end for
$\mathbf{B}_{\mathrm{u}}(\mathbf{m}) \leftarrow \max_{\sigma_i \in \mathbb{S}} \mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i)$
$\mathbf{S}_{\mathrm{u}}(\mathbf{m}) \leftarrow \operatorname{argmax}_{\sigma_i \in \mathbb{S}} \mathbf{B}_{\mathrm{u}}(\mathbf{m};\sigma_i)$
$\mathbf{B}_{\mathrm{lm}}(\mathbf{m}) \leftarrow$ local maximizers of $\mathbf{B}_{\mathrm{u}}(\mathbf{m})$
$\hat{\mathbf{I}}_{\mathrm{b}} \leftarrow$ Blob reconstruction based on $\mathbf{B}_{\mathrm{u}}(\mathbf{m})$, $\mathbf{B}_{\mathrm{lm}}(\mathbf{m})$ and $\mathbf{S}_{\mathrm{u}}(\mathbf{m})$
$\mathbf{I}_{\mathrm{br}} \leftarrow \mathbf{I} - \hat{\mathbf{I}}_{\mathrm{b}}$

scheme by employing blob reduction as a preprocessing step for six conventional denoising methods, thereby getting six denoising methods based on blob reduction. In this section, firstly, we test the USG method for blob reconstruction and blob reduction on an image containing synthetic blobs and white Gaussian noise. Secondly, in order to confirm whether or not the proposed denoising scheme can tackle the real noise occurring in high-ISO long-exposure images, we employ the developed denoising methods, i.e., the BR-BF, BR-NLM, BR-NLMC, BR-CBM3D, BR-MWNNM and BR-TWSC methods, to remove real blob-like noise.

5.5.1. Experiments on removing synthetic blobs and noise

To highlight the advantages of the USG method for blob characterization, on a synthetic image, we use the proposed USG method, which is summarized in Algorithm 3, to characterize, reconstruct and remove the blobs. Subsequently, we further remove the white Gaussian noise, using the BR-NLM method as a representative method.

The synthetic image, as shown in Fig. 5.9(a), contains blobs whose scales vary from 3 to 4, accompanied by some non-blob structures, like edges, bar-shaped lines, corners and terminations. The noisy synthetic image is obtained by adding Gaussian white noise with zero mean and a variance of 0.005. As for the parameter settings of the USG method, the multiple scales are taken from the scale set $\mathbb{S} = \left\{3, \frac{10}{3}, \frac{11}{3}, 4\right\}$, which covers the scales of the synthetic blobs, while the directions are selected from $\pi/8$ to π with a step of $\pi/8$. The blob reconstruction and blob reduction results are shown in Figs. 5.9(b) and 5.9(c).

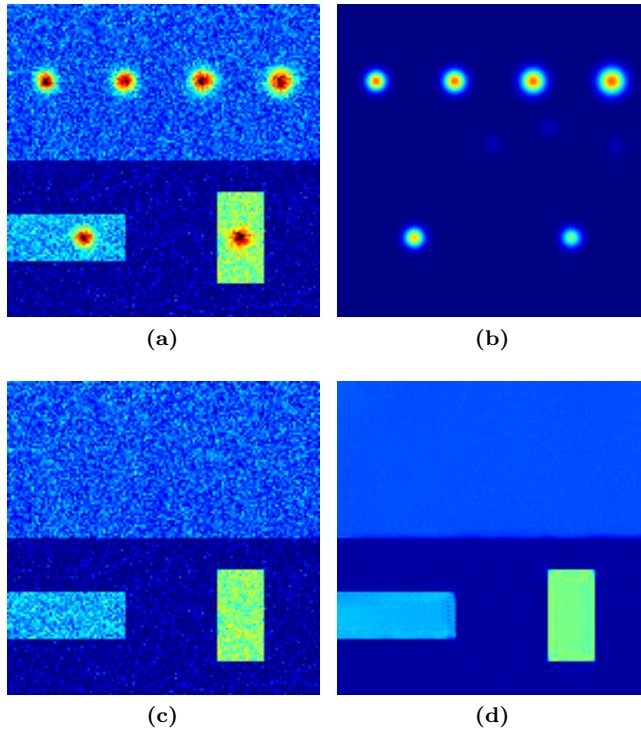


Figure 5.9: Results of blob reconstruction (b) and blob reduction (c) obtained by the USG method on a synthetic image (a) as well as the denoising result obtained by the BR-NLM method (d). For a better visualization, the images are displayed using the heat maps of intensity.

It can be seen that the blobs in the synthetic images have been reconstructed and removed successfully. Moreover, the denoising result of the BR-NLM method displayed in Fig. 5.9(d) demonstrates that the designed denoising scheme can remove both blobs and white Gaussian noise effectively.

For comparison, we use several widely adopted blob detection methods, including the TH, MLoG and gLoG methods, to characterize the blobs. The parameters in these methods are set according to the characteristics of the synthetic blobs and the original implementation. For reproducibility, the parameter settings of each method are listed as follows:

- TH: The morphological structuring element is set as a disk-shaped element. The scale of the structuring element is set to be 4.0 (i.e., the radius is 13 pixels).
- MLoG: The multiple scales are taken from 3 to 4 with a step of 1/3.

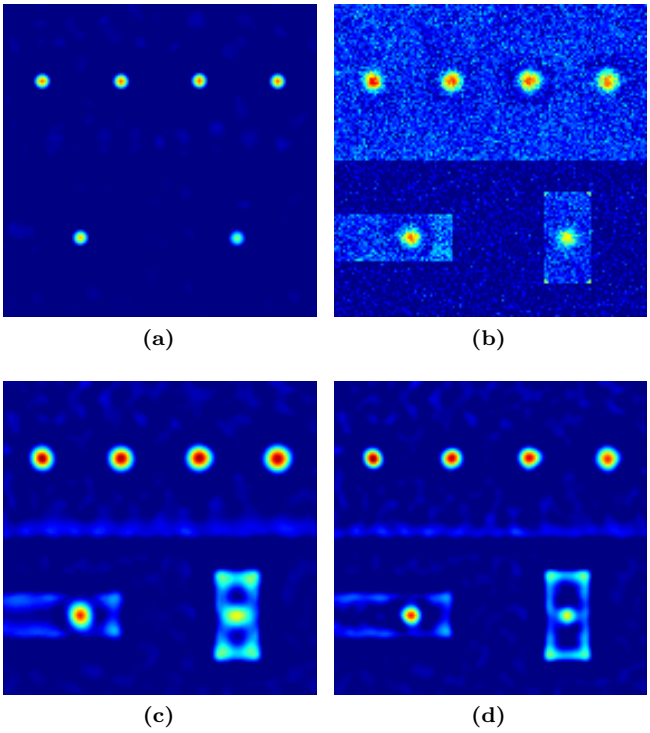


Figure 5.10: Responses obtained by the USG method (a), the TH method (b), the MLoG method (c) and the gLoG method (d). For a better visualization, the images are displayed using the heat maps of intensity.

- gLoG: The multiple scales are taken from 3 to 4 with a step of 1/3. The directions are selected from $\pi/9$ to π with a step of $\pi/9$.

The responses of the different methods are shown in Fig. 5.10. In contrast to the response of the USG method shown in Fig. 5.10(a), which has significant values only for blobs, the responses yielded by the competing methods also have significant values for non-blob structures. The TH method is sensitive to noise, while the MLoG and gLoG methods yield significant responses at locations of edges, corners, etc. Therefore, these competing methods can hardly be used for precise blob characterization.

5.5.2. Experiments on removing real noise

Experimental setup

In this section, we test the developed denoising methods, i.e., the BR-BF, BR-NLM, BR-NLMC, BR-CBM3D, BR-MWNNM and BR-TWSC methods, to remove real high-ISO long-exposure noise. To demonstrate the effectiveness of the blob reduction, we also compare the developed denoising methods with the conventional denoising methods.

In the selected conventional methods, most of the parameters are set according to the original papers. For the sake of fairness, some parameters are also optimally set by selecting the parameters that yield the best result. Specifically, to make the results reproducible, the parameter settings are listed as follows:

- BF: The degree of smoothing is optimally selected from 0.001 to 0.060 with a step of 0.001.
- NLM: The size of the similarity square neighborhood and the size of the search window are set as 7 and 21, respectively (Buades et al., 2005). The parameter determining the filtering strength is optimally selected from 0.050 to 0.080 with a step of 0.005.
- NLMC[1]: The size of the similarity square neighborhood and the size of the search window are set as 11 and 31, respectively (Goossens et al., 2008). The noise standard deviation is optimally selected from 1 to 50 with a step of 1.
- CBM3D[2]: The block size, the sliding step and the length of the search neighborhood are set as 8, 3 and 39, respectively (Dabov et al., 2007). The algorithm is carried out in A_{opp} color space (Ghimpeţeanu et al., 2016). The standard deviation, i.e., the assumed noise intensity, is optimally selected from 1 to 50 with a step of 1.
- MWNNM[3]: The size of local patches, the size of the search window, the number of non-local similar patches, the initial penalty parameter and the number of iterations are set as 6, 40, 70, 6 and 2, respectively (Xu et al., 2017a). The noise estimation parameter is optimally selected from 0.5 to 5.0 with a step of 0.5.
- TWSC[4]: The size of local patches, the size of the search window, the number of similar patches, the initial penalty parameter, the penalty

[1] https://quasar.ugent.be/bgoossen/download_nlmeans/
[2] http://www.cs.tut.fi/~foi/GCF-BM3D/
[3] https://github.com/csjunxu/MCWNNM-ICCV2017/
[4] https://github.com/csjunxu/TWSC-ECCV2018/

parameter update factor and the maximum number of iterations are set as 6, 60, 90, 0.5, 1.1 and 10, respectively (Xu et al., 2018b). The noise estimation parameter is optimally selected from 0.5 to 5.0 with a step of 0.5.

In the proposed blob reduction method, we adopt RGB color space and configure the direction set as $\mathbb{D} = \{\pi \cdot i/8 \mid i \in 1, 2, \ldots, 8\}$. The scale set is configured according to the size of the blob noise. For a blob noise structure having a radius r ($r \geq 3$), the corresponding kernel scale is supposed to be $(r - 1)/3$. In this experiment, we configure the scale set as $\mathbb{S} = \left\{\frac{2}{3}, 1, \frac{4}{3}, \frac{5}{3}\right\}$.

When conducting experiments on standard images, we also adopt two other methods for comparison, i.e. the AD method (Black et al., 1998) and the denoising method using deep convolutional neural networks (DnCNN) (Zhang et al., 2017a). Furthermore, in the experiments of real-world image denoising, we adopt a state-of-the-art low-light image enhancement method named LECARM (Ren et al., 2019). The parameters in the DnCNN[5] and LECARM[6] methods are set as the default values. When performing the AD methods, we set the number of diffusion iterations N_{itr} in two ways. The optimal anisotropic diffusion (oAD) method is configured with the N_{itr} that yields the best quantitative evaluation result, while the strong anisotropic diffusion (sAD) method is configured with $N_{\text{itr}} = 7$.

All the experiments are conducted in a Matlab (R2014b) environment and on a computer with Intel Core(TM) i7-3770 CPU 3.40GHz × 2 and RAM 16.00GB.

Performance on standard images

We first carry out the experiments on noisy versions of the selected standard images, which are obtained by adding real high-ISO long-exposure noise. The real noise is obtained by taking black photos using a digital single-lens reflex camera with high ISO sensitivity and long exposure time. As shown in Fig. 5.11, the selected standard images include the *Baboon*, *Barbara*, *Lena*, *Peppers*[7] and *Sailboat*[8] (a.k.a. *Sailboat on lake*), all of which have an identical resolution $512 \times 512 \times 3$. These images reflect a diversity of image content and are extensively used in the field of image processing. In this way, we have both the ground truth clean image and noisy images, and as such, we are able to obtain a quantitative evaluation.

[5] https://github.com/cszn/DnCNN
[6] https://github.com/baidut/LECARM
[7] https://homepages.cae.wisc.edu/~ece533/images
[8] http://sipi.usc.edu/database/database.php?volume=misc

Table 5.1: PSNR (dB) of denoising results.

Method	*Baboon*	*Barbara*	*Lena*	*Peppers*	*Sailboat*
oAD	26.79	28.52	28.89	27.01	28.12
sAD	23.61	26.64	28.40	26.98	26.80
DnCNN	26.39	27.96	28.47	26.87	27.89
BF	27.01	29.42	29.52	27.58	28.54
BR-BF	28.45	31.66	32.18	**31.07**	30.46
NLM	24.92	29.69	29.32	27.32	27.34
BR-NLM	25.51	31.52	31.15	30.00	28.20
NLMC	26.40	29.38	29.30	26.36	27.24
BR-NLMC	28.30	30.79	31.58	30.55	30.11
CBM3D	26.94	30.03	29.46	27.32	28.19
BR-CBM3D	**28.55**	**32.59**	32.14	30.94	30.42
MWNNM	27.12	30.06	29.30	27.56	28.42
BR-MWNNM	28.44	32.31	31.91	31.02	30.41
TWSC	27.05	30.07	29.59	27.47	28.31
BR-TWSC	**28.55**	32.40	**32.27**	31.03	**30.53**

In this experiment, we adopt the widely used Peak Signal-to-Noise Ratio (PSNR) as a quantitative evaluation metric, which is defined by:

$$\textsc{Psnr} = 10\log_{10}\left(\frac{|\mathbf{I}_{\mathrm{c}}|}{\sum\limits_{\mathbf{m}\in\mathbf{I}_{\mathrm{c}}}\left(\mathbf{I}_{\mathrm{c}}(\mathbf{m})-\mathbf{I}_{\mathrm{dn}}(\mathbf{m})\right)^2}\right), \tag{5.33}$$

where $\mathbf{I}_{\mathrm{c}}$ represents the true image, $\mathbf{I}_{\mathrm{dn}}$ the denoising result and $|\mathbf{I}_{\mathrm{c}}|$ the number of pixels in $\mathbf{I}_{\mathrm{c}}$. Note that both the $\mathbf{I}_{\mathrm{c}}$ and $\mathbf{I}_{\mathrm{dn}}$ have been transformed into the intensity range of $[0, 1]$. For image denoising, a higher PSNR is preferred.

The obtained quantitative evaluation results are reported in Table 5.1 and the visual denoising results are shown in Figs. 5.11, 5.12 and 5.13. It can be seen that all methods struggle with the complexity of the task. The oAD method, for example, cannot remove the noise effectively. Although the sAD method can attenuate the blob noise better than the oAD method, it expectedly suppresses image details, which leads to a significant decrease of the PSNR values. The DnCNN method also underperforms in removing blob noise. This is reasonable, because the generalization of deep convolutional neural networks

Table 5.2: Execution time (s) of the different methods on each image.

Method	*Baboon*	*Barbara*	*Lena*	*Peppers*	*Sailboat*
oAD	0.08	0.08	0.08	0.23	0.08
sAD	0.38	0.38	0.39	0.40	0.42
DnCNN	16.54	16.51	16.52	16.63	17.05
BF	0.05	0.05	0.05	0.05	0.06
BR-BF	2.13	1.88	1.98	1.88	1.82
NLM	140.54	140.83	141.28	141.18	140.80
BR-NLM	142.65	142.11	143.49	143.22	142.07
NLMC	61.56	60.74	61.72	60.80	60.98
BR-NLMC	63.33	62.64	63.56	62.53	62.96
CBM3D	4.75	4.64	5.21	5.22	5.15
BR-CBM3D	6.44	6.26	7.11	7.12	6.90
MWNNM	286.58	277.91	277.14	281.39	280.17
BR-MWNNM	288.59	279.17	278.98	283.28	281.94
TWSC	136.44	134.06	134.70	138.11	136.13
BR-TWSC	138.39	135.68	136.72	140.02	137.88

largely depends on the ability in memorizing training data, and the original DnCNN is essentially trained by images corrupted by additive white Gaussian noise (Guo et al., 2019). Compared with the AD and DnCNN methods, the BF, NLM, NLMC, CBM3D, MWNNM and TWSC methods perform slightly better in terms of PSNR and visual denoising results. However, these methods, including the TWSC method that has been designed for removing real-world noise, still have limitations in removing blob noise. In contrast, the methods incorporating blob reduction achieve a considerable performance in removing blob noise. This is confirmed on all the test images in terms of PSNR and visual denoising results, which demonstrates that the use of blob reduction can effectively benefit high-ISO long-exposure noise removal.

The execution time of each method is reported in Tab. 5.2. The oAD, sAD and BF methods are comparatively more efficient to execute, while the MWNNM and BR-MWNNM methods are the most time-consuming among all the methods tested. Note that compared with the original denoising methods, our methods incorporating blob reduction obtain better performances at the expense of acceptable additional computation (around 2 seconds in terms of runtime). Moreover, in our method, the convolution operations performed by the proposed USOG kernels are essentially linear filtering. The filtering

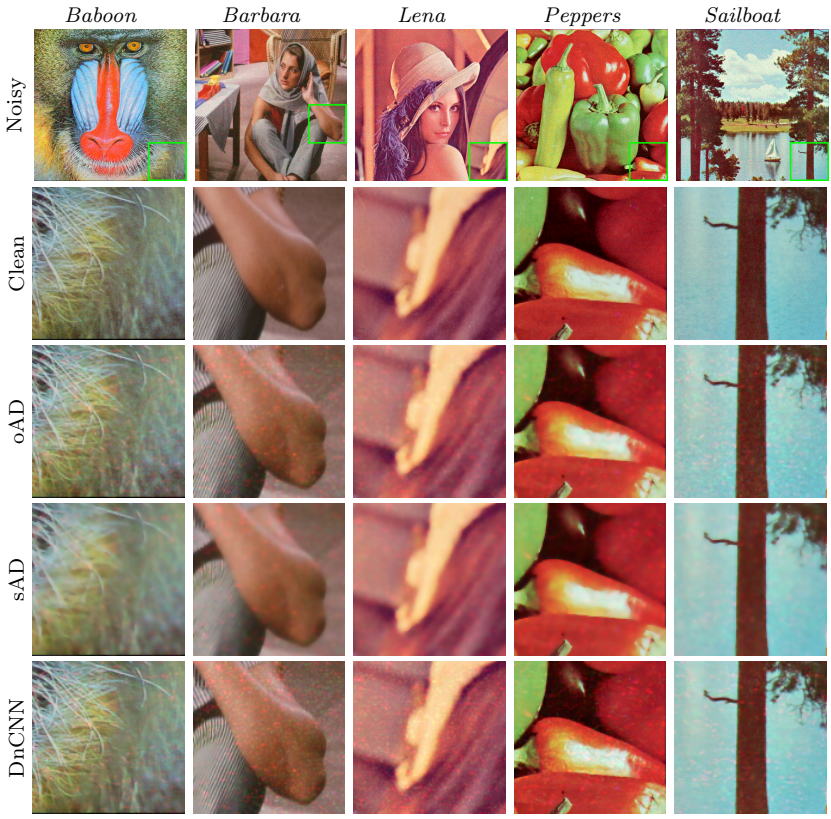


Figure 5.11: Noisy standard images corrupted by real noise along with the ground truth, and the denoising results obtained by the oAD, sAD, and DnCNN methods. Please zoom electronically for a better view.

procedures among different kernels are mutually independent. Thus, our method can be further accelerated by parallel computing.

Nonetheless, our methods still have limitations in denoising areas with plenty of textural structures. For instance, the denoising results on the image *Baboon* show a limited performance in terms of PSNR (less than 30dB). This is because the methods have to make a difficult compromise between removing more noise and preserving more structural details.

Performance on real-world images

We further apply the proposed denoising methods as well as the competing methods to real-world noisy images. As shown in Fig. 5.14, these noisy images are highly corrupted by high-ISO long-exposure noise that is difficult to remove

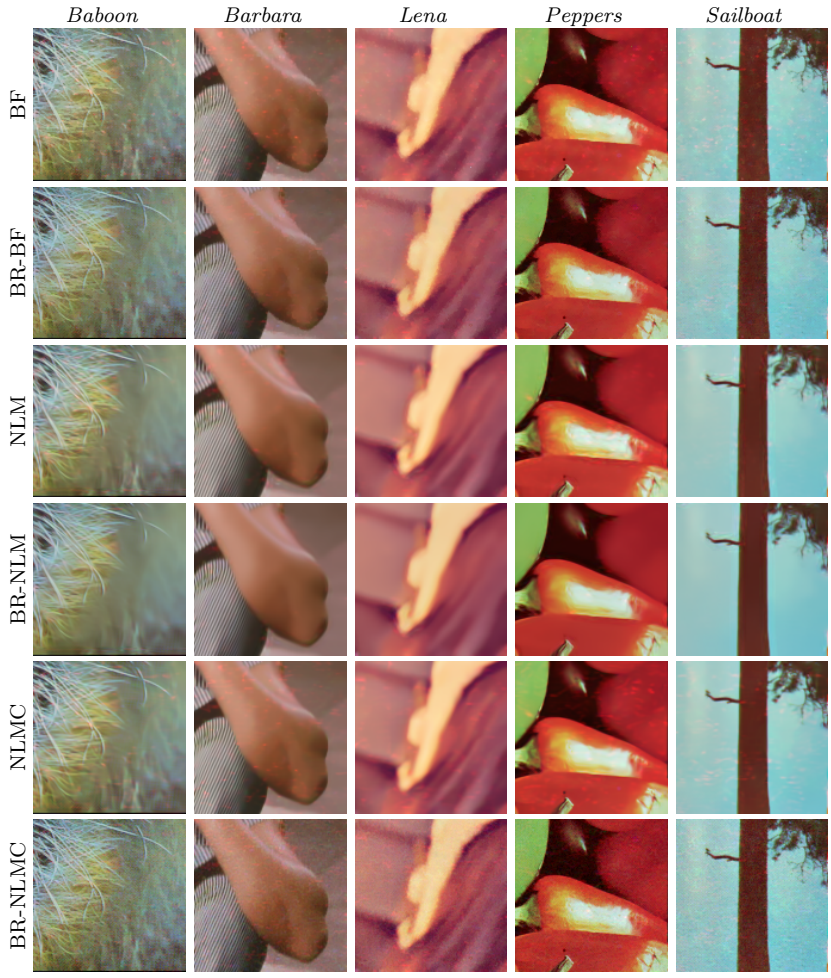


Figure 5.12: Denoising results on the noisy standard images obtained by the BF, NLM, NLMC methods and the proposed methods incorporating blob reduction. Please zoom electronically for a better view.

Figure 5.13: Denoising results on the noisy standard images obtained by the CBM3D, MWNNM, TWSC methods and the corresponding methods incorporating blob reduction. Please zoom electronically for a better view.

Figure 5.14: Five real-world noisy images (top row) (Courtesy: Peter K. Burian, Dave Johnson and Ziwei Liu (Liu et al., 2014)) and the processing results of the sAD, DnCNN and LECARM methods. Please zoom electronically for a better view.

by the sAD and DnCNN methods. Moreover, this type of noise can hardly be reduced by the low-light enhancement LECARM method.

The denoising results obtained by the BR-based methods as well as the original methods are displayed in Figs. 5.15 and 5.16. The original BF, NLM, NLMC, CBM3D, MWNNM and TWSC methods also underperform in removing blob noise. Comparatively, the methods incorporating blob reduction yield results with better visual quality. This is consistent with the experimental results obtained on the standard images. For example, when denoising the real-world noisy Image #5 shown in the top right of Fig. 5.14, the original BF, NLM, NLMC, CBM3D, MWNNM and TWSC methods fail in removing the heavy noise. In their denoising results, the blob noise is blurred and mingled, which damages the quality of the images. By contrast, the methods incorporating blob reduction remove more noise. This is mainly because the blob reduction procedure can significantly reduce the mass of the blob noise while retaining the image contents and structures, which highly benefits a subsequent denoising procedure. It is worth noting that our methods might underperform when FPN is extremely spatially correlated. For instance, when processing the real-world noisy Image #5, our methods fail to remove

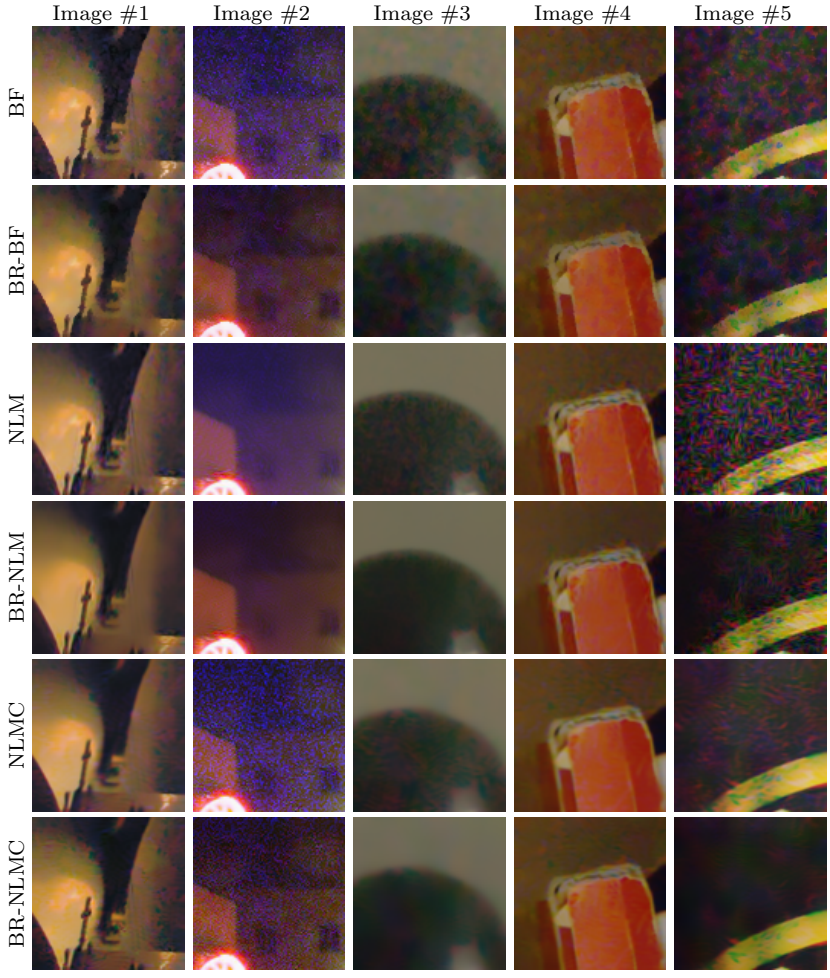


Figure 5.15: Denoising results on five real-world noisy images obtained by the BF, NLM, NLMC methods and the corresponding methods incorporating blob reduction. Please zoom electronically for a better view.

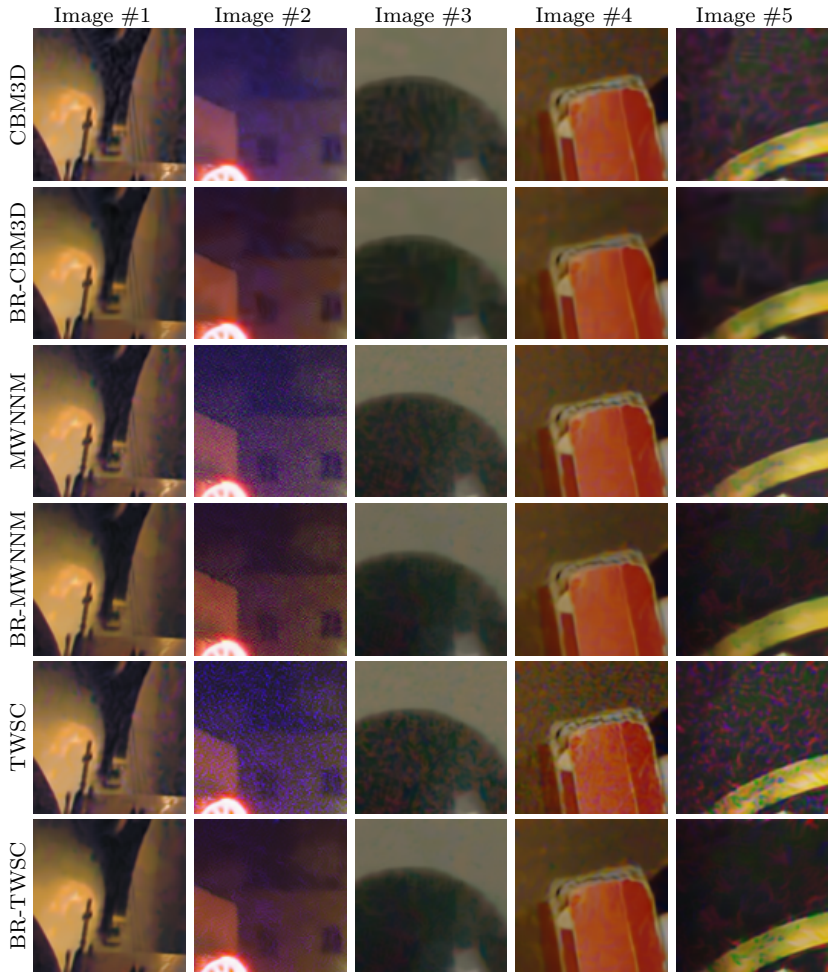


Figure 5.16: Denoising results on five real-world noisy images obtained by the CBM3D, MWNNM, TWSC methods and the corresponding methods incorporating blob reduction. Please zoom electronically for a better view.

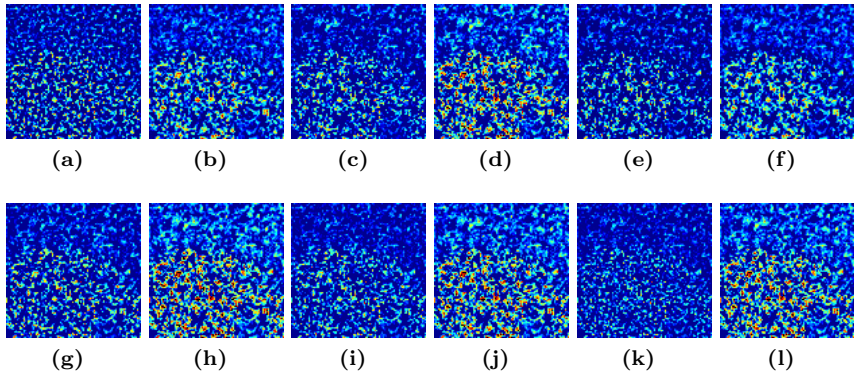


Figure 5.17: Illustration of the removed noise on the real-world noisy Image #5 (red channel) obtained by the (a) BF, (b) BR-BF, (c) NLM, (d) BR-NLM, (e) NLMC, (f) BR-NLMC, (g) CMB3D, (h) BR-CBM3D, (i) MWNNM, (j) BR-MWNNM, (k) TWSC and (l) BR-TWSC methods, respectively. For a better visualization, the images are displayed using the heat maps of intensity.

some noisy spots. This is because such heavy noise does not agree with the assumptions of our blob characterization method.

In Fig. 5.17, we also show the removed noise (a.k.a. method noise (Buades et al., 2005)) obtained on the red channels of the real-world noisy Image #3 displayed in Fig. 5.14. These maps of the removed noise also demonstrate that the blob reduction helps remove more blob noise.

Therefore, it can be inferred that the BF, NLM, NLMC, CBM3D, MWNNM and TWSC methods can benefit from the proposed blob reduction procedure in tackling high-ISO long-exposure noise. Although not providing a perfect solution, the proposed blob reduction method helps remove the high-ISO long-exposure noise in a cheap way.

5.6. Conclusions

In this chapter, we have presented a computational method to quantitatively measure the blob characteristics, using the proposed unilateral second-order Gaussian kernels. This method not only identifies the blob position, the prominence and the scale, but also suppresses non-blob structures well, and as such, this method can facilitate the implementation of the blob reconstruction and blob reduction. Moreover, to tackle the blob-like noise that occurs in high-ISO long-exposure images, we have developed a denoising scheme by employing a blob reduction procedure for each of the selected conventional

denoising methods. The experimental results have demonstrated that in high-ISO long-exposure image denoising, the methods incorporating blob reduction outperform the original conventional methods.

6 Iterative Laplacian-of-Gaussian filtering with application to blob detection

Detecting overlapping blob objects is a classical, yet challenging problem. In this chapter, we present a novel blob detection method based on iterative Laplacian-of-Gaussian filtering and unilateral second-order Gaussian kernels. The iteration of the Laplacian of Gaussian reduces the degree of overlap, facilitating a subsequent blob extraction procedure. The unilateral second-order Gaussian kernels yield responses only for blob objects, and the blob objects can therefore be pinpointed by a thresholding step. The experimental results demonstrate that the proposed method shows a promising performance in detecting fluorescence microscopy cells and electron micrograph nanoparticles.

The material of this chapter is based on the following publication:

- Wang, G., Lopez-Molina, C., and De Baets, B. (2020). Automated blob detection using iterative Laplacian of Gaussian filtering and unilateral second-order Gaussian kernels. *Digital Signal Processing*, 96:102592

6.1. Motivation

In image processing, blobs can be defined as small structures whose visual properties, e.g. brightness or color, are different from those in their surrounding region (Koenderink, 1984). Many objects in images show a blob-like appearance, and as such, blob detection has found applications in a wide variety of fields, such as cell counting (Zhang et al., 2015b), vanishing point detection (Kong et al., 2013b), bubble extraction (Zhang et al., 2012), quantum dot recognition (Xu et al., 2014a), and so on.

Over the past several decades, a variety of blob detection methods have been developed. A significant portion of these methods employ the Laplacian of Gaussian, or the approximating Difference of Gaussian, to enhance blob structures. The monoscale Laplacian of Gaussian kernel, which was proposed by Marr and Hildreth (1980), essentially measures the local contrast by subtracting the surrounding intensity from the central intensity at a given scale, thereby yielding positive and significant responses to blob structures. In the context of

the then-new scale-space theory, Lindeberg (1998b) proposed a method based on multiscale Laplacian of Gaussian filtering (LoG filtering), which has become a standard approach for blob detection. In this method, a bank of normalized Laplacian of Gaussian kernels (LoG kernels) covering all possible scales is used to convolve the image, yielding a maximum convolutional response at the *characteristic scale* for each blob. Lindeberg also used the determinant of the Hessian matrix (Lindeberg, 1998b) to detect blobs, embodying the idea that positions at which the determinant of the Hessian matrix is large and positive are likely to be blobs. Elaborating on Lindeberg's work and the theory of multiscale computer vision, several other authors presented more advanced proposals. Kong et al. (2013a) proposed a method based on generalized Laplacian of Gaussian kernels, which estimates the scale, shape and orientation of the observed blob by a bank of multiscale and anisotropic kernels. The method presented in (Moon et al., 2013) enhances blobs using the likelihood of blob-like structures as well as the magnitude of the eigenvalues at each pixel, which are computed from the eigenvalues of the Hessian matrix in scale-space. Zhang et al. (2015b) developed a method using both LoG filtering and the Hessian matrix to estimate the (overall) optimum scale. Also, following the works in (Moon et al., 2013) and (Frangi et al., 1998), they estimated the regional likelihood of blobness using the trace and determinant of the Hessian matrix, which is computationally cheaper than the method relying on eigenvalues (Moon et al., 2013). But the method in (Zhang et al., 2015b) works under the premise that the blobs are homogeneous in size and the degree of overlap is fairly low. Despite a lot of successful applications, the methods reviewed above have limitations in separating overlapping (a.k.a. adjacent or occluded) blobs, which occur ubiquitously in real imagery, e.g. cell or molecule images (Xing and Yang, 2016).

Driven by experimental and practical demands, many approaches have been presented to address overlapping blobs. Arslan et al. (2013), for example, proposed a method to separate overlapping blobs using contours, but this method underperforms when the overlapping blobs have similar grayscale intensities, or when the objects are rich in texture. The H-dome transformation (Smal et al., 2010), which suppresses the pixels whose relative height/depth is less than a given H-value in a grayscale image, is an effective tool to find regional maxima/minima in an image. Hence, it can be used for blob detection. To get rid of the interferences from noise or texture, the H-dome transformation is often paired with a blob enhancement filter, distance transformation, etc. But this method is sensitive to the choice of H-value. A small H-value might fail to separate overlapping blobs, while a large H-value might lead to false negative results. An alternative is the method based on wavelets (Püspöki et al., 2016), which can identify both the position and size of the blob to be

detected. This method has been implemented as an ImageJ plugin with a user-friendly interface. Besides, Descombes (2017) proposed a multiple blob detection method using a marked point process framework. A global energy function is built on the local contrast measure and the overlap degree. Subsequently, a so-called multiple births and deaths algorithm (Descombes et al., 2009) is employed to minimize the global energy function. It is reported that this method can deal with partially overlapping objects well. Nevertheless, the non-convexity of the function to be minimized usually leads to a high computational cost (Descombes, 2017; Ortner et al., 2008).

Instead of analyzing grayscale images, some methods separate the overlapping blobs in binarized images that are obtained by thresholding. For instance, Dewan et al. (2014) binarize a grayscale image using either Poisson-distribution-based minimum error thresholding (Fan, 1998) or Otsu thresholding (Otsu, 1979). Subsequently, in the binarized image, the result of the distance transformation is pixelwise computed as the distance between each zero pixel and the nearest nonzero pixel. On the distance transformation result, the H-dome approach is used to find the blob markers which facilitates a subsequent watershed segmentation procedure. Note that the performance of watershed segmentation highly depends on the detection of blob markers (Park et al., 2013), since the watershed algorithm usually suffers from oversegmentation (Xu et al., 2014a). There are some other methods that detect blobs in binarized images, such as the method based on ultimate erosion (Yang et al., 2006), the method based on ultimate erosion for convex sets (Park et al., 2013), the method based on bounded erosion and the fast radial symmetry transform (Zafari et al., 2015), etc. However, methods designed for binary blobs entail the assumption that the foreground blob objects are distinguishable enough to be segmented from the background (Park et al., 2013; Zafari et al., 2015).

Despite the vast literature, detecting overlapping blob objects still remains a challenging topic. To address this problem, we propose an automated blob detection method combining iterative Laplacian-of-Gaussian (iLoG) filtering and unilateral second-order Gaussian (USG) kernels (Wang et al., 2017b). Firstly, we present a multiscale normalization method for LoG kernels, thus proposing iLoG filtering to attenuate the overlapping regions of the adjacent blobs. We also explore the issue of scale setting in such an iterative process. Secondly, we investigate the potential of USG kernels, which topographically measure the minimum local contrast of a region among all directions, for separating overlapping blobs in the response of iLoG filtering. We also explain how to set the scale set appropriately. Subsequently, the blob detection result is obtained by applying a thresholding procedure to the response map yielded by USG kernels. This method can tackle both isolated blobs and partially

overlapping blobs. We have applied the proposed method to both fluorescence microscopy cell images and electron micrograph nanoparticle images to evaluate its performance on blob detection.

The remainder of this chapter is organized as follows. Section 6.2 revisits the LoG kernel. In Section 6.3, we reduce the degree of overlap by iLoG filtering, and subsequently extract the positions of blobs using USG kernels. The experimental validation is presented in Section 6.4, while the conclusions are listed in Section 6.5 concludes this chapter.

6.2. Related work

In this section, we revisit the definition and formulation of the LoG kernel, which has enjoyed a great popularity in blob detection. Conventionally, the multiscale LoG kernel is built on the normalized Gaussian kernel proposed by Lindeberg (1998b):

$$\hat{g}(\mathbf{x};\sigma) = (-1)^{\eta} \cdot \sigma^2 \cdot g(\mathbf{x};\sigma)\,, \tag{6.1}$$

where $g(\mathbf{x};\sigma)$ is the Gaussian kernel formulated in Eq. (3.2) and $\eta \in \{0,1\}$ allows the kernel to be applicable to both bright ($\eta = 1$) and dark ($\eta = 0$) structures.

Accordingly, the LoG kernel for detecting bright blobs (Lindeberg, 1998b) is given by:

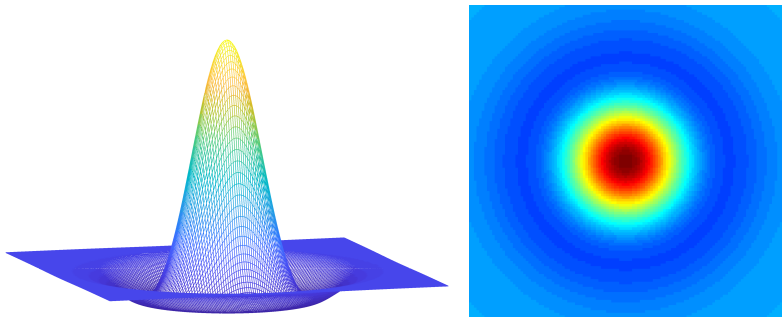
$$\begin{aligned}\nabla^2 \hat{g}(\mathbf{x};\sigma) &= -\sigma^2 \cdot \left(\frac{\partial^2 g(\mathbf{x};\sigma)}{\partial x^2} + \frac{\partial^2 g(\mathbf{x};\sigma)}{\partial y^2} \right) \\ &= -\frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^4} \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\sigma^2} \right). \end{aligned} \tag{6.2}$$

In (Lindeberg, 1998b), Lindeberg theoretically proved that, for the modelled blobs, including symmetric/non-symmetric Gaussian blobs and sine blobs, the response of LoG filtering reaches the maximum at the characteristic scale, and as such, the response at the characteristic scale is usually selected as the measure of the blob strength (a.k.a. blob saliency). For a given 2D signal $\mathbf{I}(\mathbf{x})$, the final response of LoG filtering is obtained by:

$$\hat{\mathbf{J}}(\mathbf{x}) = \max_{\sigma \in \mathbb{S}} \; \nabla^2 \hat{g}(\mathbf{x};\sigma) * \mathbf{I}(\mathbf{x})\,, \tag{6.3}$$

where $\mathbb{S}$ denotes a scale set that contains the characteristic scale.

As illustrated in Fig. 6.1, the LoG kernel is isotropic, i.e., rotationally



(a) (b)

Figure 6.1: Illustration of a conventional Laplacian of Gaussian kernel. (a) Three-dimensional visualization; (b) Planar visualization.

symmetric. Despite its popularity, the conventional LoG kernel still has limitations. It essentially measures the average local contrast between the central region and the surrounding, and as such, the LoG kernel yields significant responses at locations where the grayscale intensity changes sharply, including edges, ridges, corners and blobs.

6.3. Method for detecting overlapping blobs

In this section, an automated blob detection method is proposed to tackle both isolated and overlapping blobs. In this method, iterative LoG filtering is used to enhance the isolated objects while reducing the degree of overlap of the overlapping blobs. Subsequently, USG kernels are employed to suppress non-blob structures. By incorporating a thresholding procedure, a full-fledged blob detection method is developed. For a better visualization, the proposed method is elaborated on an example image (Lehmussola et al., 2007) that contains both isolated and overlapping blobs.

6.3.1. Reducing the degree of overlap by iterative Laplacian of Gaussian filtering

As mentioned earlier, the LoG kernel yields significant responses for regions where the image intensity changes sharply, especially for blobs, and has therefore been used for blob detection. In literature, existing LoG-based methods use LoG filtering for blob enhancement, but they can hardly separate overlapping blob objects.

In essence, the LoG kernel measures the average local contrast over all

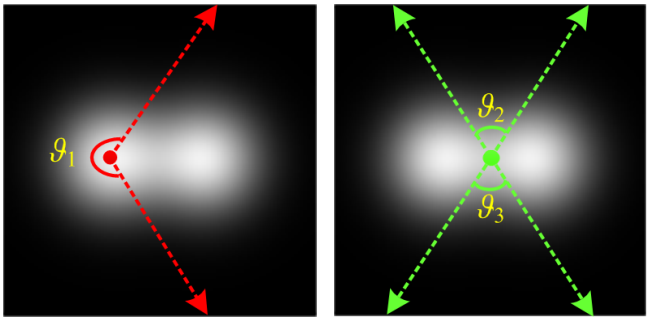


Figure 6.2: Comparison of the protrusion region and overlapping region in terms of average local contrast. (a) The protrusion region has positive and significant local contrast over the orientation range ϑ_1; (b) The overlapping region has positive and significant local contrast over the orientation range $\vartheta_2 + \vartheta_3$.

directions. For partially overlapping objects, there exists both a protrusion region and an overlapping region for each blob. Compared to the overlapping region, the protrusion region has a larger positive local contrast between the interior and the surrounding. For example, as shown in Fig. 6.2, the protrusion region has a positive and significant local contrast over the orientation range ϑ_1, while the overlapping region has a positive and significant local contrast over the orientation range $\vartheta_2 + \vartheta_3$, and obviously, we usually have $\vartheta_1 > \vartheta_2 + \vartheta_3$. Therefore, the LoG kernel tends to yield larger responses for protrusion regions than for overlapping regions, and this difference can be accumulated by an iterative procedure. Capitalizing on this fact, iterative LoG filtering is applied to the image to retain the isolated blobs while attenuating the overlapping regions of overlapping blobs.

The use of iLoG filtering brings up the problem of scale determination. Before addressing this problem, we introduce a blob model, and, subsequently, rescale Lindeberg's LoG kernel.

Firstly, like many conventional methods, we model a blob by a 2D Gaussian function, which is formulated by:

$$\mathbf{\Lambda}_0(\mathbf{x}) = p_0 \exp\left(-\frac{(\mathbf{x}-\mathbf{x}_0)^{\mathrm{T}}(\mathbf{x}-\mathbf{x}_0)}{2\omega_0^2}\right) + b_0\,, \tag{6.4}$$

where $\mathbf{x}_0$ represents the center position, $p_0 \in]0,\,1]$ the prominence, ω_0 the blob scale and $b_0 \in [0,\,1[$ the base level (background intensity). A modelled Gaussian blob is shown in Fig. 6.3(a). For simplicity, we set $\mathbf{x}_0 = [0, 0]^{\mathrm{T}}$, and

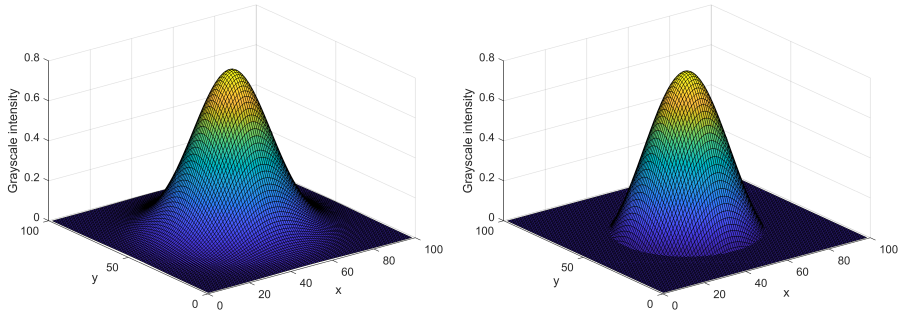


Figure 6.3: A modelled Gaussian blob ($b_0 = 0$, $p_0 = 0.8$ and $\omega_0 = 15$) as well as its corresponding response yielded by an LoG kernel at the scale $\sigma = 15$.

accordingly, the blob model formulated by Eq. (6.4) becomes

$$\mathbf{\Lambda}_0(\mathbf{x}) = p_0 \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\omega_0^2}\right) + b_0\,. \tag{6.5}$$

The spatial extension of a Gaussian blob, which can be represented by the radius R_0, is usually determined by the scale, and can therefore be approximated by

$$R_0 \approx 3\omega_0\,. \tag{6.6}$$

This is rooted in the fact that 99% of the mass of a Gaussian is concentrated within three standard deviations of its mean (Kong et al., 2013a).

Secondly, we rescale Lindeberg's LoG kernel in Eq. (6.2), thereby obtaining the kernel that would be used in iLoG filtering. The rescaled kernel formulated as follows:

$$\begin{aligned}\nabla^2 g_{\mathrm{iL}}(\mathbf{x};\sigma) &= 2\cdot\nabla^2\hat{g}(\mathbf{x};\sigma)\\ &= -\frac{x^2+y^2-2\sigma^2}{\pi\sigma^4}\exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\sigma^2}\right).\end{aligned} \tag{6.7}$$

The rescaling will make the response of the kernel precisely reflect the prominence of the blob to be detected, as we will explain in this work.

Having both the blob model and the rescaled LoG kernel, we obtain the response of the initial LoG filtering by convolving $\mathbf{\Lambda}_0$ with $\nabla^2 g_{\mathrm{iL}}(\mathbf{x};\sigma)$, which

is given by:

$$\begin{aligned}\mathbf{J}_0(\mathbf{x};\sigma) &= \nabla^2 g_{\mathrm{iL}}(\mathbf{x};\sigma) * \mathbf{\Lambda}_0(\mathbf{x}) \\ &= \frac{\partial^2}{\partial x^2}\left(g_{\mathrm{iL}}(\mathbf{x};\sigma) * \mathbf{\Lambda}_0(\mathbf{x})\right) + \frac{\partial^2}{\partial y^2}\left(g_{\mathrm{iL}}(\mathbf{x};\sigma) * \mathbf{\Lambda}_0(\mathbf{x})\right) \\ &= -\frac{2p_0\omega_0^2\sigma^2}{\omega_0^2+\sigma^2}\left(\frac{\partial^2}{\partial x^2}\exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2(\omega_0^2+\sigma^2)}\right) + \frac{\partial^2}{\partial y^2}\exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2(\omega_0^2+\sigma^2)}\right)\right) \\ &= -2p_0\omega_0^2\sigma^2 \cdot \frac{x^2+y^2-2(\omega_0^2+\sigma^2)}{(\omega_0^2+\sigma^2)^3}\exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2(\omega_0^2+\sigma^2)}\right). \qquad (6.8)\end{aligned}$$

By setting $\mathbf{x} = \mathbf{0}$, the response at the blob center is easily computed as follows:

$$\mathbf{J}_0(\mathbf{0};\sigma) = \frac{4p_0\omega_0^2\sigma^2}{(\omega_0^2+\sigma^2)^2}\,. \qquad (6.9)$$

To find the characteristic scale of the blob, we intend to find the σ at which $\mathbf{J}_0(\mathbf{0};\sigma)$ yields a maximum response in scale-space (Wang and De Baets, 2019b). For $\sigma \in \mathbb{R}_+$, it is easy to verify that the second-order derivative of $\mathbf{J}_0(\mathbf{0};\sigma)$ with respect to σ is negative. Thus, we determine the maximum value of $\mathbf{J}_0(\mathbf{0};\sigma)$ in scale-space by computing the first-order derivative of $\mathbf{J}_0(\mathbf{0};\sigma)$ with respect to σ and then setting it to zero. Specifically, the first-order derivative of $\mathbf{J}_0(\mathbf{0};\sigma)$ with respect to σ is given by:

$$\frac{\partial \mathbf{J}_0(\mathbf{0};\sigma)}{\partial\sigma} = 8p_0\omega_0^2\sigma\frac{\omega_0^4-\sigma^4}{(\omega_0^2+\sigma^2)^4}\,. \qquad (6.10)$$

By setting this partial derivative to 0, we obtain $\sigma = \omega_0$. This means that, for a given Gaussian blob with scale ω_0, LoG kernels yield the maximum response in scale-space at the scale

$$\sigma^* = \omega_0\,. \qquad (6.11)$$

In addition, having $\sigma^* = \omega_0$, we compute the maximum response of $\mathbf{J}_0(\mathbf{0};\sigma)$ in scale-space as follows:

$$\mathbf{J}_0(\mathbf{0};\sigma^*) = p_0\,. \qquad (6.12)$$

That is, convolving a Gaussian blob with the rescaled LoG kernel formulated in Eq. (6.7), we obtain the maximum response p_0, which equals the prominence of the original blob, at the scale $\sigma^* = \omega_0$. This is the essential reason why we rescale Lindeberg's LoG kernel in Eq. (6.7).

Consequently, according to Eq. (6.8), the resulting response of the LoG kernel at the scale $\sigma^* = \omega_0$ is:

$$\mathbf{J}_0(\mathbf{x}; \sigma^*) = -\frac{p_0(x^2 + y^2 - 4\omega_0^2)}{4\omega_0^2} \cdot \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{4\omega_0^2}\right). \tag{6.13}$$

For detecting bright blobs, only positive values in $\mathbf{J}_0(\mathbf{x}; \sigma^*)$ are kept as the blob strength map. Thus, the resulting response $\mathbf{\Lambda}_1$ is obtained as follows:

$$\mathbf{\Lambda}_1(\mathbf{x}) = \max\left(\mathbf{J}_0(\mathbf{x}; \sigma^*),\, 0\right), \tag{6.14}$$

which is visualized in Fig. 6.3(b). It can be seen that $\mathbf{\Lambda}_1$ also tends to appear as a Gaussian blob. Since $\mathbf{\Lambda}_1$ is rotationally symmetric, its radius R_1 can be computed from the zero points of $\mathbf{J}_0(\mathbf{x}; \sigma^*)$ on the y-axis. Setting both $\mathbf{J}_0(\mathbf{x}; \sigma^*) = 0$ and $y = 0$, we get the radius of $\mathbf{\Lambda}_1$ as:

$$R_1 = 2\omega_0\,. \tag{6.15}$$

Hence, if we approximate $\mathbf{\Lambda}_1$ as a Gaussian blob, according to Eq. (6.6), the scale is approximated as:

$$\omega_1 \approx \frac{R_1}{3} = \frac{2\omega_0}{3}\,. \tag{6.16}$$

This means that there is a scale decay (decrease) for the Gaussian blob in the initial LoG filtering process. The comparison of the spatial extension between $\mathbf{\Lambda}_0$ and $\mathbf{\Lambda}_1$ is also demonstrated in Fig. 6.4, in which we can see that the profile radius of $\mathbf{\Lambda}_1$ is narrower than that of $\mathbf{\Lambda}_0$.

According to Eqs. (6.11) and (6.16), when we filter $\mathbf{\Lambda}_1$ using the rescaled LoG kernels, the characteristic scale is supposed to be

$$\sigma^{**} \approx \omega_1 = \frac{2\sigma^*}{3}\,. \tag{6.17}$$

More specifically, if the characteristic scale σ^* is used in the first round of LoG filtering, then, in the second LoG filtering, the scale should be set to $\sigma^{**} = 2/3\sigma^*$.

In real-life imagery, blobs are usually heterogeneous in size. In order to cope with this problem, we use a scale set $\mathbb{S}^{(1)}$ that covers the values in the vicinity of the characteristic scale σ^* for the first LoG filtering. Then, according to the discussion above, in the following LoG filtering, the proper scales should

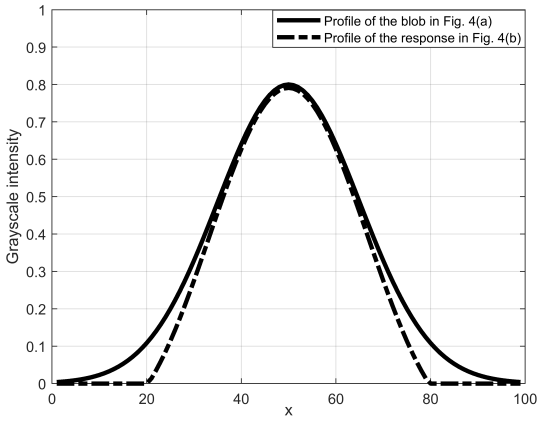


Figure 6.4: The horizontal intensity profiles through the center of Figs. 6.3(a) and 6.3(b).

be set as

$$\mathbb{S}^{(k)} = \varsigma^{k-1} \cdot \mathbb{S}^{(1)} \ , \tag{6.18}$$

where $k \in \{1, 2, \ldots, K\}$ denotes the k-th iteration, K is the total number of iterations and ς represents the scale decay factor. According to Eqs. (6.16) and (6.17), ς is supposed to be $2/3$.

We illustrate a practical example in Fig. 6.5, which contains fourteen blob objects with a high degree of overlap. We apply iLoG filtering on the image, obtaining responses in which the degree of overlap is reducing as the number of iterations increases. After two or three iterations of LoG filtering, the blob objects are fairly separable from each other.

It is worth noting that too many rounds of LoG filtering may lead to the extinction of some blobs, since the spatial extension of the resulting response decreases as the number of iterations increases. Fortunately, in most practical cases, two or three iterations are able to reduce the degree of overlap significantly.

However, in the final response of iLoG filtering, some blobs may be still topo-graphically connected to each other with saddle regions. This problem will be addressed in the next step.

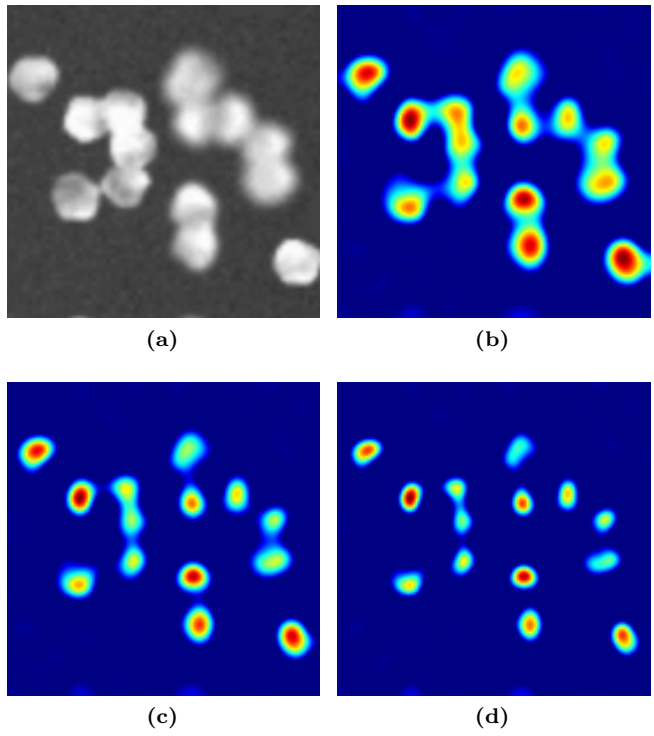
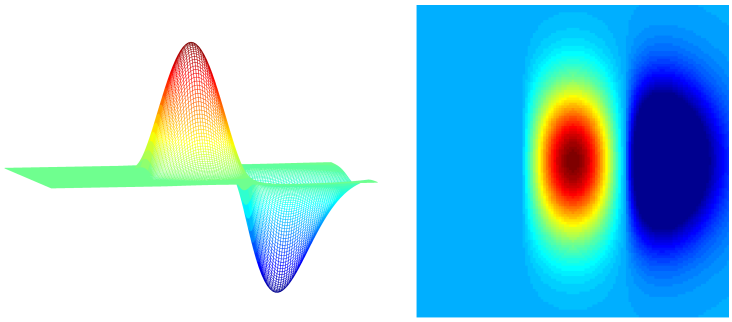


Figure 6.5: Illustration of the responses of iLoG filtering. (a)-(d): The original image, the response of the first LoG filtering, the response of the second LoG filtering and the response of the third LoG filtering.

6.3.2. Non-blob structure suppression using unilateral second-order Gaussian kernels

We have used iLoG filtering to attenuate the overlapping regions of partially overlapping blobs. Nevertheless, some blobs in the response map obtained by iLoG filtering may still remain connected, which hinders a threshold-based segmentation of blobs. Conventionally, the H-dome transformation (Smal et al., 2010) is used to suppress the saddle regions. But the choice of H-value is arduous, and the H-dome transformation is not good at suppressing noise, as mentioned in Section 6.1. Instead, we use USG kernels (Wang et al., 2017b), which yield responses only for blobs, to extract the blobs while suppressing non-blob structures, including the saddle regions.

The USG kernel has already been applied to blob noise removal in Chapter 5. Here, we further exploit its potential for separating partially overlapping blobs. For ease of presentation, based on the prior work, we restate the USG kernel



(a) (b)

Figure 6.6: Illustration of a USG kernel. (a) Three-dimensional visualization; (b) Planar visualization.

for a blob detection task. In particular, we explain how to appropriately set the scales for the USG kernels used for convolving the response of iLoG filtering.

The expression of the USG kernel has been given by Eq. (5.16). Figure 6.6 displays a single USG kernel. Essentially, a single USG kernel measures the local contrast along a given orientation at a specific scale. For each USG kernel, its response to an image $\mathbf{I}(\mathbf{x})$ is obtained by:

$$\mathbf{U}(\mathbf{x};\sigma,\theta) = \kappa_{\mathrm{u}}(\mathbf{x};\sigma,\theta) * \mathbf{I}(\mathbf{x})\,, \tag{6.19}$$

where $\theta \in [0, 2\pi[$ denotes the direction taken from a direction set $\mathbb{D}_{\mathrm{u}}$.

We next turn to the problem of scale selection for USG kernels. Similar to Eq. (6.5), we model a Gaussian blob $\mathbf{\Lambda}_K$ by:

$$\mathbf{\Lambda}_K(\mathbf{x}) = p_K \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\omega_K^2}\right) + b_K\,, \tag{6.20}$$

where $p_K \in \,]0,\,1]$ denotes the prominence, ω_K is the blob scale and $b_K \in [0,\,1[$ stands for the base level. Convolving $\mathbf{\Lambda}_K$ with a USG kernel along the direction $\theta = 0$, we have the response at the center of the blob:

$$\begin{aligned}\mathbf{U}_K(\mathbf{0};\sigma) &= \kappa_{\mathrm{u}}(\mathbf{x};\sigma,\theta) * \mathbf{\Lambda}_K(\mathbf{x}) \,\big|_{\theta=0,\mathbf{x}=\mathbf{0}} \\ &= \frac{4p_K\omega_K^2\sigma^2}{(\omega_K^2+\sigma^2)^2}\,.\end{aligned} \tag{6.21}$$

Computing its derivative with respect to σ, we have:

$$\frac{\partial \mathbf{U}_K(\mathbf{0};\sigma)}{\partial \sigma} = 8p_K\omega_K^2\sigma\frac{\omega_K^4-\sigma^4}{(\omega_K^2+\sigma^2)^4}\,. \tag{6.22}$$

By setting this partial derivative to 0, we obtain $\sigma = \omega_K$, which means that, for a given Gaussian blob with scale ω_K, USG kernels yield the maximum response in scale-space at the scale

$$\sigma^\dagger = \omega_K\,, \tag{6.23}$$

and, according to Eq. (6.21), the maximum response is:

$$\begin{aligned}\mathbf{U}_K(\mathbf{0};\sigma^\dagger) &= \frac{4p_K\omega_K^2\sigma^2}{(\omega_K^2+\sigma^2)^2}\,\Bigg|_{\sigma=\omega_K}\\ &= p_K\,.\end{aligned} \tag{6.24}$$

We learn that if we convolve USG kernels with a Gaussian blob having scale ω_K and prominence p_K, the maximum response p_K is obtained at the scale $\sigma^\dagger = \omega_K$.

As discussed earlier, for a Gaussian blob with scale ω_0, after K rounds of LoG filtering, the scale has been approximately reduced to $\omega_K = \varsigma^K \cdot \omega_0$. Therefore, having the scale set $\mathbb{S}^{(1)}$ that is used in the first LoG filtering, we should set the scale set of the USG kernels to be:

$$\mathbb{S}_{\mathrm{u}} = \varsigma^K \cdot \mathbb{S}^{(1)}\,. \tag{6.25}$$

Having the USG kernels as well as the appropriate scale set, we are able to use the USG kernels to suppress non-blob structures. Topographically speaking, a salient blob has a positive local contrast in all directions, which makes blobs distinguishable from all the other image structures. In other words, compared to non-blob structures, the minimum local contrast of a blob among all directions is still positive and significant. This is also the premise of the method based on the determinant of Hessian, the H-Dome method (Vincent, 1993), the Top-Hat method (Breen et al., 1991), etc. To capitalize on this fact, at a given scale, USG kernels are used to measure the local contrast of a region in all directions and subsequently select the minimum value as the response. Consequently, the non-blob structures can be suppressed. Formalizing this proposal, we obtain the final response of the USG kernels by:

$$\mathbf{U}(\mathbf{x}) = \max_{\sigma\in\mathbb{S}_{\mathrm{u}}}\ \min_{\theta\in\mathbb{D}_{\mathrm{u}}}\ \mathbf{U}(\mathbf{x};\sigma,\theta)\,. \tag{6.26}$$

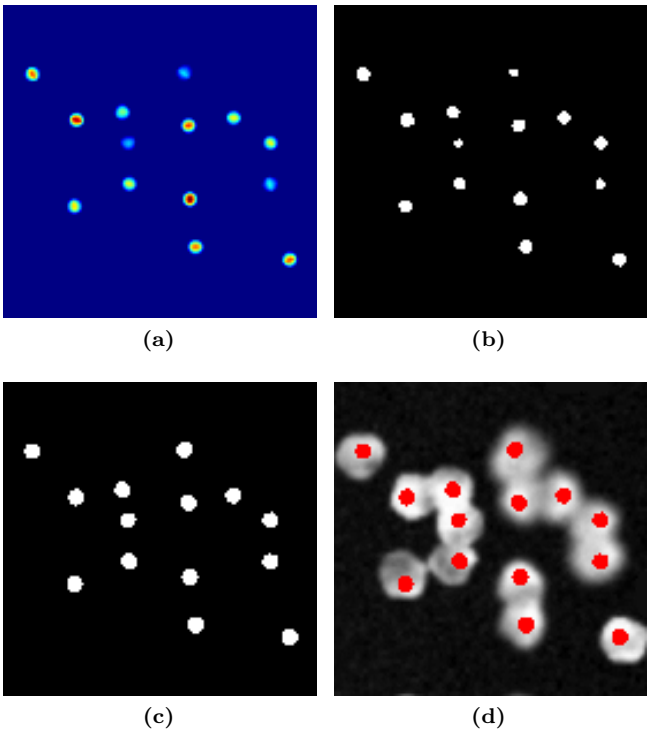


Figure 6.7: Illustration of the responses of USG kernels as well as the blob detection results. (a) The response of the USG kernels obtained on the image shown in Fig. 6.5(d); (b) The Otsu thresholding result of (a); (c) The Rosin thresholding result of (a); (d) The extracted blob markers (in red) superimposed on the original image.

Figure 6.7(a) shows the response of the USG kernels obtained on the response map shown in Fig. 6.5(d). As can be seen, only the central blob regions show significant responses, while all the non-blob structures have been suppressed.

Subsequently, we can easily obtain the blob detection result $\mathbf{I}_{\text{bw}}$ by applying a thresholding technique to the response of USG kernels. We illustrate the binarization results obtained by Otsu thresholding (Otsu, 1979) and Rosin thresholding (Rosin, 2001) in Figs. 6.7(b) and 6.7(c). Both thresholding techniques are able to segment the blobs from the background. As can be see in Fig. 6.7(d), all of the fourteen blobs have been pinpointed correctly, regardless of the fairly high degree of overlap. Furthermore, for a finer blob locating, the centroids of each 8-connected component in $\mathbf{I}_{\text{bw}}$ are selected to form the map of blob centers $\mathbf{I}_{\text{bc}}$.

Algorithm 4 The proposed method for blob detection

Require: Image $\mathbf{I}$, the initial scale set $\mathbb{S}^{(1)}$, the direction set $\mathbb{D}_{\mathrm{u}}$, the number of iterations K, the scale decay factor ς

Ensure: The map of blob centers $\mathbf{I}_{\mathrm{bc}}$

while $k \leq K$ **do**
  $\mathbb{S}^{(k)} = \varsigma^{k-1} \cdot \mathbb{S}^{(1)}$
  for each $\sigma \in \mathbb{S}^{(k)}$ **do**
    $\mathbf{J}(\mathbf{m};\sigma) \leftarrow \nabla^2 g_{\mathrm{iL}}(\mathbf{m};\sigma) * \mathbf{I}(\mathbf{m})$
  end for
  $\mathbf{J}(\mathbf{m}) \leftarrow \max_{\sigma \in \mathbb{S}^{(k)}} \mathbf{J}(\mathbf{m};\sigma)$
  $\mathbf{I}(\mathbf{m}) \leftarrow \max\big(\mathbf{J}(\mathbf{m}),\ 0\big)$
end while
$\mathbb{S}_{\mathrm{u}} = \varsigma^{K} \cdot \mathbb{S}^{(1)}$
for each $\sigma \in \mathbb{S}_{\mathrm{u}}$ **do**
  for each $\theta \in \mathbb{D}_{\mathrm{u}}$ **do**
    $\mathbf{U}(\mathbf{m};\sigma,\theta) \leftarrow \kappa_{\mathrm{u}}(\mathbf{m};\sigma,\theta) * \mathbf{I}(\mathbf{m})$
  end for
end for
$\mathbf{U}(\mathbf{m}) \leftarrow \max_{\sigma \in \mathbb{S}_{\mathrm{u}}} \min_{\theta \in \mathbb{D}_{\mathrm{u}}} \mathbf{U}(\mathbf{m};\sigma,\theta)$
$\mathbf{I}_{\mathrm{bw}} \leftarrow$ Rosin thresholding on $\mathbf{U}(\mathbf{m})$;

Ensure: $\mathbf{I}_{\mathrm{bc}} \leftarrow$ centroid of each 8-connected component in $\mathbf{I}_{\mathrm{bw}}$.

In digital image processing, we employ the discrete versions of all the kernels mentioned above. Each discrete kernel is obtained by sampling the continuous kernel in 2D integer coordinates, in which $\mathbf{m} = [m_x, m_y]^{\mathrm{T}}$ denotes the image coordinates.

6.4. Experimental validation

Having presented the proposed blob detection method based on iLoG filtering and the use of USG kernels (iLoG-USG method), we are now in a position to evaluate its practical performance. In this section, we apply our method, which is summarized in Algorithm 4, to both fluorescence microscopy images (Lehmussola et al., 2007) as well as electron micrography nanoparticle images (Park et al., 2013). Moreover, we compare our method with competing methods to make the validation more convincing.

6.4.1. Evaluation on an example image

We first test our method on an example image taken from (Descombes, 2017), which is shown in Fig. 6.8(a). This image contains a collection of blobs on

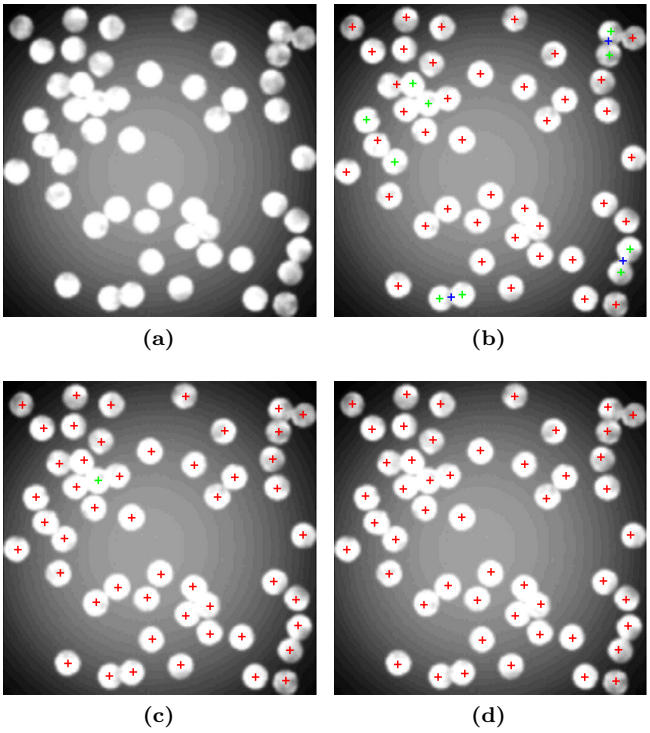


Figure 6.8: Example image containing a collection of blobs on a heterogeneous background (a) and the blob detection results obtained by the MLoG method (b), the MPP method (c) and the iLoG-USG method (d). The red, green and blue crosses denote the correctly detected, falsely rejected and falsely detected results, respectively.

a heterogeneous background. We compare the proposed method with the MLoG method, which is quite related to our work, and the method based on a marker point process framework (MPP method) (Descombes, 2017). For a fair comparison, the parameters of the MLoG method and our method have been adjusted optimally to yield the best results, while the detection result of the MPP method is taken from the original work (Descombes, 2017).

As can be seen in Fig. 6.8, all three methods are able to detect the isolated blobs, despite the heterogeneous background. Nevertheless, the MLoG method fails to identify most of the overlapping blobs. While the MPP method misses one blob that has a high degree of overlap, the proposed method detects all the blobs successfully. This manifests that the proposed method can tackle the detection of overlapping blobs on a heterogeneous background.

6.4.2. Evaluation of robustness to noise

To illustrate the noise-robustness of our method, we perform the iLoG-USG method on noisy versions[1] of the image shown in Fig. 6.5a. Firstly, we obtain detection results on images corrupted by zero-mean Gaussian noise. As displayed in Fig. 6.9, when the variance of the Gaussian noise increases to 0.7^2, the iLoG-USG method can still yield a good detection result. This method yields an incorrect detection when this variance reaches 0.8^2. Secondly, we evaluate the robustness of our method to speckle noise. Speckle noise can be simulated by multiplying the image intensity by uniformly distributed zero-mean noise with variance ξ^2_{spk}. As shown in Fig. 6.10, the iLoG-USG method can yield a good detection result on the image corrupted by speckle noise with a variance $\xi^2_{\text{spk}} = 1.0^2$. When the variance increases to 1.1^2, an incorrect detection occurs. Moreover, we apply our method on an image corrupted by Poisson noise. The detection result is displayed in Fig. 6.11. It can be seen that our method successfully detects all the blob objects despite the presence of Poisson noise. From the experimental results presented above, we conclude that the iLoG-USG method can effectively tackle noisy images.

6.4.3. Evaluation on fluorescence microscopy cell images

In this experiment, we test our method on synthetic fluorescence microscopy cell images (Lehmussola et al., 2007; Ruusuvuori et al., 2008) taken from the publicly available SIMCEP dataset [2]. From the subsets with increasing *probability of clustering* (PoC), including the PoC 0%, PoC 15%, PoC 30% and PoC 45%, we sample two images from each subset, as shown in Figs. 6.12 and 6.13. One can see that there are populations of cells with heterogeneous grayscale intensity. Moreover, in the images with high PoC, quite a few cells are situated close to each other with a high degree of overlap.

We also compare our method with several competing methods, including the Top-Hat method (TH) (Breen et al., 1991), the LoG method (Lindeberg, 1998b), the determinant of Hessian method (DoH) (Lindeberg, 1998b), the H-dome method (HD) (Vincent, 1993; Xu and Lu, 2013), the method combining the LoG and the H-dome (LH) (Smal et al., 2010), the gLoG method (Kong et al., 2013a), the MSSEF method (Jaiswal et al., 2015) and the fast wavelet-based spot detection (FWSD) method (Püspöki et al., 2016).

All of the methods are implemented on a personal computer configured with Intel Core i7-3770 CPU (3.40GHz) with 16-GB RAM. The FWSD method runs in ImageJ, while the other methods run in Matlab R2018a.

[1] https://www.mathworks.com/help/images/ref/imnoise.html
[2] http://www.cs.tut.fi/sgn/csb/simcep/benchmark/

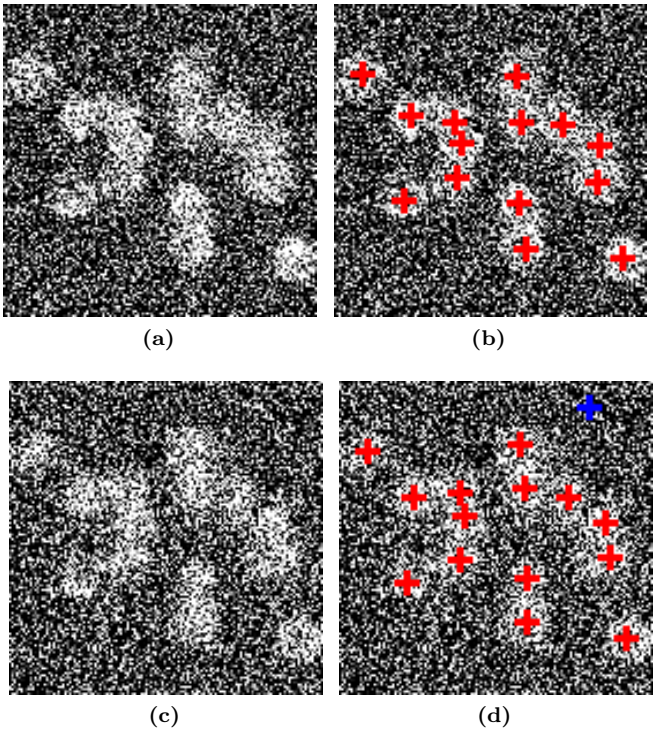
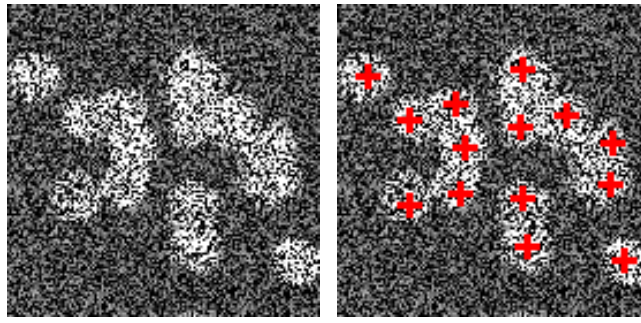


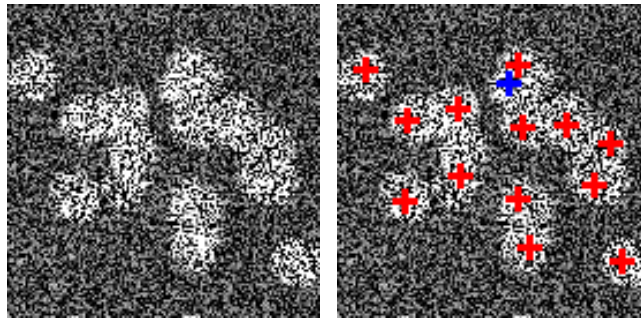
Figure 6.9: Detection results on images corrupted by Gaussian noise. (a-b) Image corrupted by zero-mean Gaussian noise with a variance of 0.7^2 and the detection result; (c-d) Image corrupted by zero-mean Gaussian noise with a variance of 0.8^2 and the detection result. The red and blue crosses denote the correct and incorrect detections, respectively.

To obtain a quantitative evaluation, we still adopt *precision*, *recall* as well as the F-measure (Basset et al., 2015) as evaluation measures. The method to compute the F-measure in this experiment is slightly different from those used in Chapter 3 and Chapter 4. In this experiment, the comparison between the cell detection result and the ground truth is formulated in terms of success and failure. Precision is the quotient of the number of correctly detected cell objects and the number of all the declared cell objects in the output. It represents the probability that the detected cells are valid. Recall is the quotient of the number of correctly detected cell objects and the number of all true cell objects in the ground truth. It represents the probability that the true cells have been correctly detected. Therefore, the precision and recall are computed as follows:

$$\text{PREC} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FP}}} \tag{6.27}$$

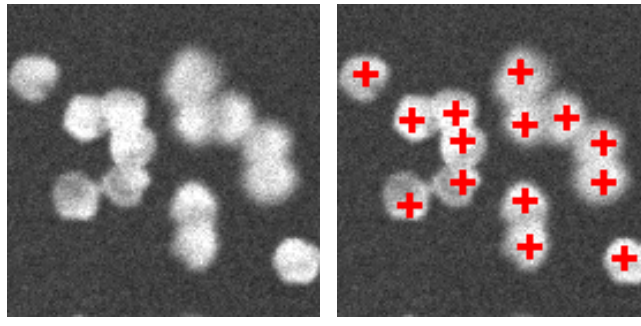


(a) (b)



(c) (d)

Figure 6.10: Detection results on images corrupted by speckle noise. (a-b) Image corrupted by speckle noise with a variance of 1.0^2 and the detection result; (c-d) Image corrupted by speckle noise with a variance of 1.1^2 and the detection result. The red and blue crosses denote the correct and incorrect detections, respectively.



(a) (b)

Figure 6.11: Detection result (b) on an image corrupted by Poisson noise (a).

and

$$\text{Rec} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FN}}}\,, \tag{6.28}$$

in which N_{TP}, N_{FP} and N_{FN} represent the numbers of true positive (correctly detected) objects, false positive (falsely detected) and false negative (falsely rejected) objects, respectively. Accordingly, we obtain the F-measure by computing the harmonic mean of the precision (Prec) and recall (Rec).

For each method, we select the parameters that yield the best overall performance. Since the cells in the fluorescence microscopy images are fairly homogeneous in size, for methods based on multiscale kernels, we consider the scales from 3.5 to 6.0 with a step of 0.5. Also, taking into account the time efficiency, we select as few scales as possible. For the sake of reproducibility, the parameter settings for all of the methods are listed below:

- LoG: The scale set is configured as $\mathbb{S} = \{4.5, 5.0\}$.
- DoH: The scale set is configured as $\mathbb{S} = \{5.0, 5.5, 6.0\}$.
- TH: The radius of morphological structuring element is set as 20.
- HD: The H-value (relative height) is set as 0.6.
- LoG-HD: The scale set of LoG kernels is configured as $\mathbb{S} = \{4.5, 5.0\}$, while the H-value is set as 0.2.
- gLoG: The scale set is configured as $\mathbb{S} = \{4.5, 5.0\}$, the direction set is originally configured as $\mathbb{D} = \{\pi \cdot i/8 \mid i \in \{0, 1, 2, \ldots, 7\}\}$, and the parameter controlling the blob-center detection accuracy and the blob eccentricity is originally set as 1.
- MSSEF: The scale set is configured as $\mathbb{S} = \{5.0, 4.5, 4.0, 3.5\}$ and the thresholding parameter is set as 0.5.
- FWSD: The minimum radius, the maximum radius and the overlap tolerance are set as 9 pixels, 15 pixels and 10 pixels, respectively. The minimum contrast is set as 0.1.
- The proposed method: The initial scale set and the direction set are configured as $\mathbb{S}^{(1)} = \{5.0, 5.5\}$ and $\mathbb{D}_{\text{u}} = \{\pi \cdot i/2 \mid i \in \{0, 1, 2, 3\}\}$, respectively. The number of iterations of LoG filtering is set as $K = 3$, while the scale decay factor is set as $\varsigma = 2/3$.

In a preprocessing step, the blue channel of each original fluorescence microscopy image is selected to form the grayscale image. In addition, in the binarization procedure of all the methods except the MSSEF method, we have used Otsu thresholding (Otsu, 1979) and Rosin thresholding (Rosin, 2001).

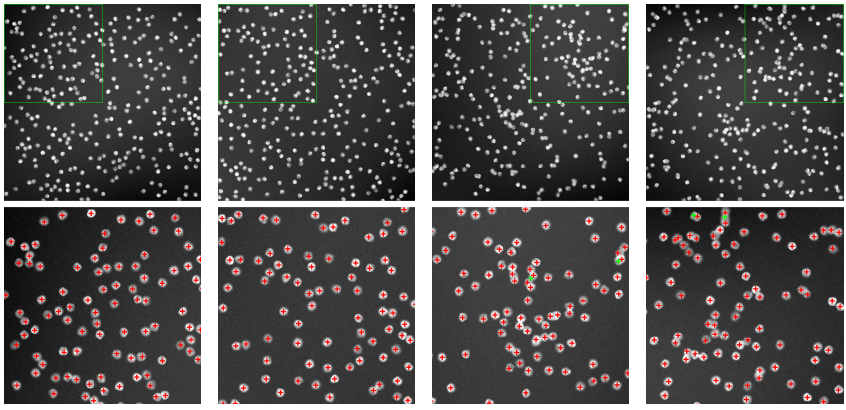


Figure 6.12: Cell detection results obtained by the proposed method on the Images PoC 0%-1 (a), PoC 0%-2 (b), PoC 15%-1 (c) and PoC 15%-2 (d). The red, green and blue crosses denote the TP, FN and FP results, respectively. For a better illustration, the images in the first row are the blue channel of the original fluorescence microscopic images. In each column, the image in the second row shows the zoomed-in patches corresponding to the green windows in the first row.

For each method, the thresholding technique yielding the better performance is used to produce the final results.

We report the quantitative evaluation results in Tab. 6.1 and illustrate the cell detection results obtained by our method in Figs. 6.12 and 6.13. All of the methods yield a high precision, which indicates that all of the methods can retain the cell objects while suppressing the background. The high precision is also due to a high signal-to-noise ratio in the original images. Nevertheless, in terms of the recall and F-measure, the proposed method outperforms all of the competing methods. On the images with low PoC, most of the methods yield acceptable detection results, while the TH method yields slightly lower values of the recall and F-measure. As the PoC increases, the performances of the competing methods decrease significantly, while the proposed method still obtains a high recall and high F-measure value. Compared with our method, although some methods obtain slightly higher precision values in several cases, but their recall and F-measure values are much lower. These experimental results demonstrate that our method can tackle both isolated and partially overlapping blobs.

With respect to the time efficiency, the average runtime of each method for processing a single fluorescence microscopy cell image is reported in Tab. 6.2. Among the methods, the TH method is the most efficient method to run, while

Table 6.1: Evaluation results obtained by each method on each fluorescence microscopy cell image.

Images	Metrics	Methods								
		LoG	DoH	TH	HD	LoG-HD	gLoG	MSSEF	FWSD	iLoG-USG
PoC 0%-1	F	0.990	0.997	0.967	0.983	0.998	0.992	0.987	0.987	**1.000**
	Prec	0.997	0.997	1.000	0.990	1.000	0.984	0.993	0.997	1.000
	Rec	0.983	0.997	0.937	0.977	0.997	1.000	0.980	0.977	1.000
PoC 0%-2	F	0.995	0.995	0.980	0.980	0.997	0.988	0.988	0.980	**1.000**
	Prec	1.000	1.000	1.000	0.990	0.997	0.987	0.993	0.993	1.000
	Rec	0.990	0.990	0.960	0.970	0.997	0.990	0.983	0.967	1.000
PoC 15%-1	F	0.901	0.963	0.861	0.897	0.976	0.945	0.898	0.942	**0.990**
	Prec	0.996	1.000	1.000	0.984	0.997	1.000	0.996	0.985	1.000
	Rec	0.834	0.929	0.756	0.824	0.956	0.895	0.817	0.902	0.980
PoC 15%-2	F	0.935	0.967	0.875	0.902	0.974	0.949	0.914	0.928	**0.988**
	Prec	1.000	1.000	1.000	0.992	1.000	0.996	1.000	0.992	1.000
	Rec	0.878	0.936	0.777	0.828	0.949	0.905	0.841	0.872	0.976
PoC 30%-1	F	0.897	0.952	0.804	0.829	0.956	0.917	0.855	0.899	**0.988**
	Prec	0.983	1.000	1.000	0.981	0.996	0.988	1.000	0.983	0.996
	Rec	0.824	0.909	0.673	0.718	0.919	0.856	0.747	0.828	0.979
PoC 30%-2	F	0.891	0.941	0.803	0.852	0.957	0.9023	0.858	0.910	**0.984**
	Prec	0.987	1.000	1.000	0.986	1.000	0.9836	0.991	0.988	0.990
	Rec	0.813	0.889	0.670	0.750	0.917	0.8333	0.757	0.844	0.979
PoC 45%-1	F	0.903	0.943	0.812	0.883	0.957	0.931	0.862	0.888	**0.982**
	Prec	0.996	1.000	1.000	0.952	0.993	0.996	1.000	0.979	1.000
	Rec	0.826	0.892	0.670	0.823	0.924	0.875	0.757	0.813	0.965
PoC 45%-2	F	0.890	0.949	0.775	0.824	0.964	0.928	0.835	0.876	**0.982**
	Prec	1.000	1.000	1.000	0.990	1.000	0.992	0.995	0.971	0.996
	Rec	0.802	0.903	0.632	0.705	0.931	0.872	0.719	0.799	0.969

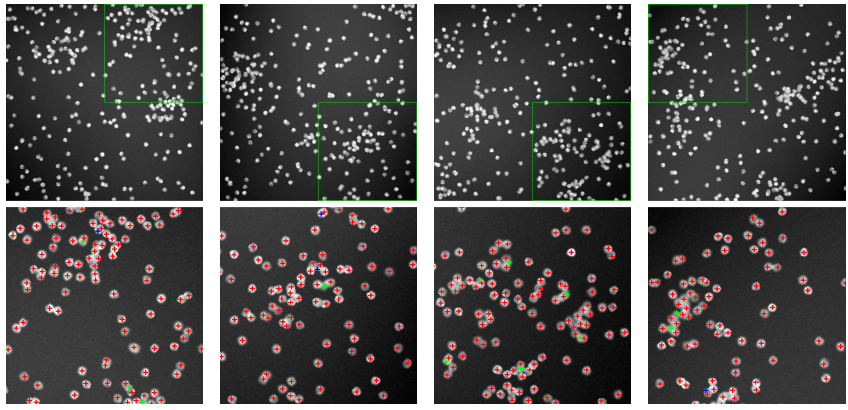


Figure 6.13: Cell detection results obtained by the proposed method on the Images PoC 30%-1 (a), PoC 30%-2 (b), PoC 45%-1 (c) and PoC 45%-2 (d). The red, green and blue crosses denote the TP, FN and FP results, respectively. For a better illustration, the images in the first row are the blue channel of the original fluorescence microscopic images. In each column, the image in the second row shows the zoomed-in patches corresponding to the green windows in the first row.

Table 6.2: The average runtime (s) of each method for processing the fluorescence microscopy cell images. The symbol † denotes the execution time on the ImageJ platform.

Methods	LoG	DoH	TH	HD	LoG-HD	gLoG	MSSEF	FWSD	iLoG-USG
Runtime	0.17	0.22	0.14	0.20	0.19	0.42	0.24	$5.28^{\dagger}$	0.41

the FWSD method consumes the highest execution time. The LoG, DoH, HD, LoG-HD and MSSEF methods are also relatively efficient, consuming approximately half of the runtime of the gLoG method or our method. Nevertheless, the computational cost of the proposed method is usually affordable, and, moreover, the proposed method achieves the best performance among all of the methods.

6.4.4. Evaluation on nanoparticle images

In addition to the experiment on fluorescence microscopy cell images, we further apply the proposed method to a nanoparticle dataset (Park et al., 2013). The microscopy images contain nanoparticles that have a medium or high degree of overlap (Zafari et al., 2015). In each image, the total number of nanoparticles is provided by the original work (Park et al., 2013) and can be validated by manual counting. The detection of nanoparticles has been addressed in literature.

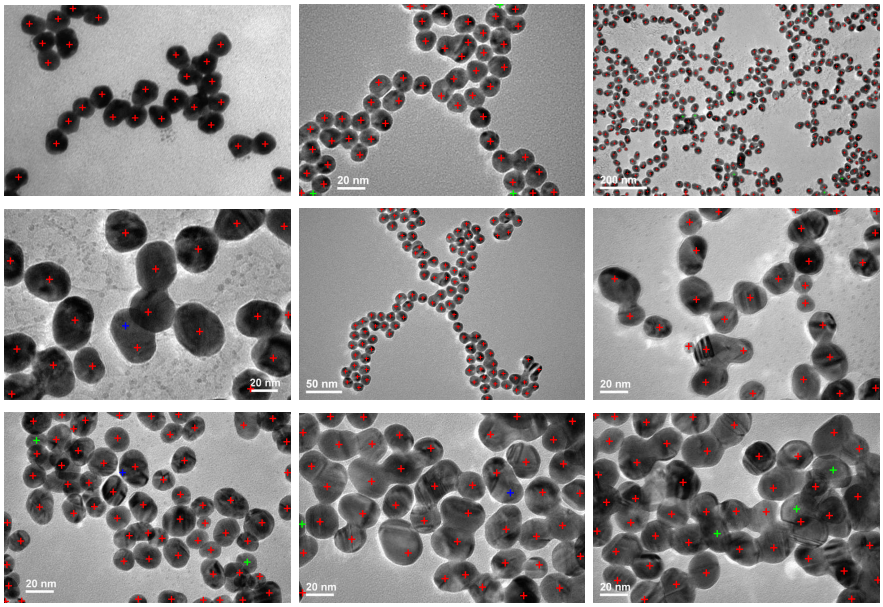


Figure 6.14: Nanoparticle detection results of the proposed method. The red, green and blue crosses denote the TP, FN and FP results, respectively.

Existing methods include the iterative voting method (IVM) (Parvin et al., 2007), the morphological multiscale decomposition method (MMD) (Schmitt and Hasse, 2009), the sliding band filter method (SBF) (Quelhas et al., 2010), the ultimate erosion for convex sets method (UECS) (Park et al., 2013) and the state-of-the-art method that is based on bounded erosion and the fast radial symmetry transform (BEFRS) (Zafari et al., 2015). We compare our method with these existing methods. Following the works in (Park et al., 2013) and (Zafari et al., 2015), we evaluate the performance of each method in terms of the number of correctly detected particles.

The sizes of the nanoparticles are not identical in different images. For each image, we individually set the proper initial scale set and accordingly obtain the evaluation results.

Table 6.3 shows the evaluation results of nanoparticle detection from (Park et al., 2013) and (Zafari et al., 2015), updated with the results of the proposed method, while Fig. 6.14 shows all the object detection results that are obtained by the proposed method. As we can see, SBF, BEFRS as well as the proposed method have correctly detected all the objects in Image #1. On all the remaining images, the proposed method achieves the best performance among all the methods. It is worth noting that compared to the cases with a

Table 6.3: The number of the correctly detected particles by different methods on each nanoparticle image.

Images	Degree of overlap	Number of particles	**Methods**					
			IVM	MMD	SBF	UECS	BEFRS	Proposed
#1	Medium	28	25	20	**28**	26	**28**	**28**
#2	Medium	52	45	48	43	48	46	**49**
#3	Medium	459	227	429	262	437	425	**452**
#4	Medium	19	16	16	6	17	18	**19**
#5	Medium	108	85	92	99	103	104	**108**
#6	Medium	29	21	23	19	25	25	**29**
#7	High	63	42	40	42	54	52	**61**
#8	High	44	27	28	28	34	37	**43**
#9	High	45	24	22	25	33	33	**42**
Total	-	847	512	718	552	777	768	**831**

medium degree of overlap, the advantage of the proposed method over the competing methods is more significant in the cases with a high degree of overlap. In Fig. 6.14, one can see that most of the nanoparticles have been detected successfully, despite of the serious overlap, the high density and the heterogeneous particle sizes.

6.5. Conclusions

There is a significant demand for detecting overlapping blob objects using automated image analysis techniques. To address this classical, yet challenging problem, in this chapter, we have presented a novel blob detection method based on iterative Laplacian-of-Gaussian filtering and unilateral second-order Gaussian kernels. The iteration of the Laplacian-of-Gaussian filtering reduces the degree of overlap, facilitating a subsequent blob extraction procedure. The employed unilateral second-order Gaussian kernels yield responses only for blob objects, and as such, the blob objects can be pinpointed by a thresholding step. The experimental results have demonstrated that the proposed method shows a promising performance in detecting fluorescence microscopy cells and electron micrograph nanoparticles, even when there is a high degree of overlap.

PART III

EXPLORATION OF DEEP CONVOLUTIONAL NETWORKS FOR IMAGE ANALYSIS

7 Automated *Artemia* detection and counting

The brine shrimp *Artemia* is an important organism in aquaculture, and the number of studies on *Artemia* is increasing. *Artemia* detection and counting is a fundamental task in *Artemia* image analysis. To facilitate this task, we propose an automated detection and counting method in this chapter. Our method consists of a UNet-based marker proposal network and a CNN-based target classifier, and we therefore term it as the Marker-CNN method. The marker proposal network introduces the image segmentation scheme into object detection. It can generate target candidates, separate highly adjacent objects and obtain the object structural information simultaneously. The target classifier determines the category of each target candidate, thereby yielding the *Artemia* detection and counting results. Moreover, we have compiled an *Artemia* detection and counting dataset to train and test the proposed method. Experimental results manifest that the proposed Marker-CNN method can accurately detect and count the *Artemia* objects in images.

The material of this chapter is based on the following publication:

- Wang, G., Van Stappen, G., and De Baets, B. (2019e). Automated detection and counting of *artemia* using U-shaped fully convolutional networks and deep convolutional networks. *Expert Systems with Applications*, Under review

7.1. Motivation

The brine shrimp *Artemia* is a genus of aquatic crustaceans. *Artemia* cysts are robust to store and are convenient to hatch into nauplii, which have a high nutritional content. Therefore, *Artemia* nauplii are extensively used as a kind of cost-effective live diet for fish and crustacean larvae, while Artemia cysts are also an essential and expensive commodity in larval aquaculture (Le et al., 2018). *Artemia* is also considered as a useful organism for stress response studies since it has been naturally found in a variety of harsh environments worldwide (El-Magsodi et al., 2016). Besides, *Artemia* has been employed as a test organism in toxicological assays and various other biological disciplines. Since *Artemia* cysts and nauplii are very small in size, they are usually observed by a stereo-microscope (Ates et al., 2013). Currently, most of the *Artemia* image analysis tasks are carried out manually. Such a typical image

analysis task is to detect and count the number of objects. For example, when assessing the hatching rate in *Artemia* incubation, which is an important commercial quality criterion, the number of cysts and nauplii should be counted separately(Kim and Cho, 2013). However, manual analysis is time-consuming and labor-intensive. It is very difficult to analyze high-throughput imagery data manually (Meijering et al., 2016). With the rapid development of computer vision techniques and their applications in image analysis (Geweid et al., 2019), it would be highly desired to have access to automated *Artemia* detection and counting methods.

Over the past several decades, many automated object detection and counting methods have been proposed (Kamilaris and Prenafeta-Boldú, 2018). Examples are automated methods for counting cells (Choudhry, 2016), fruits (Chen et al., 2017), people (Hashemzadeh and Farajzadeh, 2016), etc. Early methods detect objects using hand-crafted features. For example, many conventional pedestrian detection methods employ the histograms of oriented gradients to make the target separable from the background (Pang et al., 2011). Another example is the cell detection approach based on multiscale Laplacian-of-Gaussian features (Kong et al., 2013a). Generally, methods based on hand-crafted features have a good interpretability, and the detection procedure can be well controlled by a few key parameters (Kong et al., 2013a). Nevertheless, these methods are limited in describing semantic information, and as such, they might underperform when the appearances of the targets vary significantly (Ronneberger et al., 2015).

Recently, relying on their powerful feature learning and representation abilities, deep learning techniques have been widely used for object detection (Han et al., 2018). For generic object detection, the goal is to plot the bounding box, which represents the position and spatial extent of an object, while determining the category of this object (Szegedy et al., 2013). These methods can be mainly grouped into two classes: methods based on regression and methods based on region proposals. Methods based on regression compute the confidence scores of bounding boxes and categories in a single stage (Liu et al., 2016). For instance, Redmon et al. (2016) proposed a one-stage method named You-Only-Look-Once (YOLO). This method divides an image into grid cells. Capitalizing on the feature map extracted by deep convolutional neural networks (Krizhevsky et al., 2012), this method computes the confidence scores for both multiple bounding boxes and multiple categories in each grid cell. Thus, a grid cell with a high score for both a bounding box and a category is determined as a positive detection result. The YOLO method has a good computational efficiency. It detects objects using relatively larger *anchor boxes*, hence it produces comparatively fewer false positives in the background.

Nonetheless, this scheme assumes that each grid cell only contains a single object, and therefore, the YOLO method sometimes leads to errors of localization and false negatives for small objects (Han et al., 2018).

Detection methods based on region proposals localize the objects by simulating the attention mechanism of the human visual system. With a coarse-to-fine scheme, these methods usually include procedures of region proposing, feature extraction, classification and bounding box regression. A representative method is the one proposed by Girshick et al. (2016), which combines a region proposal network with a deep convolutional neural network (R-CNN method). The R-CNN method uses *anchor boxes* and selective search to generate thousands of region proposals. It also extracts features using a deep convolutional neural network, which facilitates a subsequent category classification procedure. With the confidence scores of categories generated by a support vector machine, the R-CNN method refines the bounding boxes, thereby yielding the detection results. Since the R-CNN method carefully considers each potential region, it usually achieves a high degree of recall. Nevertheless, this scheme entails a considerable redundancy and a heavy computation (Girshick, 2015; Ren et al., 2016). A more detailed review on generic object detection can be found in (Han et al., 2018).

The aforementioned deep convolutional neural network (CNN) is a powerful tool for image classification (Rawat and Wang, 2017). In literature, many types of CNN architectures have been developed, such as the AlexNet (Krizhevsky et al., 2012), Visual-Geomery-Group-16 (VGG16) (Simonyan and Zisserman, 2015), Residual Net (He et al., 2016), etc. In practical applications, the CNN has also enjoyed a great success in the fields of disease diagnosis (Vo et al., 2019), remote sensing (Kussul et al., 2017), autonomous driving (Bresson et al., 2017), etc. Nevertheless, since a CNN makes decisions using the global information of the input image, it has limitations in object localization. Therefore, it is inappropriate to straightforwardly employ a CNN for image segmentation (Long et al., 2015). To explore deep learning techniques in pixel-level prediction, Long et al. (2015) proposed an architecture named fully convolutional network (FCN) by transforming the fully connected layers in a CNN into convolutional layers. In addition, to obtain an output map that has the same size as the input, the FCN upsamples the last layer of the CNN by a deconvolution operation. Building on the FCN, Ronneberger et al. (2015) designed a U-shaped fully convolutional network (UNet). The UNet has both contracting (a.k.a. downsampling or encoding) convolutional layers and expanding (a.k.a. upsampling or decoding) convolutional layers. The pooling maps yielded by the layers in the contracting path are used in the expanding path to progressively obtain an output that has the same

size as the input. Thus far, the UNet has been successfully applied in road extraction (Zhang et al., 2018), liver segmentation (Liu et al., 2019), cell counting (Ronneberger et al., 2015), etc. Nonetheless, the UNet is mostly used for image segmentation, and as such, it cannot be readily used for counting objects belonging to multiple categories (Ronneberger et al., 2015).

As mentioned earlier, the demand for automated *Artemia* detection and counting is increasing. However, to the best of our knowledge, this problem has been seldom addressed so far. The only existing *Artemia* counting method is the one proposed by Kim and Cho (2013). Their method roughly obtains the foreground objects, including the cysts, nauplii and artefacts, using a binarization technique. Some small noisy points are removed by morphological operations. Subsequently, in the binary image, the method determines the connected components with a high circularity as cysts, and the connected components with a low circularity as nauplii. However, this method would yield many false results when the background is not clean. Moreover, this method can hardly separate adjacent objects, and would thus lead to a serious detection error. Besides, although many automated methods have been developed for detecting moths (Ding and Taylor, 2016), fish (French et al., 2015), shrimps (Kesvarakul et al., 2017; Kaewchote et al., 2018), etc., they cannot be readily transferred to detect and count *Artemia*. This is because the *Artemia* objects in images are usually highly adjacent. Moreover, the appearance of *Artemia* varies significantly over the different growth stages. Therefore, it is quite necessary to design a tailor-made method for analyzing *Artemia* images.

In this chapter, we aim at an automated method that can detect and count *Artemia* objects accurately and efficiently. Inspired by the scheme of the R-CNN method, we propose a two-stage method by combining a target marker proposal network with a CNN-based classifier (Marker-CNN method). In the first stage, instead of generating region proposals by blind anchor boxes, we design a target marker proposal network that has a UNet architecture. The markers are represented by thick trunk skeletons of the objects. We design the marker proposal network because of the following reasons. Firstly, compared with the conventional bounding box annotation (Ren et al., 2016), or the full instance mask annotation (He et al., 2017), the thick trunk skeleton marker annotation is easier for annotators, which can be accomplished by one stroke. Secondly, compared with bounding boxes, the thick trunk skeleton markers indicate not only the positions and extents, but also the spatial structures. Therefore, our marker proposal network can effectively separate the *Artemia* objects that are highly adjacent. Thirdly, our marker proposal network can accurately indicate target candidates, which significantly facilitates

a subsequent classification procedure. Then, in the second stage, we design a CNN-based network to classify the target candidates into the categories or label as a non-target. Furthermore, we have compiled a dataset to train and test the proposed method.

The remainder of this chapter is organized as follows. Section 7.2 recalls several related works. In Section 7.3, we present an *Artemia* detection and counting dataset, and subsequently, we elaborate the proposed Marker-CNN method. In Section 7.4, we report and discuss the experimental results. Finally, conclusions are listed in Section 7.5.

7.2. Preliminaries on convolutional neural networks

In this section, we present two concepts that are highly related to our work: deep convolutional neural networks and U-shaped fully convolutional networks.

7.2.1. Deep convolutional neural networks

The CNN is one of the most popular architectures among the deep learning techniques (Rawat and Wang, 2017). A typical CNN architecture consists of a sequence of convolutional layers, pooling layers and fully connected layers (LeCun et al., 2015). A convolutional layer is composed of convolutional neurons (LeCun and Bengio, 1995). Each neuron is represented by a trainable kernel. The response of each kernel is obtained by convolving the kernel with the output of the previous layer. The convolutional response is then passed through a non-linear activation unit, e.g., a rectified linear unit (Hinton, 2010), to obtain the feature map. In this way, different neurons yield different feature maps. Compared with the scheme considering an image as a vector (Kang and Palmer-Brown, 2008), convolutional operations can better exploit the spatial information of the image, since spatially neighboring pixels are often highly correlated in content (LeCun et al., 2015). Besides, convolutional operations can obtain a good translation invariance that can benefit the image classification greatly. In a pooling layer, the input map is partitioned into non-overlapping regions, and subsequently, the values in each region are aggregated into a single value. As a result, the dimension of the feature map is significantly reduced (Boureau et al., 2010). Moreover, a pooling operation can make the model more robust to spatial shifts. A fully connected layer multiplies the layer input by a weight matrix and then adds a bias vector, thereby exploiting the extracted high-level features. In the training procedure, all the parameters in

Figure 7.1: Illustration of the *Artemia* growth stages.

the CNN are iteratively updated by the backpropagation mechanism (LeCun et al., 1989).

7.2.2. U-shaped fully convolutional networks

In view of the great success of CNN in object classification, many works have tried to apply CNN for image segmentation, which is also a fundamental task in the computer vision filed. A straightforward scheme is to classify all the image patches cropped by a sliding window (Ciresan et al., 2012), but this scheme inevitably leads to a serious redundancy and a heavy computation. To address such shortcomings, the FCN (Long et al., 2015), which replaces the fully connected layers in the CNN with convolutional layers, was proposed for image segmentation. In order to make the sizes of the output and input identical, the FCN method upsamples the output of the last convolutional layer by a deconvolution operation, while exploiting the pooling results that are yielded by the intermediate layers of the CNN (Oliveira et al., 2018). The UNet extends the FCN architecture into a U-shaped network(Ronneberger et al., 2015), which has both contracting convolutional layers and expanding convolutional layers. The pooling maps yielded by the layers in the contracting path are also used in the expanding path to progressively obtain an output with the same size as the input. In this way, the UNet model can exploit the localization information from the intermediate CNN layers and the extracted high-level features simultaneously.

7.3. Automated *Artemia* detection and counting method

In many *Artemia* studies, e.g., in a commercial quality assessment of *Artemia* hatching, accurate numbers of *Artemia* cysts and nauplii are highly desired (Lopes-dos Santos et al., 2019). As shown in Fig. 7.1, an *Artemia* life cycle mainly includes the cyst, nauplius/metanauplius, juvenile and adult stages. The *Artemia* juveniles and adults are comparatively large in the field

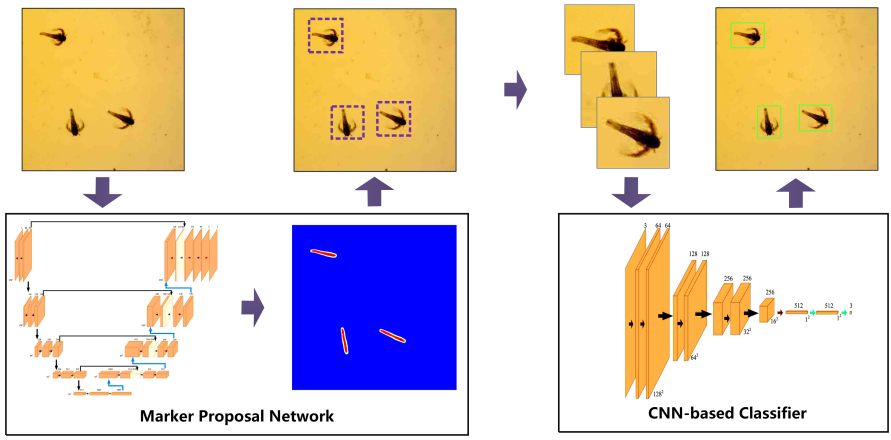


Figure 7.2: Illustration of the proposed Marker-CNN method for *Artemia* detection and counting.

of view in a microscope, and therefore, the number of *Artemia* adults (usually less than 5) in the image can be easily obtained by a glance. In this chapter, we therefore focus on detecting and counting *Artemia* objects that are in the cyst stage, nauplius stage (the first growth stage after the cyst stage) and metanauplius stage (the subsequent growth stage to the nauplius stage). Hundreds or thousands of cysts or nauplii/metanauplii may be present in a microscopy image. For convenience, we use the term nauplius to collectively refer to the terms nauplius and metanauplius. As mentioned earlier, automated *Artemia* detection and counting is a challenging task. The *Artemia* objects in images might be seriously adjacent, and moreover, the appearances of *Artemia* objects vary significantly over the different growth stages. Aiming at automated *Artemia* detection and counting, we employ deep learning techniques, which can exploit semantic information and high-level features, to overcome these problems. We design a two-stage method using the UNet architecture and the CNN architecture. In the first stage, we design a UNet-based marker proposal network. This module can indicate target candidates while separating adjacent objects. In the second stage, we develop a CNN-based classifier to classify the target candidates into the categories or label as a non-target, thereby obtaining the detection results. The flowchart of the proposed Marker-CNN method is illustrated in Fig. 7.2. To train and test this method, we have compiled an *Artemia* detection and counting (ArtDeCo) dataset[1].

[1] https://github.com/GangWangUgent/ArtDeCo/

7.3.1. The *Artemia* detection and counting dataset

The method we designed for detecting and counting *Artemia* objects is based on supervised learning. Before elaborating the proposed method, we introduce an *Artemia* detection and counting dataset. We captured *Artemia* images by a stereo-microscope (SMZ1270, Nikon®) connected to a lens of Nikon® (Plan Apo). The *Artemia* objects were fixed with lugol solution (1%). Using the acquired images, we built two subsets, an *Artemia* marker dataset and an *Artemia* classification dataset.

In the *Artemia* marker dataset, there are 70 images (with a resolution of $512 \times 512 \times 3$) containing 1353 cysts and 2421 nauplii. The corresponding ground truth marker maps have been manually labelled, as shown in Fig. 7.3. For each *Artemia* cyst and nauplius, only the thick trunk skeleton is labelled as a positive marker. As discussed earlier, compared with conventional annotation approaches, such as bounding boxes and full instance masks, our annotation approach has several advantages. Our annotation approach is easier to carry out. Instead of finding a rectangular bounding box that fully encloses an object, or labelling all the pixels of an object mask, we mark an object by only one stroke. In addition, it has been validated that skeleton extraction can benefit the instance segmentation task (Shen et al., 2017), since the skeleton embodies information on both the object position and the spatial structure. Moreover, in our annotation method, labels of thick trunk skeletons can facilitate the separation of highly adjacent objects. Furthermore, it is worth pointing out that thick trunk skeletons are more appropriate than thin skeletons in our framework, because the latter occupy far less pixels than thick ones, thus avoiding the problem of pixel-class imbalance (Sironi et al., 2016). Although some approaches, e.g., the training strategy using the focal loss function (Lin et al., 2017), have been proposed to relieve the problem of pixel-class imbalance, thin skeletons are very fragile to extract, and fractured thin skeletons would indicate superfluous target regions for a specific target.

In the *Artemia* classification dataset, there are 2100 images (with a resolution of $128 \times 128 \times 3$), including 900 cyst images, 900 nauplius images and 300 non-target images. Figure 7.4 displays several sample images for each category. Besides, we collected 80 images (with a resolution of $512 \times 512 \times 3$) for evaluating the performance in object detection and counting. These test images contain 1336 cysts and 3335 nauplii in total.

7.3.2. The marker proposal network

Our *Artemia* detection and counting method mainly consists of two modules: the marker proposal network and the classifier for target identification. The

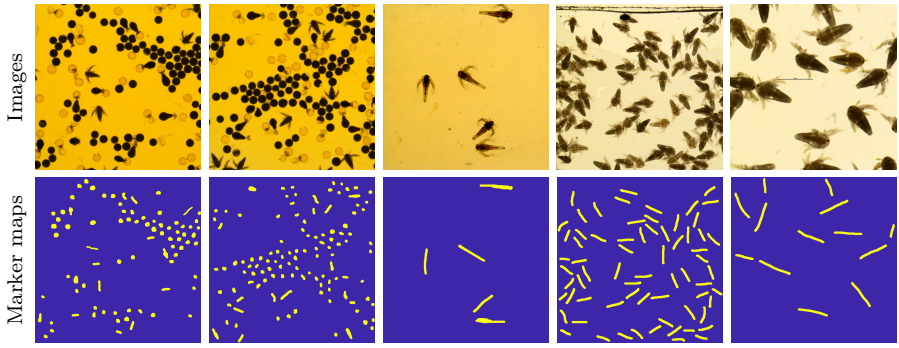


Figure 7.3: Sample *Artemia* images and the corresponding labelled marker maps for training the marker proposal network.

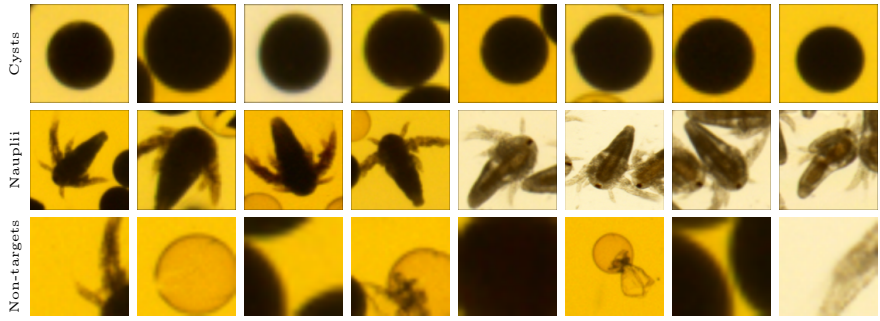


Figure 7.4: Sample images for training the CNN-based target classifier. Top row: Samples of cysts; Second row: Samples of nauplii; Bottom row: Samples of non-targets.

markers are represented by the thick trunk skeletons of objects, which embody information on the positions, extents and spatial structures. Instead of using numerous blind anchor boxes (Girshick et al., 2016), we extract the target markers through a semantic image segmentation procedure.

As reviewed earlier, the UNet has been widely used in image segmentation, due to its good properties in object localization and shape description. Therefore, we design a UNet-based architecture to yield the markers. The architecture of our marker proposal network is displayed in Fig. 7.5. It can be seen that the overall network is nearly symmetric, containing a contracting path and an expanding path.

The contracting path is built on a sequence of convolutional layers and *max pooling* layers (Krizhevsky et al., 2012). Each convolutional layer performs a convolution operation on the input maps and the trainable kernels. Each kernel extracts the features of a receptive field from the previous layer, thereby

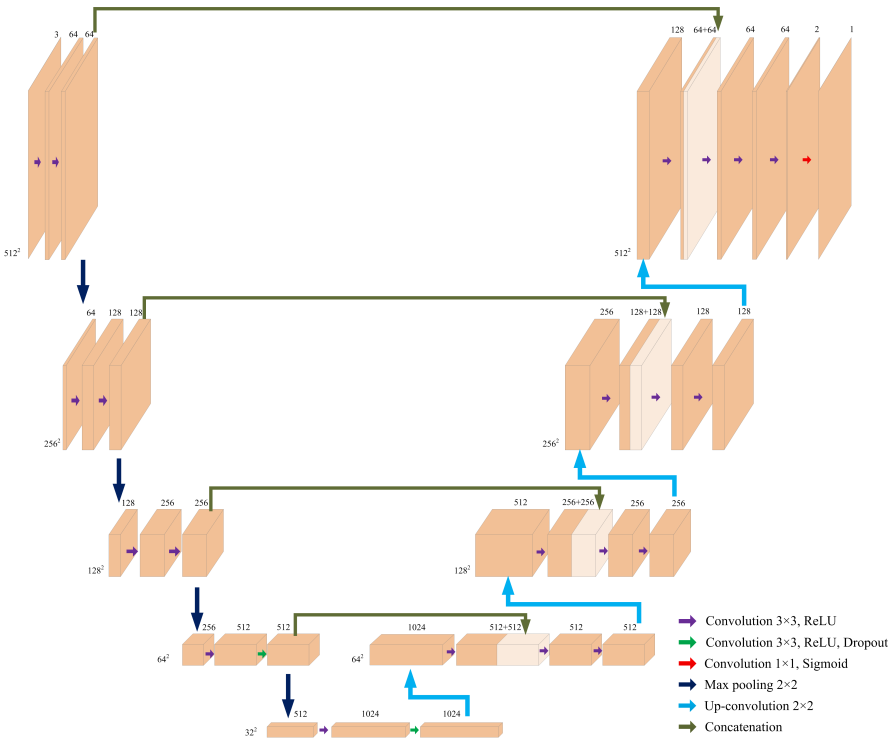


Figure 7.5: Architecture of the UNet-based marker proposal network.

contributing a convolutional response map. Let $\mathbf{I}^{(l-1)}$ be an input map of the l-th convolutional layer. For a trainable kernel κ in the l-th layer, the convolutional response map $\mathbf{Y}^{(l)}$ is obtained as

$$\mathbf{Y}^{(l)} = \mathbf{I}^{(l-1)} * \kappa \,, \tag{7.1}$$

where $*$ denotes a convolution operation. The trainable kernels are updated on each iteration through the backpropagation mechanism (LeCun et al., 1989). In each convolutional layer, the number of kernels is configured as shown in Fig. 7.5. In addition, since a convolution operation leads to a size decrease for the processed map, we employ a replication padding when performing the convolution operation, thereby making the sizes of the input map and the convolutional response map identical. For each convolutional response map, we obtain the corresponding feature map based on a non-linear activation function. We employ the rectified linear unit (ReLU) (Hinton, 2010) as the activation function, in view of its advantages in training speed. In the ReLU, only positive values are retained. That is, after passing through the ReLU,

the convolutional response $\mathbf{Y}^{(l)}$ becomes

$$\mathbf{I}^{(l)}(\mathbf{m}) = \mathcal{N}_{\mathrm{r}}\left(\mathbf{Y}^{(l)}(\mathbf{m})\right) = \max\left(\mathbf{Y}^{(l)}(\mathbf{m}), 0\right) , \tag{7.2}$$

where $\mathbf{m}$ stands for the image coordinates and $\mathcal{N}_{\mathrm{r}}(\cdot)$ represents the ReLU function, as defined earlier. Moreover, we adopt the max pooling operation (Boureau et al., 2010) to reduce the dimensionality of the feature maps. In a max pooling layer, the elements of the output are obtained by sliding a square window over the input map with a specified sliding stride, while selecting the maximum value within the window. In our model, the size of the window and the sliding stride are set to be 2×2 and 2, respectively. The UNet model we designed consists of five contracting steps. At each contracting step, the number of feature maps is doubled. Furthermore, to address the problem of overfitting, we employ the so-called dropout technique (Srivastava et al., 2014), which sets the output of each hidden neuron to zero with probability 50%. The neurons that are dropped out make no contribution to the forward pass process and their weights are not updated during the backpropagation process. In our UNet model, the dropout technique is used for the layers illustrated in Fig. 7.5.

The expanding path is built on a stack of convolutional layers and up-convolution layers. The convolutional layers work in the same way as those in the contracting path. An up-convolution layer doubles the size of each input map by adding zero pixels with a stride, and, subsequently, yields the output map by a convolutional operation. At each expanding step, the number of feature maps is halved. In particular, in order to better exploit the localization information, the feature maps yielded in the contracting path are concatenated with the corresponding feature maps in the expanding layers, as illustrated in Fig. 7.5.

At the end, the prediction map $\mathbf{Z}$ is obtained by a sigmoid function that is given by

$$\mathbf{Z}(\mathbf{m}) = \frac{1}{1 + \exp\left(-\mathbf{Z}_{\mathrm{in}}(\mathbf{m})\right)} , \tag{7.3}$$

where $\mathbf{Z}_{\mathrm{in}}$ denotes the input of the last layer.

The parameters in the UNet model are trained by minimizing the loss function that reflects the difference between the ground truth label map and the yielded prediction map. Since the segmentation in our method is a binarization problem, we define the loss function based on weighted binary cross entropy. Given a prediction map $\mathbf{Z}$ and its corresponding ground truth label map $\mathbf{T}$,

the pixel-weighted cross-entropy loss is computed as follows:

$$\mathcal{L}_{\mathrm{pc}}(\mathbf{Z},\mathbf{T}) = -\frac{1}{N_{\mathrm{pm}}}\sum_{\mathbf{m}\in\mathbf{T}} \mathbf{T}(\mathbf{m})\log\left(\mathbf{Z}(\mathbf{m})\right) + w_{\mathrm{pc}}\cdot\left(1-\mathbf{T}(\mathbf{m})\right)\log\left(1-\mathbf{Z}(\mathbf{m})\right)\,, \tag{7.4}$$

where

$$w_{\mathrm{pc}} = \frac{\sum_{\mathbf{m}\in\mathbf{T}} \mathbf{T}(\mathbf{m})}{N_{\mathrm{pm}} - \sum_{\mathbf{m}\in\mathbf{T}} \mathbf{T}(\mathbf{m})} \tag{7.5}$$

denotes the loss weight of a pixel and N_{pm} represents the number of pixels in the prediction map.

7.3.3. The target classifier

It has been widely acknowledged that the CNN is powerful in object classification. Since object detection is a procedure combining object localization and classification, the CNN has also been incorporated in many object detection methods. As mentioned earlier, the R-CNN object detection method (Girshick et al., 2016) employs a CNN module to extract the features and uses a support vector machine to determine whether a region of interest belongs to a target category or not. In Section 7.3.2, we have designed the marker proposal network to obtain the target candidates. Next, we develop a classifier to classify the target candidates using the CNN.

Capitalizing on the extensively used VGG16 architecture (Simonyan and Zisserman, 2015), we design a CNN-based target classifier as displayed in Fig. 7.6. Our CNN-based classifier consists of seven convolutional layers, three max pooling layers and three fully connected layers. The convolutional layers and max pooling layers (Krizhevsky et al., 2012) work in the same way as those in the UNet-based marker proposal network. To avoid a size decrease during the convolution operation, we also apply a replication padding in the convolutional layers.

In the last part of our CNN-based classifier, by flattening (i.e., vectorizing) the feature maps, the fully connected layers connect every neuron in one layer to every neuron in another layer, thereby implementing a high-level reasoning for object classification. Mathematically, a fully connected layer is represented as:

$$\mathbf{r}_{\mathrm{fc}}^{(l)} = \mathcal{N}_{\mathrm{r}}\left(\mathbf{W}^{(l)}\mathbf{r}_{\mathrm{fc}}^{(l-1)} + \mathbf{s}^{(l)}\right)\,, \tag{7.6}$$

where $\mathbf{r}_{\mathrm{fc}}^{(l)}$ stands for the l-th fully connected layer, $\mathbf{s}^{(l)}$ is the bias vector

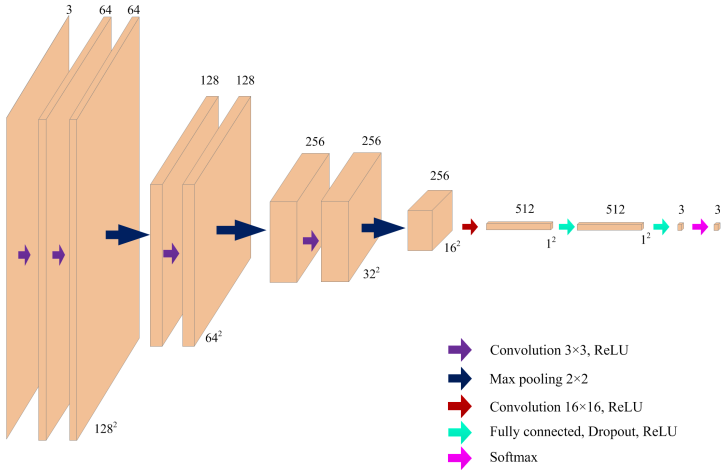


Figure 7.6: Architecture of our CNN-based target classifier.

and $\mathbf{W}^{(l)}$ denotes the weight matrix. Each element of $\mathbf{W}^{(l)}$ represents the weight between a pair of connected neurons. Since fully connected layers are usually prone to overfitting, we also employ the dropout technique (Srivastava et al., 2014) in the fully connected layers. The probability of setting the output of a hidden neuron to zero is also configured as 50%. Besides, the size of the input images is set as $128 \times 128 \times 3$.

The training strategy for the CNN-based target classifier is to minimize the cross-entropy loss function representing the dissimilarity between the ground truth category labels and the predicted confidence scores (i.e., probabilities). Using the *one-hot* encoding scheme, we transform each category label into a $J \times 1$ vector of zeros while setting the j-th digit to be 1, where J is the number of categories including the non-target. Here, the categories of non-target, cyst and nauplius are represented by $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$, respectively. The predicted probability for the j-th category is obtained by a *softmax* function (Bishop, 2006), which is given by

$$z_i = \frac{e^{\varrho_i}}{\sum_{j=1}^{J} e^{\varrho_j}} , \tag{7.7}$$

where i and j denote the category indices, and ϱ_j stands for the output value of the last fully connected layer for the j-th category. Accordingly, the cross-entropy loss is calculated as

$$\mathcal{L}_{\text{ce}}(\mathbf{z}, \mathbf{t}) = -\sum_{j=1}^{J} \upsilon_j \log z_j , \tag{7.8}$$

where $\mathbf{z}$ and $\mathbf{t}$ represent the predicted probability vector and the ground truth label vector, respectively, υ_j is the indicator that the input image belongs to the j-th category, and z_j is the *softmax* output for the j-th category.

7.4. Experiments and results

We have proposed an automated *Artemia* detection and counting method using a UNet-based marker proposal network and a CNN-based target classifier (Marker-CNN method). Having presented the method and the compiled dataset, we train the Marker-CNN model, and, subsequently, evaluate its performance. For comparison, we also present a method using the marker-controlled watershed segmentation (MCWS method), a technique that has been widely employed to detect objects in microscope images (Koyuncu et al., 2016).

7.4.1. Training procedure

The whole model of the Marker-CNN method has been implemented in Python 3.5.6 running on a PC configured with Intel i7-6800K, 64GB RAM and GEFORCE GTX 1080Ti GPU. We train the UNet-based marker proposal network and the CNN-based target classifier separately.

In order to overcome the overfitting problem, besides the aforementioned dropout technique, we also employ the method of data augmentation (Ding et al., 2016). We artificially enlarge the training image dataset by means of scaling, rotation, translation and reflection. Note that no shearing transformation is used, since it would change the eccentricity values of the cyst objects. For the marker proposal network training data, we apply the same spatial transformation to both the image and the corresponding marker map to make the images and the label maps consistently matching. For illustration, we show the augmented images and the corresponding marker maps in Fig. 7.7. Likewise, we augment the training data for object classification while preserving the corresponding category labels.

When training the UNet-based marker proposal network, we initialize the weights in each layer by the method presented in (He et al., 2015). The size of the input images is transformed into $512 \times 512 \times 3$. The output is a 512×512 grayscale map. We use the Adaptive Moment Estimation method (Adam) (Kingma and Ba, 2015) to train the UNet model, setting the learning rate and the two decay rates as 10^{-4}, 0.9 and 0.999, respectively. As for the training batch size, we have separately trained the UNet with a batch size of 2, 4 and 8. All the settings lead to a good convergence of the training

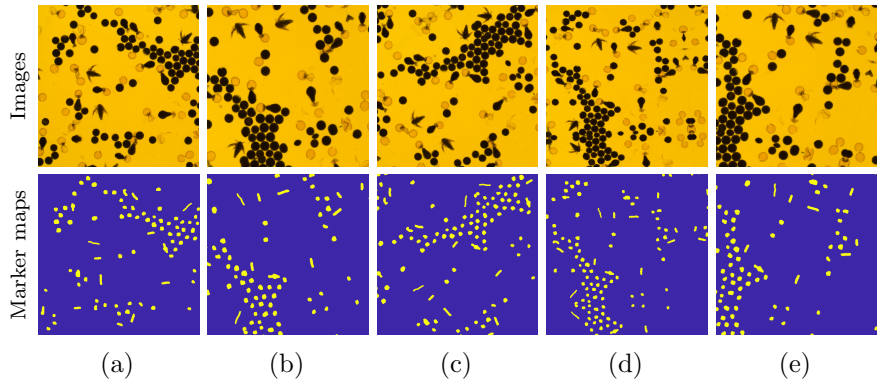


Figure 7.7: Illustration of the data augmentation results. (a) Original training image and the corresponding ground truth marker map; (b)-(e) The augmented training images and the corresponding ground truth marker maps.

loss. Here, we set the batch size as 4 and set the total number of epochs as 200. To show the evolution of the prediction ability of the marker proposal network during the training procedure, in Fig. 7.8, we display the prediction maps obtained on three sample images when the number of epochs is 5, 10, 50 and 200, respectively. It can be seen that the marker proposal network learns to segment the foreground object in the initial phase. As the number of epochs increases, the marker proposal network tends to yield large values at the locations in the vicinity of the trunk skeletons. Eventually, after 200 epochs, the marker proposal network is able to yield markers for the *Artemia* objects.

When training the CNN-based classifier, we also initialize the weights in each layer by the method presented in (He et al., 2015). The size of the input images is transformed into $128 \times 128 \times 3$. The number of output categories is three, including the non-target, cyst and nauplius. We still adopt the Adam optimizer (Kingma and Ba, 2015) to train the classifier, but setting the learning rate and two decay rates as 10^{-3}, 0.9 and 0.999, respectively. Besides, we have trained the network with a batch size of 8, 16, 32 and 64, respectively. These settings are all able to yield a good convergence of the classification accuracy. We set the batch size as 64 and set the total number of epochs as 200. Figure 7.9 shows the curves of the training accuracy and validation accuracy with respect to the number of epochs. It can be seen that after 200 epochs of training, the network has obtained a good classification accuracy. Here, the classification accuracy is computed as

$$\text{Acc} = \frac{N_{\text{cd}}}{N_{\text{ao}}}, \tag{7.9}$$

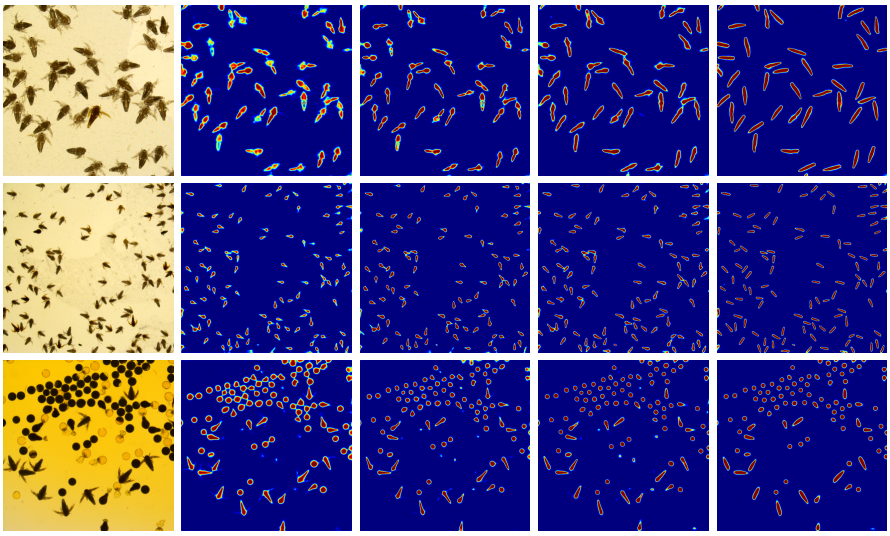


Figure 7.8: Evolution of the prediction maps yielded by the UNet as the number of epochs increases. (a) Original images; (b)-(e) The prediction maps that are obtained when the number of iterations is 5, 10, 50 and 200, respectively.

where N_{cd} denotes the number of correctly detected objects and N_{ao} represents the number of all the objects to be detected.

7.4.2. The watershed-based method for comparison

To highlight the superiority of the Marker-CNN method, we compare our method with a hand-crafted method based on the marker-controlled watershed technique (Rivest et al., 1992).

Watershed segmentation is a widely used image processing technique. Topographically, this technique views a grayscale image in three dimensions. It finds the watershed lines that separate the locations with locally minimum values into different connected components. In the marker-controlled watershed method, which was proposed to address the oversegmentation problem, the locations with locally minimum values that belong to the same object are connected by a so-called internal marker, while the locations with locally minimum values in the background are connected by external markers (González and Woods, 2010). In the MCWS method for *Artemia* detection and counting, the internal markers are obtained by a morphological erosion operation followed by a morphological reconstruction operation (Xu et al., 2011), while the external markers are obtained by skeletonizing the binarized background (Herbin et al.,

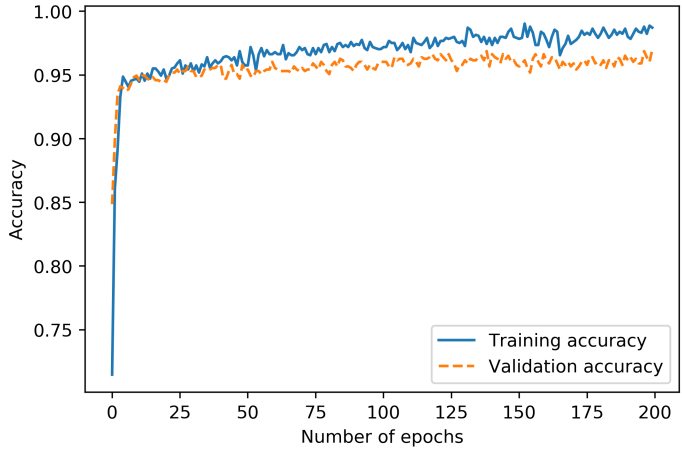


Figure 7.9: Training accuracy and validation accuracy of the target classifier with respect to the number of training epochs.

1996). Subsequently, we compute the gradient map of the image, and set the marker areas in the gradient map as local minima. Then, the watershed technique is employed to segment the foreground targets and to separate adjacent objects as much as possible. Eventually, the connected components obtained by the watershed transformation are classified into categories according to the degree of circularity (Kim and Cho, 2013), which is defined as

$$\textsc{Cir} = \frac{4\pi A_r}{P_r^2}\,, \tag{7.10}$$

where A_r and P_r denote the area and perimeter of the connected component, respectively. Empirically, a connected component with a degree of circularity that is lower than 0.70 is determined as a cyst. Otherwise, it is classified as a nauplius.

7.4.3. Performance evaluation

In order to evaluate the *Artemia* detection and counting performance of our method, we apply our method and the MCWS method to 80 real-life *Artemia* images, which contain 1336 cysts and 3335 nauplii in total. As shown in Fig. 7.10(a), there are quite a few cysts and nauplii that are seriously adjacent. There are also artefacts in the backgrounds.

With respect to the evaluation measures, we compute the detection and counting accuracy using Eq. (7.9). In addition, since the identification of each category can be considered as a binary classification procedure, we employ

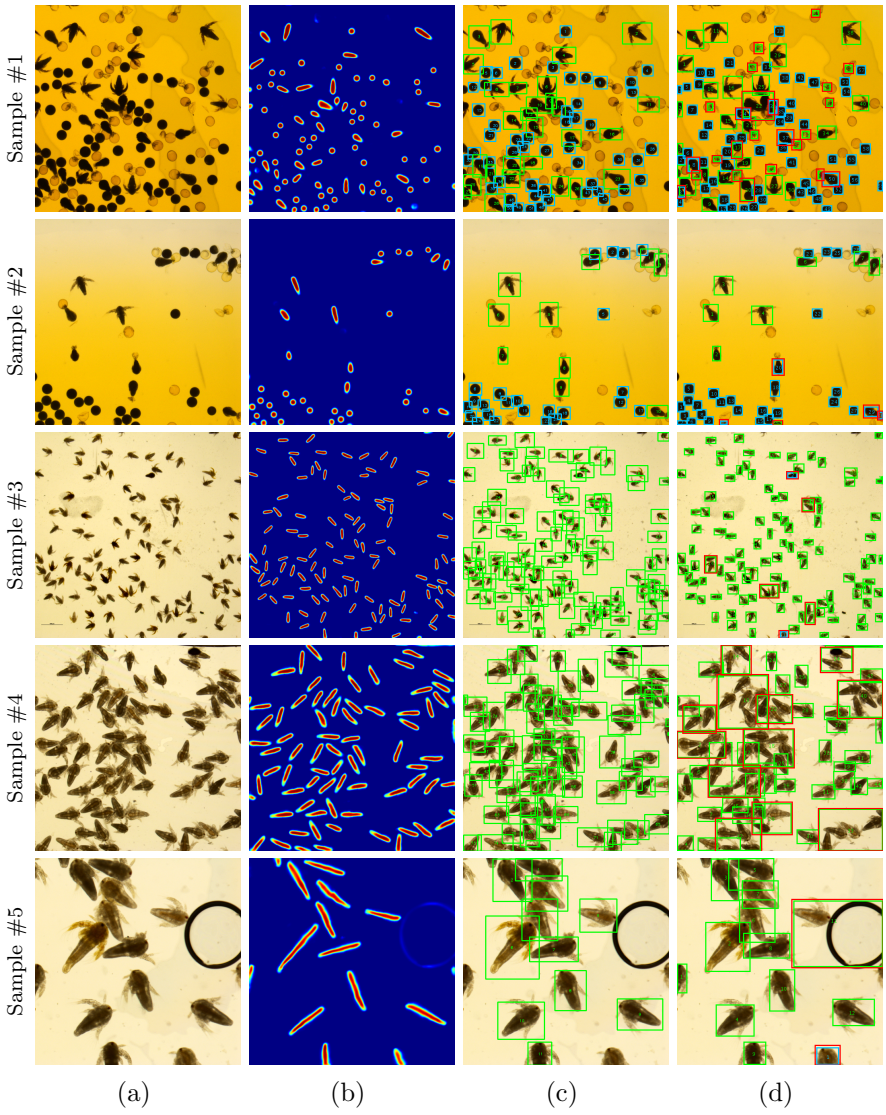


Figure 7.10: Illustration of the marker maps and detection results yielded by our method, and the detection results obtained by the MCWS method. (a) Original images; (b) Marker maps yielded by the marker proposal network; (c) Detection results obtained by the Marker-CNN method; (d) Detection results obtained by the MCWS method. The blue and green bounding boxes indicate the detection results of cysts and nauplii, respectively, while the red rectangles indicate the incorrect detection results.

Table 7.1: Evaluation results in terms of detection accuracy.

Methods	Subcategory		Overall
	Cyst	Nauplius	
MCWS	95.5%	85.4%	88.5%
Marker-CNN	99.9%	99.5%	99.6%

the precision and recall as evaluation measures, which have been given by Eqs. (6.27) and (6.28). Accordingly, we compute the harmonic mean of the precision and recall, thereby obtaining the F-measure.

The quantitative evaluation results are reported in Tabs. 7.1 and 7.2, while some visual detection results are displayed in Fig. 7.10. The MCWS method obtains an acceptable performance in detecting and counting *Artemia* cysts, but underperforms in detecting and counting *Artemia* nauplii. This is because the MCWS method mainly exploits low-level features. For the *Artemia* cysts that have a regular round shape, the MCWS method works well in object separation and recognition. Nevertheless, the MCWS method cannot semantically describe the appearance of *Artemia* nauplii, and therefore, it underperforms in detecting and counting the nauplius objects, especially the adjacent ones. As can be seen in Fig. 7.10(d), the MCWS method yields quite a few false detection results.

Comparatively, the Marker-CNN method achieves a much better performance in detecting and counting *Artemia* objects, regardless of a high degree of adjacency, variations of the object size and the interference from non-target objects in the background. This is because the Marker-CNN method benefits from both the marker proposal network and the target classifier. On the one hand, for a given image, the marker proposal module yields markers only for the *Artemia* objects, while suppressing the non-target artefacts and the background. The obtained markers localize potential targets, separate the adjacent objects and roughly indicate the sizes of the targets. We show the intermediate results in Fig. 7.10(b). On the other hand, the target candidates are further confirmed and classified by the CNN-based classifier. Furthermore, the final detection and counting results that are displayed in Fig. 7.10(c) validate the effectiveness of the Marker-CNN in detecting and counting *Artemia* objects.

With respect to time efficiency, for each method, we report its average runtime of processing a single image in Tab. 7.3. As can be seen, the Marker-CNN is quite efficient to run. With the help of GPU acceleration, our method processes each image with an average runtime of 0.16s, much faster than the

Table 7.2: Evaluation results in terms of precision, recall and F-measure.

Methods	Cyst			Nauplius		
	Prec	Rec	F	Prec	Rec	F
MCWS	0.890	0.955	0.922	0.864	0.854	0.859
Marker-CNN	0.994	0.999	0.996	0.999	0.995	0.997

MCWS method.

Table 7.3: Evaluation results in terms of average runtime (s).

Methods	MCWS	Marker-CNN
Runtime	1.62	0.16

7.5. Conclusions

The brine shrimp *Artemia* is an important organism in aquaculture, and the number of studies on *Artemia* is increasing. *Artemia* detection and counting is a fundamental task in *Artemia* image analysis. To facilitate this task, we have proposed an automated detection and counting method in this chapter. Our method consists of a UNet-based marker proposal network and a CNN-based target classifier, and we therefore term it as the Marker-CNN method. The marker proposal network introduces the image segmentation scheme into object detection. It can generate target candidates, separate highly adjacent objects and obtain the object structural information simultaneously. The target classifier determines the category of each target candidate, thereby yielding the *Artemia* detection and counting results. Moreover, we have compiled an *Artemia* detection and counting dataset to train and test the proposed method. Experimental results have manifested that the proposed Marker-CNN method can accurately detect and count the *Artemia* objects in images.

8 Automated *Artemia* length measurement

Length measurement plays an important role in many *Artemia* studies. In this chapter, we propose an automated length measurement method using UNet and NASAG kernels. The UNet model is used to extract the length measuring line structure, while NASAG kernels are employed to transform the length measuring line structure into a thin measuring line with minimal loss of the length measure. For comparison, we also follow traditional approaches and present a non-learning-based method using mathematical morphology and polynomial curve fitting. In the experiments, we evaluate the proposed method as well as the competing methods on the compiled *Artemia* length measurement dataset. The experimental results confirm that the proposed method can accurately measure the length of *Artemia* objects in images.

The material of this chapter is based on the following publication:

- Wang, G., Van Stappen, G., and De Baets, B. (2019d). Automated *artemia* length measurement using U-shaped fully convolutional networks and second-order anisotropic Gaussian kernels. *Computers and Electronics in Agriculture*, Accepted

8.1. Motivation

Image-based length measurement plays an indispensable role in many aquaculture or marine studies (Hao et al., 2015), such as species classification (Chuang et al., 2016), wild creature monitoring (Muñoz-Benavent et al., 2018), fishery surveillance (French et al., 2015), quality grading (Misimi et al., 2008), etc. These methods are generally divided into two categories: manual processing methods and automated methods. Manual processing methods are time-consuming and labor-intensive, and as such, they are less preferred than automated methods when having to process large volumes of imagery data (Meijering et al., 2016).

In literature, some automated fish length measurement methods have already been developed. A straightforward and simplified idea is to consider a fish as a rigid body. For example, the method proposed by Hsieh et al. (2011) determines the length measuring line by finding the straight line passing through the snout and the tail. Another example is the method proposed

in (Balaban et al., 2010), which measures the salmon fish length using a rectangular box fitting the fish body well (Lee et al., 2013). However, in many practical applications, these methods underperform when the fish body shows a curved appearance.

Comparatively, a more authentic scheme is to extract a curve along the lengthways centerline of the fish body as the length measuring line. Strachan (1993) designed an equipment consisting of a conveyor and a camera. For each fish in the image, the middle points of the body transverse lines are connected to form a length measuring line. This method has been successfully applied in fish sorting (Strachan, 1994) and species recognition (White et al., 2006). Nevertheless, using a polyline to approximate the length will inevitably incur a considerable measurement error. Huang et al. (2016) proposed a method that delineates the length measuring line based on a recursive morphological erosion process. Similarly, the method presented in (Saberioon and Císař, 2018) determines the length measuring line by applying the distance transform on the fish silhouette. This method represents the fish silhouette by zero and the background by non-zero. Subsequently, it uses the distance transform algorithm to compute the Euclidean distance between each zero pixel and its nearest non-zero pixel. The resulting image contains ridges and the ridge connecting the head point and the tail point is determined as the length measuring line. Besides, there are also methods that determine the length measuring line using morphological thinning or skeletonization (Han et al., 2009; Pan et al., 2009). However, these methods are based on the premise that the contour of the silhouette is smooth, and as such, these methods are vulnerable to structures such as fish fins. To overcome this shortcoming, the method in (Miranda and Romero, 2017) determines the length measuring line by fitting a polynomial curve using all the points within the fish silhouette. However, this method underperforms when the fish body shows a significant distortion.

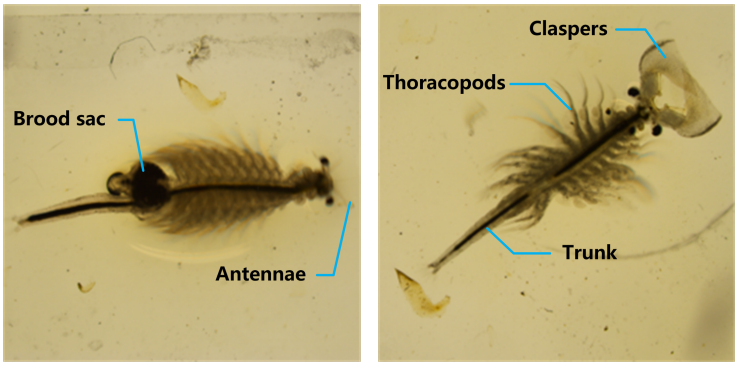
Not only in fish studies, but also in *Artemia* studies, an accurate length measurement is desired (Toi et al., 2013). We have introduced some biological background of *Artemia* in Chapter 7. In many *Artemia* studies, the length information is considered a key dependent variable or feature (Balachandar and Rajaram, 2019). For instance, in a controlled pond *Artemia* production, which has become an effective way to meet the growing demand of *Artemia* cysts supply, the individual *Artemia* length is usually adopted as a metric in evaluating the feeding strategies of intensive *Artemia* culture (Lopes-dos Santos et al., 2019). Despite an increasing demand for *Artemia* length measurement, to the best of our knowledge, the problem of automated *Artemia* length measurement has so far not been addressed in literature. Traditionally, the

length measuring line is delineated following a manual approach. As mentioned earlier, this method is time-consuming and labor-intensive, having limitations in processing large sets of imagery. Moreover, due to the distortion of non-rigid bodies, the variation over growth stages, the variation over species and the interference from the antennae and other appendages, existing methods that have been applied for fish length measurement cannot be readily transferred to measure the *Artemia* length. Therefore, it is desired to develop an automated *Artemia* length measurement method in the *Artemia* research field.

For automated *Artemia* length measurement, it is pivotal to determine the length measuring line semantically and accurately. Considering the length measuring line as a positive foreground, the task of measuring line extraction can be viewed as a semantic segmentation problem. Relying on their powerful feature learning and representation ability, approaches using deep convolutional neural networks have achieved considerable progress in the image segmentation filed. Representative examples include the method using deep convolutional neural networks (CNN) (Ciresan et al., 2012), the method using fully convolutional networks (FCN) (Long et al., 2015) and the method using U-shaped fully convolutional networks (UNet) (Ronneberger et al., 2015). We have introduced these networks in previous chapters.

In this chapter, aiming at measuring the length of *Artemia* in images, we propose a method to delineate the length measuring line, using UNet and normalized adaptive second-order anisotropic Gaussian (NASAG) kernels. The trained UNet model is used to extract a length measuring line structure, while the NASAG kernels are employed to transform the thick measuring line structure into a thin measuring line. For comparison, we also follow several existing fish length measurement approaches and design a method using mathematical morphology and polynomial curve fitting (MMPCF method). This method firstly extracts the principal lengthways morphological skeleton, and subsequently uses the points on the skeleton to fit a polynomial curve. The polynomial curve is then considered the length measuring line. Besides, we have compiled an *Artemia* length measurement dataset to train and test our method.

The remainder of this chapter is organized as follows. In Section 8.2, we present an *Artemia* length measurement dataset and an automated length measurement method, while in Section 8.3 we report and discuss the experimental results. Section 8.4 lists our conclusions.



(a) (b)

Figure 8.1: Illustration of an adult female *Artemia* (a) and an adult male *Artemia* (b).

8.2. Materials and methods

In this section, we firstly introduce the *Artemia* length measurement dataset containing 150 train images and 100 test images[1]. Subsequently, we propose a method for measuring the *Artemia* length in images.

8.2.1. The *Artemia* length measurement dataset

The length of *Artemia* is considered an important feature for indicating the growing status, and, as such, length measure data are highly desired in many *Artemia* studies (Lopes-dos Santos et al., 2019). Nonetheless, image-based *Artemia* length measurement is a challenging task, since the biological morphology of *Artemia* individuals varies significantly with the growth stage. As introduced in Chapter 7, an *Artemia* life cycle mainly includes the dormant cyst, nauplius/metanauplius, juvenile and adult stages. The whole life cycle may last several weeks. For the nauplii/metanauplii, juveniles and adults, automated *Artemia* length measurement would suffer from the interference of the antennae, thoracopods, claspers, brood sac, etc., as shown in Fig. 8.1. Therefore, automated *Artemia* length measurement requires not only low-level image features, but also semantic information. Aiming at developing an automated length measurement method, we compiled a set of images containing *Artemia Franciscana* individuals of different gender and in different growth stages (including nauplius/metanauplius, juvenile and adult).

For each *Artemia* individual to be measured, the full body was fixed

[1] https://github.com/GangWangUgent/Artemia_length_measurement/

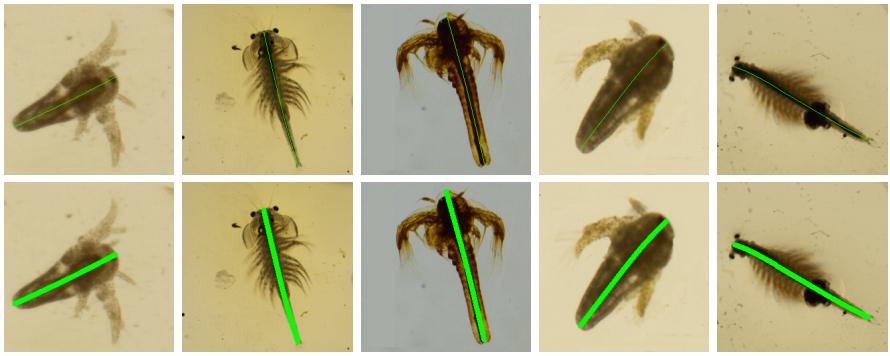


Figure 8.2: Sample label maps of length measuring lines (in green) superimposed on the original images. First row: Label maps of thin measuring lines. Second row: Label maps of thick length measuring lines.

with lugol solution (1%). Subsequently, the image was acquired by a stereomicroscope (SMZ1270, Nikon®) connected to a lens of Nikon® (Plan Apo). The relationship between pixels and the real-life length was obtained by the software NIS Elements (version 4.40, Nikon®).

We aim at accurate length measurements for *Artemia* individuals. However, the real-life length measures of *Artemia* individuals vary significantly with the growth stage. Therefore, in view of a normalized performance evaluation, we report the *Artemia* individual length in terms of pixels in this chapter.

We obtained 250 color images, including 85 juvenile/adult images and 165 nauplius/metanauplius images. Comparatively, we collected more nauplius/metanauplius images, because the appearance of the *Artemia* body varies more significantly during the nauplius/metanauplius stage. Subsequently, we transformed each image into a resized version that has a resolution of $256 \times 256 \times 3$. We split the 250 images into two subsets: a training subset (150 images) and a test subset (100 images).

To produce the label maps of length measuring lines, we manually delineated the length measuring line for each *Artemia* individual. The *Artemia* length is considered the length of the measuring line, as shown in Fig. 8.2. Moreover, for each *Artemia* image, besides a label map of the thin (one-pixel in width) measuring line, we also delineated a thick measuring line structure, as shown in Fig. 8.2. The reason is that thin measuring lines are inappropriate for training an image segmentation model (Sironi et al., 2016). As a pixel on a length measuring line is visually similar to its neighboring pixels, it is difficult to accurately segment a one-pixel wide line from an *Artemia* body. We will also address this in terms of the experimental results in Section 8.3.1.

8.2.2. Automated *Artemia* length measurement using U-shaped fully convolutional networks

Next, we design a UNet model for extracting a length measuring line structure from an image. Subsequently, we transform the length measuring line structure into a thin measuring line using NASAG kernels.

The architecture of the designed UNet model

The architecture of the UNet model we designed is displayed in Fig. 8.3. As can be seen, the overall model is similar to the UNet used in Chapter 7. The contracting path is built on a succession of convolutional layers and *max pooling* layers (Krizhevsky et al., 2012). In each convolutional layer, the number of kernels is configured as reported in Fig. 8.3. In addition, we also employ a replication padding to make the sizes of the input map and the response map identical. The rectified linear unit (Hinton, 2010) is selected as the activation function. The size of the window and the sliding stride are set to be 2×2 and 2, respectively. The UNet model we designed consists of five contracting steps. Furthermore, the so-called *dropout* technique (Srivastava et al., 2014) is also used to address the problem of overfitting. In the designed UNet model, the dropout technique is used for the layers illustrated in Fig. 8.3.

The expanding path is built on a succession of convolutional layers and up-convolution layers. The convolutional layers work in the same way as those in the contracting path. The up-convolution layers double the size of each input map. In particular, in order to better exploit the localization information, the feature maps yielded in the contracting path are concatenated with the corresponding feature maps in the expanding layers, as illustrated in Fig. 8.3.

At the end, the prediction map $\mathbf{P}$ is obtained by a sigmoid function as formulated in Eq. 7.3.

The UNet training process

A UNet model usually contains a high number of parameters. To prevent the UNet model from overfitting, a large amount of training data is required. Since there are only 150 images for training, we augmented the training data by means of scaling, rotation, translation, shearing and reflection. To make the images and the label maps consistently matching, we applied the same spatial transformation to the image and the corresponding label map. In this way, we obtained 3000 augmented images and corresponding label maps.

We train the UNet by minimizing the loss function that represents the

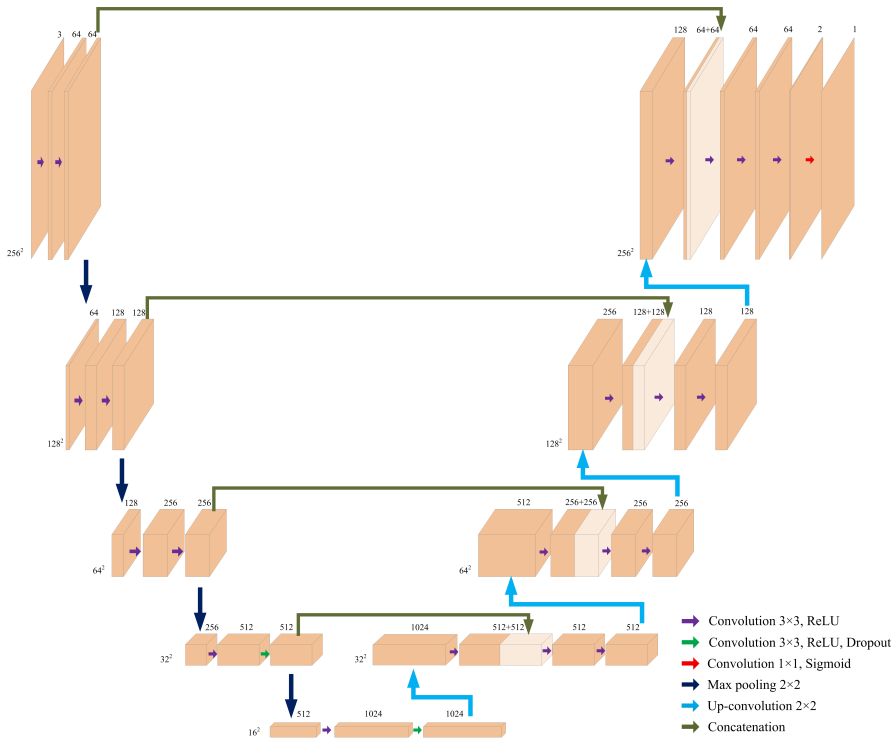


Figure 8.3: Architecture of the UNet for centerline area segmentation.

difference between the ground truth label map and the yielded prediction map. Since the segmentation in our method is a binarization problem, we define the loss function based on pixel-weighted cross-entropy. We use the adaptive moment estimation (Adam) method to train the UNet model (Kingma and Ba, 2015). The learning rate and the two decay rates are set as 10^{-4}, 0.9 and 0.999, respectively.

The weights in each layer are initialized by the method presented in (He et al., 2015). As for the training batch size (BS), we have separately trained the UNet with BS = 4 and BS = 8. Both settings lead to a good convergence, as shown in Fig. 8.4. In our method, we set the batch size as BS = 4 and the total number of epochs as 200.

To present the evolution of the prediction ability during the training procedure, we display in Fig. 8.5 the prediction map obtained on a sample test image when the number of epochs is 5, 20, 50, 100 and 200, respectively. The UNet model learns to segment the foreground object in the initial phase. As the number of epochs increases, the UNet model tends to yield large values at the locations of the lengthways central region. Eventually, after 200 epochs,

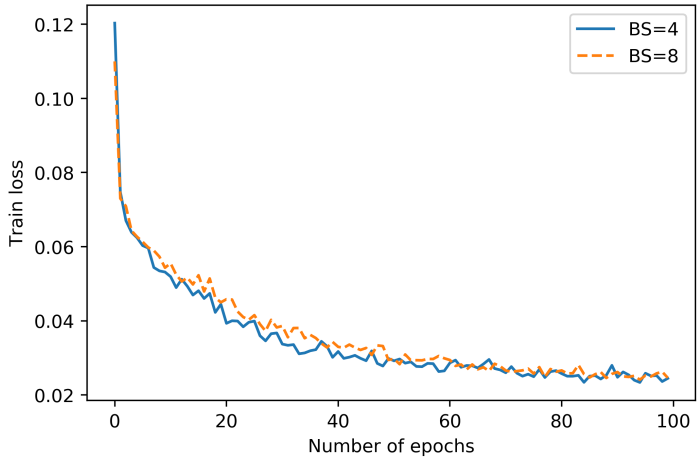


Figure 8.4: Curves of train losses obtained by two different settings of batch size.

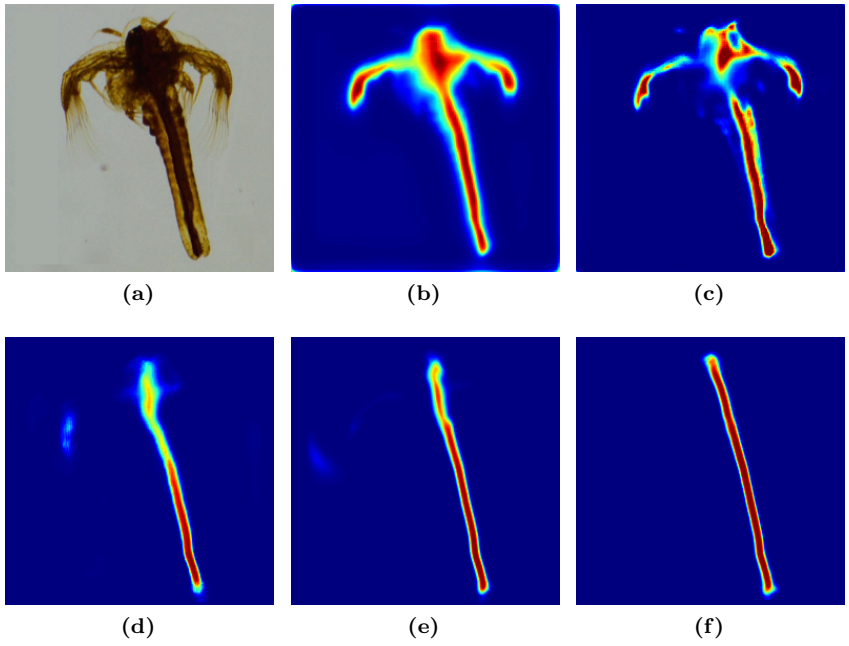


Figure 8.5: Evolution of the prediction map as the number of epochs increases. (a) Original image; (b)-(f) The prediction map that is obtained when the number of iterations is 5, 20, 50, 100 and 200, respectively.

the UNet model is able to segment the length measuring line structure for a given *Artemia* object.

The UNet model has been implemented and trained on Python 3.5.6 running on a PC configured with Intel i7-6800K, 64GB RAM and GEFORCE GTX 1080Ti GPU.

Obtaining the thin length measuring line

The prediction map yielded by the UNet shows a thick measuring line structure, as displayed in Fig. 8.5(f). It is required to transform the length measuring line structure into a thin measuring line. A straightforward method is to binarize the prediction map (Otsu, 1979) and subsequently employ a morphological skeletonization process (Herbin et al., 1996; Kong and Rosenfeld, 1996; González and Woods, 2010) to obtain the thin skeleton, e.g., the length measuring line. However, this will lead to a loss of length measure at the endpoints of the line. We desire a method that can obtain the thin centerline with minimal loss of length measure.

In literature, NASAG kernels have already been used for line detection (Wang et al., 2019c). They can delineate the thin centerline of a thick curvilinear structure that is heterogeneous in terms of width, direction and prominence. Therefore, we here employ NASAG kernels to extract the thin measuring line from the prediction map yielded by the UNet.

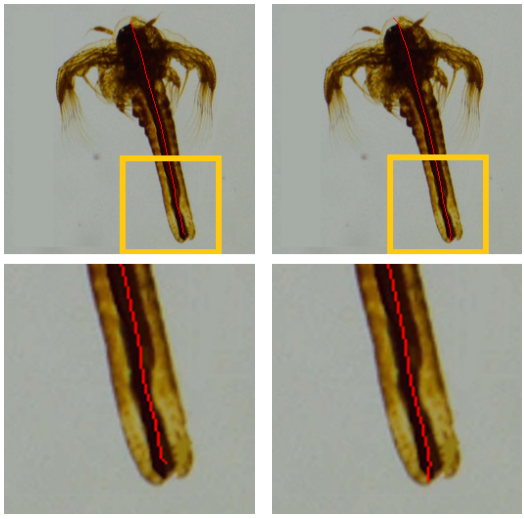
Having the prediction map $\mathbf{P}$ yielded by the UNet, we obtain the final response of NASAG kernels as follows:

$$\mathbf{L}_{\mathrm{l}}(\mathbf{m}) = \max_{\sigma_i \in \mathbb{S}} \max_{\theta_j \in \mathbb{D}} g''_{\mathrm{L}}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j) * \mathbf{P}(\mathbf{m}) \,, \tag{8.1}$$

where $g''_{\mathrm{L}}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j)$ denotes the discrete version of the NASAG kernel. Moreover, the direction map $\mathbf{\Theta}_{\mathrm{l}}(\mathbf{m})$ is obtained as:

$$\mathbf{\Theta}_{\mathrm{l}}(\mathbf{m}) = \underset{\theta_j \in \mathbb{D}}{\operatorname{argmax}} \max_{\sigma_i \in \mathbb{S}} g''_{\mathrm{L}}(\mathbf{m}; \sigma_i, \varphi_i, \theta_j) * \mathbf{P}(\mathbf{m}) \,. \tag{8.2}$$

Subsequently, based on $\mathbf{L}_{\mathrm{l}}(\mathbf{m})$ and $\mathbf{\Theta}_{\mathrm{l}}(\mathbf{m})$, we use the NMS technique (Rosenfeld et al., 1972) to obtain the thin measuring line. For each pixel, if the value of $\mathbf{L}_{\mathrm{l}}(\mathbf{m})$ at the current pixel position $\mathbf{m}$ is the largest value compared to the other pixels along the direction $\mathbf{\Theta}_{\mathrm{l}}(\mathbf{m})$, the value will be retained. Otherwise, it will be nullified. The result of applying the NMS technique to Fig. 8.5(f) is shown in Fig. 8.6(b). For comparison, we also display the result obtained by the morphological skeletonization (MS) method in Fig. 8.6(a). As can be seen, the MS method leads to a loss of length measure at the endpoints of the length measuring line, while the NASAG method yields a length measuring line with minimal loss of length measure.



(a) **(b)**

Figure 8.6: The results of measuring line extraction (red lines superimposed on the original image) obtained by the MS method (a) and the NASAG method (b) on the image shown in Fig. 8.5(f). For a better demonstration, a zoom-in view of the regions of interest (indicated by windows) is displayed in the second row.

8.2.3. A method using mathematical morphology and polynomial curve fitting

For comparison, we also follow several existing fish length measurement approaches (Huang et al., 2016; Miranda and Romero, 2017), and design a method using mathematical morphology and polynomial curve fitting. This method is based on the premise that the silhouette of the object to be measured is usually symmetric. Specifically, we implement the MMPCF method in three steps: silhouette segmentation, silhouette alignment and measuring line fitting. For a better presentation, we elaborate our method by processing an example *Artemia* image displayed in Fig. 8.7(a).

Silhouette segmentation and alignment

It is observed that the *Artemia* bodies are significantly different from the background. Therefore, in most cases, the foreground silhouette can be extracted using the Otsu thresholding technique (Otsu, 1979) on the grayscale version of the original image. For illustration, Fig. 8.7(b) shows the grayscale image and Fig. 8.7(c) displays the segmented silhouette.

The *Artemia* in the captured image appears along a random direction,

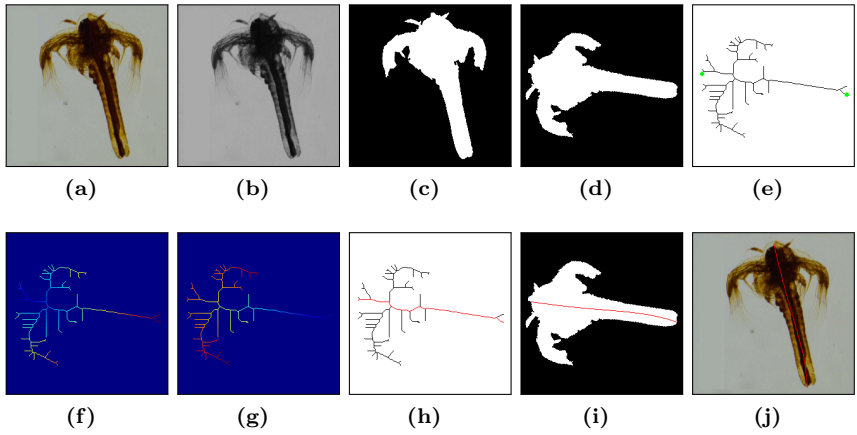


Figure 8.7: Demonstration of the MMPCF method. (a) The original image; (b) The grayscale version; (c) The foreground silhouette; (d) The horizontally aligned silhouette; (e) The result of skeletonization, the starting point and the endpoint (in green); (f) The result of geodesic distance transform, with the starting point as the seed location; (g) The result of geodesic distance transform, with the endpoint as the seed location; (h) The principal morphological skeleton (in red); (i) The fitted polynomial curve overlaid on the aligned silhouette; (j) The length measuring line overlaid on the original image.

which will hinder a subsequent procedure of polynomial curve fitting. To align the silhouette horizontally, we determine the principal direction of the silhouette based on the image moments, which have been extensively used in many computer vision systems (Karakasis et al., 2015). For a given binary image $\mathbf{K}$, the ij-th order image moment is computed as follows (Prokop and Reeves, 1992):

$$M_{ij} = \sum_{\mathbf{m} \in \mathbf{K}} m_x^i m_y^j \cdot \mathbf{K}(\mathbf{m}) \,, \tag{8.3}$$

where $[m_x, m_y]^{\mathrm{T}}$ denotes the image coordinates. Based on the image moments, we are able to compute an equivalent ellipse corresponding to the silhouette, the covariance matrix $\mathbf{V}$ of which is computed as follows:

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_{1,1} & \mathbf{V}_{1,2} \\ \mathbf{V}_{2,1} & \mathbf{V}_{2,2} \end{bmatrix} \tag{8.4}$$

where

$$\mathbf{V}_{1,1} = \frac{M_{00} M_{20} - M_{10}^2}{M_{00}^2} \,,$$

$$\mathbf{V}_{1,2} = \mathbf{V}_{2,1} = \frac{M_{00}M_{11} - M_{10}M_{10}}{M_{00}^2}\,,$$
$$\mathbf{V}_{2,2} = \frac{M_{00}M_{02} - M_{01}^2}{M_{00}^2}\,. \tag{8.5}$$

According to the covariance matrix, the principal direction of the main body is computed as

$$\tilde{\vartheta} = \frac{1}{2}\arctan\frac{2\cdot\mathbf{V}_{1,2}}{\mathbf{V}_{1,1} - \mathbf{V}_{2,2}}\,. \tag{8.6}$$

In addition, we compute the standard deviations ζ_{h} and ζ_{l} of the locations within $\mathbf{K}$ along the horizontal and longitudinal axis, respectively. Since the standard deviation along the lengthways direction is larger than that along the transverse direction, the principal direction in Eq. (8.6) is modified as

$$\hat{\vartheta} = \begin{cases} \tilde{\vartheta} - \dfrac{\pi}{2}|\tilde{\vartheta}| & \text{, if } \zeta_{\mathrm{h}} < \zeta_{\mathrm{l}} \\ \tilde{\vartheta} & \text{, otherwise} \end{cases}\,. \tag{8.7}$$

Consequently, using the principal direction $\hat{\vartheta}$, we obtain the horizontally aligned *Artemia* silhouette, which is denoted by $\mathbf{K}_{\mathrm{h}}$. As an illustration, the $\mathbf{K}_{\mathrm{h}}$ shown in Fig. 8.7(d) is the result of rotating $\mathbf{K}$ by $\hat{\vartheta}$.

Polynomial curve fitting

Having the aligned *Artemia* silhouette, we next describe the approach to delineate the length measuring line. As mentioned earlier, some authors have proposed methods for extracting the lengthways centerline of a fish body, which is considered an approximation of the length measuring line. However, these existing methods cannot be readily applied to *Artemia* length measurement, due to the serious distortion of non-rigid bodies, the variations over growth stages and the interference from the antennae, telopodite and endopodite, etc. In our method, we determine the length measuring line of the *Artemia* silhouette as follows.

We apply a morphological opening operation (González and Woods, 2010) to $\mathbf{K}_{\mathrm{h}}$ to remove the antenna areas. Subsequently, we select the rightmost and leftmost pixels as the starting point and endpoint, respectively. Then, to find more points that are close to the lengthways centerline of the full silhouette, we compute the skeleton map $\mathbf{K}_{\mathrm{sk}}$ of $\mathbf{K}_{\mathrm{h}}$ as follows:

$$\mathbf{K}_{\mathrm{sk}} = \bigcup_{j=0}^{N_{\mathrm{sk}}} (\mathbf{K}_{\mathrm{h}} \ominus \kappa_{\mathrm{se}})^{(j)} \setminus \left[(\mathbf{K}_{\mathrm{h}} \ominus \kappa_{\mathrm{se}})^{(j+1)} \oplus \kappa_{\mathrm{se}}\right]\,, \tag{8.8}$$

where κ_{se} denotes the disk structuring element with a radius of 2, j stands for the j-th successive operation and

$$N_{\mathrm{sk}} = \max\left\{ j \,|\, (\mathbf{K}_{\mathrm{h}} \ominus \kappa_{\mathrm{se}})^{(j)} \neq \varnothing \right\} . \tag{8.9}$$

For demonstration, in Fig. 8.7(e), we show the skeleton map $\mathbf{K}_{\mathrm{sk}}$ obtained on Fig. 8.7(d). In $\mathbf{K}_{\mathrm{sk}}$, we determine the principal morphological skeleton by finding the shortest skeleton passing through both the starting point and the endpoint. Setting the starting point or the endpoint as the seed location, we compute two maps of the geodesic distance transform (Cárdenes et al., 2010) on $\mathbf{K}_{\mathrm{sk}}$, as illustrated in Figs. 8.7(f) and 8.7(g). In the sum of the two geodesic distance transform maps, the principal morphological skeleton is identified as the set of locations at which the intensities are regional minima, as displayed in Fig. 8.7(h).

Having the points that are close to the lengthways centerline of the silhouette, we fit a four-order polynomial curve in a least-squares sense, as illustrated in Fig. 8.7(i). Furthermore, to reduce the error brought by the rotation procedure, we rotate the obtained polynomial curve by -$\hat{\vartheta}$. Eventually, we obtain the length measuring line of the *Artemia* silhouette, as shown in Fig. 8.7(j).

8.3. Experiments and results

In the previous section, we have proposed an automated method to measure the *Artemia* length in images using UNet and NASAG kernels (UNet+NASAG method). In this section, we apply the UNet+NASAG method to real *Artemia* images to evaluate its performance.

8.3.1. Comparison between two models trained by different types of label maps

The UNet model for extracting the length measuring line structure plays a pivotal role in our method. In Section 8.2.2, we have trained a UNet model by label maps of thick measuring lines. Alternatively, it is also possible to train the UNet using label maps of thin measuring lines. In literature, the method in (Shen et al., 2017) trains a model for skeleton extraction using label maps of thin skeletons. To compare the performance of the two schemes, we separately train a UNet model using label maps of thin measuring lines ($\mathrm{UNet}^{\mathrm{tn}}$) and a UNet model using label maps of thick measuring lines ($\mathrm{UNet}^{\mathrm{tk}}$). For a fair comparison, the parameters in the $\mathrm{UNet}^{\mathrm{tn}}$ and $\mathrm{UNet}^{\mathrm{tk}}$ are set the same.

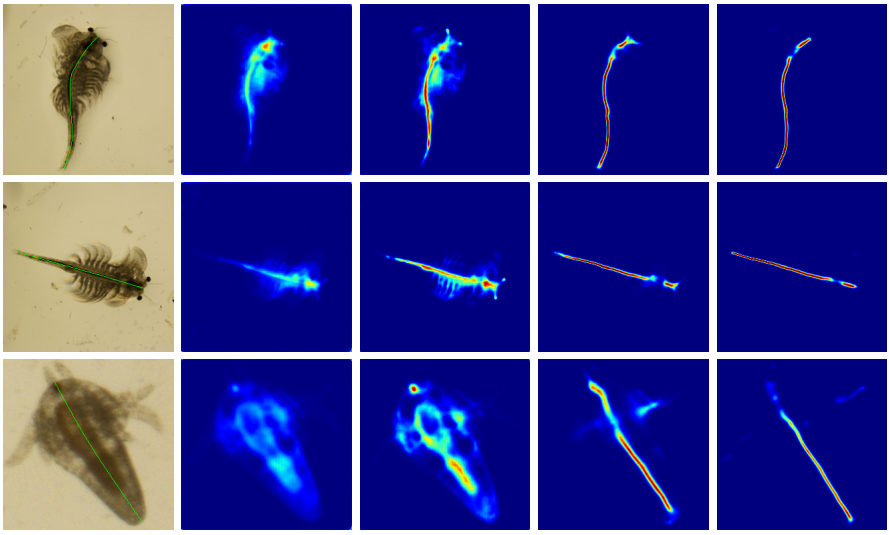


Figure 8.8: Sample images and the prediction maps yielded by the UNet$^{\mathrm{tn}}$. First column: The label maps of thin measuring lines superimposed on the original image. Second column to fifth column: The prediction maps obtained on the sample images when the number of epochs is 5, 10, 50 and 200, respectively.

In Fig. 8.8, we display samples of prediction maps yielded by the UNet$^{\mathrm{tn}}$, while in Fig. 8.9, we show samples of prediction maps yielded by the UNet$^{\mathrm{tk}}$. It can be seen that the UNet$^{\mathrm{tn}}$ sometimes yields a prediction map in which the length measuring line structures are disconnected. Even worse, the UNet$^{\mathrm{tn}}$ sometimes fails to yield a full measuring line structure. In contrast, the UNet$^{\mathrm{tk}}$ successfully yields the length measuring line structure for all the three sample images.

8.3.2. Length measurement evaluation

We apply our method to 100 real-life *Artemia* images, the GT length measuring lines of which have been labelled manually. Several example images as well as their GT label maps are displayed in Fig. 8.2. The *Artemia* objects in these images are in different growth stages and are situated in inhomogeneous backgrounds. Some *Artemia* objects have seriously distorted shapes. To highlight the role of the NASAG kernels in our method, we also compare the proposed UNet+NASAG method with the method that uses the morphological skeletonization technique (González and Woods, 2010) to transform the prediction map of the UNet into a thin measuring line (UNet+MS method).

In addition, we compare our method with two existing methods that are originally designed for fish length measurement, including the method that

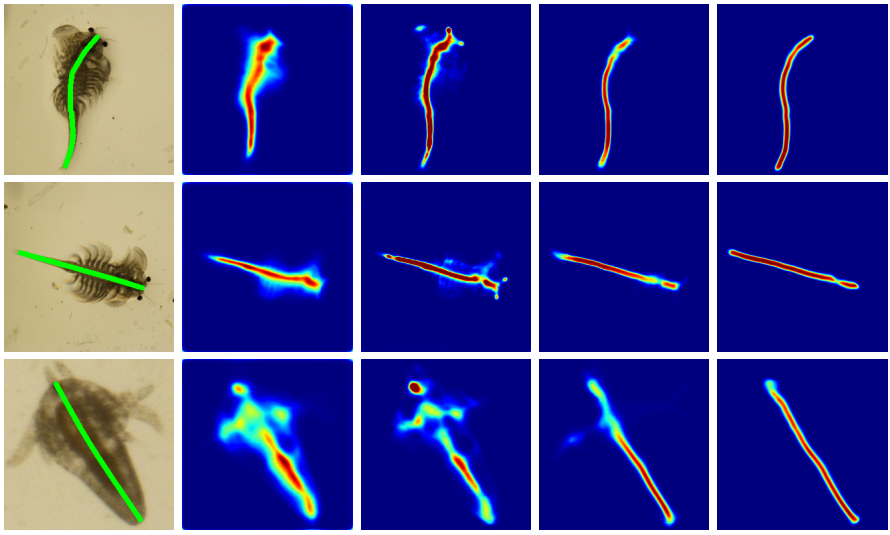


Figure 8.9: Sample images and the prediction maps yielded by the UNet$^{\mathrm{tk}}$. First column: The label maps of thick measuring lines (in green) superimposed on the original image. Second column to fifth column: The prediction maps obtained on the sample images when the number of epochs is 5, 10, 50 and 200, respectively.

uses the mid-line points yielded by a recursive erosion (MPRE method) (Huang et al., 2016) and the method employing a polynomial curve fitting process on the silhouette (PCFS method) (Miranda and Romero, 2017).

To obtain quantitative evaluation results, we adopt three widely used evaluation measures. The first one is the root mean square error (RMSE), which measures the differences between the estimated values and the true values as follows:

$$\textsc{Rmse} = \sqrt{\frac{1}{N_{\mathrm{im}}} \sum_{i=1}^{N_{\mathrm{im}}} \left(Z_{\mathrm{e}}^{(i)} - Z_{\mathrm{t}}^{(i)} \right)^2}, \tag{8.10}$$

where $Z_{\mathrm{e}}^{(i)}$ and $Z_{\mathrm{t}}^{(i)}$ denote the i-th estimated length and the i-th true length, respectively, and N_{im} represents the number of images in the dataset. The second evaluation measure is the mean absolute error (MAE), which represents the average absolute difference between the estimated length values and the true length values, as follows:

$$\textsc{Mae} = \frac{1}{N_{\mathrm{im}}} \sum_{i=1}^{N_{\mathrm{im}}} \left| Z_{\mathrm{e}}^{(i)} - Z_{\mathrm{t}}^{(i)} \right| . \tag{8.11}$$

Table 8.1: Quantitative evaluation results in terms of RMSE, MAE and MAPE obtained by different methods.

Methods	RMSE (pixels)	MAE (pixels)	MAPE (%)
MPRE	41.45	25.96	12.02
PCFS	27.69	14.98	6.91
MMPCF	13.89	7.27	3.11
UNet+MS	11.53	10.57	4.76
UNet+NASAG	**3.77**	**2.67**	**1.16**

We also adopt the mean absolute percentage error (MAPE) as an evaluation measure, which is defined as:

$$\text{MAPE} = \frac{1}{N_{\text{im}}} \sum_{i=1}^{N_{\text{im}}} \frac{\left| Z_{\text{e}}^{(i)} - Z_{\text{t}}^{(i)} \right|}{Z_{\text{t}}^{(i)}} \times 100\% \,. \tag{8.12}$$

The quantitative evaluation results are reported in Tab 8.1. The MPRE and PCFS methods, which work well for fish length measurement, underperform for *Artemia* length measurement. This is mainly because the two methods are vulnerable to the structures of antenna, telopodite, endopodite, etc. In addition, although the designed MMPCF method obtains an acceptable performance in terms of the MAE and MAPE, its RMSE value is fairly large, which indicates that the performance is unstable. Comparatively, the UNet+MS and UNet+NASAG methods obtain a more stable performance. In particular, among all the methods, the UNet+NASAG method obtains the best performance in terms of the three evaluation measures.

In Fig. 8.10, we also show the length measuring line extraction results of different methods obtained on sample images. As can be seen, the methods based on mathematical morphology operations, including the MPRE, PCFS and MMPCF methods, are able to tackle the object with a symmetric silhouette. However, they usually underperform when processing images like Sample #4 and Sample #5, the objects in which have asymmetric silhouettes. In contrast, the UNet+MS and UNet+NASAG methods work well for measuring the objects that have asymmetric silhouettes. Furthermore, it can be seen that the UNet+MS method leads to a loss of length measure in the morphological skeletonization procedure, while the UNet+NASAG method measures the *Artemia* length accurately. This manifests that the employed NASAG kernels transform the prediction map yielded by the UNet into a thin measuring line with minimal loss of length measure.

Figure 8.10: Comparison of different methods for measuring line extraction. First row: The label maps of measuring lines (in green) superimposed on the sample images. Second row to bottom row: The results of length measuring line extraction (in red) obtained by different methods.

8.4. Conclusions

The brine shrimp *Artemia* is widely used as a live food for fish and shellfish larviculture, and the *Artemia* length measurement plays an important role in many *Artemia* studies. Aiming at automated *Artemia* length measurement, we have compiled a dataset that contains 250 *Artemia* images and the ground truth maps of length measuring lines. We have proposed an automated length measurement method using UNet and NASAG kernels. In this method, the trained UNet model is used to extract the length measuring line structure, while the NASAG kernels are employed to transform the length measuring line structure into a thin measuring line with minimal loss of length measure. For comparison, we have also followed traditional approaches and presented a non-learning-based method using mathematical morphology and polynomial curve fitting. In the experiments, we have evaluated the proposed method as well as the competing methods on the *Artemia* length measurement dataset. The experimental results have confirmed that the proposed method can accurately measure the length of *Artemia* objects in images.

PART IV

EPILOGUE

9 Conclusions and future work

With the increasing use of imaging systems in scientific studies, automated image analysis has attracted much attention over the past decades, due to its superiority in processing efficiency and assessment objectivity over manual image analysis. Aiming at more automated image analysis techniques, in this thesis, we have investigated several Gaussian-based convolutional kernels, and accordingly, have proposed several methods to accomplish particular tasks. In addition, to facilitate *Artemia* studies, we have also developed two automated methods for analyzing *Artemia* images. Below, we summarize the main conclusions that can be drawn from this thesis, and, subsequently, list several pending issues that might be included in future work.

9.1. Conclusions

In this thesis, we have investigated three Gaussian-based convolutional kernels (i.e., the NFAG kernel, NASAG kernel and USG kernel) and have proposed a novel filtering method (i.e., iLoG filtering). In addition, to facilitate *Artemia* studies, we have developed two automated methods for analyzing *Artemia* images: an automated *Artemia* detection and counting method and an automated *Artemia* length measurement method. All of the proposed methods have been validated on in-house datasets.

In Chapter 3, we have proposed a multiscale version of NFAG kernels. These kernels are able to quantitatively measure the edge strength, edge direction and edge scale simultaneously, while reducing the impact of noise. They are also quite reliable in detecting edges with heterogeneous widths. In addition, building on these kernels, we have also proposed a contour detection method. In this method, the anisotropic edge strength is computed using NFAG kernels. Using a hierarchical set of superpixel maps, the superpixel contrast maps at different hierarchy levels are also computed. Subsequently, the contour strength map is obtained as the product of the anisotropic edge strength map and the average of the hierarchical superpixel contrast maps. The experimental results obtained on widely used datasets have validated the superiority of our contour detection method over the competing methods. For further expanding the use of NFAG kernels, we have also presented a superpixel segmentation method using NFAG kernels. The NFAG-based anisotropic edge strength is incorporated into a distance measure, which computes the dissimilarity between two neighboring superpixels by combining the edge strength and

the color distance. In this way, we have modified an existing superpixel segmentation method. Experimental results on three datasets have manifested that our superpixel segmentation method is less dataset-dependent than the competing methods. Furthermore, it has also been illustrated that our method can facilitate a subsequent saliency detection task.

In Chapter 4, we have studied and presented a line detection method based on NASAG kernels. This method is reliable to detect lines that are heterogeneous in width and prominence. Moreover, this method can be easily adapted to noisy environments. A scale-adaptive anisotropy factor has also been designed to attenuate the anisotropy stretch effect. Experimental results on a publicly available biological image dataset as well as a noisy version thereof have demonstrated that compared with the selected competing methods, our line detection method can significantly improve the robustness to noise while consuming an acceptable execution time.

In Chapter 5, we have proposed the USG kernel, and accordingly, have presented a method to quantitatively measure the blob characteristics. This method not only identifies the blob position, prominence and scale, but also suppresses non-blob structures well, and thus, this method can facilitate the implementation of blob reconstruction and blob reduction. Moreover, to tackle the blob-like noise that occurs in high-ISO long-exposure images, we have developed a denoising scheme by employing a blob reduction procedure for each of the selected conventional denoising methods. The experimental results have demonstrated that in high-ISO long-exposure image denoising, the methods incorporating blob reduction outperform the original conventional methods.

In Chapter 6, in order to accurately detect overlapping blob objects, we have presented a novel blob detection method based on iLoG filtering and USG kernels. The iteration of the Laplacian of Gaussian reduces the degree of overlap, facilitating a subsequent blob extraction procedure. The USG kernels yield responses only for blob objects, and as such, the blob objects can be pinpointed by a thresholding step. The experimental results have demonstrated that the proposed method shows a promising performance in detecting fluorescence microscopy cells and electron micrograph nanoparticles, even when there is a high degree of overlap.

In Chapter 7, we have addressed the issue of automated *Artemia* detection and counting, which is highly desired in many *Artemia* studies. We have proposed a method consisting of a UNet-based marker proposal network and a CNN-based target classifier, which is therefore termed as the Marker-CNN method. The marker proposal network introduces the image segmentation scheme into object detection. This network can generate target candidates,

separate seriously adjacent objects and obtain the object structural information simultaneously. The target classifier determines the category of each target candidate, thereby yielding the *Artemia* detection and counting results. Moreover, we have collected an *Artemia* detection and counting dataset to train and test the proposed method. Experimental results have manifested that the proposed Marker-CNN method can accurately detect and count the *Artemia* objects in images.

In Chapter 8, aiming at an accurate *Artemia* length measurement in images, we have proposed an automated method using UNet and NASAG kernels. In this method, the UNet model is used to extract the length measuring line structure, while the NASAG kernels are employed to transform the length measuring line structure into a thin measuring line with minimal loss of length measure. In addition, to train and test the proposed method, we have collected an *Artemia* length measurement dataset. For comparison, we have also followed traditional approaches and have presented a method based on mathematical morphology and polynomial curve fitting. In the experiments, we have evaluated the proposed method as well as the competing methods on the collected *Artemia* length measurement dataset. The experimental results have confirmed that the proposed method can accurately measure the length of *Artemia* objects in images.

9.2. Potential research directions

In this thesis, we have studied and proposed several automated image analysis methods using hand-crafted features and deep learning techniques. Besides the contributions to solving particular problems, our studies have indicated several potential research directions, which might be included in future work.

9.2.1. Texture suppression using superpixels for contour detection

Contour detection is a fundamental problem in image analysis and a great many of contour detection methods have been developed. In terms of contour detection performance, there is still a significant gap between hand-crafted methods and human abilities. The interference in contour detection is mainly from textures, since textural regions also show intensity/color discontinuities. In literature, several efforts have been made for texture suppression. Some methods suppress textures by simulating the mechanism of human visual systems (Yang et al., 2014; Akbarinia and Parraga, 2018), but these methods usually entail a heavy computation. Another kind of methods estimate the texture strength using spatial sparseness measurement (Alpert et al., 2012;

Yang et al., 2015a), and then, use the texture strength map to suppress textural edges. It has been validated that contour detection can benefit from a sparseness-constraint. Nevertheless, existing methods compute the spatial sparseness measure using a regular window, which makes the measurement not content-aware and shape-adaptive. Actually, contours are the borders between the objects and the background or between different objects, and thus, contours are highly related to image contents and object shapes. In this thesis, we have presented a contour detection method that exploits shape information by superpixels. Despite its good properties, our method still has limitations in exploiting textural features. One possible way to improve our method is to modify the superpixel contrast measure. Currently, we obtain superpixel contrast maps using the mean color of each superpixel. Despite its good time efficiency, this approach does not exploit textural information well. Therefore, in a modified version, we could incorporate texture dissimilarity into the superpixel contrast measure. Another possible approach to improve the current method is to incorporate a spatial sparseness measurement. A map of spatial sparseness measurement reflects the intensity of textures, and thus, we could employ it to further suppress the textural responses in the contour strength map that is yielded by our current method.

9.2.2. Line detection based on image segmentation and line thinning

Line detection is a long-standing problem in image analysis, and has been widely used for detecting roads (Bae et al., 2015), blood vessels (Obara et al., 2012a), etc. Among the existing methods, both hand-crafted methods and deep learning methods have their own advantages and disadvantages. Nonetheless, in terms of detection accuracy, deep learning methods comparatively perform better, and thus, they have attracted increasing attention recently. Typical existing methods consider line detection as a regression problem (Sironi et al., 2016). Despite the significant improvements over the competing methods, these methods usually yield isolated erroneous responses, discontinuities and topological errors (Sironi et al., 2015). One reason is that they use the distance transformation result of the thin centerlines as the ground truth, and accordingly, the trained model tends to yield fragile predicted lines. We have also argued this problem in this thesis. Experimental results have manifested that thick line structures are more robust to learn and represent than thin line structures. Besides, in Chapters 4 and 8, we have validated the efficacy of the UNet and NASAG kernels in line structure extraction or line structure thinning. These works suggest that it is meaningful to jointly use UNet and NASAG kernels for detecting line-like structures, such as fungi in microscopy images and rivers in remote sensing images. Currently,

the challenging problems in line detection include weak line extraction and spatial structure preservation. Besides approaches of data augmentation (Peng et al., 2018), structure learning (Dollár and Zitnick, 2015), etc., we can also address these problems by adjusting the training data. The thin centerlines in the ground truth can be quantitatively diffused into thick line structures by NASAG kernels. The obtained thick line structures can be used for training the UNet. It has been argued that training by thick line structures can ease the problem of pixel-class imbalance and can facilitate structure preservation. Since a UNet model trained by such training data would accordingly yield segmentation results that contain thick line structures, we can use NASAG kernels to transform these structures into thin lines.

9.2.3. Noise-aware and content-aware image denoising

Image denoising has been intensively studied for quite a long time. Despite a great many of achievements, most of the conventional methods are designed and tested for removing artificially modelled noise instead of real-world noise. Thus, these methods might underperform when dealing with real-world noise (Xu et al., 2018a). Moreover, many existing methods work well only if the noise level is known beforehand. Their performance would be degraded significantly if the noise level is incorrectly estimated. Quite a few methods have been developed to address real-world noise (Chen et al., 2018b) and to estimate the noise level (Dong et al., 2017), but thus far, there is no method for identifying and estimating high-ISO long-exposure noise. The blob characterization method we have proposed in Chapter 5 might provide cues for solving this problem.

Content-aware processing is a technique in which the processing operation varies adaptively according to the image contents (Rivera et al., 2012). Existing content-aware denoising methods are mainly used for preserving image details during the denoising procedure. Actually, a content-aware method for high-ISO long-exposure image denoising is also meaningful. This is because when reducing high-ISO long-exposure noise, our method removes blob structures indiscriminately, and therefore, some semantic blob objects, e.g., traffic lights in the far background, would also be removed. Although this kind of situations is generally rare, it would be desired to fix this potential vulnerability in some scenarios. A possible solution is to use CNN to develop a method that can detect semantic blob regions (Li et al., 2017). Such regions would be excluded in a subsequent blob reduction procedure, so that the denoising method would not remove the semantic objects.

9.2.4. Extensions of the automated *Artemia* analysis methods

Many biological studies consider image data as a primary source of information in unraveling the mechanisms of the observed organism (Meijering et al., 2016), and many such studies desire automated techniques for an accurate, efficient and objective image analysis. In this thesis, we have proposed two automated methods for performing *Artemia* image analysis tasks, which would benefit some *Artemia* studies. Nevertheless, extra work is still required to make these methods more adaptable.

Firstly, the *Artemia* detection method elaborated in Chapter 7 is currently used for detecting and counting cysts and nauplii. This method can be further extended for detecting *Artemia* objects belonging to other categories, including the juvenile, male adult and female adult. To this end, we can compile an extended dataset that contains annotated *Artemia* objects belonging to other categories. Also, the number of output categories should be adjusted by increasing the output of the Marker-CNN model.

Secondly, the colors of *Artemia* objects and the background are highly related to the used fixatives. The proposed *Artemia* analysis methods can be improved by making them more adaptable to different fixative colors. Moreover, *Artemia* objects appear nearly transparent when no fixative is applied. Methods for analyzing nearly transparent *Artemia* objects are also desired in practice. To achieve this goal, we can design a tailor-made image enhancement method, and compile a dataset containing *Artemia* objects that appear nearly transparent.

Thirdly, most of the images in the compiled datasets have clean backgrounds. To make the designed object detection and counting method more robust to undesired artefacts, we can collect and annotate more images that contain undesired artefacts, e.g., salt crystals, bioflocs and other zooplankton organisms.

Fourthly, the performance of the currently designed methods might be improved by using more advanced neural networks. For example, some recent works have manifested that a UNet architecture that is built on residual neural networks (He et al., 2016) might help to improve the segmentation performance (Zhang et al., 2018).

Besides, it would be interesting to expand the use of the proposed methods into some *Artemia* video analysis tasks, e.g., *Artemia* tracking. *Artemia* tracking is the process of keeping watch on the *Artemia* objects of interest. There are three key steps to accomplish this task: object detection, object

tracking from frame to frame, and behavior analysis using the object tracks. It is believed that the proposed *Artemia* detection method can make contributions to the first two steps. For example, on the initial video frame, the *Artemia* detection method can indicate all the possible targets and distribute an identification number for each target, which would facilitate the subsequent tracking and behavior analysis.

9.2.5. Transfer use of the developed *Artemia* image analysis methods

In this thesis, we have developed an *Artemia* detection/counting method and an *Artemia* length measurement method. To train and test the methods, we have also compiled the corresponding datasets. As stated earlier, many biological studies desire automated techniques for image analysis. Therefore, it would be interesting to transfer the developed methods to process images of other species that have a visual similarity to *Artemia*, e.g., shrimps or copepods. To achieve such objectives, the following approaches can be taken into account.

Firstly, when developing methods for analyzing images of other species, we can adopt transfer learning schemes (Tajbakhsh et al., 2016) and use the pre-trained models. The initial layers of the deep convolutional networks are mainly used for extracting and representing low-level features, and as such, these layers can be applied to images of other species. Using pre-trained models is also helpful for relieving the over-fitting problem, especially when the sizes of the datasets are small. We can fine-tune either the whole pre-trained models or the last several layers of the pre-trained models. In the fine-tuning procedure, we can use completely compiled datasets containing annotated images of other species. Besides, we can also use expandable datasets whose training data can be expanded by the users' interactive annotation incrementally.

Secondly, when annotating images of other species, we can adopt active learning techniques (Zhou et al., 2017) that are able to interactively query the annotations from the users. Among the unlabelled images, active learning techniques help us indicate the samples that should be labelled preferentially, and thus, the model to be trained can achieve a good performance with comparatively less annotated data.

9.2.6. Open-source software and image material

In bioscience engineering, there are many studies that require automated methods to analyze the acquired image data (Xing et al., 2018). In response to such demand, some open-source platforms (or toolkits) have been released

publicly. A representative platform is *Fiji*, a further developed version of the long-standing *ImageJ* (Schindelin et al., 2012). In the *Fiji* platform, the internal image analysis algorithms are transparent and can be further modified if desired. Most of the algorithms are based on hand-crafted features, so their use can be controlled by adjusting a few key parameters. Having an increasing number of algorithms, this platform provides a software library for a rapid transfer from conventional algorithms to practical image-analysis tools.

With the development of deep learning, many bioimage analysis methods using deep neural networks have been proposed. Many particular tasks have been addressed in literature, but only a small portion of these methods make their source codes and datasets publicly available, which leads to obstacles to reproduce or further improve these methods. Generally, models built on deep learning methods are more complex than those built on hand-crafted methods. The performance of a specific deep learning method might vary significantly with the settings of hyper-parameters, and as such, it is rather difficult to build a generic deep learning platform for solving a wide range of problems. Nevertheless, it is quite meaningful to develop a deep-learning-based platform (Gibson et al., 2018) that can solve a group of similar tasks (e.g., irregular cell/nucleus counting), or to build a platform for a specific field (e.g., *Artemia* image analysis). In this thesis, although we have accomplished several *Artemia* analysis tasks, there still remains a lot of work to build a full *Artemia* image analysis platform. To make contributions to such a platform, we have made the source codes of our *Artemia* detection & counting method and the *Artemia* length measurement method publicly available[1]. Moreover, we hope that our methods would contribute to some similar tasks, e.g., shrimp length measurement (Kesvarakul et al., 2017).

It is also encouraged that researchers in the bioscience engineering field could make more image materials open to the computer science community. On the one hand, image materials are pivotal for data-driven methods, and opening up more image datasets would help to attract more attention from software developers. On the other hand, although a lot of methods have been developed in literature, many of these methods still have room for improvement. Opening up image datasets would facilitate comparisons between conventional methods and new methods. Recently, there are more and more publicly available bioimage datasets, which have solicited a lot of solutions to the particular problems (Ronneberger et al., 2015). However, these existing datasets are far from enough. To attract more attention to *Artemia* analysis tasks, we have released our in-house datasets, the *Artemia* detection & counting dataset and the *Artemia* length measurement dataset, open to the public[1].

[1] https://github.com/GangWangUgent?tab=repositories/

Appendices

A Appendix

A.1. Proof of Eq. (3.13)

Let

$$\begin{aligned}\tilde{\mathbf{x}} &= [\tilde{x}, \tilde{y}]^{\mathrm{T}} \\ &= \begin{bmatrix} \cos\theta_0 & \sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 \end{bmatrix} \mathbf{x}\end{aligned}$$

be the rotated plane coordinates, $\tilde{\mathbf{u}} = [\tilde{u}, \tilde{v}]^{\mathrm{T}}$ an alternative notation of $\tilde{\mathbf{x}}$, and $\mathbf{u} = [u, v]^{\mathrm{T}}$ an alternative notation of $\mathbf{x}$. Then, the edge model formulated in Eq. (3.10) becomes

$$\boldsymbol{\Xi}_0(\mathbf{x}) = g(\mathbf{x}; \omega_0) * (c_0 \ \mathcal{H}(\tilde{x}) + b_0) \ .$$

In this edge model, $g(\mathbf{x}; \omega_0)$ is a rotational symmetric Gaussian function, and as such, it holds that

$$\begin{aligned}g(\mathbf{x}; \omega_0) &= g(\tilde{\mathbf{x}}; \omega_0) \\ &= \frac{1}{2\pi\omega_0^2} \exp\left(-\frac{\tilde{\mathbf{x}}^{\mathrm{T}}\tilde{\mathbf{x}}}{2\omega_0^2}\right) .\end{aligned}$$

Besides, the first derivative of $g(\tilde{\mathbf{x}}; \omega_0)$ w.r.t. x is given by

$$\begin{aligned}g'(\tilde{\mathbf{x}}; \omega_0) &= -\frac{\tilde{x}}{\omega_0^2} g(\tilde{\mathbf{x}}; \omega_0) \\ &= -\frac{\tilde{x}}{2\pi\omega_0^4} \exp\left(-\frac{\tilde{x}^2 + \tilde{y}^2}{2\omega_0^2}\right) .\end{aligned}$$

With respect to the anisotropic Gaussian kernel, by setting the value of θ as θ_0, we have

$$\begin{aligned}g(\mathbf{x}; \sigma, \varphi, \theta_0) &= \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{1}{2\sigma^2}\mathbf{x}^{\mathrm{T}}\mathbf{R}_{\theta_0}^{\mathrm{T}} \begin{bmatrix} 1 & 0 \\ 0 & \varphi^{-2} \end{bmatrix} \mathbf{R}_{\theta_0}\mathbf{x}\right) \\ &= \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{\tilde{\mathbf{x}}^2 + \varphi^{-2}\tilde{\mathbf{y}}^2}{2\sigma^2}\right) .\end{aligned}$$

Moreover, based on the algebraic properties of a convolution operation,

given two signals f and g, it holds that

$$\begin{aligned} f * g' &= g' * f \\ &= f' * g \\ &= (f * g)' \ . \end{aligned}$$

Then, we compute the E_{non} in Eq. (3.13) as follows:

$$\begin{aligned}
&E_{\text{non}} \\
&= \max_{\theta} \left| g'(\mathbf{x};\sigma,\varphi,\theta) * \mathbf{\Xi}_0(\mathbf{x}) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \left| g'(\mathbf{x};\sigma,\varphi,\theta) * \mathbf{\Xi}_0(\mathbf{x}) \right|_{\mathbf{x}=\mathbf{0},\theta=\theta_0} \\
&= \left| \mathbf{\Xi}_0(\mathbf{x}) * g'(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \left| (c_0 H(\tilde{x}) + b_0) * g(\tilde{\mathbf{x}};\omega_0) * g'(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \left| (c_0 H(\tilde{x}) + b_0) * g(\tilde{\mathbf{x}};\omega_0) * g'(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \left| (c_0 H(\tilde{x}) + b_0) * g'(\tilde{\mathbf{x}};\omega_0) * g(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \left| \big(c_0 H(\tilde{x}) * g'(\tilde{\mathbf{x}};\omega_0) + b_0 * g'(\tilde{\mathbf{x}};\omega_0)\big) * g(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \left| \big(c_0 H(\tilde{x}) * g'(\tilde{\mathbf{x}};\omega_0) + 0\big) * g(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= c_0 \left| \left(\iint_{\mathbb{R}^2} H(0-\tilde{u}) \cdot g'(\tilde{\mathbf{u}};\omega_0) \mathrm{d}\tilde{\mathbf{u}} \right) * g(\tilde{\mathbf{u}};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= c_0 \left| \left(-\frac{1}{2\pi\omega_0^4} \int_{-\infty}^{\tilde{u}} \tilde{u} \exp\left(-\frac{\tilde{u}^2}{2\omega_0^2}\right) \mathrm{d}\tilde{u} \int_{-\infty}^{+\infty} \exp\left(-\frac{\tilde{v}^2}{2\omega_0^2}\right) \mathrm{d}\tilde{v} \right) * g(\tilde{\mathbf{u}};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= c_0 \left| \left(-\frac{1}{2\pi\omega_0^4} \cdot \left(-\omega_0^2 \exp\left(-\frac{\tilde{u}^2}{2\omega_0^2}\right)\right) \cdot \sqrt{2\pi}\omega_0 \right) * g(\tilde{\mathbf{u}};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= c_0 \left| \frac{1}{\sqrt{2\pi}\omega_0} \exp\left(-\frac{\tilde{u}^2}{2\omega_0^2}\right) * g(\tilde{\mathbf{u}};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= c_0 \left| \frac{1}{\sqrt{2\pi}\omega_0} \exp\left(-\frac{\tilde{u}^2}{2\omega_0^2}\right) * \left(\frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{\tilde{u}^2 + \varphi^{-2}\tilde{v}^2}{2\sigma^2}\right)\right) \right|_{\mathbf{x}=\mathbf{0}} \\
&= c_0 \left| \iint_{\mathbb{R}^2} \frac{1}{\sqrt{2\pi}\omega_0} \exp\left(-\frac{(0-\tilde{x})^2}{2\omega_0^2}\right) \cdot \left(\frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{\tilde{x}^2 + \varphi^{-2}\tilde{y}^2}{2\sigma^2}\right)\right) \mathrm{d}\tilde{\mathbf{x}} \right| \\
&= c_0 \left| \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\omega_0} \exp\left(-\frac{(0-\tilde{x})^2}{2\omega_0^2}\right) \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\tilde{x}^2}{2\sigma^2}\right) \mathrm{d}\tilde{x} \right| \cdot \\
&\quad \left| \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\varphi\sigma} \exp\left(-\frac{\tilde{y}^2}{2(\varphi\sigma)^2}\right) \mathrm{d}\tilde{y} \right| \\
&= c_0 \frac{1}{\sqrt{2\pi(\omega_0^2 + \sigma^2)}} \cdot 1 \\
&= \frac{c_0}{\sqrt{2\pi(\omega_0^2 + \sigma^2)}} \, .
\end{aligned}$$

A.2. Proof of Eq. (3.15)

We compute the E_{norm} in Eq. (3.15) as follows:

$$
\begin{aligned}
&E_{\text{norm}} \\
&= \max_{\theta} \left| g'_{\text{E}}(\mathbf{x};\sigma,\varphi,\theta) * \mathbf{\Xi}_0(\mathbf{x}) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \left| \mathbf{\Xi}_0(\mathbf{x}) * g'_{\text{E}}(\mathbf{x};\sigma,\varphi,\theta) \right|_{\mathbf{x}=\mathbf{0},\theta=\theta_0} \\
&= \left| (c_0 H(\tilde{x}) + b_0) * g(\tilde{\mathbf{x}};\omega_0) * \left(\beta\sigma^{2\gamma} \cdot g'(\mathbf{x};\sigma,\varphi,\theta_0)\right) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta\sigma^{2\gamma} \left| (c_0 H(\tilde{x}) + b_0) * g(\tilde{\mathbf{x}};\omega_0) * g'(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta\sigma^{2\gamma} \left| (c_0 H(\tilde{x}) + b_0) * g'(\tilde{\mathbf{x}};\omega_0) * g(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta\sigma^{2\gamma} \left| \left(c_0 H(\tilde{x}) * g'(\tilde{\mathbf{x}};\omega_0) + b_0 * g'(\tilde{\mathbf{x}};\omega_0)\right) * g(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta\sigma^{2\gamma} \left| \left(c_0 H(\tilde{x}) * g'(\tilde{\mathbf{x}};\omega_0) + 0\right) * g(\mathbf{x};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta c_0 \sigma^{2\gamma} \left| \left(\iint_{\mathbb{R}^2} H(0-\tilde{u}) \cdot g'(\tilde{\mathbf{u}};\omega_0) \mathrm{d}\tilde{\mathbf{u}} \right) * g(\tilde{\mathbf{u}};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta c_0 \sigma^{2\gamma} \left| \left(-\frac{1}{2\pi\omega_0^4} \int_{-\infty}^{\tilde{u}} \tilde{u} \exp\left(-\frac{\tilde{u}^2}{2\omega_0^2}\right) \mathrm{d}\tilde{u} \int_{-\infty}^{+\infty} \exp\left(-\frac{\tilde{v}^2}{2\omega_0^2}\right) \mathrm{d}\tilde{v} \right) * g(\tilde{\mathbf{u}};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta c_0 \sigma^{2\gamma} \left| \left(-\frac{1}{2\pi\omega_0^4} \cdot \left(-\omega_0^2 \exp\left(-\frac{\tilde{u}^2}{2\omega_0^2}\right)\right) \cdot \sqrt{2\pi}\omega_0 \right) * g(\tilde{\mathbf{u}};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta c_0 \sigma^{2\gamma} \left| \frac{1}{\sqrt{2\pi}\omega_0} \exp\left(-\frac{\tilde{u}^2}{2\omega_0^2}\right) * g(\tilde{\mathbf{u}};\sigma,\varphi,\theta_0) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta c_0 \sigma^{2\gamma} \left| \frac{1}{\sqrt{2\pi}\omega_0} \exp\left(-\frac{\tilde{u}^2}{2\omega_0^2}\right) * \left(\frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{\tilde{u}^2+\varphi^{-2}\tilde{v}^2}{2\sigma^2}\right)\right) \right|_{\mathbf{x}=\mathbf{0}} \\
&= \beta c_0 \sigma^{2\gamma} \left| \iint_{\mathbb{R}^2} \frac{1}{\sqrt{2\pi}\omega_0} \exp\left(-\frac{(0-\tilde{x})^2}{2\omega_0^2}\right) \cdot \left(\frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{\tilde{x}^2+\varphi^{-2}\tilde{y}^2}{2\sigma^2}\right)\right) \mathrm{d}\tilde{\mathbf{x}} \right| \\
&= \beta c_0 \sigma^{2\gamma} \left| \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\omega_0} \exp\left(-\frac{(0-\tilde{x})^2}{2\omega_0^2}\right) \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\tilde{x}^2}{2\sigma^2}\right) \mathrm{d}\tilde{x} \right| \\
&\quad \cdot \left| \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\varphi\sigma} \exp\left(-\frac{\tilde{y}^2}{2(\varphi\sigma)^2}\right) \mathrm{d}\tilde{y} \right| \\
&= \beta c_0 \sigma^{2\gamma} \cdot \frac{1}{\sqrt{2\pi(\omega_0^2+\sigma^2)}} \cdot 1 \\
&= \beta c_0 \frac{\sigma^{2\gamma}}{\sqrt{2\pi(\omega_0^2+\sigma^2)}} .
\end{aligned}
$$

A.3. Proof of Eq. (4.11)

For bright line detection, we have $\eta = 1$. Without loss of generality, we consider the case $\theta_0 = 0$ and $\mathbf{x}_0 = 0$. Then, the line model formulated in Eq. (4.3) becomes

$$\begin{aligned}\boldsymbol{\Gamma}_0(\mathbf{x}) &= p_0 \cdot \exp\left(-\frac{(\tilde{\mathbf{x}} - \mathbf{x_0})^{\mathrm{T}}(\tilde{\mathbf{x}} - \mathbf{x_0})}{2\omega_0^2}\right) + b_0 \Bigg|_{\theta_0=0, \mathbf{x}_0=0} \\ &= p_0 \cdot \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\omega_0^2}\right) + b_0 \,.\end{aligned}$$

In this case, the anisotropic Gaussian kernel becomes

$$\begin{aligned}g(\mathbf{x}; \sigma, \varphi, \theta) &= \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{1}{2\sigma^2}\mathbf{x}^{\mathrm{T}}\mathbf{R}_\theta^{\mathrm{T}}\begin{bmatrix}1 & 0 \\ 0 & \varphi^{-2}\end{bmatrix}\mathbf{R}_\theta\mathbf{x}\right)\Bigg|_{\theta=0} \\ &= \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{x^2 + \varphi^{-2}y^2}{2\sigma^2}\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{2\pi}\varphi\sigma} \exp\left(-\frac{\varphi^{-2}y^2}{2\sigma^2}\right) .\end{aligned}$$

Accordingly, the normalized SAG kernel $g''_{\mathrm{L}}(\mathbf{x}; \sigma, \varphi, \theta)$ is given by

$$\begin{aligned}g''_{\mathrm{L}}(\mathbf{x}; \sigma, \varphi, \theta) &= (-1)^\eta \cdot \beta \cdot \sigma^{2\gamma} \cdot g''(\mathbf{x}; \sigma, \varphi, \theta) \\ &= -1 \cdot \beta\sigma^{2\gamma} \cdot \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{x^2 + \varphi^{-2}y^2}{2\sigma^2}\right) \\ &= -\beta\sigma^{2\gamma} \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{2\pi}\varphi\sigma} \exp\left(-\frac{\varphi^{-2}y^2}{2\sigma^2}\right) .\end{aligned}$$

Then, the proof of Eq. (4.11) is given as follows:

$$\begin{aligned}&L \\ &= \mathbf{L}(\mathbf{x}; \sigma, \varphi, \theta)|_{\mathbf{x}=\mathbf{0}, \theta=0} \\ &= g''_{\mathrm{L}}(\mathbf{x}; \sigma, \varphi, \theta) * \boldsymbol{\Gamma}_0(\mathbf{x})|_{\mathbf{x}=\mathbf{0}, \theta=0} \\ &= \left(-\beta\sigma^{2\gamma} \cdot \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot g(\mathbf{x}; \sigma, \varphi, \theta)\right) * \left(p_0 \cdot \exp\left(-\frac{x^2}{2\omega_0^2}\right) + b_0\right)\Bigg|_{\mathbf{x}=\mathbf{0}, \theta=0} \\ &= \left(p_0 \cdot \exp\left(-\frac{x^2}{2\omega_0^2}\right) + b_0\right) * \left(-\beta\sigma^{2\gamma} \cdot \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot g(\mathbf{x}; \sigma, \varphi, \theta)\right)\Bigg|_{\mathbf{x}=\mathbf{0}, \theta=0} \\ &= -\beta\sigma^{2\gamma} \cdot \left(p_0 \cdot \exp\left(-\frac{x^2}{2\omega_0^2}\right) + b_0\right) * \left(\left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot g(\mathbf{x}; \sigma, \varphi, \theta)\right)\Bigg|_{\mathbf{x}=\mathbf{0}, \theta=0}\end{aligned}$$

$$
\begin{aligned}
&= -\beta\sigma^{2\gamma} \cdot \iint_{\mathbb{R}^2} \left(p_0 \cdot \exp\left(-\frac{(0-u)^2}{2\omega_0^2} \right) + b_0 \right) \cdot \left(\left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \cdot g(\mathbf{u}; \sigma, \varphi, \theta) \right) \mathrm{d}\mathbf{u} \Big|_{\theta=0} \\
&= -\beta\sigma^{2\gamma} \iint_{\mathbb{R}^2} \left(p_0 \exp\left(-\frac{u^2}{2\omega_0^2} \right) + b_0 \right) \cdot \left(\left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \cdot \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{u^2 + \varphi^{-2}v^2}{2\sigma^2} \right) \right) \mathrm{d}\mathbf{u} \\
&= -\beta\sigma^{2\gamma} \iint_{\mathbb{R}^2} \left(p_0 \exp\left(-\frac{u^2}{2\omega_0^2} \right) \right) \cdot \left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \cdot \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{u^2 + \varphi^{-2}v^2}{2\sigma^2} \right) \mathrm{d}\mathbf{u} \\
&\quad + b_0 \cdot \iint_{\mathbb{R}^2} \left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \cdot \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{u^2 + \varphi^{-2}v^2}{2\sigma^2} \right) \mathrm{d}\mathbf{u} \\
&= -p_0\beta\sigma^{2\gamma} \iint_{\mathbb{R}^2} \exp\left(-\frac{u^2}{2\omega_0^2} \right) \cdot \left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \cdot \frac{1}{2\pi\varphi\sigma^2} \exp\left(-\frac{u^2 + \varphi^{-2}v^2}{2\sigma^2} \right) \mathrm{d}\mathbf{u} + 0 \\
&= -p_0\beta\sigma^{2\gamma} \iint_{\mathbb{R}^2} \left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \cdot \exp\left(-\frac{u^2}{2\omega_0^2} \right) \cdot \frac{1}{2\pi\varphi\sigma^2} \cdot \exp\left(-\frac{u^2 + \varphi^{-2}v^2}{2\sigma^2} \right) \mathrm{d}\mathbf{u} \\
&= -p_0\beta\sigma^{2\gamma} \iint_{\mathbb{R}^2} \left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \exp\left(-\frac{u^2}{2\omega_0^2} \right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2} \right) \frac{1}{\sqrt{2\pi}\varphi\sigma} \exp\left(-\frac{\varphi^{-2}v^2}{2\sigma^2} \right) \mathrm{d}\mathbf{u} \\
&= -p_0\beta\sigma^{2\gamma} \int_{-\infty}^{+\infty} \left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \cdot \exp\left(-\frac{u^2}{2\omega_0^2} \right) \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2} \right) \mathrm{d}u \\
&\quad \cdot \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\varphi\sigma} \exp\left(-\frac{\varphi^{-2}v^2}{2\sigma^2} \right) \mathrm{d}v \\
&= -p_0\beta\sigma^{2\gamma} \int_{-\infty}^{+\infty} \left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2} \right) \exp\left(-\frac{u^2}{2\omega_0^2} \right) \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2} \right) \mathrm{d}u \cdot 1 \\
&= -p_0\beta\sigma^{2\gamma} \left(\int_{-\infty}^{+\infty} \frac{u^2}{\sigma^4} \cdot \exp\left(-\frac{u^2}{2\omega_0^2} \right) \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2} \right) \mathrm{d}u \right) \\
&\quad + p_0\beta\sigma^{2\gamma} \left(\int_{-\infty}^{+\infty} \frac{1}{\sigma^2} \cdot \exp\left(-\frac{u^2}{2\omega_0^2} \right) \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2} \right) \mathrm{d}u \right) \\
&= -p_0\beta\sigma^{2\gamma} \left(\sigma^{-2}\omega_0^3(\omega_0^2 + \sigma^2)^{-\frac{3}{2}} - \sigma^{-2}\omega_0(\omega_0^2 + \sigma^2)^{-\frac{1}{2}} \right) \\
&= \beta p_0 \omega_0 \sigma^{2\gamma} (\omega_0^2 + \sigma^2)^{-\frac{3}{2}}
\end{aligned}
$$

A.4. Proof of Eq. (5.9)

For bright blob detection, we have $\eta = 1$. Without loss of generality, we consider the case $\mathbf{x}_0 = 0$. Then, the blob model becomes

$$\mathbf{\Lambda}_0(\mathbf{x}) = p_0 \cdot \exp\left(-\frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{2\omega_0^2}\right) + b_0\,.$$

We compute B in Eq. (5.9) as follows:

$$\begin{aligned}
&B\\
&= g''_{\mathrm{B}}(\mathbf{x};\sigma) * \mathbf{\Lambda}_0(\mathbf{x})|_{\mathbf{x}=\mathbf{0}}\\
&= g''_{\mathrm{B}}(\mathbf{x};\sigma) * \left(p_0 \cdot \exp\left(-\frac{x^2+y^2}{2\omega_0^2}\right) + b_0\right)\Bigg|_{\mathbf{x}=\mathbf{0}}\\
&= \left(g''_{\mathrm{B}}(\mathbf{x};\sigma) * \left(p_0 \cdot \exp\left(-\frac{x^2+y^2}{2\omega_0^2}\right)\right) + g''_{\mathrm{B}}(\mathbf{x};\sigma) * b_0\right)\Bigg|_{\mathbf{x}=\mathbf{0}}\\
&= \left(g''_{\mathrm{B}}(\mathbf{x};\sigma) * \left(p_0 \cdot \exp\left(-\frac{x^2+y^2}{2\omega_0^2}\right)\right) + 0\right)\Bigg|_{\mathbf{x}=\mathbf{0}}\\
&= \left(-1 \cdot \beta\sigma^{2\gamma} \cdot \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\right) * \left(p_0 \cdot \exp\left(-\frac{x^2+y^2}{2\omega_0^2}\right)\right)\Bigg|_{\mathbf{x}=\mathbf{0}}\\
&= -p_0\beta\sigma^{2\gamma}\left(\left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\right) * \exp\left(-\frac{x^2+y^2}{2\omega_0^2}\right)\Bigg|_{\mathbf{x}=\mathbf{0}}\\
&= -p_0\beta\sigma^{2\gamma}\iint_{\mathbb{R}^2}\left(\frac{(0-u)^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot \frac{1}{2\pi\sigma^2} \exp\left(\frac{(0-u)^2+(0-v)^2}{-2\sigma^2}\right) \cdot \exp\left(-\frac{u^2+v^2}{2\omega_0^2}\right) \mathrm{d}\mathbf{u}\Bigg|_{\mathbf{x}=\mathbf{0}}\\
&= -p_0\beta\sigma^{2\gamma}\iint_{\mathbb{R}^2}\left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2+v^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{u^2+v^2}{2\omega_0^2}\right) \mathrm{d}\mathbf{u}\\
&= -p_0\beta\sigma^{2\gamma}\iint_{\mathbb{R}^2}\left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2}{2\sigma^2}\right)\exp\left(-\frac{v^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{u^2}{2\omega_0^2}\right)\exp\left(-\frac{v^2}{2\omega_0^2}\right) \mathrm{d}\mathbf{u}\\
&= -p_0\beta\sigma^{2\gamma}\int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi}\sigma}\left(\frac{u^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \cdot \exp\left(-\frac{u^2}{2\sigma^2}\right)\exp\left(-\frac{u^2}{2\omega_0^2}\right) \mathrm{d}u\\
&\quad \cdot \int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{v^2}{2\sigma^2}\right)\exp\left(-\frac{v^2}{2\omega_0^2}\right) \mathrm{d}v\\
&= -p_0\beta\sigma^{2\gamma}\left(\omega_0(\omega_0^2+\sigma^2)^{-\frac{1}{2}}\right)\left(\sigma^{-2}\omega_0^3(\omega_0^2+\sigma^2)^{-\frac{3}{2}} - \sigma^{-2}\omega_0(\omega_0^2+\sigma^2)^{-\frac{1}{2}}\right)\\
&= -p_0\beta\sigma^{2\gamma}\left(\sigma^{-2}\omega_0^4(\omega_0^2+\sigma^2)^{-2} - \sigma^{-2}\omega_0^2(\omega_0^2+\sigma^2)^{-1}\right)\\
&= \beta p_0\sigma^{2\gamma-2}\omega_0^2(\omega_0^2+\sigma^2)^{-1}\left(1 - \omega_0^2(\omega_0^2+\sigma^2)^{-1}\right)
\end{aligned}$$

Bibliography

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

Akbarinia, A. and Parraga, C. A. (2018). Feedback and surround modulated boundary detection. *International Journal of Computer Vision*, 126(12):1367–1380.

Akyüz, A. O. and Reinhard, E. (2007). Noise reduction in high dynamic range imaging. *Journal of Visual Communication and Image Representation*, 18(5):366–376.

Alpert, S., Galun, M., Brandt, A., and Basri, R. (2012). Image segmentation by probabilistic bottom-up aggregation and cue integration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):315–327.

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916.

Arslan, S., Ersahin, T., Cetin-Atalay, R., and Gunduz-Demir, C. (2013). Attributed relational graphs for cell nucleus segmentation in fluorescence microscopy images. *IEEE Transactions on Medical Imaging*, 32(6):1121–1131.

Ates, M., Daniels, J., Arslan, Z., and Farah, I. O. (2013). Effects of aqueous suspensions of titanium dioxide nanoparticles on *Artemia salina*: Assessment of nanoparticle aggregation, accumulation, and toxicity. *Environmental Monitoring and Assessment*, 185(4):3339–3348.

Atick, J. J. and Redlich, A. N. (1992). What does the retina know about natural scenes? *Neural Computation*, 4(2):196–210.

Azzari, L., Borges, L. R., and Foi, A. (2018). Modeling and estimation of signal-dependent and correlated noise. In *Denoising of Photographic Images and Video*, pages 1–36. Springer.

Bae, Y., Lee, W.-H., Choi, Y., Jeon, Y. W., and Ra, J. B. (2015). Automatic road extraction from remote sensing images based on a normalized second derivative map. *IEEE Geoscience and Remote Sensing Letters*, 12(9):1858–1862.

Balaban, M. O., Ünal Şengör, G. F., Soriano, M. G., and Ruiz, E. G. (2010). Using image analysis to predict the weight of Alaskan salmon of different species. *Journal of Food Science*, 75(3):E157–E162.

Balachandar, S. and Rajaram, R. (2019). Influence of different diets on the growth, survival, fecundity and proximate composition of brine shrimp *Artemia franciscana* (kellog, 1906). *Aquaculture Research*, 50(2):376–389.

Bao, P., Zhang, L., and Wu, X. (2005). Canny edge detection enhancement by scale multiplication. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1485–1490.

Basset, A., Boulanger, J., Salamero, J., Bouthemy, P., and Kervrann, C. (2015). Adaptive spot detection with optimal scale selection in fluorescence microscopy images. *IEEE Transactions on Image Processing*, 24(11):4512–4527.

Batool, N. and Chellappa, R. (2015). Fast detection of facial wrinkles based on Gabor features using image morphology and geometric constraints. *Pattern Recognition*, 48(3):642–658.

Benmansour, F. and Cohen, L. D. (2011). Tubular structure segmentation based on minimal path method and anisotropic enhancement. *International Journal of Computer Vision*, 92(2):192–210.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Black, M. J., Sapiro, G., Marimont, D. H., and Heeger, D. (1998). Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, 7(3):421–432.

Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the International Conference on Machine Learning*, pages 111–118.

Breen, E., Joss, G., and Williams, K. (1991). Locating objects of interest within biological images: The Top Hat box filter. *Journal of Computer-Assisted Microscopy*, 3:97–102.

Bresson, G., Alsayed, Z., Yu, L., and Glaser, S. (2017). Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220.

Brooks, T., Mildenhall, B., Xue, T., Chen, J., Sharlet, D., and Barron, J. T. (2019). Unprocessing images for learned raw denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11036–11045.

Buades, A., Coll, B., and Morel, J. M. (2005). A non-local algorithm for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 60–65.

Bühler, J., Rishmawi, L., Pflugfelder, D., Huber, G., Scharr, H., Hülskamp, M., Koornneef, M., Schurr, U., and Jahnke, S. (2015). Phenovein-A tool for leaf vein segmentation and analysis. *Plant Physiology*, 169(4):2359–2370.

Calderero, F. and Marques, F. (2010). Region merging techniques using information theory statistical measures. *IEEE Transactions on Image Processing*, 19(6):1567–1586.

Candamo, J., Kasturi, R., Goldgof, D., and Sarkar, S. (2009). Detection of thin lines using low-quality video from low-altitude aircraft in urban settings. *IEEE Transactions on Aerospace and Electronic Systems*, 45(3):937–949.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698.

Cárdenes, R., Alberola-López, C., and Ruiz-Alzola, J. (2010). Fast and accurate geodesic distance transform by ordered propagation. *Image and Vision Computing*, 28(3):307–316.

Chang, S. G., Yu, B., and Vetterli, M. (2000a). Adaptive wavelet thresholding for image denoising and compression. *IEEE Transactions on Image Processing*, 9(9):1532–1546.

Chang, S. G., Yu, B., and Vetterli, M. (2000b). Adaptive wavelet thresholding for image denoising and compression. *IEEE Transactions on Image Processing*, 9(9):1532–1546.

Chatterjee, P., Joshi, N., Kang, S. B., and Matsushita, Y. (2011). Noise suppression in low-light images through joint denoising and demosaicing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 321–328.

Chaudhuri, S., Chatterjee, S., Katz, N., Nelson, M., and Goldbaum, M. (1989). Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Transactions on Medical Imaging*, 8(3):263–269.

Chen, C., Chen, Q., Xu, J., and Koltun, V. (2018a). Learning to see in the dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3291–3300.

Chen, J., Chen, J., Chao, H., and Yang, M. (2018b). Image blind denoising with generative adversarial network based noise modeling. In *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition, pages 3155–3164.

Chen, S. W., Shivakumar, S. S., Dcunha, S., Das, J., Okon, E., Qu, C., Taylor, C. J., and Kumar, V. (2017). Counting apples and oranges with deep learning: A data-driven approach. *IEEE Robotics and Automation Letters*, 2(2):781–788.

Choudhry, P. (2016). High-throughput method for automated colony and cell counting by digital image analysis based on edge detection. *PloS One*, 11(2).

Chuang, M.-C., Hwang, J.-N., and Williams, K. (2016). A feature learning and object recognition framework for underwater fish images. *IEEE Transactions on Image Processing*, 25(4):1862–1872.

Ciresan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 2843–2851.

Coleman, S. A., Scotney, B. W., and Suganthan, S. (2010). Edge detecting for range data using Laplacian operators. *IEEE Transactions on Image Processing*, 19(11):2814–2824.

Cornelis, B., Ružić, T., Gezels, E., Dooms, A., Pižurica, A., Platiša, L., Cornelis, J., Martens, M., De Mey, M., and Daubechies, I. (2013). Crack detection and inpainting for virtual restoration of paintings: The case of the Ghent altarpiece. *Signal Processing*, 93(3):605–619.

Csillik, O. (2017). Fast segmentation and classification of very high resolution remote sensing data using SLIC superpixels. *Remote Sensing*, 9(3):24301–24319.

Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. (2007). Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095.

Descombes, X. (2017). Multiple objects detection in biological images using a marked point process framework. *Methods*, 115:2–8.

Descombes, X., Minlos, R., and Zhizhina, E. (2009). Object extraction using a stochastic birth-and-death dynamics in continuum. *Journal of Mathematical Imagfing and Vision*, 33(3):347–359.

Dewan, M. A., Ahmad, M. O., and Swamy, M. N. (2014). A method for automatic segmentation of nuclei in phase-contrast images based on intensity,

convexity and texture. *IEEE Transactions on Biomedical Circuits and Systems*, 8(5):716–728.

Diciotti, S., Lombardo, S., Coppini, G., Grassi, L., Falchini, M., and Mascalchi, M. (2010). The LoG characteristic scale: A consistent measurement of lung nodule size in CT imaging. *IEEE Transactions on Medical Imaging*, 29(2):397–409.

Ding, J., Chen, B., Liu, H., and Huang, M. (2016). Convolutional neural network with data augmentation for SAR target recognition. *IEEE Geoscience and Remote Sensing Letters*, 13(3):364–368.

Ding, L. and Goshtasby, A. (2001). On the Canny edge detector. *Pattern Recognition*, 34(3):721–725.

Ding, W. and Taylor, G. (2016). Automatic moth detection from trap images for pest management. *Computers and Electronics in Agriculture*, 123:17–28.

Do, M. N. and Vetterli, M. (2005). The contourlet transform: an efficient directional multiresolution image representation. *IEEE Transactions on Image Processing*, 14(12):2091–2106.

Dollár, P. and Zitnick, C. L. (2015). Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1558–1570.

Dong, L., Zhou, J., and Tang, Y. Y. (2017). Noise level estimation for natural images based on scale-invariant kurtosis and piecewise stationarity. *IEEE Transactions on Image Processing*, 26(2):1017–1030.

Duval-Poo, M. A., Odone, F., and De Vito, E. (2015). Edges and corners with shearlets. *IEEE Transactions on Image Processing*, 24(11):3768–3780.

El-Magsodi, M. O., Baruah, K., Norouzitallab, P., Bossier, P., Sorgeloos, P., and Van Stappen, G. (2016). Hydration/dehydration cycles imposed on *Artemia* cysts influence the tolerance limit of nauplii against abiotic and biotic stressors. *Aquaculture International*, 24(2):429–439.

Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745.

Elder, J. H. and Zucker, S. W. (1998). Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):699–716.

Fan, J. (1998). Notes on Poisson distribution-based minimum error thresholding. *Pattern Recognition Letters*, 19(5-6):425–431.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.

Ferraz, A., Mallet, C., and Chehata, N. (2016). Large-scale road detection in forested mountainous areas using airborne topographic lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 112:23–36.

Florack, L. M., ter Haar Romeny, B. M., Koenderink, J. J., and Viergever, M. A. (1992). Scale and the differential structure of images. *Image and Vision Computing*, 10(6):376–388.

Foi, A., Trimeche, M., Katkovnik, V., and Egiazarian, K. (2008). Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–1754.

Frangi, A. F., Niessen, W. J., Vincken, K. L., and Viergever, M. A. (1998). Multiscale vessel enhancement filtering. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 130–137.

French, G., Fisher, M. H., Mackiewicz, M., and Needle, C. L. (2015). Convolutional neural networks for counting fish in fisheries surveillance video. *Proceedings of the Workshop on Machine Vision of Animals and Their Behaviour*, pages 1–10.

Gebäck, T. and Koumoutsakos, P. (2009). Edge detection in microscopy images using curvelets. *BMC Bioinformatics*, 10(1):75.

Geweid, G. G., Elsisy, M., Faragallah, O. S., and Fazel-Rezai, R. (2019). Efficient tumor detection in medical images using pixel intensity estimation based on nonparametric approach. *Expert Systems with Applications*, 120:139–154.

Ghimpeţeanu, G., Batard, T., Bertalmío, M., and Levine, S. (2016). A decomposition framework for image denoising algorithms. *IEEE Transactions on Image Processing*, 25(1):388–399.

Gibson, E., Li, W., Sudre, C., Fidon, L., Shakir, D. I., Wang, G., Eaton-Rosen, Z., Gray, R., Doel, T., Hu, Y., et al. (2018). NiftyNet: A deep-learning platform for medical imaging. *Computer Methods and Programs in Biomedicine*, 158:113–122.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158.

Godard, C., Matzen, K., and Uyttendaele, M. (2018). Deep burst denoising. In *Proceedings of the European Conference on Computer Vision*, pages 538–554.

Goldberg, A. V. and Kennedy, R. (1995). An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71(2):153–177.

González, R. C. and Woods, R. E. (2010). *Digital Image Processing*. Pearson Education.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goossens, B., Luong, H., Aelterman, J., Pižurica, A., and Philips, W. (2012). Realistic camera noise modeling with application to improved HDR synthesis. *EURASIP Journal on Advances in Signal Processing*, 2012(1):171.

Goossens, B., Luong, H., Pizurica, A., and Philips, W. (2008). An improved non-local denoising algorithm. In *Proceedings of the International Workshop on Local and Non-Local Approximation in Image Processing*, pages 143–156.

Goossens, B., Pizurica, A., and Philips, W. (2009). Removal of correlated noise by modeling the signal of interest in the wavelet domain. *IEEE Transactions on Image Processing*, 18(6):1153–1165.

Grigorescu, C., Petkov, N., and Westenberg, M. A. (2003). Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on Image Processing*, 12(7):729–739.

Guo, S., Yan, Z., Zhang, K., Zuo, W., and Zhang, L. (2019). Toward convolutional blind denoising of real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1712–1722.

Guo, X., Li, Y., and Ling, H. (2017). LIME: Low-light image enhancement via illumination map estimation. *IEEE Transactions on Image Processing*, 26(2):982–993.

Han, J., Honda, N., Asada, A., and Shibata, K. (2009). Automated acoustic method for counting and sizing farmed fish during transfer using DIDSON. *Fisheries Science*, 75(6):1359.

Han, J., Zhang, D., Cheng, G., Liu, N., and Xu, D. (2018). Advanced deep-learning techniques for salient and category-specific object detection: A survey. *IEEE Signal Processing Magazine*, 35(1):84–100.

Hao, M., Yu, H., and Li, D. (2015). The measurement of fish size by machine vision-A review. In *Proceedings of the International Conference on Computer and Computing Technologies in Agriculture*, pages 15–32.

Haralick, R. M. (1983). Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing*, 22(1):28–38.

Haralick, R. M., Shanmugam, K., and Dinstein, I. H. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, (6):610–621.

Hashemzadeh, M. and Farajzadeh, N. (2016). Combining keypoint-based and segment-based features for counting people in crowded scenes. *Information Sciences*, 345:199–216.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Helman, A. (2005). *The Finest Peaks-Prominence and Other Mountain Measures*. Trafford Publishing.

Herbin, M., Bonnet, N., and Vautrot, P. (1996). A clustering method based on the estimation of the probability density function and on the skeleton by influence zones: Application to image processing. *Pattern Recognition Letters*, 17(11):1141–1150.

Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines vinod nair. In *Proceedings of the International Conference on Machine Learning*, pages 807–814.

Hopfield, J. J. (1988). Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5):3–10.

Hsieh, C.-L., Chang, H.-Y., Chen, F.-H., Liou, J.-H., Chang, S.-K., and Lin, T.-T. (2011). A simple and effective digital imaging approach for tuna fish length measurement compatible with fishing operations. *Computers and Electronics in Agriculture*, 75(1):44–51.

Hu, Z., Wu, Z., Zhang, Q., Fan, Q., and Xu, J. (2013). A spatially-constrained color–texture model for hierarchical VHR image segmentation. *IEEE Geoscience and Remote Sensing Letters*, 10(1):120–124.

Huang, D.-A., Kang, L.-W., Wang, Y.-C. F., and Lin, C.-W. (2014). Self-learning based image decomposition with applications to single image denoising. *IEEE Transactions on Multimedia*, 16(1):83–93.

Huang, T.-W., Hwang, J.-N., and Rose, C. S. (2016). Chute based automated fish length measurement and water drop detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1906–1910.

Hui, R. and O'Sullivan, M. (2009). *Fiber Optic Measurement Techniques*. Academic Press.

Jacob, M. and Unser, M. (2004). Design of steerable filters for feature detection using Canny-like criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1007–1019.

Jaiswal, A., Godinez, W. J., Eils, R., Lehmann, M. J., and Rohr, K. (2015). Tracking virus particles in fluorescence microscopy images using multi-scale detection and multi-frame association. *IEEE Transactions on Image Processing*, 24(11):4122–4136.

Jerman, T., Pernuš, F., Likar, B., and Špiclin, Ž. (2015). Beyond Frangi: An improved multiscale vesselness filter. In *Proceedings of SPIE Medical Imaging 2015: Image Processing*, volume 9413, pages 1–11.

Kaewchote, J., Janyong, S., and Limprasert, W. (2018). Image recognition method using Local Binary Pattern and the Random forest classifier to count post larvae shrimp. *Agriculture and Natural Resources*, 52(4):371–376.

Kamilaris, A. and Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90.

Kang, M. and Palmer-Brown, D. (2008). A modal learning adaptive function neural network applied to handwritten digit recognition. *Information Sciences*, 178(20):3802–3812.

Karakasis, E. G., Amanatiadis, A., Gasteratos, A., and Chatzichristofis, S. A. (2015). Image moment invariants as local features for content based image

retrieval using the bag-of-visual-words model. *Pattern Recognition Letters*, 55:22–27.

Kesvarakul, R., Chianrabutra, C., and Chianrabutra, S. (2017). Baby shrimp counting via automated image processing. In *Proceedings of the International Conference on Machine Learning and Computing*, pages 352–356.

Kim, S. and Cho, H.-Y. (2013). Automatic estimation of *Artemia* hatching rate using an object discrimination method. *Ocean and Polar Research*, 35(3):239–247.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

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

Kong, H., Akakin, H. C., and Sarma, S. E. (2013a). A generalized Laplacian of Gaussian filter for blob detection and its applications. *IEEE Transactions on Cybernetics*, 43(6):1719–1733.

Kong, H., Sarma, S. E., and Tang, F. (2013b). Generalizing Laplacian of Gaussian filters for vanishing-point detection. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):408–418.

Kong, T. Y. and Rosenfeld, A. (1996). *Topological Algorithms for Digital Image Processing*. Elsevier.

Koschan, A. and Abidi, M. (2005). Detection and classification of edges in color images. *IEEE Signal Processing Magazine*, 22(1):64–73.

Koyuncu, C. F., Akhan, E., Ersahin, T., Cetin-Atalay, R., and Gunduz-Demir, C. (2016). Iterative h-minima-based marker-controlled watershed for cell nucleus segmentation. *Cytometry Part A*, 89(4):338–349.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 1097–1105.

Krylov, V. A. and Nelson, J. D. (2014). Stochastic extraction of elongated curvilinear structures with applications. *IEEE Transactions on Image Processing*, 23(12):5360–5373.

Kussul, N., Lavreniuk, M., Skakun, S., and Shelestov, A. (2017). Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 14(5):778–782.

Law, M. W., Tay, K., Leung, A., Garvin, G. J., and Li, S. (2012). Dilated

divergence based scale-space representation for curve analysis. In *Proceedings of the European Conference on Computer Vision*, pages 557–571. Springer.

Le, T. H., Hoa, N. V., Sorgeloos, P., and Van Stappen, G. (2018). *Artemia* feeds: A review of brine shrimp production in the Mekong Delta, Vietnam. *Reviews in Aquaculture*, 1:1–7.

Le Pennec, E. and Mallat, S. (2005). Sparse geometric image representations with bandelets. *IEEE Transactions on Image Processing*, 14(4):423–438.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*, pages 255–258.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Lee, D.-G., Cha, B.-J., Park, S.-W., Kwon, M.-G., Xu, G.-C., and Kim, H.-J. (2013). Development of a vision-based automatic vaccine injection system for flatfish. *Aquacultural Engineering*, 54:78–84.

Lehmussola, A., Ruusuvuori, P., Selinummi, J., Huttunen, H., and Yli-Harja, O. (2007). Computational framework for simulating fluorescence microscope images with cell populations. *IEEE Transactions on Medical Imaging*, 26(7):1010–1016.

Li, J., Liang, X., Wei, Y., Xu, T., Feng, J., and Yan, S. (2017). Perceptual generative adversarial networks for small object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1222–1230.

Li, X., Chen, H., Qi, X., Dou, Q., Fu, C.-W., and Heng, P.-A. (2018). H-DenseUNet: Hybrid densely connected UNet for liver and tumor segmentation from CT volumes. *IEEE Transactions on Medical Imaging*, 37(12):2663–2674.

Li, X. and Chen, T. (1994). Nonlinear diffusion with multiple edginess thresholds. *Pattern Recognition*, 27(8):1029–1037.

Li, Y., Wang, S., Tian, Q., and Ding, X. (2015a). A survey of recent advances in visual feature detection. *Neurocomputing*, 149:736–751.

Li, Z., Ahmed, E., Eltawil, A. M., and Cetiner, B. A. (2015b). A beam-

steering reconfigurable antenna for WLAN applications. *IEEE Transactions on Antennas and Propagation*, 63(1):24–32.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988.

Lindeberg, T. (1993). Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318.

Lindeberg, T. (1998a). Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):117–156.

Lindeberg, T. (1998b). Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116.

Lindeberg, T. (2013). Scale selection properties of generalized scale-space interest point detectors. *Journal of Mathematical Imaging and Vision*, 46(2):177–210.

Lindeberg, T. (2015). Image matching using generalized scale-space interest points. *Journal of Mathematical Imaging and Vision*, 52(1):3–36.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88.

Liu, C., Freeman, W. T., Szeliski, R., and Kang, S. B. (2006). Noise estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 901–908.

Liu, M., Tuzel, O., Ramalingam, S., and Chellappa, R. (2011). Entropy rate superpixel segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2097–2104.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37.

Liu, X., Yu, Y., Liu, B., and Li, Z. (2013). Bowstring-based dual-threshold computation method for adaptive Canny edge detector. In *Proceedings of the International Conference Image and Vision Computing New Zealand*, pages 13–18.

Liu, Y., Cheng, M., Hu, X., Wang, K., and Bai, X. (2017). Richer convolutional

features for edge detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5872–5881.

Liu, Z., Song, Y.-Q., Sheng, V. S., Wang, L., Jiang, R., Zhang, X., and Yuan, D. (2019). Liver CT sequence segmentation based with improved U-Net and graph cut. *Expert Systems with Applications*, 126:54–63.

Liu, Z., Yuan, L., Tang, X., Uyttendaele, M., and Sun, J. (2014). Fast burst images denoising. *ACM Transactions on Graphics*, 33(6):232:1–232:9.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

Lopes-dos Santos, R., Groot, R., Liying, S., Bossier, P., and Van Stappen, G. (2019). Halophilic bacteria as a food source for the brine shrimp *artemia*. *Aquaculture*, 500:631–639.

Lopez-Molina, C., Bustince, H., and De Baets, B. (2016). Separability criteria for the evaluation of boundary detection benchmarks. *IEEE Transactions on Image Processing*, 25(3):1047–1055.

Lopez-Molina, C., De Baets, B., and Bustince, H. (2013). Quantitative error measures for edge detection. *Pattern Recognition*, 46(4):1125–1139.

Lopez-Molina, C., Montero, J., Bustince, H., and De Baets, B. (2018). Self-adapting weighted operators for multiscale gradient fusion. *Information Fusion*, 44:136–146.

Lopez-Molina, C., Vidal-Diez de Ulzurrun, G., Baetens, J. M., Van Den Bulcke, J., and De Baets, B. (2015). Unsupervised ridge detection using second order anisotropic Gaussian kernels. *Signal Processing*, 116:55–67.

Luisier, F., Blu, T., and Unser, M. (2011). Image denoising in mixed Poisson-Gaussian noise. *IEEE Transactions on Image Processing*, 20(3):696–708.

Maggioni, M., Sánchez-Monge, E., and Foi, A. (2014). Joint removal of random and fixed-pattern noise through spatiotemporal video filtering. *IEEE Transactions on Image Processing*, 23(10):4282–4296.

Mäkinen, Y., Azzari, L., and Foi, A. (2019). Exact transform-domain noise variance for collaborative filtering of stationary correlated noise. In *Proceedings of the IEEE International Conference on Image Processing*, pages 185–189.

Mallat, S. and Hwang, W. L. (1992). Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643.

Mallat, S. and Zhong, S. (1992). Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (7):710–732.

Marr, D. and Hildreth, E. (1980). Theory of edge detection. In *Proceedings of Royal Society of London*, volume B207, pages 187–217.

Martin, D. R. (2003). *An Empirical Approach to Grouping and Segmentation*. PhD thesis, University of California, Berkeley.

Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549.

Matrecano, M., Poggi, G., and Verdoliva, L. (2012). Improved BM3D for correlated noise removal. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 129–134.

McIlhagga, W. (2011). The Canny edge detector revisited. *International Journal of Computer Vision*, 91(3):251–261.

Meijering, E., Carpenter, A. E., Peng, H., Hamprecht, F. A., and Olivo-Marin, J.-C. (2016). Imagining the future of bioimage analysis. *Nature Biotechnology*, 34(12):1250–1255.

Meijering, E., Jacob, M., Sarria, J.-C., Steiner, P., Hirling, H., and Unser, M. (2004). Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry Part A*, 58(2):167–176.

Miranda, J. M. and Romero, M. (2017). A prototype to measure rainbow trout's length using image processing. *Aquacultural Engineering*, 76:41–49.

Misimi, E., Erikson, U., and Skavhaug, A. (2008). Quality grading of atlantic salmon (*Salmo salar*) by computer vision. *Journal of Food Science*, 73(5):E211–E217.

Moon, W. K., Shen, Y.-W., Bae, M. S., Huang, C.-S., Chen, J.-H., and Chang, R.-F. (2013). Computer-aided tumor detection based on multi-scale blob detection algorithm in automated breast ultrasound images. *IEEE Transactions on Medical Imaging*, 32(7):1191–1200.

Moranduzzo, T. and Melgani, F. (2013). Automatic car counting method for unmanned aerial vehicle images. *IEEE Transactions on Geoscience and Remote Sensing*, 52(3):1635–1647.

Muñoz-Benavent, P., Andreu-García, G., Valiente-González, J. M., Atienza-Vanacloig, V., Puig-Pons, V., and Espinosa, V. (2018). Enhanced fish

bending model for automatic tuna sizing using computer vision. *Computers and Electronics in Agriculture*, 150:52–61.

Nam, S., Hwang, Y., Matsushita, Y., and Joo Kim, S. (2016). A holistic approach to cross-channel image noise modeling and its application to image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1683–1691.

Obara, B., Fricker, M., Gavaghan, D., and Grau, V. (2012a). Contrast-independent curvilinear structure detection in biomedical images. *IEEE Transactions on Image Processing*, 21(5):2572–2581.

Obara, B., Grau, V., and Fricker, M. D. (2012b). A bioimage informatics approach to automatically extract complex fungal networks. *Bioinformatics*, 28(18):2374–2381.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

Oliveira, A., Pereira, S., and Silva, C. A. (2018). Retinal vessel segmentation based on fully convolutional neural networks. *Expert Systems with Applications*, 112:229–242.

Oliveira, H. and Correia, P. L. (2013). Automatic road crack detection and characterization. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):155–168.

Ortner, M., Descombe, X., and Zerubia, J. (2008). A marked point process of rectangles and segments for automatic analysis of digital elevation models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):105–119.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

Pan, P., Li, J., Lv, G., Yang, H., Zhu, S., and Lou, J. (2009). Prediction of shelled shrimp weight by machine vision. *Journal of Zhejiang University Science B*, 10(8):589–594.

Pan, X., Ye, Y., Wang, J., Gao, X., He, C., Wang, D., Jiang, B., and Li, L. (2014). Complex composite derivative and its application to edge detection. *SIAM Journal on Imaging Sciences*, 7(4):2807–2832.

Pang, Y., Yuan, Y., Li, X., and Pan, J. (2011). Efficient HOG human detection. *Signal Processing*, 91(4):773–781.

Park, C., Huang, J. Z., Ji, J. X., and Ding, Y. (2013). Segmentation, inference and classification of partially overlapping nanoparticles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):669–681.

Parvin, B., Yang, Q., Han, J., Chang, H., Rydberg, B., and Barcellos-Hoff, M. H. (2007). Iterative voting for inference of structural saliency and characterization of subcellular events. *IEEE Transactions on Image Processing*, 16(3):615–623.

Pele, O. and Werman, M. (2010). The Quadratic-Chi histogram distance family. In *Proceedings of the European Conference on Computer Vision*, pages 749–762.

Peng, X., Tang, Z., Yang, F., Feris, R. S., and Metaxas, D. (2018). Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2226–2234.

Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639.

Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., and Toyama, K. (2004). Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3):664–672.

Pižurica, A., Portilla, J., Hirakawa, K., and Egiazarian, K. (2013). Advanced statistical tools for enhanced quality digital imaging with realistic capture models.

Plotz, T. and Roth, S. (2017). Benchmarking denoising algorithms with real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1586–1595.

Po, D.-Y. and Do, M. N. (2006). Directional multiscale modeling of images using the contourlet transform. *IEEE Transactions on Image Processing*, 15(6):1610–1620.

Pont-Tuset, J., Arbelaez, P., Barron, J. T., Marques, F., and Malik, J. (2017). Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):128–140.

Portilla, J., Strela, V., Wainwright, M. J., and Simoncelli, E. P. (2003). Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions Image Processing*, 12(11).

Prewitt, J. M. (1970). Object enhancement and extraction. *Picture Processing and Psychopictorics*, 10(1):15–19.

Prokop, R. J. and Reeves, A. P. (1992). A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphical Models and Image Processing*, 54(5):438–460.

Püspöki, Z., Sage, D., Ward, J. P., and Unser, M. (2016). Spotcaliper: Fast wavelet-based spot detection with accurate size estimation. *Bioinformatics*, 32(8):1278–1280.

Pyo, Y. I., Park, R. H., and Chang, S. (2011). Noise reduction in high-ISO images using 3-D collaborative filtering and structure extraction from residual blocks. *IEEE Transactions on Consumer Electronics*, 57(2):687–695.

Qin, Y., Feng, M., Lu, H., and Cottrell, G. W. (2018). Hierarchical cellular automata for visual saliency. *International Journal of Computer Vision*, 126(7):751–770.

Qin, Y., Lu, H., Xu, Y., and Wang, H. (2015). Saliency detection via cellular automata. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 110–119.

Quelhas, P., Marcuzzo, M., Mendonça, A. M., and Campilho, A. (2010). Cell nuclei and cytoplasm joint segmentation using the sliding band filter. *IEEE Transactions on Medical Imaging*, 29(8):1463–1473.

Rabie, T. (2004). Adaptive hybrid mean and median filtering of high-ISO long-exposure sensor noise for digital photography. *Journal of Electronic Imaging*, 13(2):264.

Rasouli, A., Kotseruba, I., and Tsotsos, J. K. (2017). Agreeing to cross: How drivers and pedestrians communicate. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 264–269.

Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449.

Ray, K. (2013). Unsupervised edge detection and noise detection from a single image. *Pattern Recognition*, 46(8):2067–2077.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

Remez, T., Litany, O., Giryes, R., and Bronstein, A. M. (2018). Class-aware

fully convolutional Gaussian and Poisson denoising. *IEEE Transactions on Image Processing*, 27(11):5707–5722.

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

Ren, Y., Ying, Z., Li, T. H., and Li, G. (2019). LECARM: Low-light image enhancement using the camera response model. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(4):968–981.

Rivera, A. R., Ryu, B., and Chae, O. (2012). Content-aware dark image enhancement through channel division. *IEEE Transactions on Image Processing*, 21(9):3967–3980.

Rivest, J.-F., Beucher, S., and Delhomme, J. (1992). Marker-controlled segmentation: An application to electrical borehole imaging. *Journal of Electronic Imaging*, 1(2):136–143.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241.

Rosenfeld, A. and Thurston, M. (1971). Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, C-20(5):562–569.

Rosenfeld, A., Thurston, M., and Lee, Y.-H. (1972). Edge and curve detection: Further experiments. *IEEE Transactions on Computers*, 100(7):677–715.

Rosin, P. L. (2001). Unimodal thresholding. *Pattern Recognition*, 34(11):2083–2096.

Ruck, D. W., Rogers, S. K., Kabrisky, M., Oxley, M. E., and Suter, B. W. (1990). The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298.

Ruusuvuori, P., Äijö, T., Chowdhury, S., Garmendia-Torres, C., Selinummi, J., Birbaumer, M., Dudley, A. M., Pelkmans, L., and Yli-Harja, O. (2010). Evaluation of methods for detection of fluorescence labeled subcellular objects in microscope images. *BMC Bioinformatics*, 11(1):248.

Ruusuvuori, P., Lehmussola, A., Selinummi, J., Rajala, T., Huttunen, H., and Yli-Harja, O. (2008). Benchmark set of synthetic images for validating cell image analysis algorithms. In *Proceedings of the European Signal Processing Conference*, pages 1–5.

Saberioon, M. and Císař, P. (2018). Automated within tank fish mass estimation using infrared reflection system. *Computers and Electronics in Agriculture*, 150:484–492.

Saglam, A. and Baykan, N. A. (2017). Sequential image segmentation based on minimum spanning tree representation. *Pattern Recognition Letters*, 87:155–162.

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., Tinevez, J.-Y., White, D., Hartenstein, V., Eliceiri, K., Tomancak, P., and Cardona, A. (2012). Fiji: An open-source platform for biological-image analysis. *Nature Methods*, 9(7):676.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schmitt, O. and Hasse, M. (2009). Morphological multiscale decomposition of connected regions with emphasis on cell clusters. *Computer Vision and Image Understanding*, 113(2):188–201.

Selesnick, I. W., Baraniuk, R. G., and Kingsbury, N. G. (2005). The dual-tree complex wavelet transform. *IEEE Signal Processing Magazine*, 22(6):123–151.

Shao, L., Yan, R., Li, X., and Liu, Y. (2014). From heuristic optimization to dictionary learning: A review and comprehensive comparison of image denoising algorithms. *IEEE Transactions on Cybernetics*, 44(7):1001–1013.

Shapley, R. and Hawken, M. J. (2011). Color in the cortex: Single-and double-opponent cells. *Vision Research*, 51(7):701–717.

Shen, W., Zhao, K., Jiang, Y., Wang, Y., Bai, X., and Yuille, A. (2017). DeepSkeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images. *IEEE Transactions on Image Processing*, 26(11):5298–5311.

Shi, J., Yan, Q., Xu, L., and Jia, J. (2016). Hierarchical image saliency detection on extended CSSD. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):717–729.

Shui, P. and Wang, F. (2017). Anti-impulse-noise edge detection via anisotropic morphological directional derivatives. *IEEE Transactions on Image Processing*, 26(10):4962–4977.

Shui, P. and Zhang, W. (2012). Noise-robust edge detector combining isotropic and anisotropic Gaussian kernels. *Pattern Recognition*, 45(2):806–820.

Shui, P. and Zhang, W. (2013). Corner detection and classification using anisotropic directional derivative representations. *IEEE Transactions on Image Processing*, 22(8):3204–3218.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*.

Sironi, A., Lepetit, V., and Fua, P. (2015). Projection onto the manifold of elongated structures for accurate extraction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 316–324.

Sironi, A., Türetken, E., Lepetit, V., and Fua, P. (2016). Multiscale centerline detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1327–1341.

Smal, I., Loog, M., Niessen, W., and Meijering, E. (2010). Quantitative comparison of spot detection methods in fluorescence microscopy. *IEEE Transactions on Medical Imaging*, 29(2):282–301.

Smith, S. M. and Brady, J. M. (1997). SUSAN—A new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78.

Sobel, I. (1970). *Camera Models and Machine Perception*. PhD thesis, Stanford University.

Sochen, N., Kimmel, R., and Malladi, R. (1998). A general framework for low level vision. *IEEE Transactions on Image Processing*, 7(3):310–318.

Sofka, M. and Stewart, C. V. (2006). Retinal vessel centerline extraction using multiscale matched filters, confidence and edge measures. *IEEE Transactions on Medical Imaging*, 25(12):1531–1546.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Starck, J.-L., Candès, E. J., and Donoho, D. L. (2002). The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6):670–684.

Steger, C. (1998). An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):113–125.

Steger, C. (2013). Unbiased extraction of lines with parabolic and Gaussian profiles. *Computer Vision and Image Understanding*, 117(2):97–112.

Strachan, N. (1993). Length measurement of fish by computer vision. *Computers and Electronics in Agriculture*, 8(2):93–104.

Strachan, N. (1994). Sea trials of a computer vision based fish species sorting and size grading machine. *Mechatronics*, 4(8):773–783.

Stutz, D., Hermans, A., and Leibe, B. (2018). Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27.

Sun, X., Wu, P., and Hoi, S. C. (2018). Face detection using deep learning: An improved faster RCNN approach. *Neurocomputing*, 299:42–50.

Szegedy, C., Toshev, A., and Erhan, D. (2013). Deep neural networks for object detection. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 2553–2561.

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., and Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312.

Tiirola, J. (2019). A learning based approach to additive, correlated noise removal. *Journal of Visual Communication and Image Representation*, 62:286–294.

Toi, H. T., Boeckx, P., Sorgeloos, P., Bossier, P., and Van Stappen, G. (2013). Bacteria contribute to *Artemia* nutrition in algae-limited conditions: A laboratory study. *Aquaculture*, 388:1–7.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 98, pages 836–846.

Torre, V. and Poggio, T. A. (1986). On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):147–163.

Tsin, Y., Ramesh, V., and Kanade, T. (2001). Statistical calibration of CCD imaging process. In *Proceedings the IEEE International Conference on Computer Vision*, volume 1, pages 480–487.

Vajda, S., Rangoni, Y., and Cecotti, H. (2015). Semi-automatic ground truth generation using unsupervised clustering and limited manual labeling: Application to handwritten character recognition. *Pattern Recognition Letters*, 58:23–28.

Vidal-Diez de Ulzurrun, G., Baetens, J. M., Van den Bulcke, J., Lopez-Molina, C., De Windt, I., and De Baets, B. (2015). Automated image-based analysis of spatio-temporal fungal dynamics. *Fungal Genetics and Biology*, 84:12–25.

Vincent, L. (1993). Morphological grayscale reconstruction in image analy-

sis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201.

Viola, P., Jones, M., et al. (2001). Robust real-time object detection. *International Journal of Computer Vision*, 4(34-47):4.

Vo, D. M., Nguyen, N.-Q., and Lee, S.-W. (2019). Classification of breast cancer histology images using incremental boosting convolution networks. *Information Sciences*, 482:123–138.

Wang, C., Elazab, A., Wu, J., and Hu, Q. (2017a). Lung nodule classification using deep feature fusion in chest radiography. *Computerized Medical Imaging and Graphics*, 57:10–18.

Wang, F. and Shui, P. (2016). Noise-robust color edge detector using gradient matrix and anisotropic Gaussian directional derivative matrix. *Pattern Recognition*, 52:346–357.

Wang, G. and De Baets, B. (2017). Edge detection based on the fusion of multiscale anisotropic edge strength measurements. In *Proceedings of the Conference of the European Society for Fuzzy Logic and Technology*, volume 3, pages 530–536.

Wang, G. and De Baets, B. (2019a). Contour detection based on anisotropic edge strength and hierarchical superpixel contrast. *Signal, Image and Video Processing*, 13(8):1657–1665.

Wang, G. and De Baets, B. (2019b). Superpixel segmentation based on anisotropic edge strength. *Journal of Imaging*, 5(6):57.

Wang, G., Lopez-Molina, C., and De Baets, B. (2017b). Blob reconstruction using unilateral second order Gaussian kernels with application to high-ISO long-exposure image denoising. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4817–4825.

Wang, G., Lopez-Molina, C., and De Baets, B. (2019a). High-ISO long-exposure image denoising based on quantitative blob characterization. *IEEE Transactions on Image Processing*, Under revision.

Wang, G., Lopez-Molina, C., and De Baets, B. (2019b). Multiscale edge detection using first-order derivative of anisotropic Gaussian kernels. *Journal of Mathematical Imaging and Vision*, 61(8):1096–1111.

Wang, G., Lopez-Molina, C., and De Baets, B. (2020). Automated blob detection using iterative Laplacian of Gaussian filtering and unilateral second-order Gaussian kernels. *Digital Signal Processing*, 96:102592.

Wang, G., Lopez-Molina, C., Vidal-Diez de Ulzurrun, G., and De Baets, B. (2019c). Noise-robust line detection using normalized and adaptive second-order anisotropic Gaussian kernels. *Signal Processing*, 160:252–262.

Wang, G., Van Stappen, G., and De Baets, B. (2019d). Automated *artemia* length measurement using U-shaped fully convolutional networks and second-order anisotropic Gaussian kernels. *Computers and Electronics in Agriculture*, Accepted.

Wang, G., Van Stappen, G., and De Baets, B. (2019e). Automated detection and counting of *artemia* using U-shaped fully convolutional networks and deep convolutional networks. *Expert Systems with Applications*, Under review.

Wang, M., Liu, X., Gao, Y., Ma, X., and Soomro, N. Q. (2017c). Superpixel segmentation: A benchmark. *Signal Processing: Image Communication*, 56:28–39.

Wei, X., Yang, Q., Gong, Y., Ahuja, N., and Yang, M. (2018). Superpixel hierarchy. *IEEE Transactions on Image Processing*, 27(10):4838–4849.

White, D. J., Svellingen, C., and Strachan, N. J. (2006). Automated measurement of species and length of fish by computer vision. *Fisheries Research*, 80(2-3):203–210.

Xiao, C., Staring, M., Wang, Y., Shamonin, D. P., and Stoel, B. C. (2013). Multiscale bi-Gaussian filter for adjacent curvilinear structures detection with application to vasculature images. *IEEE Transactions on Image Processing*, 22(1):174–188.

Xie, S. and Tu, Z. (2017). Holistically-nested edge detection. *International Journal of Computer Vision*, 125(1-3):3–18.

Xing, F., Xie, Y., Su, H., Liu, F., and Yang, L. (2018). Deep learning in microscopy image analysis: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4550–4568.

Xing, F. and Yang, L. (2016). Robust nucleus/cell detection and segmentation in digital pathology and microscopy images: A comprehensive review. *IEEE Reviews in Biomedical Engineering*, 9:234–263.

Xu, J., Xiang, L., Liu, Q., Gilmore, H., Wu, J., Tang, J., and Madabhushi, A. (2016). Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. *IEEE Transactions on Medical Imaging*, 35(1):119–130.

Xu, J., Zhang, L., and Zhang, D. (2018a). External prior guided internal prior

learning for real-world noisy image denoising. *IEEE Transactions on Image Processing*, 27(6):2996–3010.

Xu, J., Zhang, L., and Zhang, D. (2018b). A trilateral weighted sparse coding scheme for real-world image denoising. In *Proceedings of the European Conference on Computer Vision*, pages 20–36.

Xu, J., Zhang, L., Zhang, D., and Feng, X. (2017a). Multi-channel weighted nuclear norm minimization for real color image denoising. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1105–1113.

Xu, L. and Lu, H. (2013). Automatic morphological measurement of the quantum dots based on marker-controlled watershed algorithm. *IEEE Transactions on Nanotechnology*, 12(1):51–56.

Xu, L., Lu, H., and Zhang, M. (2014a). Automatic segmentation of clustered quantum dots based on improved watershed transformation. *Digital Signal Processing*, 34:108–115.

Xu, Q., Varadarajan, S., Chakrabarti, C., and Karam, L. J. (2014b). A distributed Canny edge detector: Algorithm and FPGA implementation. *IEEE Transactions on Image Processing*, 23(7):2944–2960.

Xu, S., Liu, H., and Song, E. (2011). Marker-controlled watershed for lesion segmentation in mammograms. *Journal of Digital Imaging*, 24(5):754–763.

Xu, Y., Carlinet, E., Géraud, T., and Najman, L. (2017b). Hierarchical segmentation using tree-based shape spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):457–469.

Xu, Z., Shin, B.-S., and Klette, R. (2015). Accurate and robust line segment extraction using minimum entropy with Hough transform. *IEEE Transactions on Image Processing*, 24(3):813–822.

Yang, J. and Shi, Y. (2014). Towards finger-vein image restoration and enhancement for finger-vein recognition. *Information Sciences*, 268:33–52.

Yang, K., Gao, S., Guo, C., Li, C., and Li, Y. (2015a). Boundary detection using double-opponency and spatial sparseness constraint. *IEEE Transactions on Image Processing*, 24(8):2565–2578.

Yang, K., Li, M., Liu, Y., Cheng, L., Huang, Q., and Chen, Y. (2015b). River detection in remotely sensed imagery using Gabor filtering and path opening. *Remote Sensing*, 7(7):8779–8802.

Yang, K. F., Li, C. Y., and Li, Y. J. (2014). Multifeature-based surround inhibition improves contour detection in natural images. *IEEE Transactions on Image Processing*, 23(12):5020–5032.

Yang, X., Li, H., and Zhou, X. (2006). Nuclei segmentation using marker-controlled watershed, tracking using mean-shift, and Kalman filter in time-lapse microscopy. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(11):2405–2414.

Yi, S., Labate, D., Easley, G. R., and Krim, H. (2009). A shearlet approach to edge analysis and detection. *IEEE Transactions on Image Processing*, 18(5):929–941.

You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016). Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659.

Yu, P., Qin, A., and Clausi, D. A. (2012). Unsupervised polarimetric SAR image segmentation and classification using region growing with edge penalty. *IEEE Transactions on Geoscience and Remote Sensing*, 50(4):1302–1317.

Yuan, J., Gleason, S. S., and Cheriyadat, A. M. (2013). Systematic benchmarking of aerial image segmentation. *IEEE Geoscience and Remote Sensing Letters*, 10(6):1527–1531.

Zafari, S., Eerola, T., Sampo, J., Kälviäinen, H., and Haario, H. (2015). Segmentation of overlapping elliptical objects in silhouette images. *IEEE Transactions on Image Processing*, 24(12):5942–5952.

Zhang, G., Jia, X., and Hu, J. (2015a). Superpixel-based graphical model for remote sensing image mapping. *IEEE Transactions on Geoscience and Remote Sensing*, 53(11):5861–5871.

Zhang, H., Fritts, J. E., and Goldman, S. A. (2008). Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, 110(2):260–280.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. (2017a). Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155.

Zhang, L. and Bao, P. (2002). Edge detection by scale multiplication in wavelet domain. *Pattern Recognition Letters*, 23(14):1771–1784.

Zhang, M. and Gunturk, B. K. (2008). Multiresolution bilateral filtering for image denoising. *IEEE Transactions on Image Processing*, 17(12):2324–2333.

Zhang, M., Wu, T., and Bennett, K. M. (2015b). Small blob identification in medical images using regional features from optimum scale. *IEEE Transactions on Biomedical Engineering*, 62(4):1051–1062.

Zhang, W., Zhao, Y., Breckon, T. P., and Chen, L. (2017b). Noise robust image edge detection based upon the automatic anisotropic Gaussian kernels. *Pattern Recognition*, 63:193–205.

Zhang, W.-H., Jiang, X., and Liu, Y.-M. (2012). A method for recognizing overlapping elliptical bubbles in bubble image. *Pattern Recognition Letters*, 33(12):1543–1548.

Zhang, Z., Liu, Q., and Wang, Y. (2018). Road extraction by deep residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753.

Zhou, Z., Shin, J., Zhang, L., Gurudu, S., Gotway, M., and Liang, J. (2017). Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7340–7351.

Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *Proceedings of the European Conference on Computer Vision*, pages 391–405.

Curriculum Vitae

Personalia and contact

Name	Gang Wang
Place of birth	Shandong, P. R. China
Nationality	P. R. China
ORCID	https://orcid.org/0000-0002-1916-6110
E-mail	gang.wang@ugent.be g_wang@foxmail.com

Educations

- 2011-2013: MSc. in Navigation, Guidance and Control, Mechanical Engineering College, Shijiazhuang, China.
- 2007-2011: BSc. in Communication Engineering, Dalian Maritime University, Dalian, China.

Employment

- 2015-present: Full-time researcher at the Research Unit Knowledge-Based Systems, Department of Data Analysis and Mathematical Modelling, Ghent University, Ghent, Belgium.

Research interests

Image processing; Computer vision; Machine learning

Languages

- Chinese: Native speaker
- English: Fluent

Programming languages

MATLAB; Python; Keras; TensorFlow; C/C++

Awards during the doctoral research

- 2016: *First Prize* of the IDS2016 Competition on Image Denoising at the Second International Conference on Intelligent Decision Science held in Dubai, United Arabic Emirates.
- 2017: *Best Student Paper Award Nomination* at the Tenth Conference of the European Society for Fuzzy Logic and Technology held in Warsaw, Poland.
- 2019: *Best Student Paper Nomination* at the 31st Benelux Conference on Artificial Intelligence held in Brussels, Belgium.

Scientific outputs during the doctoral research

Publications in international journals

- **Wang, G.**, Lopez-Molina, C., Vidal-Diez de Ulzurrun, G., and De Baets, B. (2019f). Noise-robust line detection using normalized and adaptive second-order anisotropic Gaussian kernels. *Signal Processing*, 160:252-262
- **Wang, G.** and De Baets, B. (2019a). Contour detection based on anisotropic edge strength and hierarchical superpixel contrast. *Signal, Image and Video Processing*, 13(8):1657-1665
- **Wang, G.**, Lopez-Molina, C., and De Baets, B. (2019e). Multiscale edge detection using first-order derivative of anisotropic Gaussian kernels. *Journal of Mathematical Imaging and Vision*, 61(8):1096–1111
- **Wang, G.**, Lopez-Molina, C., and De Baets, B. (2020). Automated blob detection using iterative Laplacian of Gaussian filtering and unilateral second-order Gaussian kernels. *Digital Signal Processing*, 96:102592
- **Wang, G.** and De Baets, B. (2019b). Superpixel segmentation based on anisotropic edge strength. *Journal of Imaging*, 5(6):57
- **Wang, G.**, Van Stappen, G., and De Baets, B. (2019b). Automated *Artemia* length measurement using U-shaped fully convolutional networks and second-order anisotropic Gaussian kernels. *Computers and Electronics in Agriculture*, Accepted

- **Wang, G.**, Lopez-Molina, C., and De Baets, B. (2019d). High-ISO long-exposure image denoising based on quantitative blob characterization. *IEEE Transactions on Image Processing*, Under revision

- **Wang, G.**, Van Stappen, G., and De Baets, B. (2019a). Automated detection and counting of *Artemia* using U-shaped fully convolutional networks and deep convolutional networks. *Expert Systems with Applications*, Under review

Conference proceedings

- **Wang, G.**, Lopez-Molina, C., and De Baets, B. (2017). Blob reconstruction using unilateral second order Gaussian kernels with application to high-ISO long-exposure image denoising. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4817-4825

- **Wang, G.** and De Baets, B. (2017). Edge detection based on the fusion of multiscale anisotropic edge strength measurements. In *Proceedings of the Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)*, volume 3, pages 530-536

- **Wang, G.**, Lopez-Molina, C., and De Baets, B. (2016). Blob noise reduction using unilateral second order Gaussian kernels and the non-local means algorithm. In *Proceedings of the International Conference on Intelligent Decision Science (IDS)*

- **Wang, G.** and De Baets, B. (2019). Automated *Artemia* detection and length measurement using deep convolutional networks. In *Proceedings of the Benelux Conference on Artificial Intelligence (BNAIC)*