LA TROBE UNIVERSITY

DOCTORAL THESIS

Optimisation Techniques for Signal and Image Processing

Submitted by

Waseem WAHEED

B.S., Salahaddin University, 2009

M.S., La Trobe University, 2014

A thesis submitted in total fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Department of Engineering

School of Engineering and Mathematical Sciences

Melbourne, Victoria 3086

Australia

June, 2020

Statement of Authorship

Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis accepted for the award of any other degree or diploma. No other person's work has been used without due acknowledgement in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution.

Name: Waseem WAHEED
Date: 30 June 2020

Abstract

Mathematical optimization has and continues to fuel much of the success in many engineering disciplines. More specifically, it is behind a significant portion of the success in signal processing, image processing, computer vision and machine learning in general. The research behind this thesis aims at investigating important problems and developing effective optimization models and algorithms in the areas of digital signal and image processing. The important problems studied in this thesis include denoising signals defined on graphs and image filtering.

Denoising signals defined on graphs is the first problem studied in this thesis. Classic digital signal processing is concerned with signals defined at equally distanced discrete time steps which is a restricting assumption. A more recent advancement in the field of signal processing is the extension of classical signal processing techniques to the signals residing on the nodes of an irregular graph. An interesting problem is how to denoise signals defined on such irregular graphs. In this thesis, an efficient algorithm for denoising signals defined on irregular and directed graphs is developed.

Edge-aware image smoothing is the second problem studied in this thesis. Standard linear time invariant (LTI) filters are very efficient but lack the discriminative power between the different regions of the image, a desirable property of image filters. In particular, it is an important property for smoothing filters to respect the edges image and stop smoothing across significant edges. To that end, this thesis proposes a technique that can turn the results of any non edge-aware filter to an edge-aware filtered image. The proposed technique is formulated as a patch level optimization problem.

Applications of the fractional Laplacian operator to signal and image processing is the third problem investigated in this thesis. The Laplacian operator is a ubiquitous tool in the applied mathematics, in general, and in the signal and image processing literature, in particular. The utility of the Laplacian operator has fostered many generalizations, one such generalization is the fractional order Laplacian. In this thesis, a technique for the easy construction of the discrete fractional Laplacian operator is developed. The developed technique makes it easier to formulate signal and image processing algorithms that utilize the fractional Laplacian operator.

Publications

A number of papers have been published or are under review for publication in international journals based on the material and discussion in this thesis. The publications are listed as follows:

- W. Waheed and D. B. H. Tay, "Graph polynomial filter for signal denoising," in IET Signal Processing, vol. 12, no. 3, pp. 301-309, 5 2018, doi: 10.1049/iet-spr.2016.0700.
- W. Waheed, M. Al-nasrawi, and G. Deng, "Guided Adaptive Interpolation Filter" in IET Image Processing, 8 2020, doi: 10.1049/iet-ipr.2019.1577
- W. Waheed, and G. Deng, B. Liu "Discrete Laplacian operator and its applications in signal processing" in IEEE Access, doi: 10.1109/ACCESS.2020.2993577.

Acknowledgements

This work was supported by an Australian Government Research Training Program Scholarship. The thesis is the product of five years journey. It is the collaboration with many researchers that made it possible and is shaped by their contributions and generous help.

First and foremost, my deepest gratitude goes to my supervisor Dr. Guang (Dennis) Deng initially for agreeing to advise me after 1.5 years of my candidature and secondly for his tireless guidance and insightful advice at all stages of my candidature and finally for giving the freedom to pursue what I enjoyed most. Dennis's passion and deep knowledge infected me and forged my interest in image processing and optimization modelling.

I am thankful to my industry mentor at Aurecon, Dr. Slaven Marusic for providing me the opportunity and the environment to explore and to meet many researchers at various stages of their research career. During my time at Aurecon, I was able to utilize my research skills in solving real world problems that expanded my horizons.

I am thankful to my former mentors who planted the seed of passion for research in me. Dr. Samah Mostafa as my bachelors project mentor introduced me to research. Dr. David Tay as my Masters and early stage PhD advisor, introduced me to the interesting field of wavelets and graph signal processing. David's guidance laid the foundation for the following steps in my research career.

No journey is complete without friends and this one is no exception. Along this journey I had the privilege to make many great friends, some of whom have proven instrumental to my progress. I'd like to specifically thank Mukhalad Al-nasrawi, those debugging sessions, lengthy paper readings and the philosophical discussions have contributed significantly to my own development. I would also like to thank Hussien Al-bandawi for the daily walks and discussions, they helped me reduce the lengthy desk sessions.

Finally, this work would not have been possible without the tireless support and encouragement of my family along this journey. I would like to first thank my wife, Lourd Ibrahim, for putting up with my long working hours and not being able to spend more time at home, specially after our daughter,

Talia, was born. I would also like to thank my parents for their continuous support and encouragement, this work takes inspiration from their hard work and success.

Contents

Statement of Authorship iii

Abstract v

Acknowledgements ix

1 Introduction 1
- 1.1 Motivation and objective . . . 1
- 1.2 More formal introduction . . . 2
  - 1.2.1 Optimization definition . . . 2
  - 1.2.2 Convex optimization . . . 2
  - 1.2.3 Solutions to smooth convex problems . . . 3
  - 1.2.4 Non-smooth problems . . . 4
  - 1.2.5 Algorithms for non-smooth convex problems . . . 5
- 1.3 Overview of research outcomes . . . 6
- 1.4 Summary of contributions . . . 6
  - 1.4.1 Graph polynomial filter for signal denoising (Chapter 2) 6
  - 1.4.2 Guided adaptive interpolation filter (Chapter 3) . . . 7
  - 1.4.3 High-pass filter generalization of the total variation model and its performance (Chapter 4) . . . 7
  - 1.4.4 Discrete Laplacian operator and its applications in signal processing (Chapter 5) . . . 8

2 Graph Polynomial Filter for Signal Denoising 9
- 2.1 Introduction . . . 9
- 2.2 Discrete signal processing on graphs . . . 10
- 2.3 Graph signal denoising . . . 13
- 2.4 Graph polynomial filter . . . 14
  - 2.4.1 Approximate solution . . . 16
  - 2.4.2 Distributed processing and complexity comparison . . 20
- 2.5 Experimental results . . . 21

2.5.1 Temperature sensors measurement denoising 21
2.5.2 Image denoising 25
2.5.3 Discussion 26
2.6 Further comparisons 29
2.7 Chapter summary 33

3 Guided Adaptive Interpolation Filter **35**
3.1 Introduction 35
3.2 Related work 37
3.2.1 Kernel-based filters 37
3.2.2 Guided image filter 37
3.2.3 Energy minimization global filters 38
3.2.4 Interpolation based filters 39
3.2.5 Edge-preserving filtering 39
3.2.6 Our contribution 40
3.3 Guided adaptive interpolation filter 40
3.3.1 Definition 40
Analysis 43
3.3.2 Weighted adaptive interpolation filter 44
3.3.3 Filter kernel 47
3.3.4 $O(N)$ Time exact algorithm 48
3.4 Parameters setting and details smoothing 49
3.4.1 Parameter setting 49
3.4.2 The role of M 49
3.4.3 Details smoothing 60
3.5 Applications and Experimental Results 62
3.5.1 Single image haze removal 62
3.5.2 Flash/No-flash fusion denoising 65
3.5.3 Image detail enhancement 68
3.5.4 Edge detection 69
3.6 Chapter summary 70

4 High-pass filter generalization of the TV model **73**
4.1 Introduction 73
4.2 The high-pass filter generalization 74
4.3 Results 76
4.4 Chapter summary 80

5 Discrete Laplacian operator and its applications **81**
5.1 Introduction . . . 81
5.1.1 Fractional derivatives . . . 82
5.1.2 Fractional Laplacian (FL) . . . 82
5.1.3 Related works . . . 83
Fractional derivative operators in image processing . . . 83
Fractional Laplacian operators in image processing . . . 84
5.1.4 Contributions . . . 85
5.2 A Discrete fractional Laplacian operator . . . 86
5.2.1 Definition for 1D case . . . 86
5.2.2 Extension to the 2D case . . . 88
5.2.3 Computational complexity analysis . . . 88
5.2.4 Discussion . . . 89
5.3 Applications . . . 90
5.3.1 Fast trend filters in the DCT domain . . . 90
Definition . . . 90
Fractional ℓ_2 trend filter . . . 92
Fractional ℓ_1 trend filtering . . . 93
Discussion . . . 95
Numerical examples . . . 98
5.3.2 Image processing applications . . . 105
1D filtering . . . 105
Image sharpening . . . 106
Edge detection . . . 107
Shock filtering . . . 111
α scale-space . . . 117
5.4 Chapter summary . . . 119

6 Conclusion and future directions **121**

A Explicit kernel proof **125**

List of Figures

1.1 Example 2D convex functions 2
1.2 Example 2D non-smooth convex function 4

2.1 Vertex and graph spectral domains of a 2D grid graph (N=25) with a test signal defined by adding the first five eigenvectors ($\sum_{i=1}^{5} v_i$) and Gaussian noise with noise $\sigma = 0.1$. 12
2.2 Graph polynomial filter of degree ($L = 7$) designed using the least squares technique with $\lambda_{max} = 6.34$ and $\gamma = 10$. 20
2.3 Sensor Graph with N = 200 21
2.4 Sensor Graph with N = 400 22
2.5 Sensor Graph with N = 600 22
2.6 Images used in the image denoising experiment: Barbara, Lena, Mandrill, Peppers and Cameraman. 26
2.7 Visualisations of various graphs used from the GSP toolbox [45] 31
2.8 GPF vs Weighted average (FIR) 33

3.1 GAIF block-diagram. Patches at index k in the images I and M are interpolated to produce the corresponding patch in image J 41
3.2 Overlapping patches 43
3.3 Comparison between the solutions in (3.9) and (3.11). 44
3.4 Variant 1 of weighted GAIF. (a) is input image I. (b) is median filtered version of I with window size $= 11$. (c) and (e) are $\bar{\alpha}_p$ image in the case of standard GAIF (3.11) and variant 1 of the weighted GAIF (3.14) respectively. Light areas represent more contribution from I than M and darker regions represent more contribution from M than I. (d) and (f) are the results of GAIF and weighted GAIF respectively with $\epsilon = 1$. 46

3.5 Variant 2 of weighted GAIF. (a) is input image I. (b) is median filtered version of I with window size $= 11$. (c) and (e) are $\bar{\alpha}_p$ image in the case of standard GAIF (3.11) and variant 2 of the weighted GAIF (3.14) respectively. Light areas represent more contribution from I than M and darker regions represent more contribution from M than I. (d) and (f) are the results of GAIF and weighted GAIF respectively with $\epsilon = 1$. 47
3.6 Edge-preserving smoothing using GAIF filter with different kernel sizes r and values of the regularization parameter ϵ. Lower ϵ values correspond to sharper outputs. Larger patch radius r results in better edge preservation. 50
3.7 Input images used to demonstrate the role of the smooth image M. 51
3.8 The role of the smooth image M. (a) and (c) are M images produced using a Gaussian filter with $\sigma = 2$ and $\sigma = 7$ respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (a) while M being the image (a) and (c) respectively. 52
3.9 The role of the smooth image M. (a) and (c) are M images produced using a Median filter with window size 9 and 21 respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (a) while M being the image (a) and (c) respectively. 53
3.10 The role of the smooth image M. (a) and (c) are M images produced using a Gaussian filter with $\sigma = 2$ and $\sigma = 7$ respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (b) while M being the image (a) and (c) respectively. 54
3.11 The role of the smooth image M. (a) and (c) are M images produced using a Median filter with window size 9 and 21 respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (b) while M being the image (a) and (c) respectively. 55

3.12 The role of the smooth image M. (a) and (c) are M images produced using a Gaussian filter with $\sigma = 2$ and $\sigma = 7$ respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (c) while M being the image (a) and (c) respectively. 56
3.13 The role of the smooth image M. (a) and (c) are M images produced using a Median filter with window size 5 and 13 respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (c) while M being the image (a) and (c) respectively. 57
3.14 The role of the smooth image M. (a) and (c) are M images produced using a Gaussian filter with $\sigma = 2$ and $\sigma = 7$ respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (d) while M being the image (a) and (c) respectively. 58
3.15 The role of the smooth image M. (a) and (c) are M images produced using a Median filter with window size 9 and 21 respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (d) while M being the image (a) and (c) respectively. 59
3.16 Image smoothing (b) Gaussian $\sigma = 6$, (c) median 5×5 and GAIF $\epsilon = 0.04$ for all cases. The top part compares two results, to the left is a result of the indicated filter, to the right is a result of filtering with GAIF. The bottom part is zoomed-in versions of the top part, the red box is for the result of the indicated filter and the blue box is for the GAIF result. 60
3.17 Image smoothing (a) GIF $r = 5, \epsilon = 0.01$, (b) SWF $r = 5$, (c) WLS $\lambda = 0.05, \alpha = 1$ and GAIF $\epsilon = 0.04$ for all cases. The top part compares two results, to the left is a result of the indicated filter, to the right is a result of filtering with GAIF. The bottom part is zoomed-in versions of the top part, the red box is for the result of the indicated filter and the blue box is for the GAIF result. 61

3.18 Image smoothing (a) SD $\lambda = 5, \mu = 50, \nu = 400, \text{iter} = 10$, (b) RG $\sigma_s = 3, \sigma_r = 0.01, \text{iter} = 4$, (c) RTV $\lambda = 0.005, \sigma = 3$ and GAIF $\epsilon = 0.04$ for all cases. The top part compares two results, to the left is a result of the indicated filter, to the right is a result of filtering with GAIF. The bottom part is zoomed-in versions of the top part, the red box is for the result of the indicated filter and the blue box is for the GAIF result. 61
3.19 Single image de-hazing steps proposed in [97]. The black arrows refer to the original model and the red path includes the GAIF as the veil refinement step. 62
3.20 A comparison of image haze removal on road image. GAIF is used to filter the atmospheric veil estimate in the no-black pixel constraint (NBPC) technique [97]. In comparison, de-hazing using the original NBPC [97], NBPC+PA [98] and dark prior channel [71] techniques are evaluated. 63
3.21 A comparison of image haze removal on city image. GAIF is used to filter the atmospheric veil estimate in the no-black pixel constraint (NBPC) technique [97]. In comparison, de-hazing using the original NBPC [97], NBPC+PA [98] and dark prior channel [71] techniques are evaluated. 64
3.22 Flash/No flash denoising algorithm. 65
3.23 A comparison of image flash/no-flash denoising between GAIF with ($F_{base} : M = \text{boxfilter}(I, w = 50), \Omega_k = 3, \epsilon = 1, A_{NR} : M = \text{boxfilter}(I, w = 50), \Omega_k = 5, \epsilon = 20$), the joint bilateral filter [78], guided image filter [49] with ($r = 9, \epsilon = 0.0004$) and Semi-guided filter [99] with ($F_{base} : \sigma_s = 8.5, \sigma_r = 0.5, N = 5, A_{NR} : \sigma_s = 8.5, \sigma_r = 0.35, N = 5$). The proposed filter is superior to the first two and is on par with (e) but at less computational complexity. In (f) GAIF manages to capture all the important details as annotated. 66

3.24 A comparison of image flash/no-flash denoising between GAIF with (F_{base} : M = boxfilter$(I, w = 50), \Omega_k = 3, \epsilon = 1, A_{NR}$: M = boxfilter$(I, w = 50), \Omega_k = 5, \epsilon = 20$), the joint bilateral filter [78], guided image filter [49] with ($r = 9, \epsilon = 0.0004$) and Semi-guided filter [99] with (F_{base} : $\sigma_s = 8.5, \sigma_r = 0.5, N = 5, A_{NR}$: $\sigma_s = 8.5, \sigma_r = 0.35, N = 5$). The proposed filter is superior to the first two and is on par with (e) but at less computational complexity. In (f) GAIF manages to capture all the important details as annotated. 67

3.25 A comparison of Image details enhancement performance between GAIF (M = MEDFILT$(I, w = 13), \Omega_k = 3, \epsilon = 0.1$), weighted least-squares filter [47] ($\lambda = 0.125, \alpha = 1.2$) and the semi-guided filter [99] ($\sigma_s = 3.5, \sigma_r = 0.05, N = 5$). As annotated, the proposed filter excels at preserving the true edges of the input image while achieving comparable smoothing performance in other regions. 68

3.26 A comparison of Image details enhancement performance between GAIF ($M = $ MEDFILT$(I, w = 13), \Omega_k = 3, \epsilon = 0.00001$), weighted least-squares filter [47] ($\lambda = 0.125, \alpha = 1.12$) and the semi-guided filter [99] ($\sigma_s = 3.5, \sigma_r = 0.05, N = 5$). 69

3.27 Edge detection by preprocessing image using GAIF. (a) is input image, (b) is the edge map of the input image, (c) is the input image plus a Gaussian noise with $\mu = 0.5$ and $\sigma^2 = 0.05$, (d) is the edge map of the image in (c), (e) is average filtered version of the image in (c) with window size 9, (f) edge map of the image in (e), (g) is GAIF filtered image with I is the image in (c) and M is the image in (e), finally (h) is the edge map of the image in (g). 70

4.1 Amplitude response of the filter [1 -1] 74

4.2 Six types of non-stationary test signals. 76

4.3 Amplitude response of a sample of filters $H(\omega)$ used in this work of order $N = 12$. 77

4.4 Comparison between filtering a noisy signal using TV (4.2) vs using the proposed model (4.4); the proposed model doesn't suffer from the staircasing phenomenon. 80

5.1 The shrinkage effect of the fractional ℓ_2 trend filter (first term in (5.27)) using various values of α. 93
5.2 5×5 2D masks for filtering images in (5.30) (H_1, H_2) 94
5.3 Frequency response of fractional Laplacian l_α. 96
5.4 Impulse response of fractional Laplacian l_α. 97
5.5 Fractional ℓ_2 trend filtering of synthesized data. Dashed black is the synthetic data. Solid red is the original trend. Solid blue is the filter result with λ chosen to minimize the mean squared error between the filter output and the original trend. 100
5.6 Fractional ℓ_1 trend filtering of synthesized data. Dashed black is the synthetic data. Solid red is the original trend. Solid blue is the filter result with λ chosen to minimize the mean squared error between the filter output and the original trend. 101
5.7 Grayscale images (256×256) used in the denoising experiments. (a) cameraman, (b) house, (c) lena, (d) peppers, (e) pirate and (f) blonde woman. 104
5.8 Blocks signal (dashed black) filtered using fractional Laplacian l_α (solid red) with different values of α. From top to bottom are filtered signals with $\alpha \in \{0.1, 0.4, 0.7, 1\}$ 105
5.9 5×5 2D isotropic fractional Laplacian (H_α). 106
5.10 Image sharpening application. (a) is the original input image. (b) is sharpened image using the guided filter. (c) is sharpened image with $\alpha = 1$. (d) is sharpened image with $\alpha = 0.5$. Sharpened images are produced according to (5.32) with $\gamma = 2$. In the case of the guided filter, the residual of filtering is boosted ($J = I + \gamma(I - \mathrm{GF}(I))$. 107
5.11 Block diagram of the Marr-Hildreth edge detector (top) and the extended version with our fractional Laplacian (bottom). . 108
5.12 Fractional Marr-Hildreth edge detection. (a) is the original image. (b) is edge map produced using MATLAB's *edge* command (based on first order derivatives) using default values. (c) is edge map produced using the standard Marr-Hildreth. (d) is edge map produced using Fractional Marr-Hildreth with $\alpha = 0.2$. Gaussian smoothing with $\sigma = 1$ was used in this experiment. 109

5.13 Noise robustness of the fractional Marr-Hildreth edge detector. (a) is the original image. (b) is a noisy image formed by adding noise with $\sigma = 0.3$ to the original image. (c) is edge map produced using the standard Marr-Hildreth. (d) is edge map produced using fractional Marr-Hildreth with $\alpha = 0.1$. Gaussian smoothing with $\sigma = 1$ was used in this experiment. 110
5.14 α Shock filter of the top input image. Columns correspond to values of $\alpha \in \{0.6, 1\}$ from left to right. Rows correspond to values of $N \in \{1, 25, 100\}$ from left to right. The parameter λ was set to 0.1 for all results. 112
5.15 TV-L2 calculated for 600 iterations of the model in (5.36). Each curve corresponds to one value of $\alpha \in \{0.1, 0.5, 0.9\}$. 113
5.16 α Shock filter with stopping criterion Algorithm 5. Left column are input image. Middle column is the output images of shock filter with $\alpha = 1$ (Standard shock filter). Right column is the output of shock filtering with $\alpha = 0.1$. The parameter h was set to 0.05 in this experiment. 114
5.17 Scan-lines captured from the RGB (left to right) channels of the images shown in Figure 5.16. Using the fractional Laplacian results in more abstract lines. 115
5.18 Scan-line (line 350) captured from the red channel of the cat image shown in Figure 5.16. The fractional Laplacian with $\alpha = 0.1$ results in more positive and negative signals and less zeros compared to the standard Laplacian ($\alpha = 1$) 116
5.19 α scale-space filtered image on the top. Rows correspond to values of $\alpha \in \{0.2, 0.4, 0.7, 1\}$ from top to bottom. Columns correspond to values of $N \in \{1, 3, 10, 30\}$ from left to right. In the bottom right corner of each image is a plot of the α scale-space function in (5.42) with the corresponding values of α and N. 118

List of Tables

2.1 Temperature Sensors ($N = 200$) : RMSE of GTVR and GPF . . 23
2.2 Temperature Sensors ($N = 400$) : RMSE of GTVR and GPF . . 24
2.3 Temperature Sensors ($N = 600$) : RMSE of GTVR and GPF . . 24
2.4 Image Denoising (Barbara) : RMSE of GTVR and GPF 27
2.5 Image Denoising (Lena) : RMSE of GTVR and GPF 27
2.6 Image Denoising (Mandrill) : RMSE of GTVR and GPF 28
2.7 Image Denoising (Peppers) : RMSE of GTVR and GPF 28
2.8 Image Denoising (Cameraman) : RMSE of GTVR and GPF . . 29
2.9 Run-time comparison between the GTVR and GPF methods . 30
2.10 Denoising performance using Least-squares (LS) polynomial and Chebyshev polynomial (CP) 32

3.1 Computational complexities of some of the well-known filters in the literature. N is the total number of pixels in an image, $|\Omega_k|$ is number of pixels in the patch, w is the number of pixels in a window, and n is the number of iterations. 49

4.1 MAE of the six signals at various SNR levels. 79

5.1 Mean square error (MSE) of denoised image images for six images using fractional ℓ_1 trend filtering. 99
5.2 Mean square error (MSE) of denoised image images for six images using fractional ℓ_2 trend filtering. 102
5.3 Mean square error (MSE) of denoised image images for six images using BM3D. 103

List of Abbreviations

ADMM	**A**lternating **D**irection **M**ethod of **M**ultipliers
AIF	**A**daptive **I**nterpolation **F**ilter
CP	**C**hebyshev **P**olynomial
DCT	**D**iscrete **C**osine **T**ransform
DFT	**D**iscrete **F**ourier **T**ransform
DSP$_G$	**D**iscrete **S**ignal **P**rocessing on **G**raphs
FC	**F**ractional **C**alculus
FFT	**F**ast **F**ourier **T**ransform
FIR	**F**inite **I**mpulse **R**esponse
FL	**F**ractional **L**aplacian
GAIF	**G**uided **A**daptive **I**nterpolation **F**ilter
GIF	**G**uided **I**mage **F**ilter
GPF	**G**raph **P**olynomial **F**ilter
GSP	**G**raph **S**ignal **P**rocessing
GTV	**G**eneralized **T**otal **V**ariation
GTVR	**G**raph **T**otal **V**ariation **R**egularization
HDR	**H**igh **D**ynamic **R**ange
IGO	**I**terative **G**lobal **O**ptimization
IIR	**I**nfinite **I**mpulse **R**esponse
JBF	**J**oint **B**ilateral **F**ilter
LS	**L**east **S**quares
LSP	**L**east-**S**quares **P**olynomial
LTI	**L**inear **T**ime **I**nvariant
MAE	**M**ean **A**bsolute **E**rror
MEDFILT	**Med**ian **Filt**er
MRI	**M**agnetic **R**esonance **I**maging
MSE	**M**ean **S**quare **E**rror
NBPC	**N**o-**B**lack **P**ixel **C**onstraint
PDE	**P**artial **D**ifferential **E**quation
RGB-NIR	**R**ed **G**reen **B**lue - **N**ear **I**nfrared

RMSE	**R**oot **M**ean **S**quare **E**rror
RTV	**R**elative **T**otal **V**ariation
SD	**S**tatic **D**ynamic
SGBF	**S**emi-**G**uided **B**ilateral **F**ilter
SNR	**S**ignal to **N**oise **R**atio
SPSD	**S**ymmetric **P**ositive **S**emi-**D**efinite
TV	**T**otal **V**ariation
WLS	**W**eighted **L**east-**S**quares

Dedicated to the love of my life

LOURD,

and my little sweetheart

TALIA

for their unconditional love, patience and support.

Chapter 1

Introduction

1.1 Motivation and objective

Mathematical optimization has fuelled success in numerous fields with applications in engineering (e.g. control engineering [1], antenna design [2]), computer science (e.g. machine learning [3], computer vision [4, 5], signal [6] and image processing [7]), operations research [8] (e.g. logistics, scheduling and planning), economics [9], etc. Mathematical optimization is a wide field that is generally divided into a number of sub-fields based on the properties of the objectives and constrains.

The work in this thesis is mainly focused around a specific category of optimization problems, problems that are continuous, linear or non-linear, and convex in nature. What is interesting about this family of objective functions is twofold: firstly, they represent a wide range of objective functions which translates to a large number of applications. Secondly, convex objective functions are provably tractable, in other words, they have globally unique solutions.

Convex optimization is a rich area of research however, the interest in this thesis can be summarized as follows:

1. Developing, computationally efficient, and approximate, solutions to signal and image processing optimization problems.

2. Utilizing convex optimization as a mathematical modelling technique for solving signal and image processing problems

1.2 More formal introduction

To formally introduce the mathematical optimization of interest to this work, the main concepts are broken down into the following subsections.

1.2.1 Optimization definition

1.2.2 Convex optimization

Generally, a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be convex if

$$f(\epsilon x + (1-\epsilon)y) \leq \epsilon f(x) + (1-\epsilon)f(y), \qquad \forall x, y \in \mathbb{R}^n, \forall \epsilon [0,1] \tag{1.1}$$

For scalar and 2D functions, a visualization can be attained as shown in Figure 1.1

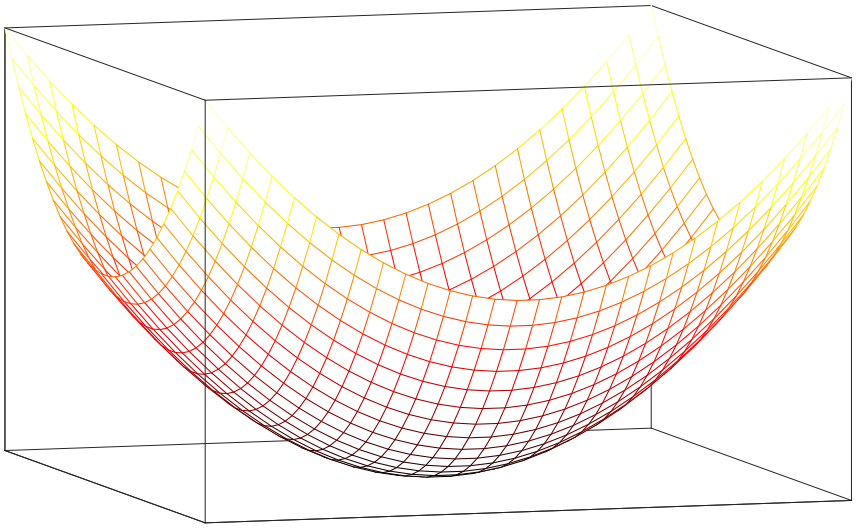


FIGURE 1.1: Example 2D convex functions

The functions of interest in this thesis take the form

$$\mathcal{E}(x,y) = \underset{x}{\operatorname{argmin}} \left\{ \mathcal{D}(x,y) + \mathcal{R}(x) \right\}, \tag{1.2}$$

where $\mathcal{D}(x,y)$ is known as the data term, which is a distance measure between the observed signal y and the reconstructed signal x, and $\mathcal{R}(x)$ is a regularizer which avoids the optimization algorithm from taking up trivial

solutions and encodes our knowledge about the solution x. A parallel interpretation to the function in (1.2) comes from Bayesian inference perspective [10], particularly, minimizing the function $\mathcal{E}(x, y)$ is equivalent to maximizing the log of the posterior function, while $\mathcal{D}(x, y)$ corresponds to the log likelihood and $\mathcal{R}(x)$ corresponds to the log prior of x which is summarized as follows

$$-\log(\mathcal{P}(x|y)) = -\log(\mathcal{P}(y|x)) - \log(\mathcal{P}(x)). \tag{1.3}$$

The model (1.2) is very general and applies to convex and non-convex optimization problems but, in this thesis, the model is assumed to be convex. What makes convex optimization interesting is that there is a theory behind it which guarantees that any local minima is also a global minima [8]. Additionally, convex problems can always be solved with polynomial time algorithms [8].

1.2.3 Solutions to smooth convex problems

The function in (1.2) is known by different names such as *objective function*, *energy function* and *loss function* to name the most common. An import property of the objective function is its differentiability which is related to the smoothness of the function. Quadratic functions are differentiable and admit closed-form solutions. Differentiable functions can also be solved by first-order or second-order methods which require knowledge of the gradient or Hessian of the objective function respectively [8].

The focus in this thesis is mainly on first order methods, in other words, methods that require knowledge of the gradient of a function. In the general case and more formally, it is important to introduce the idea of subgradient [11]

Definition 1.2.1. ***(subgradient).*** *A vector $g \in \mathbb{R}^n$ is a subgradient of $f : \mathbb{R}^n \to \mathbb{R}$ at $x \in \operatorname{dom} f$ if*

$$f(z) \geq f(x) + g^T(z - x), \forall z \in \operatorname{dom} f. \tag{1.4}$$

The set of the subgradients of f at a point x are referred to as the subdifferential of f and denoted by $\partial f(x)$. The concept of subgradients leads to the idea of subdifferentiable functions which are defined as follows:

Definition 1.2.2. ***(subdifferential).***

- *A function f is called subdifferentiable at x if there exists at least one subgradient of f at x.*
- *A function f is called subdifferentiable if it is subdifferentiable at all $x \in$ dom f.*

The subdifferential generalizes the gradient and when the function is differentiable, the subdifferential reduces to the gradient ($\partial f(x) = \{\nabla f(x)\}$) of the function and the relation in (1.4) reduces to

$$f(z) \geq f(x) + \nabla f(x)^T (z - x), \forall z \in \text{dom}\, f, \tag{1.5}$$

which is exploited in the pursuit of the global minimum through the use of the classic method of gradient descent as follows:

$$x_{k+1} = x_k - \tau \nabla f(x_k), \tag{1.6}$$

where τ is a step size.

1.2.4 Non-smooth problems

A more recent trend in optimization-based modelling has been the use of non-smooth convex objective functions where the function is not differentiable at all points such as the function in Figure 1.2. The interest in these ob-

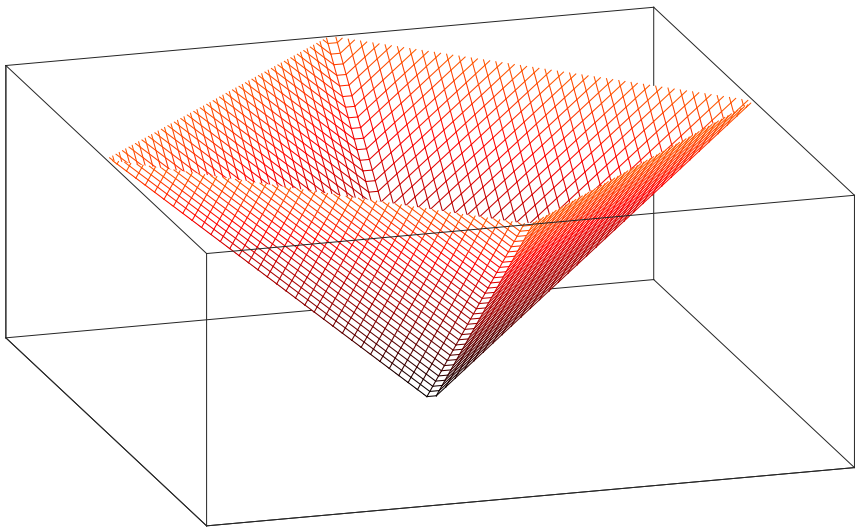


FIGURE 1.2: Example 2D non-smooth convex function

jective function comes mainly from the total variation [12] and related models, and from sparse modelling [13, 14].

1.2.5 Algorithms for non-smooth convex problems

Optimising non-smooth functions usually relies on the subdifferentials through the use of proximal operators [15, 16].

Definition 1.2.3. *(**proximal operator**). The proximal operator* $\mathbf{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$ *of the function* $f(.)$ *is defined by*

$$\mathbf{prox}_{\lambda f}(x) = \underset{y}{\operatorname{argmin}} \left(f(y) + \frac{1}{2\lambda} \|x - y\|_2^2 \right), \tag{1.7}$$

where the optimal value for the optimization problem is arrived at by iteratively evaluating the proximal operator as follows:

$$x_{k+1} = \mathbf{prox}_{\lambda f}(x_k). \tag{1.8}$$

the role of λ in (1.7) is to trade-off the two terms. The connection between the subdifferential and proximal operator is through the following relation [16]:

$$\mathbf{prox}_{\lambda f}(x) = (I + \lambda \partial f)^{-1}, \tag{1.9}$$

which is known as the *resolvent* of the subdifferential operator ∂f.

In the general case, objective functions of interest have multiple terms some of which are smooth while others are non-smooth. To approach such problems, the *alternating direction method of multipliers (ADMM)* algorithm [17] can be used. The ADMM algorithm decouples the terms of the problem as follows:

$$\text{minimize} \quad f(x) + g(y) \tag{1.10}$$

$$\text{subject to} \quad Ax + By = c \tag{1.11}$$

which is solved by iteratively alternating between solving for x and solving for y as follows:

$$x_{k+1} := \underset{x}{\operatorname{argmin}} \left(f(x) + (\rho/2) \|Ax + By_k - c + z_k\|_2^2 \right) \tag{1.12}$$

$$y_{k+1} := \underset{y}{\operatorname{argmin}} \left(g(y) + (\rho/2) \|Ax_{k+1} + By - c + z_k\|_2^2 \right) \tag{1.13}$$

$$z_{k+1} := z_k + Ax_{k+1} + By_{k+1} - c \tag{1.14}$$

where the x and y sub-problems could be solved using either a gradient-based algorithm or a proximal operator-based algorithm based on the smoothness of $f(.)$ and $g(.)$ respectively.

1.3 Overview of research outcomes

In this thesis, convex optimization was utilized as a tool to formulate and solve signal and image processing problems. The models developed are filters of signals or images. The goals of the developed filters range from denoising signals defined on graphs in Chapter 2, edge-aware smoothing images as fundamental tool for further image processing applications in Chapter 3, avoiding blocking artefacts in total variation based filtering in Chapter 4, and utilizing a discrete fractional Laplacian as a regularizer in signal and image filtering tasks in Chapter 5.

1.4 Summary of contributions

The thesis is composed of six chapters, including this Chapter. Chapters 2 to 5, present the contributions of this thesis. Related literature to each chapter is presented within the corresponding chapter. The last chapter (Chapter 6), contains the conclusions and future directions.

The contributions of the thesis are summarized as follows:

1.4.1 Graph polynomial filter for signal denoising (Chapter 2)

A technique for denoising signals defined over graphs was recently proposed in [18]. The technique is based on a regularization framework and denoising is achieved by solving an optimization problem. Matrix inversion is required and an approximate solution that avoids the direct calculation of the inverse, by using a graph filter, was proposed in [18]. The technique however, requires an eigendecomposition and the degree of the resulting filter is high. In Chapter 2, we propose a computationally efficient technique that is based on a least squares approximation of the eigenvalues of the inverse. We show that a good approximation can be achieved with a low degree graph polynomial filter without the need for any eigendecomposition. Low degree filters also have the desirable property of vertex localization (analogous to time localization). The filter gives denoising results that are very similar to that

using the exact solution and can be implemented using distributed processing.

1.4.2 Guided adaptive interpolation filter (Chapter 3)

Edge-aware smoothing has proved to be a fundamental technique for various image processing and computer vision tasks. In Chapter 3, we introduce a local, non-iterative and effective edge-preserving filter namely guided adaptive interpolation filter (GAIF). GAIF can be used as a post-processing step after any smoothing filter to improve its edge preservation performance without reformulation. GAIF has an $O(N)$ computation complexity where N being the total number of pixels in the image. To further increase the efficiency of GAIF at edge-preservation, two techniques are introduced and demonstrated. GAIF efficiency is demonstrated and compared to state-of-the-art techniques on a number of tasks including image smoothing, flash/no-flash image denoising/fusion, single image dehazing and image details enhancement.

1.4.3 High-pass filter generalization of the total variation model and its performance (Chapter 4)

The difference operator in the total-variation (TV) model can be regarded as the simplest half band high pass filter. Chapter 4 presents a new generalization of the TV model by replacing the difference operator with a high pass filter which has a user specified bandwidth. The new TV model can be efficiently implemented using the ADMM algorithm. Results from extensive experiments on six typical classes of signals show that the proposed model:

- Does not generate staircase artefacts which is problematic in the classical TV model.

- It achieves better denoising performance for particular signal classes. Especially for smooth signals with limited discontinuities.

1.4.4 Discrete Laplacian operator and its applications in signal processing (Chapter 5)

Fractional calculus has increased in popularity in recent years, as the number of its applications in different fields has increased. Compared to the traditional operations in calculus (integration and differentiation) which are uniquely defined, the fractional-order operators have numerous definitions. Furthermore, a consensus on the most suitable definition for a given task is yet to be reached. Fractional operators are defined as continuous operators and their implementation requires a discretization step. Chapter 5 presents a discrete fractional Laplacian as a matrix operator. The proposed operator is real (non-complex) which makes it computationally efficient. The construction of the proposed fractional Laplacian utilizes the DCT transform avoiding the complexity associated with the discretization step which is typical in the constructions based on signal processing. We demonstrate the utility of the proposed operator on a number of data modelling and image processing tasks.

Chapter 2

Graph Polynomial Filter for Signal Denoising

2.1 Introduction

The processing of signals defined on irregular domains is a problem faced in numerous fields such as biomedical imaging and social science. Irregular domains can be modelled effectively using techniques from graph theory. In some cases, the native domain is inherently irregular such as in sensor or transportation networks. In other cases, such as in images, even though the domain is inherently regular, a graph model can potentially capture the signal statistics enhancing the performance of filtering algorithms [19, 20]. Two frameworks have emerged recently for processing signals defined on graphs. The first framework is based on spectral graph theory [21] and is applicable for undirected graphs. The second framework is called DSP$_G$ (Discrete-Signal-Processing on Graphs) [22, 23, 24] and is applicable for both directed and undirected graphs. Several classical signal processing notions and techniques, such as filtering [25, 26], prediction, frequency analysis [27, 23], wavelets and filter banks [28, 29, 30, 31, 32, 33, 34, 35], have been extended to signals defined on graph.

Denoising is a classical problem in signal processing and there is a plethora of techniques, such as low-pass filtering and wavelet thresholding, for denoising regular domain signals. A class of techniques, which has its roots in the Tikhonov regularization method, is very popular for denoising regular domain signals. Recently, a technique that is based on the DSP$_G$ framework for denoising graph signals was proposed by Chen et al. [18]. The technique uses a quadratic form regularized formulation and is applicable for both directed and undirected graph signals. A regularization technique that is based

on the graph Laplacian was proposed [36] but it is applicable for undirected graphs only.

The solution with the regularized formulation in [18] requires the inversion of a potentially large matrix. To avoid direct inversion, a graph filtering method was proposed to approximate the inverse [18]. The technique however requires an eigendecomposition which can be computationally expensive. The degree of the graph filter is also large. In this chapter, an alternative technique is proposed to approximate the matrix inverse using a graph polynomial filter. The proposed technique avoids the need for an eigendecomposition, and the approximate filter is a low degree resulting in good vertex localization. Although the techniques proposed here and in [18] utilize matrix polynomials, the types of polynomials used are different. Experiments demonstrate that the proposed approximate technique yields similar denoising results to the exact method (direct inverse). An overview of this chapter is as follows. Basic concepts of graph signal processing are reviewed in Section 2.2. The problem of denoising signals defined over graphs using a variational regularized framework and previously proposed solutions are discussed in Section 2.3. In Section 2.4 a graph polynomial filter solution is presented together with relevant theoretical results. It is also shown how distributed processing can be used to efficiently implement the filter. Experimental results on denoising graph signals are presented in Section 2.5. Comparisons with other techniques are presented in Section 2.6. The chapter concludes in Section 2.7.

2.2 Discrete signal processing on graphs

A brief review of relevant concepts from DSP_G is presented here. More details are found in [22, 23, 24]. A graph G consists of a set of vertices V and a set of edges E that connects the vertices. A graph signal $\boldsymbol{x} \in \mathbb{R}^N$ is an association or mapping between $x_i \in \boldsymbol{x}$ and $v_i \in V$. The adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ contains the weights of the edges. The element $a_{n,m}$ is the weight of a directed edge from v_m to v_n. If $a_{n,m} = 0$ there is no edge. In general $a_{n,m} \neq a_{m,n}$ and the graph is directed. The weights are assumed to be non-negative, i.e. $a_{n,m} \geq 0$, although in the DSP_G framework this restriction is not needed. Additionally, $\boldsymbol{A}$ is assumed to be normalized such that the eigenvalue with largest magnitude $|\hat{\lambda}_N| = 1$. If $\boldsymbol{A}$ is un-normalized, normalization can be achieved by either of the two methods described below:

1. $\boldsymbol{A} \rightarrow \boldsymbol{A}/|\hat{\lambda}_N|$.
2. $\boldsymbol{A} \rightarrow \boldsymbol{D}^{-1}\boldsymbol{A}$, where $\boldsymbol{D} = \text{diag}\{\boldsymbol{A}\mathbf{1}\}$[1] and $\mathbf{1}$ is the column vector with N ones.

The second method achieves normalization as a consequence of the Perron-Frobenius theorem on non-negative matrices [37]. The adjacency matrix $\boldsymbol{A}$ is considered as the generalization of the shift operator. A linear shift invariant graph filter is a matrix polynomial

$$h(\boldsymbol{A}) = \sum_{i=0}^{L} h_i \mathbf{A}^i.$$

Graph filtering is achieved by the transform matrix product $\boldsymbol{y} = h(\boldsymbol{A})\boldsymbol{x}$. If $\mathbf{A}$ is diagonalizable, the eigendecomposition is given by

$$\boldsymbol{A} = \hat{\boldsymbol{V}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{V}}^{-1}$$

where $\hat{\mathbf{V}} = [\mathbf{v}_1\, \mathbf{v}_2 \ldots \mathbf{v}_N]$ is the matrix of eigenvectors and $\hat{\boldsymbol{\Lambda}} = \text{diag}(\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_N)$ is the diagonal matrix of eigenvalues, where the eigenvalues $\hat{\lambda}_i$ are sorted in an increasing order. If $\mathbf{A}$ is not diagonalizable, then $\boldsymbol{A} = \hat{\boldsymbol{V}}\boldsymbol{J}\hat{\boldsymbol{V}}^{-1}$, where $\mathbf{J}$ is the Jordan normal form of eigenvalues and $\hat{\mathbf{V}}$ is the matrix of eigenvectors [38]. The eigenvalues can be interpreted as the graph natural frequencies and the eigenvectors can be used as the basis for spectral analysis. The graph Fourier transform is then defined as

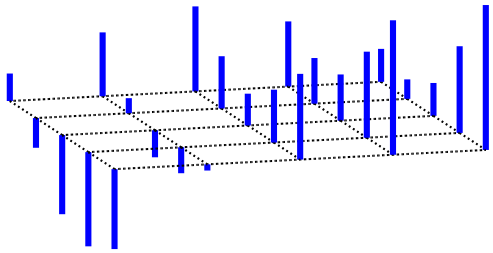
$$\hat{\boldsymbol{x}} = \boldsymbol{V}^{-1}x, \tag{2.1}$$

The element $[\hat{\boldsymbol{x}}]_i$ gives the amplitude of component $\mathbf{v}_i$ of the graph signal $\boldsymbol{x}$. The original signal $\mathbf{x}$ can then be expressed in terms of its spectral component as

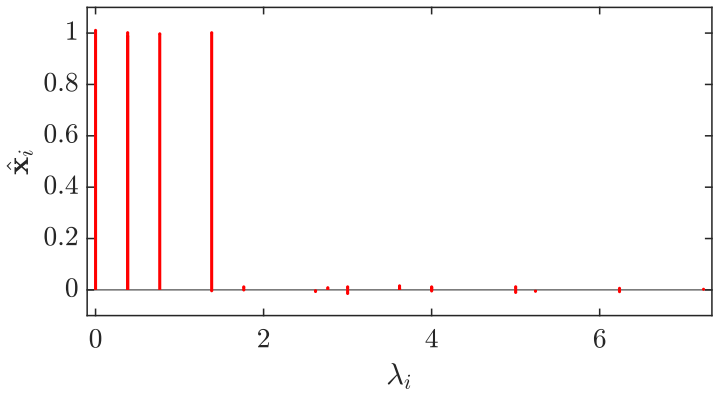
$$\boldsymbol{x} = \boldsymbol{V}\hat{\boldsymbol{x}}. \tag{2.2}$$

and this represents the inverse graph Fourier transform. Consequently, we have two alternative methods for expressing a graph signal, vertex domain representation x and the corresponding spectral domain representation $\hat{x}$. This idea is illustrated in Figure 2.1

[1] diag $\{.\}$ creates a diagonal matrix from the vector argument.



(a) Vertex domain



(b) Graph spectral domain

FIGURE 2.1: Vertex and graph spectral domains of a 2D grid graph (N=25) with a test signal defined by adding the first five eigenvectors ($\sum_{i=1}^{5} \boldsymbol{v}_i$) and Gaussian noise with noise $\sigma = 0.1$.

2.3 Graph signal denoising

The observed noisy signal defined over a graph $G = (V, E)$ is modelled as follows

$$\boldsymbol{t} = \boldsymbol{x} + \boldsymbol{n} \tag{2.3}$$

where $\boldsymbol{x}$ is the desired noiseless signal and $\boldsymbol{n}$ is random additive noise. The assumption made about $\boldsymbol{x}$ is that it is smooth with respect to the underlying graph [39]. The measure of smoothness of a graph signal proposed in [23] is the quadratic form defined as:

$$R(\boldsymbol{x})_{DSP_G} \equiv \|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x})^T(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}) \tag{2.4}$$

where it is assumed that the adjacency matrix $\boldsymbol{A}$ is normalized so that the largest magnitude eigenvalue is unity. A smooth graph signal will have a small $R(\boldsymbol{x})_{DSP_G}$ value and this is the assumption made of the desired signal $\boldsymbol{x}$.

A common approach to denoising is via a variational regularized framework pioneered by Tikhonov for ill-posed problems. Consider the functional

$$E(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{t}\|_2^2 + \gamma R(\boldsymbol{x}) \tag{2.5}$$

where the first term measures the fidelity, the second term measures the degree of non-smoothness and the parameter γ controls the trade-off between the two measures. Two types of quadratic regularizer function $R(\boldsymbol{x})$ have been proposed in the literature. One is based on the DSP_G framework [22] and uses $R(\boldsymbol{x})_{DSP_G}$ in (2.4) as the regularizer. The other is based on spectral graph theory for undirected graphs [21] and the regularizer is given by $R(\boldsymbol{x})_{SGT} = \boldsymbol{x}^T\mathbf{L}\boldsymbol{x}$ [40, 36], where $\mathbf{L}$ is the Laplacian matrix of the undirected graph. In this work, the regularizer in (2.4) is considered, which works for both undirected and directed graphs.

The denoised signal is obtained by solving the following optimization problem

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\text{argmin}} \left\{ \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{t}\|_2^2 + \gamma\|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \right\} \tag{2.6}$$

The objective function is quadratic in $\boldsymbol{x}$ and admits the following exact solution:

$$\hat{\boldsymbol{x}} = (\boldsymbol{I} + \gamma(\boldsymbol{I} - \boldsymbol{A})^T(\boldsymbol{I} - \boldsymbol{A}))^{-1}\boldsymbol{t} \tag{2.7}$$

The solution in (2.7) is exact but requires the inversion of an $N \times N$ matrix. If the number of vertices N is large, it becomes computationally expensive as the inversion requires at least $O(N^2)$ arithmetic operations even if $\boldsymbol{A}$ is sparse. This has motivated the authors in [18] to propose a method that avoids direct inversion of the matrix. The idea is to construct a suitable polynomial $g(\lambda)$ such that the matrix polynomial $g(\boldsymbol{A})$, i.e. graph filter, gives the matrix inverse. It is shown in [18] that such a filter can be constructed if $\boldsymbol{A}^T$ is diagonalizable: $\boldsymbol{A}^T = \hat{\mathbf{V}}\mathbf{D}\hat{\mathbf{V}}^{-1}$, where $\mathbf{D}$ is a diagonal matrix and $\hat{\mathbf{V}}$ is the eigenvector matrix of $\boldsymbol{A}$, i.e. $\boldsymbol{A} = \hat{\boldsymbol{V}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{V}}^{-1}$. Note that the eigenvectors of $\boldsymbol{A}^T$ and $\boldsymbol{A}$ are assumed to be the same and this implies that $\boldsymbol{A}^T$ and $\boldsymbol{A}$ commute, in multiplication, with each other, i.e. $\boldsymbol{A}$ is a normal matrix. Then a polynomial of degree N can be found such that $h(\boldsymbol{A}) = \boldsymbol{A}^T$ and the graph filter is an N degree polynomial that interpolates the values

$$g(\hat{\lambda}_n) = (1 + \gamma(1 - \hat{\lambda}_n - h(\hat{\lambda}_n) + h(\hat{\lambda}_n)\hat{\lambda}_n)) \tag{2.8}$$

for $n = 1, \ldots, N$. In general $\boldsymbol{A}^T$ is not exactly diagonalizable and an approximation is used:

$$\boldsymbol{A}^T \approx h(\boldsymbol{A}) = \hat{\mathbf{V}}\hat{\mathbf{D}}\hat{\mathbf{V}}^{-1} \tag{2.9}$$

where $\hat{\mathbf{D}}$ is the solution to the optimization problem

$$\hat{\mathbf{D}} = \underset{\mathbf{D}\in\mathcal{D}}{\operatorname{argmin}} \left\| \hat{\boldsymbol{V}}\boldsymbol{D}\hat{\boldsymbol{V}}^{-1} - \boldsymbol{A}^T \right\|_F^2. \tag{2.10}$$

Now $\mathcal{D}$ is the set of all diagonal matrices and F denotes the Frobenius norm.

2.4 Graph polynomial filter

The method in [18] (as described above) requires knowledge of the eigenvector matrix $\hat{\mathbf{V}}$ for the problem (2.10) and also for constructing $h(\boldsymbol{A}) = \hat{\mathbf{V}}\hat{\mathbf{D}}\hat{\mathbf{V}}^{-1}$. When the number of vertices N in the graph is large, the resulting graph filter ($g(\boldsymbol{A})$) degree is also large. In this section a method that avoids the need for eigendecomposition is presented. The proposed method is in the form of a

low degree polynomial filter which is derived with no assumption about the diagonalizability of $\boldsymbol{A}^T$ or $\boldsymbol{A}$.

By defining

$$\tilde{\boldsymbol{A}} \equiv \boldsymbol{I} - \boldsymbol{A} \quad \text{and} \quad \boldsymbol{B} \equiv \tilde{\boldsymbol{A}}^T \tilde{\boldsymbol{A}}, \tag{2.11}$$

equation (2.7) can be expressed as:

$$\hat{\boldsymbol{x}} = (\boldsymbol{I} + \gamma \boldsymbol{B})^{-1} \boldsymbol{t} \tag{2.12}$$

Now it can be easily verified that $\boldsymbol{B} = \boldsymbol{B}^T$ and

$$\boldsymbol{y}^T \boldsymbol{B} \boldsymbol{y} = \boldsymbol{y}^T \tilde{\boldsymbol{A}}^T \tilde{\boldsymbol{A}} \boldsymbol{y} = \|\tilde{\boldsymbol{A}} \boldsymbol{y}\|_2^2 \geq 0 \tag{2.13}$$

for any $\boldsymbol{y}$, i.e. $\boldsymbol{B}$ is a *symmetric positive semi-definite* (SPSD) matrix. An orthogonal eigendecomposition therefore exist $\boldsymbol{B} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T$, where $\boldsymbol{U}$ is the orthogonal matrix of eigenvectors and $\boldsymbol{\Lambda}$ is the diagonal matrix of non-negative eigenvalues $\lambda_n \geq 0$ for $n = 1, \ldots, N$. Now (2.12) can also be expressed as $\hat{\boldsymbol{x}} = f(\boldsymbol{B}) \boldsymbol{t}$ where

$$f(\lambda) \equiv \frac{1}{1 + \gamma \lambda} \tag{2.14}$$

and the usual definition of a matrix function applies [41]

$$f(\boldsymbol{B}) \equiv \boldsymbol{U} f(\Lambda) \mathbf{U}^T$$

Lemma 1. *If a degree N polynomial $H(\lambda) = \sum_{m=0}^{N} H_m \lambda^m$ interpolates the points $(\lambda_n, f(\lambda_n))$ for $n = 1, \ldots, N$, i.e. $H(\lambda_n) = f(\lambda_n)$, the solution (2.12) is also given by:*

$$\hat{\boldsymbol{x}} = H(\boldsymbol{B}) \boldsymbol{t} = \sum_{m=0}^{N} H_m \boldsymbol{B}^m \boldsymbol{t}. \tag{2.15}$$

Proof. Since $\boldsymbol{B} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T$ and $\boldsymbol{U} \boldsymbol{U}^T = \boldsymbol{U}^T \mathbf{U} = \mathbf{I}$, (2.12) can be written as

$$\begin{aligned}
\hat{\boldsymbol{x}} &= (\boldsymbol{U} \boldsymbol{U}^{\boldsymbol{T}} + \gamma \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T)^{-1} && \boldsymbol{t} \\
&= (\boldsymbol{U} (\boldsymbol{I} + \gamma \boldsymbol{\Lambda}) \boldsymbol{U}^T)^{-1} && \boldsymbol{t} \\
&= \boldsymbol{U} (\boldsymbol{I} + \gamma \boldsymbol{\Lambda})^{-1} \boldsymbol{U}^T && \boldsymbol{t} \\
&= \boldsymbol{U} \mathrm{diag}((1 + \gamma \lambda_n)^{-1}) \boldsymbol{U}^{\boldsymbol{T}} \boldsymbol{t} \\
&= \boldsymbol{U} \mathrm{diag}\{f(\lambda_n)\} \boldsymbol{U}^{\boldsymbol{T}} && \boldsymbol{t} \\
&= \boldsymbol{U} \mathrm{diag}\{h(\lambda_n)\} \boldsymbol{U}^{\boldsymbol{T}} && \boldsymbol{t} \\
&= H(\boldsymbol{B}) && \boldsymbol{t}
\end{aligned}$$

□

Lemma 1 shows that the exact solution can be obtained using the graph filter $H(\boldsymbol{B})$, when the polynomial $H(\lambda)$ interpolates $f(\lambda)$ at the eigenvalues points $\lambda = \lambda_n$. To construct the interpolating polynomial, knowledge of all the eigenvalues is required. This is computationally expensive for large N. The large filter degree N also leads to increased implementation complexity. An approximate solution is proposed next to reduce the computational complexity.

2.4.1 Approximate solution

Instead of a polynomial of degree N a lower degree $L << N$ polynomial is considered

$$\hat{H}(\lambda) = \sum_{m=0}^{L} \hat{h}_m \lambda^m \tag{2.16}$$

Unlike $H(\lambda)$ in Lemma 1, the polynomial $\hat{H}(\lambda)$ will only *approximately* interpolate the eigenvalues points $(\lambda_n, f(\lambda_n))$. The matrix polynomial $\hat{H}(\mathbf{B})$ is used as an approximation to the inverse

$$\hat{H}(\mathbf{B}) \approx f(\boldsymbol{B}) = (\boldsymbol{I} + \gamma \boldsymbol{B})^{-1}$$

The commonly used Frobenius norm is considered as the criterion for approximation.

Lemma 2. *The Frobenius norm of the approximation error is given*

$$\left\| \hat{H}(\mathbf{B}) - f(\boldsymbol{B}) \right\|_F^2 = \sum_{n=1}^{N} (\hat{H}(\lambda_n) - f(\lambda_n))^2 \equiv e(\boldsymbol{\Lambda})$$

Proof. Both $\hat{H}(\mathbf{B})$ and $f(\boldsymbol{B})$ are symmetric matrices and can be diagonalized with the orthogonal matrix $\mathbf{U}$ (eigenvector matrix of $\mathbf{B}$). Therefore

$$\hat{H}(\mathbf{B}) - f(\boldsymbol{B}) = \mathbf{U}(\hat{H}(\boldsymbol{\Lambda}) - f(\boldsymbol{\Lambda}))\mathbf{U}^T \tag{2.17}$$

Now the Frobenius norm of a matrix $\mathbf{E}$ is invariant under pre- or post- multiplication by a unitary matrix $\mathbf{V}$, i.e. $\|\mathbf{EV}\|_F = \|\mathbf{VE}\|_F = \|\mathbf{E}\|_F$ [37, page 292].

Since both $\mathbf{U}$ and $\mathbf{U}^T$ are unitary, we have

$$\begin{aligned}\|\hat{H}(\mathbf{B}) - f(\boldsymbol{B})\|_F &= \|\mathbf{U}(\hat{H}(\mathbf{\Lambda}) - f(\mathbf{\Lambda}))\mathbf{U}^T\|_F \\ &= \|\hat{H}(\mathbf{\Lambda}) - f(\mathbf{\Lambda})\|_F\end{aligned}$$

Since both $\hat{H}(\mathbf{\Lambda})$ and $f(\mathbf{\Lambda})$ are diagonal matrices, the Frobenius norm is the sum-of-squares of the diagonals and the result follows. □

To minimize $e(\mathbf{\Lambda})$ with respect to $\hat{h}_m$, knowledge of all the eigenvalues λ_n (the spectrum) is required. Determining the spectrum is however computationally expensive for large matrices. To avoid the computation of the spectrum, the following strategy is proposed. Let $\rho(\mathbf{B}) \equiv \max_n \lambda_n$ denote the spectral radius and $\lambda_{max} (\geq \rho(\mathbf{B}))$ be an upper bound of the radius. Therefore, a bound on the spectrum exists as follows $0 \leq \lambda_n \leq \lambda_{max}$. Instead of trying to minimize the errors, i.e. $(\hat{H}(\lambda) - f(\lambda))^2$, at discrete spectral frequencies λ_n, the proposed strategy is to minimize the average errors over the entire range $\lambda \in [0, \lambda_{max}]$. This result is suboptimal but this strategy avoids having to determine the values of the discrete spectral frequencies λ_n, i.e. spectrum. Consider then the proxy objective function

$$E(\hat{\mathbf{h}}) \equiv \int_0^{\lambda_{max}} (\hat{H}(\lambda; \hat{\mathbf{h}}) - f(\lambda))^2 d\lambda \tag{2.18}$$

where $\hat{\mathbf{h}} \equiv [\hat{h}_0 \dots \hat{h}_L]$. A small $E(\hat{\mathbf{h}})$ will result in a small $e(\mathbf{\Lambda})$. With $E(\hat{\mathbf{h}})$ detailed knowledge of the spectrum is not required but only an upper bound on the spectral radius λ_{max}. Using (2.16), $E(\hat{\mathbf{h}})$ can be written as

$$E(\hat{\mathbf{h}}) = \int_0^{\lambda_{max}} \left(\sum_{m=0}^{L} \hat{h}_m \lambda^m - f(\lambda) \right)^2 d\lambda \tag{2.19}$$

which is quadratic in $\hat{\mathbf{h}}$. Now

$$\frac{\partial E}{\partial h_n} = 2 \int_0^{\lambda_{max}} \left(\sum_{m=0}^{L} \hat{h}_m \lambda^m - f(\lambda) \right) \lambda^n \, d\lambda \tag{2.20}$$

The optimum value is obtained when $\frac{\partial E}{\partial h_n} = 0$ or

$$\sum_{m=0}^{L} \hat{h}_m \int_0^{\lambda_{max}} \lambda^{m+n} d\lambda = \int_0^{\lambda_{max}} f(\lambda)\lambda^n \, d\lambda$$
$$= \int_0^{\lambda_{max}} \frac{\lambda^n}{1+\gamma\lambda} \, d\lambda$$

for $n = 0, \ldots, L$. This leads to the system of linear equations $\mathbf{C}\hat{\mathbf{h}} = \mathbf{r}$ where the elements of $\mathbf{C}$ are given by

$$c_{m,n} \equiv \int_0^{\lambda_{max}} \lambda^{m+n} d\lambda = \frac{\lambda_{max}^{m+n+1}}{m+n+1} \tag{2.21}$$

for $m, n = 0, \ldots, L$ and the elements of $\mathbf{r}$ are given by

$$r_n \equiv \int_0^{\lambda_{max}} \frac{\lambda^n}{1+\gamma\lambda} \, d\lambda$$

for $n = 0, \ldots, L$. By using the change of variable $x = 1 + \gamma\lambda$ so that $x_{max} = 1 + \gamma\lambda_{max}$,

$$r_n \equiv \frac{1}{\gamma^{n+1}} \int_1^{x_{max}} \frac{(x-1)^n}{x} \, dx$$

For $n = 0$, the integral can be easily evaluated to give

$$r_0 = \frac{1}{\gamma} \log_e(1 + \gamma\lambda_{max})$$

For $n \geq 1$, the following binomial expansion is used for the integrand

$$(x-1)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k}(-1)^k$$
$$= \sum_{k=0}^{n-1} \binom{n}{k} x^{n-k}(-1)^k + (-1)^n$$

The last term needs to be treated separately as in results in a $1/x$ integral. Using this expansion in the integral gives the following explicit expression

for r_n

$$r_n = \frac{1}{\gamma^{n+1}} \sum_{k=0}^{n-1} \begin{pmatrix} n \\ k \end{pmatrix} (-1)^k \frac{(1+\gamma\lambda_{max})^{n-k} - 1}{(n-k)} + \frac{1}{\gamma^{n+1}} (-1)^n \log_e(1+\gamma\lambda_{max})$$

for $n = 1, \ldots, L$.

If λ_{max} is sufficiently large the resulting solution $\hat{\mathbf{h}}$ is independent of the graph (spectrum). However if λ_{max} is too large the approximation may not be good over the interval $0 \leq \lambda \leq \rho(\mathbf{B})$. For a given graph and therefore $\mathbf{B}$, one approach is to apply the power method [38] to determine $\rho(\mathbf{B}) = \lambda_{max}$. The power method is computationally efficient but its convergence may be slow.

Another approach, which is non-iterative and adopted here, is to use known bounds for $\rho(\mathbf{B})$ [42, page 142] :

$$\rho(\mathbf{B}) \leq \min\left(\|\boldsymbol{B}\|_1, \|\boldsymbol{B}\|_\infty\right) \tag{2.22}$$

where

$$\|\boldsymbol{B}\|_1 \equiv \max_j \sum_i |[\boldsymbol{B}]_{i,j}|, \text{ and}$$
$$\|\boldsymbol{B}\|_\infty \equiv \max_i \sum_j |[\boldsymbol{B}]_{i,j}|,$$

are the 1-norm and ∞-norm, respectively, and can be computed easily without any iterative process. Figure 2.2 shows the degree $L = 7$ polynomial approximation to $f(\lambda) \equiv \frac{1}{1+\gamma\lambda}$ ($\gamma = 10$). The value $\lambda_{max} = 6.34$ is the upper bound of $\rho(\mathbf{B})$ for the temperature sensor graph (in Section 2.5.1) using (2.22). The relative approximation error for this case is given by

$$\frac{\|\hat{H}(\mathbf{B}) - f(\boldsymbol{B})\|_F^2}{\|f(\boldsymbol{B})\|_F^2} = 1.51\%$$

and is reasonably small. This means that the approximated inverse is close to the exact inverse. The example in Figure 2.2 shows that any eigenvalue of the inverse $f(\lambda_n)$ (for $0 \leq \lambda_n \leq \lambda_{max}$) is well approximated by $\tilde{H}(\lambda_n)$ irrespective of the location of the eigenvalue λ_n. This may seem like an 'overkill' but the approach does not require detailed knowledge of the location of the

eigenvalues. It is important to note that to achieve a specific target approximation error, the larger the value of γ, the higher the degree L required. In other words, as $L \to \infty$ the approximation error $E \to 0$.

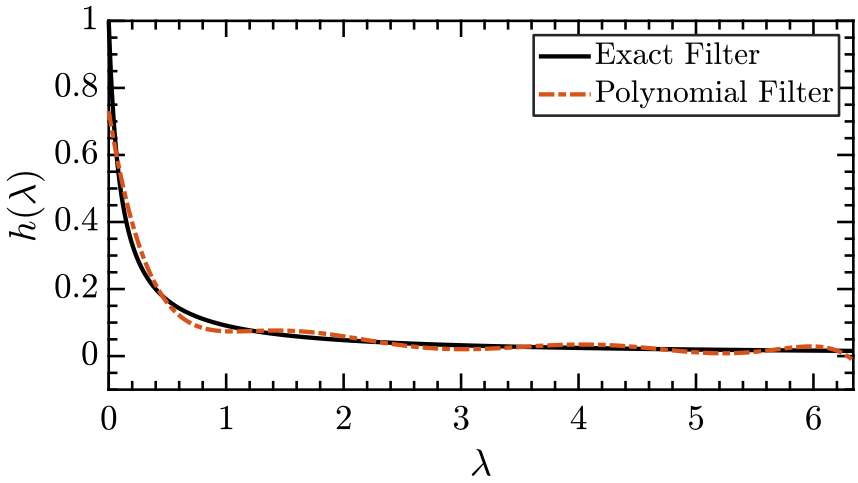


FIGURE 2.2: Graph polynomial filter of degree ($L = 7$) designed using the least squares technique with $\lambda_{max} = 6.34$ and $\gamma = 10$.

2.4.2 Distributed processing and complexity comparison

The denoising using $\hat{H}(\mathbf{B})$ can be implemented in a distributed manner where information are exchanged between neighbourhood vertices. The fundamental operator here is $\mathbf{B} = \tilde{\boldsymbol{A}}^{\boldsymbol{T}}\tilde{\boldsymbol{A}}$. Now the operator $\tilde{\boldsymbol{A}} \equiv \boldsymbol{I} - \boldsymbol{A}$ results in the difference between the signal value at a vertex and the weighted average of the signals from the one-hop in-link vertices. The operator $\tilde{\boldsymbol{A}}^T \equiv \boldsymbol{I} - \boldsymbol{A}^T$ does something similar but with one-hop out-link vertices. The operator $\mathbf{B}$ is a tandem combination of these two operators. The degree of vertex localization is determined by the degree L of the graph filter. The solution in [18] however, is a matrix polynomial in $\boldsymbol{A}$ which involves only the in-link vertices. For many practical applications, the adjacency matrix $\boldsymbol{A}$ is highly sparse, i.e. on average every row contains only K non-zero elements where K is small. Implementing the graph polynomial filter would then require $O(2LKN)$ arithmetic operations. For large N (graphs) this is significantly less than the $O(N^2)$ operations required by the direct inversion.

2.5 Experimental results

The exact solution using explicit matrix inversion in (2.7) is referred to as GTVR (Graph-Total-Variation-Regularization) method. Our proposed approximate solution using $\hat{H}(\mathbf{B})$ is referred to as the GPF (Graph-Polynomial-Filter) method. The graph filter degree is $L = 7$.

2.5.1 Temperature sensors measurement denoising

For this example, the freely available GSOD [43] dataset is used. The GSOD dataset contains daily summaries from weather stations around the world. For this experiment, temperature measurements from a subset of 617 sensors located in the US for the year 2003 are used. N random stations, representing vertices, are chosen out of 617 stations to construct a graph. The graph is constructed by connecting each vertex with its 8 nearest geographic neighbours. Three different graphs with $N = 200, 400, 600$, as shown in Figures 2.3 to 2.5, are used in the simulations.

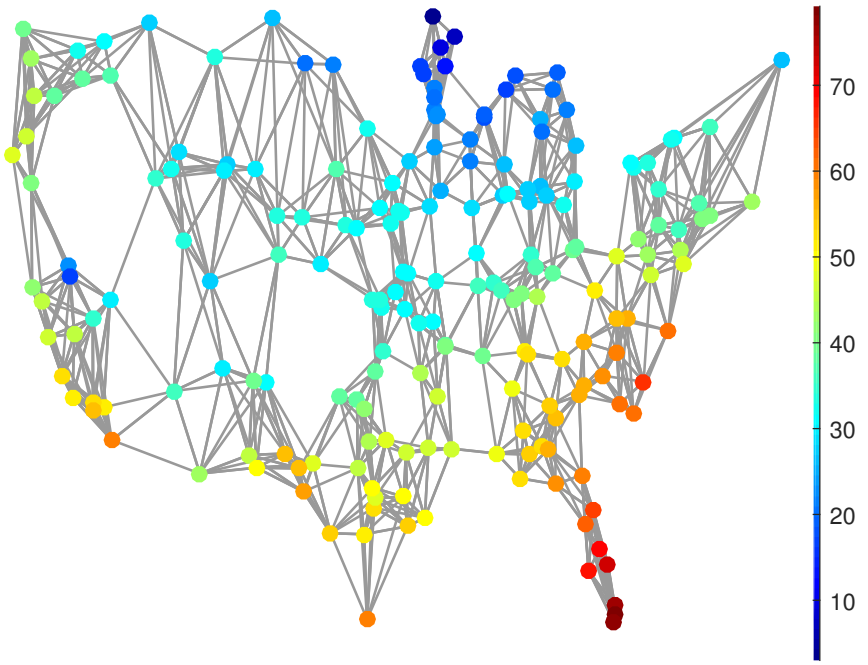


FIGURE 2.3: Sensor Graph with N = 200

The edge weight between vertices v_n and v_m is calculated as follows

$$a_{m,n} = \exp\left(-\frac{d_{m,n}}{\sigma^2}\right)$$

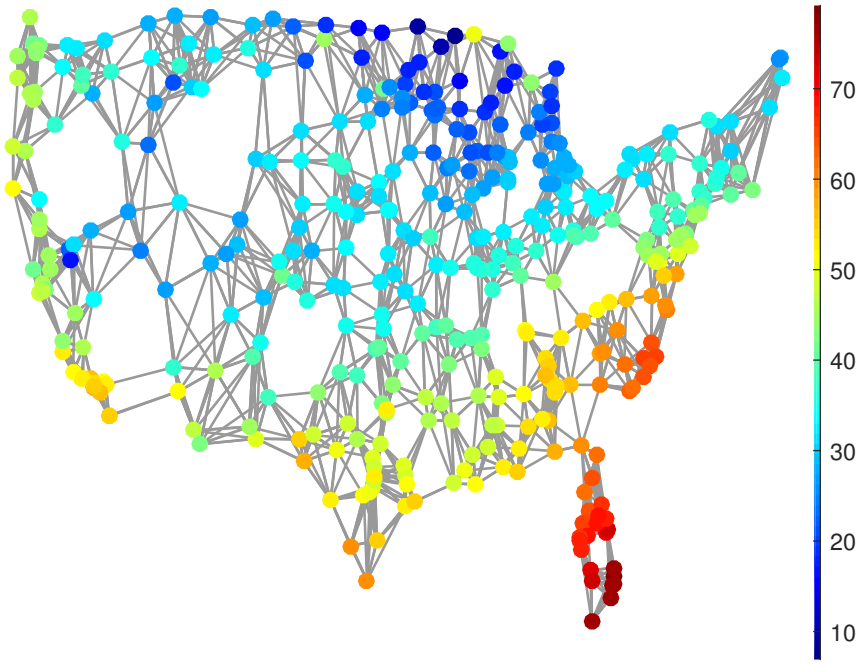


FIGURE 2.4: Sensor Graph with N = 400

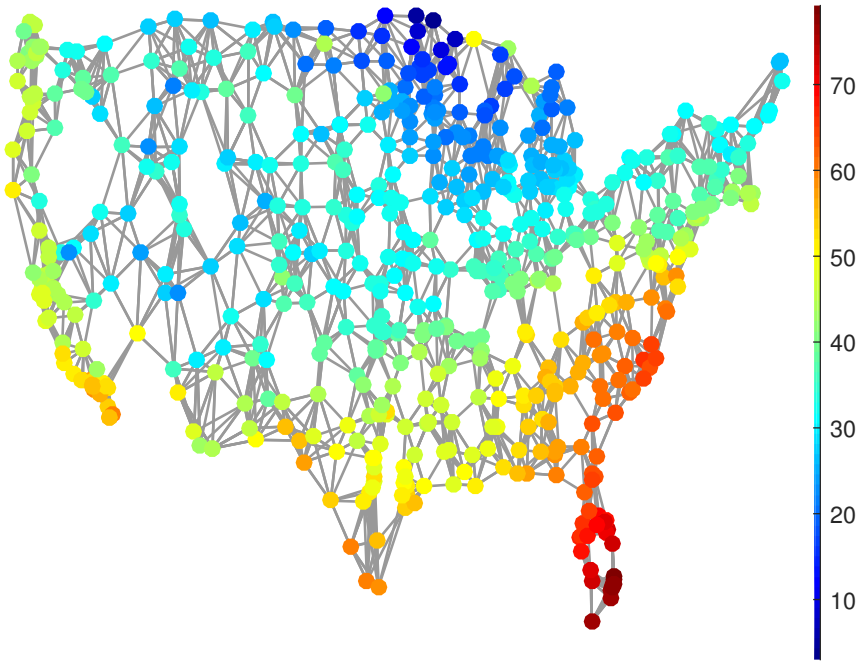


FIGURE 2.5: Sensor Graph with N = 600

where $d_{m,n}$ is the geodesic distance between vertices and

$$\sigma^2 = \frac{1}{N} \sum_{(m,n)\in E} d_{m,n}$$

The adjacency matrix is normalized as follows: $\boldsymbol{A} \rightarrow \boldsymbol{D}^{-1}\boldsymbol{A}$.

The noisy signal is synthesized by adding zero mean Gaussian noise with a range of variances to simulate varying levels of noise. To evaluate the performance, the RMSE between the ground truth signal $\mathbf{x}$ and the denoised signal $\hat{\mathbf{x}}$ is calculated using the GTVR and GPF techniques. Tables 2.1 to 2.3 show the results of denoising the temperature signals on graphs of different sizes.

TABLE 2.1: Temperature Sensors $(N = 200)$: RMSE of GTVR and GPF

γ	Technique	Noise σ 5	15	25	35
0	GPF	5.06	15.17	24.36	35.53
	GTVR	5.06	15.17	24.36	35.53
0.01	GPF	4.69	14.32	23.73	36.75
	GTVR	4.69	14.32	23.73	36.75
0.1	GPF	4.84	12.99	23.39	31.68
	GTVR	4.84	12.99	23.39	31.68
1	GPF	3.98	8.98	12.91	20.01
	GTVR	3.98	8.98	12.91	20.02
10	GPF	6.21	7.19	10.04	10.54
	GTVR	6.02	7.01	10.34	10.76
100	GPF	8.02	8.37	9.68	10.45
	GTVR	7.10	7.78	8.97	11.37

TABLE 2.2: Temperature Sensors $(N = 400)$: RMSE of GTVR and GPF

γ	Technique	Noise σ 5	15	25	35
0	GPF	5.16	14.62	24.57	36.24
	GTVR	5.16	14.62	24.57	36.24
0.01	GPF	4.96	14.40	23.65	33.81
	GTVR	4.96	14.40	23.65	33.81
0.1	GPF	4.63	13.42	23.43	30.64
	GTVR	4.63	13.42	23.43	30.64
1	GPF	3.83	8.09	13.03	18.47
	GTVR	3.83	8.09	13.03	18.48
10	GPF	5.90	7.10	9.10	12.71
	GTVR	5.77	7.12	9.34	13.22
100	GPF	7.90	8.04	9.22	11.32
	GTVR	6.58	7.30	9.37	12.42

TABLE 2.3: Temperature Sensors $(N = 600)$: RMSE of GTVR and GPF

γ	Technique	Noise σ 5	15	25	35
0	GPF	5.09	14.57	25.23	35.38
	GTVR	5.09	14.57	25.23	35.38
0.01	GPF	4.69	14.97	24.63	34.76
	GTVR	4.69	14.97	24.63	34.76
0.1	GPF	4.42	13.50	22.46	30.44
	GTVR	4.42	13.50	22.46	30.44
1	GPF	3.61	8.07	13.37	18.42
	GTVR	3.61	8.07	13.37	18.43
10	GPF	5.17	6.46	7.81	10.10
	GTVR	5.16	6.64	8.01	10.52
100	GPF	6.68	7.09	8.31	8.78
	GTVR	6.04	7.01	8.37	10.25

2.5.2 Image denoising

The underlying similarity between pixels is modelled using as a weighted graph where the vertices represent the pixels. Edges weights are calculated using the Non-Local-Means (NLM) filter weights [44]

$$a_{ij} = \exp\left(-\frac{\|\mathbf{p}(i) - \mathbf{p}(j)\|^2}{h^2}\right) \tag{2.23}$$

where $\mathbf{p}(i)$ and $\mathbf{p}(j)$ are the image patches centred at pixels i and j respectively, and h is the smoothing parameter. The adjacency matrix is normalized as follows: $\boldsymbol{A} \to \boldsymbol{A}/|\hat{\lambda}_N|$.

Noisy images are synthesized by adding zero mean Gaussian noise with a range of variances σ^2 to simulate varying levels of noise. For an image, patches that are used to determine the edge weights in (2.23) are obtained from a smoothed version of the noisy image by using a simple Gaussian filter. This smoothing is only for the purpose of obtaining the edge weights a_{ij}. The denoising using the GTVR and GPF techniques are performed on the noisy images (before the Gaussian filter). Experiments were carried out on five standard images (Barbara, Lena, Cameraman, Mandrill and Peppers) shown in Figure 2.6 and the results are shown in Tables 2.4 to 2.8, where each table contains the results for one image.

FIGURE 2.6: Images used in the image denoising experiment: Barbara, Lena, Mandrill, Peppers and Cameraman.

2.5.3 Discussion

The results in presented earlier (Section 2.5.2) show that the GPF performance is very similar to the GTVR performance except for when γ is large. For most cases the results are indistinguishable. The difference in RMSE becomes larger when the parameter γ value becomes larger. A larger γ value is required when the noise level is higher. When γ is large, it is harder to approximate the ideal function $f(\lambda)$ in (2.14) with a polynomial as there is a steep decrease in the function near the origin. A higher degree L polynomial is then required for a good approximation to the inverse. For localization and efficient implementation, it is desirable to have a small L. A simple way to ascertain if a particular L is good enough is to compare the polynomial approximation (2.16) with the ideal function in (2.14), e.g. Figure 2.2 with $\gamma = 10$. If the polynomial is a good fit to the ideal function the denoising performance using the GPF will be similar to the GTVR technique. In the examples above, it can be seen that the performance is very similar when $\gamma \leq 10$ as the $L = 7$ degree polynomial gives a good fit to the ideal function.

TABLE 2.4: Image Denoising (Barbara) : RMSE of GTVR and GPF

γ	Technique	Noise σ 5	15	25	35
0	GPF	4.95	14.82	25.05	34.80
	GTVR	4.95	14.82	25.05	34.80
0.01	GPF	5.04	15.23	24.98	34.76
	GTVR	5.04	15.23	24.98	34.76
0.1	GPF	4.79	14.64	24.68	35.01
	GTVR	4.79	14.64	24.68	35.01
1	GPF	3.31	12.99	23.83	33.47
	GTVR	3.31	12.97	23.82	33.47
10	GPF	1.27	10.63	22.00	32.14
	GTVR	1.17	9.70	20.99	31.44
100	GPF	1.22	10.06	17.06	29.58
	GTVR	0.56	5.84	11.81	25.34

TABLE 2.5: Image Denoising (Lena) : RMSE of GTVR and GPF

γ	Technique	Noise σ 5	15	25	35
0	GPF	5.03	15.00	25.36	34.08
	GTVR	5.03	15.00	25.36	34.08
0.01	GPF	5.03	14.89	25.27	34.25
	GTVR	5.03	14.89	25.27	34.25
0.1	GPF	4.73	14.43	25.02	35.40
	GTVR	4.73	14.43	25.02	35.40
1	GPF	3.31	13.44	23.70	33.79
	GTVR	3.31	13.43	23.69	33.79
10	GPF	1.58	10.67	20.10	29.78
	GTVR	1.42	9.43	18.93	28.98
100	GPF	1.22	10.00	19.97	28.24
	GTVR	0.64	6.02	14.50	23.09

TABLE 2.6: Image Denoising (Mandrill) : RMSE of GTVR and GPF

γ	Technique	Noise σ 5	15	25	35
0	GPF	5.04	15.18	24.65	34.64
	GTVR	5.04	15.18	24.65	34.64
0.01	GPF	4.91	14.96	25.20	34.48
	GTVR	4.91	14.96	25.20	34.48
0.1	GPF	4.79	14.49	24.64	35.78
	GTVR	4.79	14.49	24.64	35.78
1	GPF	3.25	13.27	22.95	33.65
	GTVR	3.25	13.26	22.95	33.65
10	GPF	1.48	10.03	20.29	29.39
	GTVR	1.34	9.08	19.25	28.34
100	GPF	1.23	8.92	18.73	32.11
	GTVR	0.52	5.51	13.84	28.54

TABLE 2.7: Image Denoising (Peppers) : RMSE of GTVR and GPF

γ	Technique	Noise σ 5	15	25	35
0	GPF	5.02	14.98	25.05	35.36
	GTVR	5.02	14.98	25.05	35.36
0.01	GPF	5.02	15.17	24.61	35.44
	GTVR	5.02	15.17	24.61	35.44
0.1	GPF	4.70	14.62	24.59	35.19
	GTVR	4.70	14.62	24.59	35.19
1	GPF	3.31	13.44	24.05	33.58
	GTVR	3.31	13.43	24.04	33.58
10	GPF	1.41	12.21	21.60	32.81
	GTVR	1.28	11.12	20.61	32.32
100	GPF	1.22	9.48	20.86	30.24
	GTVR	0.63	5.52	15.70	25.64

TABLE 2.8: Image Denoising (Cameraman) : RMSE of GTVR and GPF

γ	Technique	Noise σ 5	15	25	35
0	GPF	5.02	15.10	24.65	34.61
	GTVR	5.02	15.10	24.65	34.61
0.01	GPF	4.85	14.84	25.33	34.51
	GTVR	4.85	14.84	25.33	34.51
0.1	GPF	4.75	15.03	25.07	34.63
	GTVR	4.75	15.03	25.07	34.63
1	GPF	3.29	13.65	23.69	33.23
	GTVR	3.29	13.64	23.68	33.23
10	GPF	1.41	10.34	21.46	31.68
	GTVR	1.29	9.37	20.58	31.06
100	GPF	1.16	9.26	19.20	30.11
	GTVR	0.52	5.41	14.02	25.13

2.6 Further comparisons

The exact solution of GTVR is given by (2.12) and essentially requires the solution to the system of linear equations

$$\mathbf{H}\hat{\boldsymbol{x}} = \boldsymbol{t} \tag{2.24}$$

where $\mathbf{H} \equiv \boldsymbol{I} + \gamma \boldsymbol{B}$. For large systems, either the LDL or QR decomposition [38] of $\mathbf{H}$ can be used as part of the process of obtaining the solution. This approach is computationally more efficient than the process of calculating the inverse of $\mathbf{H}$ directly. A comparison is provided of the run-time using this exact method with the run-time using the GPF for 7 different graphs with different number of vertices. For the GPF case (2.15) (with L replacing N and $\hat{h}_m$ replacing H_m) is implemented using Horner's method of polynomial evaluation. Starting with

$$\hat{\mathbf{x}}_{L-1} = \hat{h}_L \mathbf{B}\mathbf{t} + \hat{h}_{L-1}\mathbf{t},$$

Horner's method computes the following

$$\hat{\mathbf{x}}_m = \mathbf{B}\hat{\mathbf{x}}_{m+1} + \hat{h}_m \mathbf{t} \tag{2.25}$$

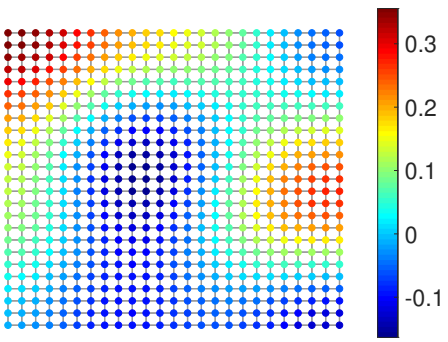
for $m = (L-2), (L-3), \ldots, 1, 0$. The GPF output is then $\hat{\boldsymbol{x}} = \hat{\mathbf{x}}_0$. The experiments were performed using Matlab on an i7-4790, 3.6GHz CPU machine and the graphs were generated using the GSP toolbox [45] which are visually represented in Figure 2.7 except for the Erdos Renyi graph, for which the nodes do not have positions which means; any visualisation of which is going to be random. The run-time comparisons are shown in Table 2.9 and it can be seen that the GPF method is computationally more efficient. When the number of vertices is large the run-time is more than one order of magnitude smaller.

TABLE 2.9: Run-time comparison between the GTVR and GPF methods

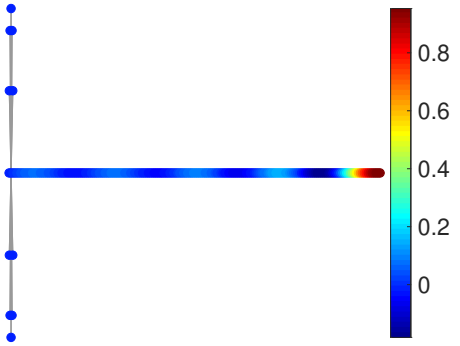
Graph	Technique	N (Number of vertices) 2500	5625	10000	15625
2D Grid	GPF	0.037	0.201	0.476	1.024
	GTVR	0.253	2.211	8.877	30.990
Comet	GPF	0.053	0.198	0.497	1.027
	GTVR	0.285	2.390	9.205	30.998
Community	GPF	0.040	0.230	0.550	1.024
	GTVR	0.347	2.262	9.106	31.014
Erdos Renyi	GPF	0.042	0.190	0.459	1.027
	GTVR	0.284	2.280	8.684	31.089
Spiral	GPF	0.074	0.207	0.462	1.067
	GTVR	0.439	3.978	15.503	56.812
Stochastic Block	GPF	0.050	0.192	0.500	1.025
	GTVR	0.286	2.189	8.934	31.040
Swiss Roll	GPF	0.158	0.305	0.638	1.255
	GTVR	0.293	2.358	11.879	95.014

Another approach to solving (2.24) that avoids calculating the inverse directly is to use the Landweber majorization-minimization iteration method. Starting with an initial $\mathbf{x}_0 = \mathbf{t}$, the method proceeds iteratively

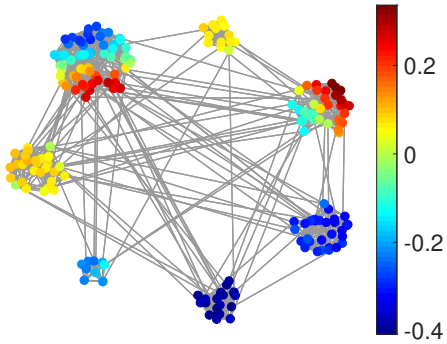
$$\hat{\mathbf{x}}_{i+1} = \hat{\mathbf{x}}_i + \frac{1}{\alpha}\mathbf{H}^T(\mathbf{t} - \mathbf{H}\hat{\mathbf{x}}_i) \tag{2.26}$$



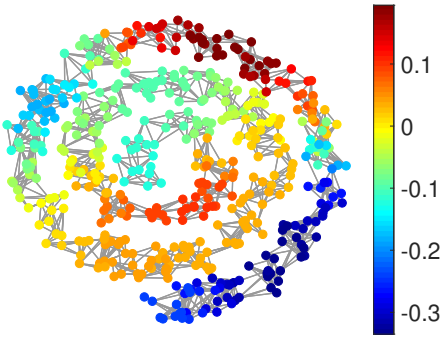
(a) 2D Grid (N = 625)



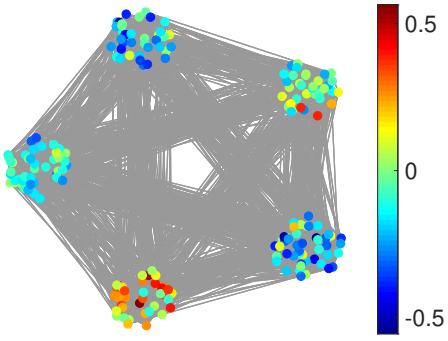
(b) Comet (N = 200)



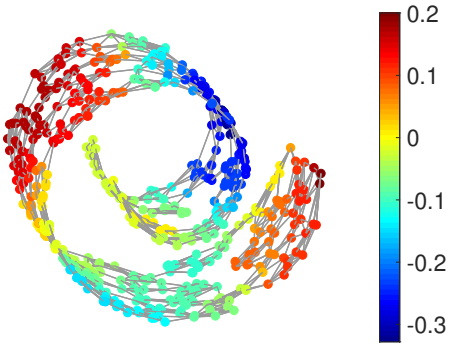
(c) Community (N = 200)



(d) Spiral (N = 500)



(e) Stochastic block (N = 200)



(f) Swiss roll (N = 500)

FIGURE 2.7: Visualisations of various graphs used from the GSP toolbox [45]

until $\hat{\mathbf{x}}_i$ converges. The hyper-parameter α affects the rate of convergence, and needs to be properly chosen, but there are no parameters with the GPF (using Horner's method (2.25)). The number of iterations required here is not fixed but with the GPF the number of iterations is determined by the degree of the filter L and is fixed. In (2.25), there is one matrix-vector multiplication but, there are two matrix-vector multiplications in (2.26) .

A least squares (LS) approximation to the function (2.14) was used to derive the GPF. An alternative to the LS approximation is the Chebyshev polynomial approximation [28]. In Table 2.10, a comparison is provided between the denoising performance of the LS and Chebyshev approximations. In most cases, the LS approximation gave better results.

TABLE 2.10: Denoising performance using Least-squares (LS) polynomial and Chebyshev polynomial (CP)

Graph	Technique	N (Number of vertices) 1849	5625	10000	15625
2D Grid	LSP	6.592	6.351	6.187	6.086
	CP	6.937	6.640	6.481	6.359
Comet	LSP	9.360	9.126	9.380	9.332
	CP	11.746	11.445	11.768	11.699
Community	LSP	4.137	3.623	3.708	3.344
	CP	4.455	3.937	4.032	3.691
Erdos Renyi	LSP	2.141	2.195	2.212	2.204
	CP	2.573	2.711	2.758	2.759
Stochastic Block	LSP	2.214	2.182	2.183	2.184
	CP	2.701	2.729	2.742	2.746
Swiss Roll	LSP	7.349	7.295	7.220	7.473
	CP	7.924	7.862	7.843	8.202

Finally, a comparison of the denoising performance of the GPF with a simple FIR filter that calculates the weighted average value of adjacent vertices as follows: $\hat{\mathbf{x}} = \mathbf{A}\mathbf{t}$ is provided. The graph used is the temperature sensor graph with 600 vertices from Section 2.5.1. The adjacency matrix is normalized as follows: $\boldsymbol{A} \rightarrow \boldsymbol{D}^{-1}\boldsymbol{A}$. The average RMSE over ten realization of the noise, for each variance value, is calculated. Figure 2.8 compares the denoising performance at different noise levels. It can be seen that the GPF gave consistently better results compared to the simple FIR filter. However the simple FIR filter has a very low complexity.

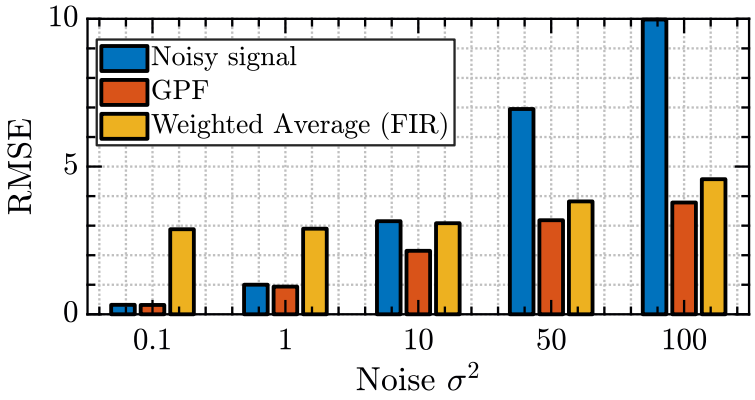


FIGURE 2.8: GPF vs Weighted average (FIR)

2.7 Chapter summary

The denoising of signals defined on graphs is formulated as a regularized optimization problem. An exact solution to the problem does exist but requires the calculation of a matrix inverse. The technique presented in this chapter is based on using a graph polynomial filter to approximate the inverse. The main idea behind the technique is to approximate the eigenvalues of the inverse using a least squares criterion. Eigendecomposition is not required and a low degree polynomial can be used to achieve results that are very similar to the exact solution. Polynomial filters have the additional advantages of the vertex-localization and the suitability for distributed processing.

Chapter 3

Guided Adaptive Interpolation Filter

3.1 Introduction

Image smoothing is a fundamental tool for several applications such as edge detection, feature extraction, and image restoration. Conventional linear time-invariant (LTI) filters are utilized to remove noise. Although these filters are computationally efficient, they are oblivious to image content and structures usually resulting in undesirable visual effects. This is due to the use of a spatially-invariant kernels which leads to smoothing or enhancing both; image structure and noise.

To address this problem, researchers have developed and studied numerous non-linear alternatives (spatially varying kernels) called edge-aware filters. The goal of edge-aware filters is to avoid smoothing across significant boundaries while eliminating the unimportant details. There are several edge-aware filters including: bilateral filter [46], weighted least square filter [47], edge-avoiding wavelets [48], guided filter [49], L_0 smoothing [50], L_1 smoothing [51], region co-variance [52], domain transform [53], local Laplacian filter [54], weighted median filter [55], fast global smoother [56], fast domain decomposition [57], the bilateral solver [58], L_0 gradient projection [59], side window guided filtering [60], guided wavelet filter [61], and filters based on adaptive mathematical morphology [62, 63].

In addition to edge-aware smoothing, these filters are broadly utilized in numerous applications in image processing and computational photography. Examples include image de-noising [64, 57], detail enhancement [65, 48], image fusion [66, 67], texture smoothing [68, 57, 56, 69, 70], single image haze removal [71], tone mapping of high dynamic range (HDR) images [47, 70,

48, 55, 72], anomaly detection in hyper-spectral images [73], object classification accuracy enhancement in hyper-spectral images [74], enhance the output of semantic segmentation algorithms [58], depth super-resolution/upsampling [58, 56], image colourization [58, 56, 48], image colour quantization [57], scale-space filtering [57, 69], style transfer [57, 55], optical flow estimation [55], compression artefacts removal [69, 59], content-aware resizing and stereo matching [55].

From the earlier review of the current state-of-the-art edge-aware smoothing algorithms, it can be noted that most of them are based on the idea of preserving distinctive structures while smoothing small scale details. Inspired by the success of recently published works on edge-aware filters and their valuable applications, the goal of this work is to propose and investigate a new edge-aware filter called guided interpolation edge-aware filter (GAIF).

This work is motivated by the guided image filter (GIF) [49] and adaptive interpolation filter (AIF) [68]. The key idea of this work is that edge-aware smoothing can be obtained by a local interpolation between the input image and a guidance image which, in the simplest case, could be a linearly smoothed version of the input by using a Gaussian filter. A fundamental difference between this work and those based on the AIF is that in this work the edges and the flat regions in the resultant image are locally selected from the original image and the guidance image, respectively, through interpolation process. On the other hand, in the AIF, the interpolation process is achieved by an iterative pixel-wise process over the entire image. Although the interpolation process in the AIF is achieved through a linear process, it is an iterative filter. As a result, the proposed filter is computationally more efficient than the AIF.

On the other hand, GIF assumes a patch-level linear model instead of the interpolation in GAIF, in other words, an output pixel is produced as a linear model of the patch centred at the corresponding pixel in a guidance image.

In the following sections, related works are summarized in Section 3.2. The mathematical model of the proposed guided adaptive interpolation filter, an algorithm to solve it, and a way to extend it are presented in Section 3.3. Section 3.4, is a discussion about the impact of parameter tuning and the smoothing performance. Applications demonstrating the efficiency of the proposed filter are presented in Section 3.5. Finally, a brief discussion and a conclusion about the results are presented in Section 3.6.

3.2 Related work

Non-linear filters can be divided into two groups based on the locality of the filtering effect; local filters which represent most of the non-linear filters in the literature [75] and global filters which are usually the solutions to optimization problems such as the weighted least-squares based (WLS) filter [47] [76] [77].

3.2.1 Kernel-based filters

Smoothing in kernel-based methods is achieved through a weighted average of the input signal values to yield each element of the output. The kernel is used to measure the similarities between pixels. These similarities are normalized and used as the weights for the averaging. Specifically, the filtered pixel denoted y_i is computed from the pixels of the input image denoted $\{x_j\}$ as shown below:

$$y_i = \sum_j W_{ij} x_j \tag{3.1}$$

where the weight W_{ij} is a function of the image to be filtered [46] or another image in the case of joint/cross-filtering [78]. Milanfar et al. [75] has presented an excellent exposition about this kind of filters. Kernel-based filters are generally considered to be local; because, a filtered pixel is computed as a weighted average of its surrounding pixels.

3.2.2 Guided image filter

The guided image filter (GIF) [49] has received a great deal of attention by the community because it has many desirable properties. For example, the filter formulation, intuitively, makes sense from a statistical regression perspective, and the algorithm is computationally efficient $O(N)$. In addition, filter results are very compelling. These qualities contributed to popularity of GIF and motivated other researchers to borrow ideas from it. Ham et al. [77], adapted the idea of guidance image to the regularization term at the global level of image rather the patch level as is done in the GIF. Li et al. [79], introduced weight to the regularization parameter to enhance the edge-awareness of the original GIF. Lu et al. [80] have proposed another weighting function which is more robust to the regularization parameter.

The original guided filter (GIF) [49] assumes a local linear patch model. A pixel at location p in the kth patch Ω_k is represented as $p \in \Omega_k$. The pixel in output image J_{pk} (the subscript k refers to the kth patch) is related to the corresponding pixel in the guidance image G_p in the following way:

$$J_{pk} = a_k G_p + b_k \qquad \forall p \in \Omega_k \tag{3.2}$$

where a_k and b_k are model parameters for the pixels in Ω_k. They are determined by solving the following optimization problem:

$$\underset{a_k, b_k}{\operatorname{argmin}}\, C(a_k, b_k) = \sum_{p \in \Omega_k} (a_k G_p + b_k - I_p)^2 + \epsilon a_k^2 \tag{3.3}$$

where ϵ is a user specified regularization parameter. Since for a square-shaped patch of $|\Omega_k|$ pixels, a pixel G_p belongs to $|\omega_p|$ overlapping patches. Each resulting in an output patch. GIF takes the average of these outputs as the final filter output J_p.

$$J_p = \frac{1}{|\omega_p|} \sum_{k \in \omega_p} J_{pk} \tag{3.4}$$

where ω_p is the set of patches to which the pixel p belongs. This is a simple model averaging process. For completeness here, it is worth mentioning that other forms of model averaging can be adopted [81].

The idea of using a pair of images to produce the output image was first described in the joint/cross bilateral filter [82] [78] which included the guidance information in an *ad-hoc* fashion. The idea made disciplined in [49] [83] by modelling the image patches as a linear transformation to the corresponding patches in the guidance image. Extensions to this idea include using two guidance images [84], and making the guidance procedure global rather than local [85].

3.2.3 Energy minimization global filters

Most of the optimization-based filters are global filters. In another word, they minimize a cost function calculated over the whole image as opposed to the patch-oriented approach. In [86], Xu et al. proposed the relative total variation (RTV) measure to distinguish between structures and texture. Later, RTV is used as a regularizer in a global optimization problem. RTV achieves good texture smoothing. Zhou et al. [69] proposed a scale-aware

measure and included it in an objective function to achieve scale-aware filtering called Iterative Global Optimization filter (IGO). Liu et al. [87] proposed a global optimization model involving truncated Huber function, the resulting model is non-convex and non-smooth, leading to some desirable properties. The authors demonstrated the effectiveness of this model on a number of tasks. A major drawback in these methods is their computational complexity which comes from solving large linear systems [56] [88] [57].

3.2.4 Interpolation based filters

Al-nasrawi et al. [68], proposed a pixel level edge-aware smoothing technique that utilises the idea of interpolation between two images, which are the observed/original image and a smooth version of observed/original. The filtering process proceeds in an iterative fashion and the interpolation weights are updated in each iteration based on the residual between the observed/original image and the current estimate image.

Unsharp masking [7] is a classical technique used to improve the sharpness of an image. Two versions of the input image are used to produce the result, a sharp negative version and a smoothed positive. The parameter in unsharp masking is usually fixed throughout the image domain. Main applications of unsharp masking revolve around contrast enhancement.

3.2.5 Edge-preserving filtering

Several edge-preserving smoothing operators have been proposed in the literature. One of the earliest of these operators is the bilateral filter [46], which has been used in numerous applications including HDR tone-mapping [89], [90] and highlight removal [91]. A major drawback in the smoothing performance of the bilateral filter is the gradient-reversal which results in halos when used for image enhancement [47]. Farbman et al. [47] tackled the gradient reversal problem by solving a global optimization problem. He et al. [49] proposed GIF, a more efficient filter, by solving a local optimization problem but the results still suffer from the halo artefacts [79]. Xu et al. proposed an L_0 adaptation of the total variation filter [12], which produces piece-wise constant results. Its performance was demonstrated on a number of applications. Ham et al. [92] proposed the static-dynamic image filter which solves a global non-convex optimization problem that involves two guidance images: the current estimate and an external image. The authors have demonstrated

its effectiveness at texture removal and depth super-resolution. However, they noted that the filter produces artefacts in flash/no-flash and RGB-NIR denoising tasks.

3.2.6 Our contribution

The novel contributions in this chapter are as follows:

1. General framework for patch-based interpolation is presented, along with algorithms for two special cases.
2. To further enhance the edge-preserving performance of the proposed filter, two weighting functions, that boost or suppress the penalty term based on the image content, are introduced and compared.
3. A relationship between GAIF and GIF is highlighted where a special case of GAIF is also a special case of GIF.

3.3 Guided adaptive interpolation filter

3.3.1 Definition

Smoothing images can readily be achieved using any linear low-pass filter such as the Gaussian filter. However, the resulting filtered image is equally smoothed everywhere regardless of the image contents. To rectify the lack of discriminating power in linear filters, this chapter proposes a local, patch-level, interpolation model. This local interpolation is between the corresponding patches of two images. The first image, denoted by I, is the raw image to be filtered. The second is a smooth image, denoted by M, produced using any linear or non-linear smoother applied on I as follows:

$$M = f(I) \tag{3.5}$$

where $f(.)$ is the smoother/filter of choice. More concretely, and adopting the same notation as the GIF, J_{pk} is the output pixel at location p due to the local model derived from the kth patch. J is used to denote the interpolation between I and its smooth version M in the patch Ω_k:

$$J_{pk} = \alpha_k I_p + (1 - \alpha_k) M_p \qquad \forall p \in \Omega_k \tag{3.6}$$

where α_k is an interpolation parameter and is assumed to be constant in the patch Ω_k, p is the index of pixels within the patch Ω_k. Each patch Ω_k has $|\Omega_k|$ pixels.The output patch J_{pk} in (3.6) is closer to I_p if α_k is higher than 0.5, and this should be the case if the patch k is part of an edge. Alternatively, the output patch J_{pk} is closer to M_p if α_k is lower than 0.5, this should be the case if the patch k is not part of an edge, i.e. smooth area. In other words, the images I and M are responsible for the edges and the smooth areas in the image J respectively.

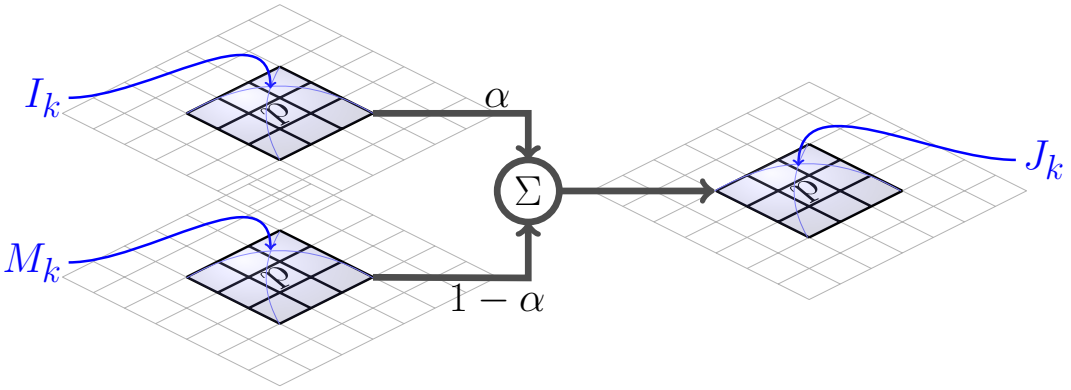


FIGURE 3.1: GAIF block-diagram. Patches at index k in the images I and M are interpolated to produce the corresponding patch in image J

To determine the interpolation coefficient α_k, a solution to (3.6) is sought by minimizing the difference between the approximate image J and the image to be filtered I. Concretely, performing the minimization of the following cost function for each patch k:

$$C(\alpha_k) = \sum_{p \in \Omega_k} |\alpha_k I_p + (1 - \alpha_k) M_p - I_p|^\gamma + \epsilon \alpha_k^2 \tag{3.7}$$

where ϵ is a regularization parameter stopping α from blowing up and controlling the amount of emphasis placed on I and M. The parameter $\gamma \in \{1, 2\}$ generalizes two distinct models which, apparently, have different solutions but are similar in performance. The reason behind that will be demonstrated. The solutions of the model (3.7) are as follows:

- $\gamma = 1$

$$\underset{\alpha_k}{\operatorname{argmin}}\, C(\alpha_k) = \sum_{p \in \Omega_k} |\alpha_k I_p + (1 - \alpha_k) M_p - I_p| + \epsilon \alpha_k^2 \tag{3.8}$$

$$\alpha_k = \min\left(1, \frac{|\Omega_k|}{2\epsilon} \mathrm{MAE}_k\right) \tag{3.9}$$

where $|\Omega_k|$ is the number of pixels in the patch k and

$$\mathrm{MAE}_k = \sum_{p \in \Omega_k} |I_p - M_p| / |\Omega_k|$$

- $\gamma = 2$

$$\underset{\alpha_k}{\operatorname{argmin}}\, C(\alpha_k) = \sum_{p \in \Omega_k} (\alpha_k I_p + (1 - \alpha_k) M_p - I_p)^2 + \epsilon \alpha_k^2 \tag{3.10}$$

$$\alpha_k = \frac{\mathrm{MSE}_k}{\mathrm{MSE}_k + \tilde{\epsilon}_k} \tag{3.11}$$

where $\tilde{\epsilon}_k = \epsilon / |\Omega_k|$ and $\mathrm{MSE}_k = \sum_{p \in \Omega_k} (I_p - M_p)^2 / |\Omega_k|$.

Just as is the case in GIF [49], the pixel at location p belongs to $|\omega_p|$ overlapping patches as shown in Figure 3.2, which results in $|\omega_p|$ output patches per pixel p, each denoted by J_{pk}. Following the model averaging principle, the average of these outputs is taken as the final output as follows:

$$J_p = \frac{1}{|\omega_p|} \sum_{k \in \omega_p} J_{pk} \tag{3.12}$$

$$= \bar{\alpha}_p I_p + (1 - \bar{\alpha}_p) M_p \tag{3.13}$$

where $\bar{\alpha}_p = \frac{1}{|\omega_p|} \sum_{k \in \omega_p} \alpha_k$.

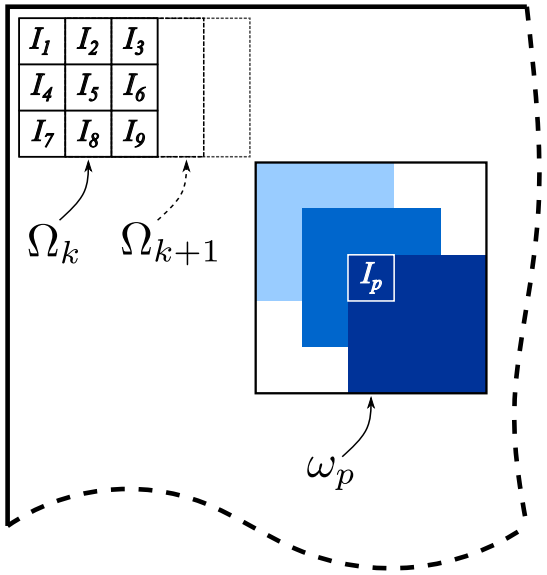


FIGURE 3.2: Overlapping patches

Analysis

The solutions in (3.9) and (3.11) are in terms of the patch MAE and MSE, respectively which are defined earlier. The square of MAE can be considered as an approximation of MSE in this specific case and this can be justified empirically. As such, to simplify the following analysis, the assumption that MSE $\approx$ MAE^2 is made. With this assumption in mind, a one-to-one comparison is facilitated. In Figure 3.3, a pair of $|\Omega_k|$ and ϵ is considered in (3.9) then the best corresponding pair of $|\Omega_k|$ and ϵ is found in (3.11) such that the second function is closest to the first in the ℓ_2 sense.

This former parameter tuning allows us to highlight the difference between the two cases when they are the closest to each other. There are observations; (1) The two filters treat a patch Ω_k of the input image in a generally similar fashion. In other words, at the very low and the very high values of MAE the two functions are equal. (2) The two filters differ around the $\frac{2\epsilon}{|\Omega_k|}$ point. Before this point, α is higher for the $\gamma = 2$ case signifying that the output patch gets a higher contribution from the input image. After the $\frac{2\epsilon}{|\Omega_k|}$ point, the opposite occurs, the output patch gets a lower contribution from the input image than in the $\gamma = 1$ case. Interestingly, the $\gamma = 1$ filter produces $\alpha = 1$ for MAE $> \frac{2\epsilon}{|\Omega_k|}$

However, it was experimentally found that it is very hard to visually discern the differences between the two images resulting from the two filters

thus, from this point onward whenever GAIF is mentioned, $\gamma = 2$ is implicitly implied.

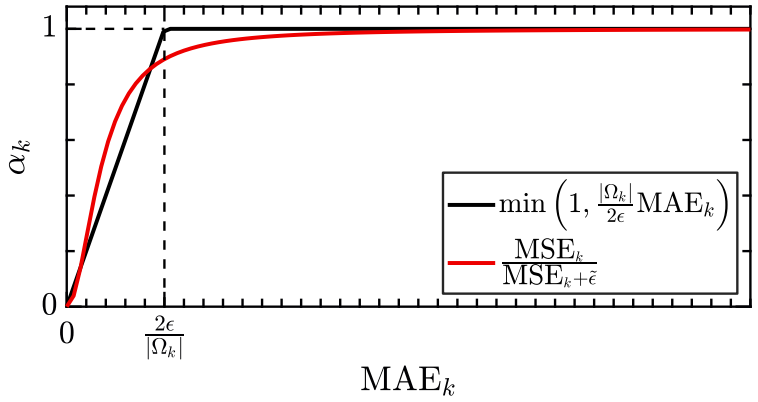


FIGURE 3.3: Comparison between the solutions in (3.9) and (3.11).

3.3.2 Weighted adaptive interpolation filter

From the earlier discussion in Section 3.3.1, the proposed scheme produces each output pixel by interpolating the patches centred around the corresponding pixels in the two input images, see Figure 3.1. The choice of which patch I_k or M_k contributes more to the output pixel is encoded as α_k. In other words, both images I and M contribute to the final GAIF filtering result J as can be seen in (3.6). More specifically, strong edges are contributed by I while smooth areas are contributed by M. This means, it is preferable to have $\alpha_k \approx 1$ at the locations of edges and $\alpha_k \approx 0$ elsewhere.

The proposed model (3.7) has a regularisation term that puts a cost on choosing $\alpha = 1$ which is what the solution would be without the extra regularization term. However, the impact of this regularization term is controlled by a single tuning parameter ϵ which is fixed for the whole image.

This leads to questioning the possibility of automatically adjusting the tuning parameter ϵ such that the impact of the regularization parameter becomes negligible, allowing the model to pick $\alpha_k \approx 1$ at the edges, and amplify the impact of the regularization parameter in smooth areas to allow the solution to be $\alpha_k \approx 0$

To this end, the following model is proposed:

$$\underset{\alpha_k}{\text{argmin}}\, C(\alpha_k) = \sum_{p\in\Omega_k} (\alpha_k I_p + (1-\alpha_k)M_p - I_p)^2 + \tilde{\epsilon}_p \alpha_k^2 \tag{3.14}$$

where $\tilde{\epsilon}_p = \epsilon\theta_p$ which means that ϵ is tuned at the pixel level rather than globally as it is the case in the initial model (3.7). Li et al. [79] proposed such formulation for the GIF. In this work, two variants of θ_p are proposed as follows:

Define a re-scaling function

$$\phi(x) = A - \frac{Ax}{\beta + |x|} \tag{3.15}$$

Variant 1:

$$\theta_p = \phi(\eta_p) \tag{3.16}$$

$$\text{where } \eta_p = \frac{1}{|\omega_p|} \sum_{k \in \omega_p} \frac{\hat{\eta}_k}{\hat{\eta}_k + c}$$

$$\text{and } \hat{\eta}_k = \frac{1}{|\Omega_k|} \sum_{p \in \Omega_k} |I_p - \mu_p|$$

Variant 2:

$$\theta_p = \phi\left(\sigma_p\left(\left\{\eta^{(i)}(I)\right\}\right)\right), \qquad \forall i \in [1,5] \tag{3.17}$$

$$\text{where } \eta^{(i)}(I) = \text{MEDFILT}(I, 3 + 2(i-1))$$

where MEDFILT represents a 2D median filter operation with the input image and window size as the first and second arguments respectively, $|\omega_p|$ is the cardinality of pixels in the region ω_p and the parameters in equation (3.15) were found and set empirically to $A = 5$ and $\beta = 0.025$ throughout this chapter.

The role of the rescaling function in (3.15) is twofold. Firstly; it flips the sign of its argument, secondly; it makes the output saturate at $2A$ for negative inputs and saturate at 0 for large positive inputs. In other words, the rescaling function $\phi(.)$ makes sure that the value of θ_p is always positive and bounded thus avoiding potential numerical issues. c in (3.16) is a small constant[1].

It is important here to note that these variants result in slightly different smoothing effects as can be clearly seen in Figure 3.4.

In variant 1, the pixels in a window are used in the computation of θ_p which measures the relative mean absolute deviation of the central pixel in a window to the mean absolute deviations of the surrounding pixels followed by the re-scaling function $\phi(x)$ to bound the scaling of α_k within the range {0,5}.

[1] c was fixed throughout this paper as $c = 1 \times 10^{-6}$

In variant 2, the input image I is filtered with five median filters of increasing windows sizes producing five different values for each pixel, the standard deviation of the five samples is re-scaled with $\phi(x)$ to produce the final scaling parameter θ_p.

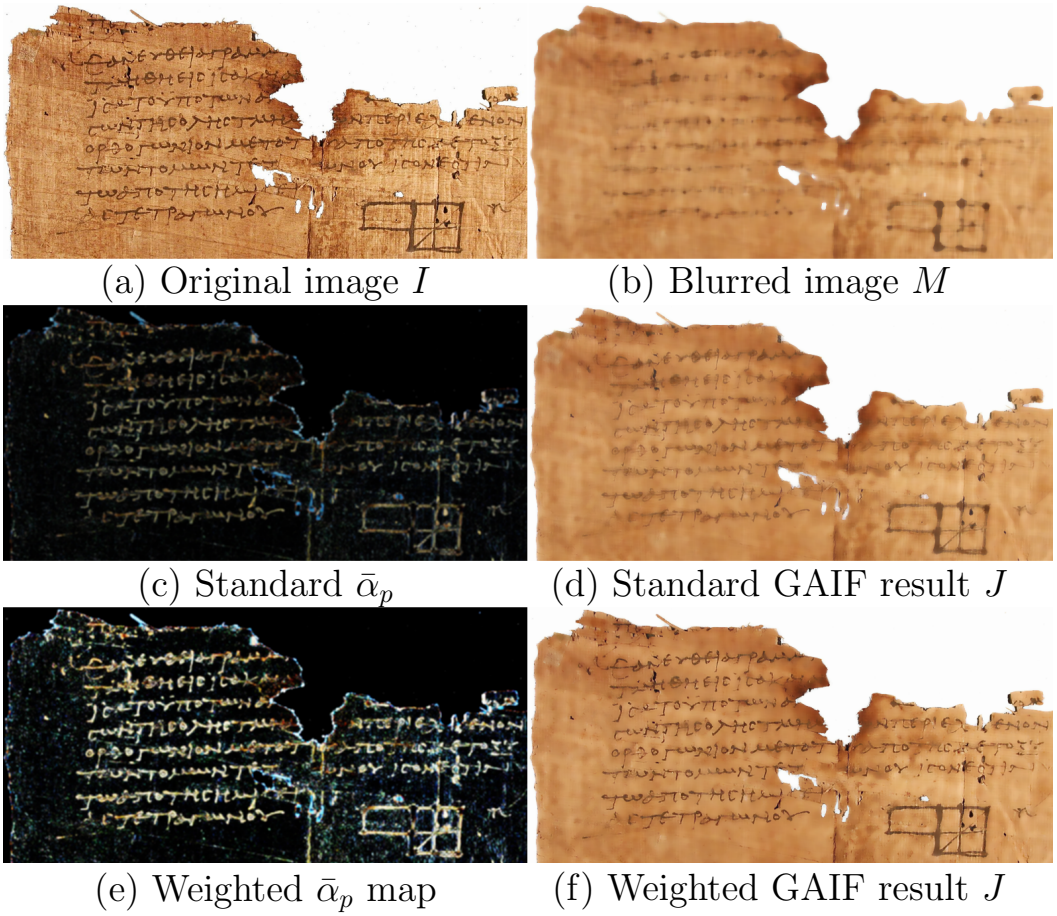
Figures 3.4 and 3.5 illustrate the way $\bar{\alpha}_p$ changes after adding the weighted regularization across the image domain. Darker regions represent areas where the filter leans towards the smooth version i.e more emphasis on M, while light regions capture important edges where the filter leans towards the original version i.e more emphasis on I.

(a) Original image I (b) Blurred image M

(c) Standard $\bar{\alpha}_p$ (d) Standard GAIF result J

(e) Weighted $\bar{\alpha}_p$ map (f) Weighted GAIF result J

FIGURE 3.4: Variant 1 of weighted GAIF. (a) is input image I. (b) is median filtered version of I with window size $= 11$. (c) and (e) are $\bar{\alpha}_p$ image in the case of standard GAIF (3.11) and variant 1 of the weighted GAIF (3.14) respectively. Light areas represent more contribution from I than M and darker regions represent more contribution from M than I. (d) and (f) are the results of GAIF and weighted GAIF respectively with $\epsilon = 1$.



(a) Original image I (b) Blurred image M

(c) Standard $\bar{\alpha}_p$ (d) Standard GAIF result J

(e) Weighted $\bar{\alpha}_p$ map (f) Weighted GAIF result J

FIGURE 3.5: Variant 2 of weighted GAIF. (a) is input image I. (b) is median filtered version of I with window size $= 11$. (c) and (e) are $\bar{\alpha}_p$ image in the case of standard GAIF (3.11) and variant 2 of the weighted GAIF (3.14) respectively. Light areas represent more contribution from I than M and darker regions represent more contribution from M than I. (d) and (f) are the results of GAIF and weighted GAIF respectively with $\epsilon = 1$.

3.3.3 Filter kernel

GAIF is a local filter, in particular, the resulting image J is a local linear combination of the input image I and a smoothed version of I namely M as follows:

$$J_p = \alpha_k I_p + (1 - \alpha_k) M_p, \quad \forall p \in \Omega_k, \tag{3.18}$$

which is considered a general model. The former model admits the special case

$$J_p = \alpha_k I_p + (1 - \alpha_k) M_k, \quad \forall p \in \Omega_k, \tag{3.19}$$

where the output pixel in modelled as local blend of the input pixel I_p and a patch level constant M_k. The output pixel of both models (3.18) and (3.19) can be written as follows:

$$J_p = \sum_{j \in \Omega_k} W_{pj}(I, M) I_j \tag{3.20}$$

where the $W_{pj}(I, M)$ is the filter kernel. The kernel depends on both I and M, where M is generally a filtered version of the image I. One example case is to filter I with a box filter to produce M where, the kernel of the model in (3.19) is similar to the self-guided case of the GIF [49] which has the following explicit formula:

$$W_{pj}(I) = \frac{1}{|\Omega_k|^2} \sum_{p \in \Omega_k} \sum_{j \in \Omega_k} \left(1 + \frac{(I_p - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon} \right) \tag{3.21}$$

Details can be found in Appendix A

3.3.4 $O(N)$ Time exact algorithm

A major advantage of the proposed filter over the global energy minimization schemes is that it is an $O(N)$ complexity exact algorithm. $O(N)$ means that the filter computational complexity depends linearly only on the number of pixels. In other words, it is independent of the window size, which allows the user to choose any windows size without additional computational cost as is the case with the bilateral filter [46], non-local means [93], SD [92], RTV [86], and the more recent filter by Wang et al. [94] just to name few. This property is shared with the guided image filter [49]. Table 3.1 is a summary of the computational complexities of some well-known techniques in the literature for comparison.

TABLE 3.1: Computational complexities of some of the well-known filters in the literature. N is the total number of pixels in an image, $|\Omega_k|$ is number of pixels in the patch, w is the number of pixels in a window, and n is the number of iterations.

Filter	Complexity
Bilateral	$O(\|\Omega_k\|.N)$
NLM	$O(\|\Omega_k\|.w.N)$
SD	$O(n.\|\Omega_k\|.N)$ Assuming that the linear system can be solved in $O(N)$
RTV	$O(n.\|\Omega_k\|.N)$
GF	$O(N)$

3.4 Parameters setting and details smoothing

3.4.1 Parameter setting

GAIF has two parameters to tune; the radius r of a patch Ω_k, which is an odd integer and can take the values $\{3, 5, 7, \text{etc}\}$, the second parameter being the regularization parameter ϵ. It is observed that increasing r results in better edge preservation. On the other hand, increasing ϵ was found to result in increasingly smoother images. Those observations can be verified in Figure 3.6.

The behaviour of GAIF at various values of ϵ and r can be explained by referring to Section 3.3.3. For the special case where M is an average filter, the relationship between the output pixels and the input pixels is encoded in the explicit formula in (3.21). From (3.21), it cab be noticed that; increasing ϵ results in an averaging effect while increasing r, which increases the number of pixels in Ω_k, increases the reliance on the image I, hence more edges.

3.4.2 The role of M

The image M in (3.6) is responsible for the smoothing effect hence, it is important to demonstrate the role it plays in the GAIF filtering. The choice of $f(.)$ in (3.5) can be considered as a tuning parameter with the highest gains in smoothing performance achieved for non edge-aware filters ($f(.)$). To this end, a comparison demonstrating the effect of using Gaussian and Median filters to construct the image M on various images, is provided. Figures 3.8 to 3.15 demonstrate the role of different M images on the filtering outcome of the four images in Figure 3.7.

FIGURE 3.6: Edge-preserving smoothing using GAIF filter with different kernel sizes r and values of the regularization parameter ϵ. Lower ϵ values correspond to sharper outputs. Larger patch radius r results in better edge preservation.

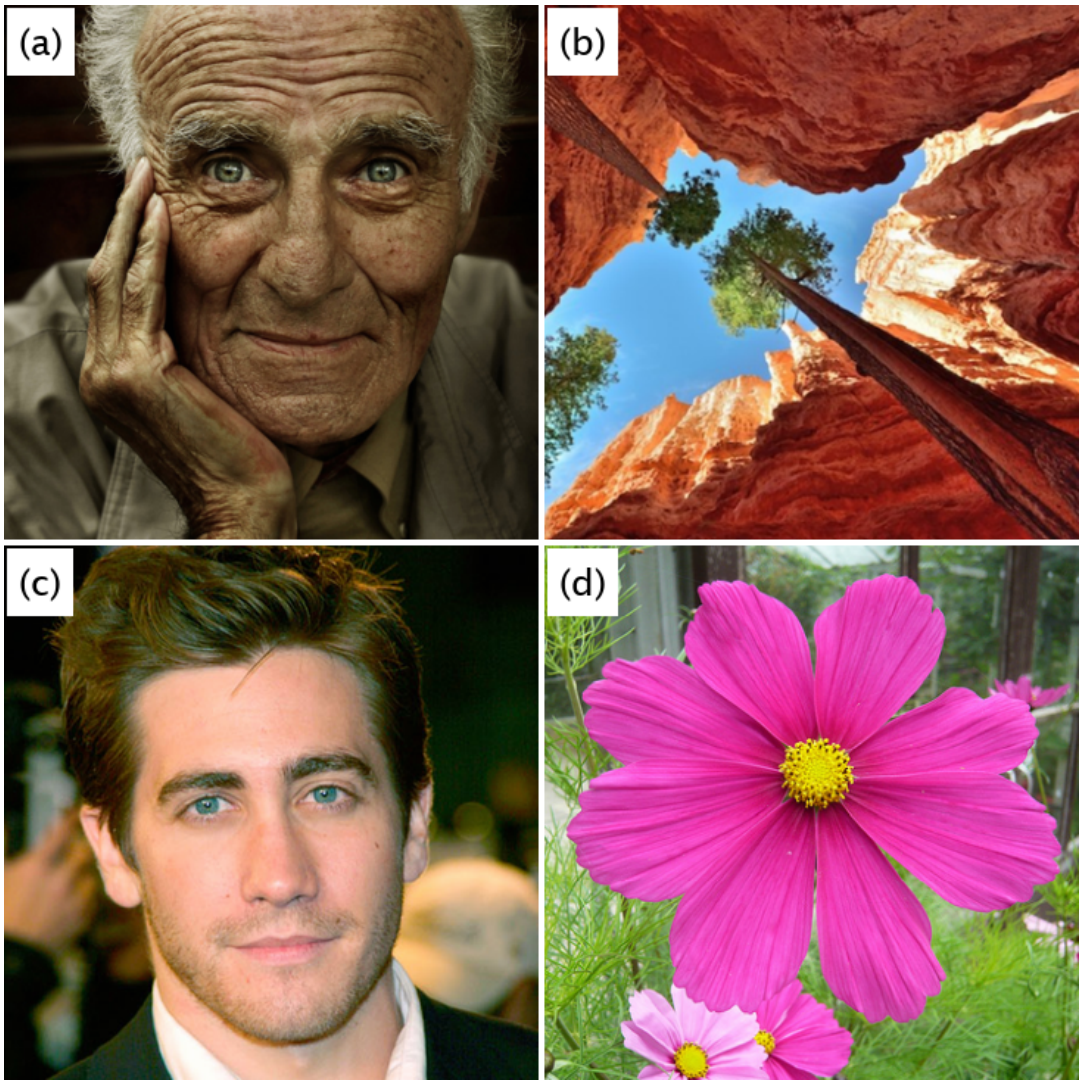


FIGURE 3.7: Input images used to demonstrate the role of the smooth image M.

FIGURE 3.8: The role of the smooth image M. (a) and (c) are M images produced using a Gaussian filter with $\sigma = 2$ and $\sigma = 7$ respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (a) while M being the image (a) and (c) respectively.

FIGURE 3.9: The role of the smooth image M. (a) and (c) are M images produced using a Median filter with window size 9 and 21 respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (a) while M being the image (a) and (c) respectively.

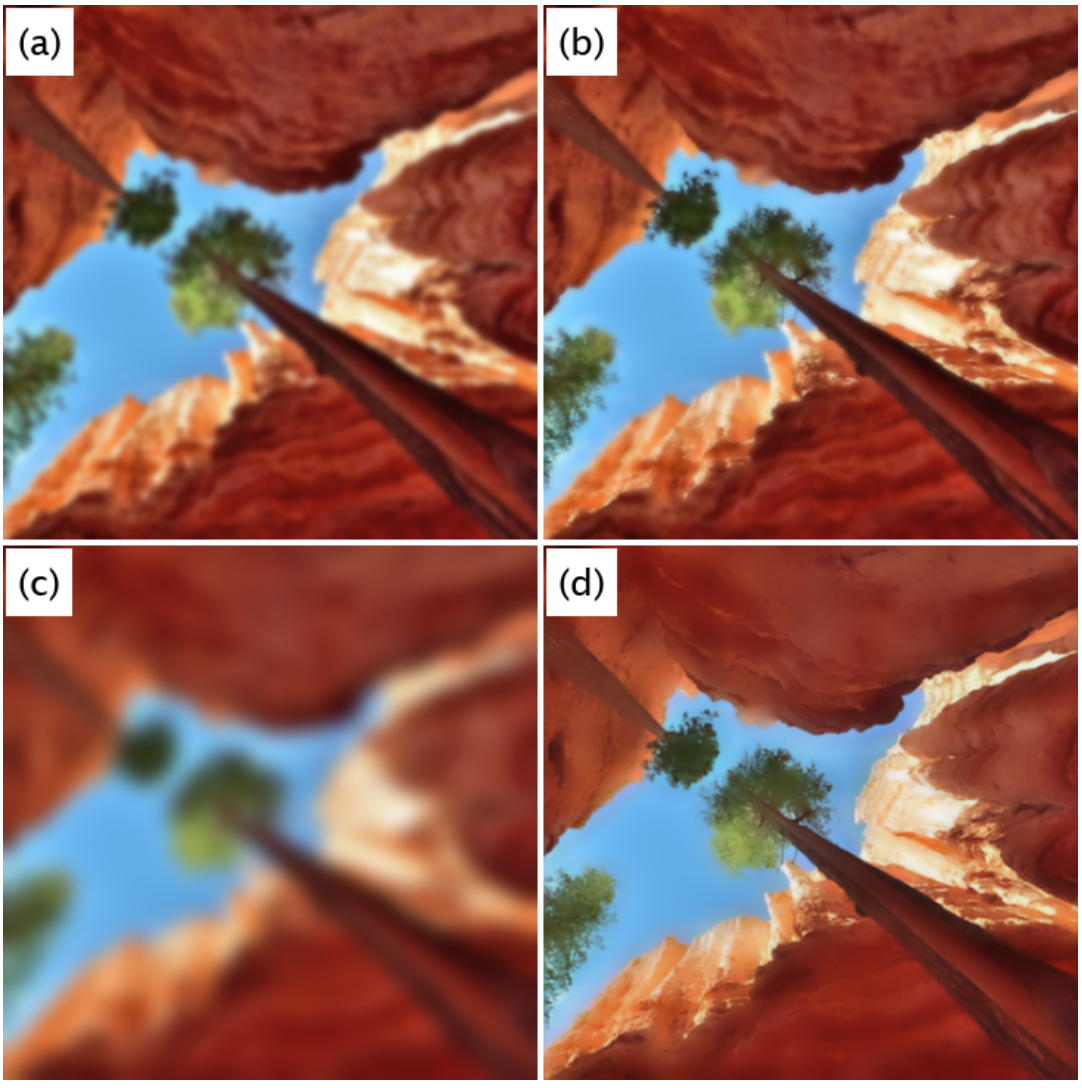


FIGURE 3.10: The role of the smooth image M. (a) and (c) are M images produced using a Gaussian filter with $\sigma = 2$ and $\sigma = 7$ respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (b) while M being the image (a) and (c) respectively.

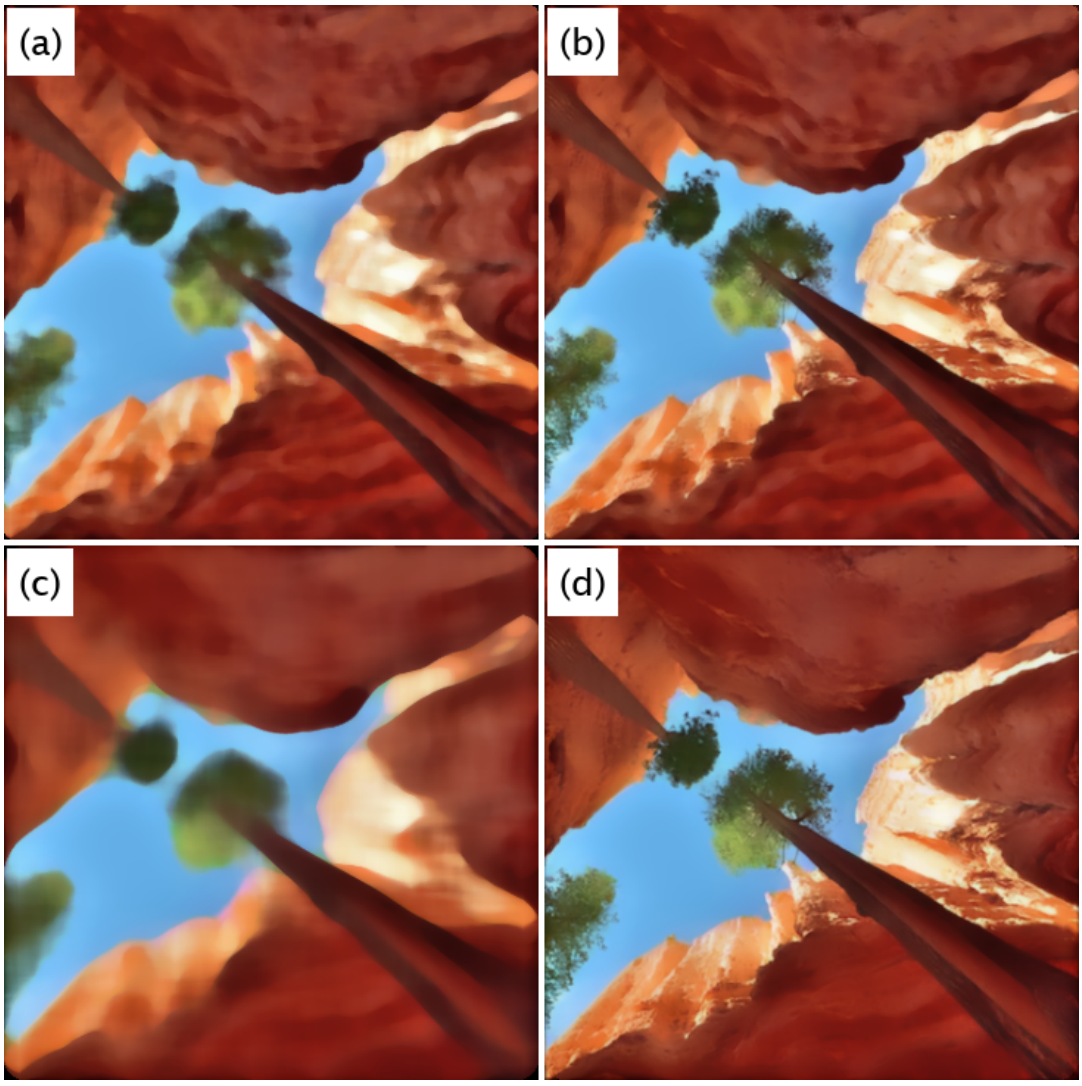


FIGURE 3.11: The role of the smooth image M. (a) and (c) are M images produced using a Median filter with window size 9 and 21 respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (b) while M being the image (a) and (c) respectively.

FIGURE 3.12: The role of the smooth image M. (a) and (c) are M images produced using a Gaussian filter with $\sigma = 2$ and $\sigma = 7$ respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (c) while M being the image (a) and (c) respectively.

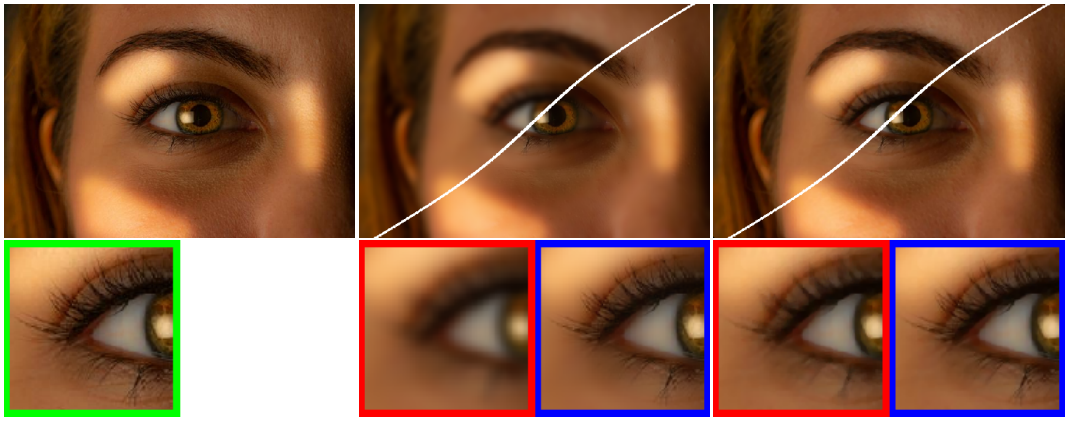
FIGURE 3.13: The role of the smooth image M. (a) and (c) are M images produced using a Median filter with window size 5 and 13 respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (c) while M being the image (a) and (c) respectively.

FIGURE 3.14: The role of the smooth image M. (a) and (c) are M images produced using a Gaussian filter with $\sigma = 2$ and $\sigma = 7$ respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (d) while M being the image (a) and (c) respectively.

FIGURE 3.15: The role of the smooth image M. (a) and (c) are M images produced using a Median filter with window size 9 and 21 respectively. (b) and (d) are GAIF filtered images with I being the original image in Figure 3.7 (d) while M being the image (a) and (c) respectively.

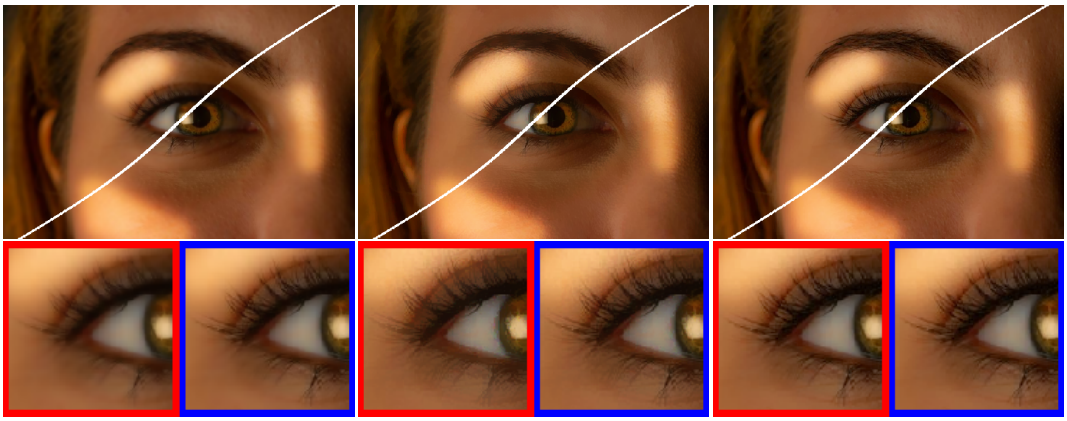
3.4.3 Details smoothing

Figures 3.16 to 3.18 are comparisons between the smoothing results. The top part merges two filtered version of the eye image, upper-left is the eye smoothed using the named filter, lower-right is the smoothed with GAIF. Here the GAIF smoothed image is a result of filtering the original image, guided by the smoothed image using the named filter. Filters used in the comparison include the classical Gaussian and median filters, guided filter (GIF) [49], sub-window box filter (SWF) [95], weighted least-squares (WLS) [47], static-dynamic filter (SD) [92], rolling guidance filter (RGF) [96] and relative total variation filter (RTV) [86]. In the close-up views, it is clear that GAIF is better preserving the eyelashes in comparison with other filters, meanwhile smoothing the other parts of the face. This result demonstrates the efficiency of GAIF at improving on the smoothing results of linear and non-linear filters. This improvement comes at a minimal computational cost and no reformulation of these filters is required.



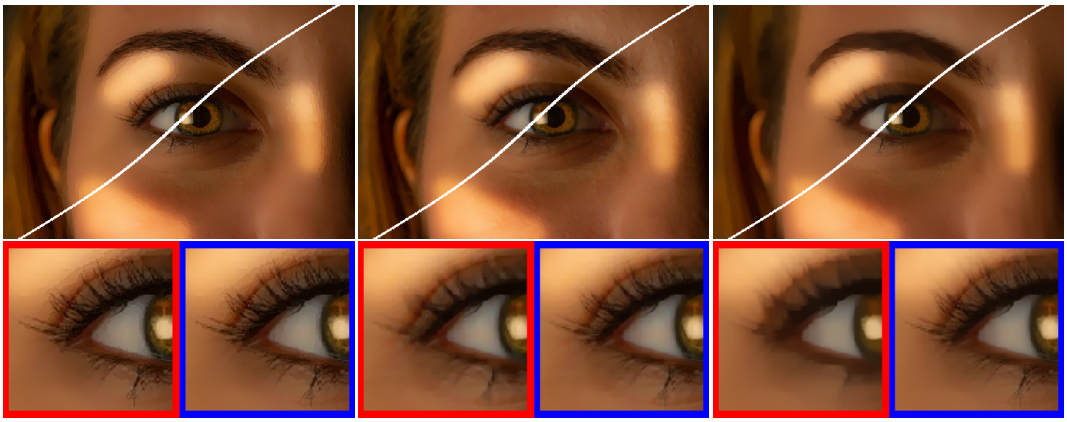
(a) Original (b) Gaussian (c) Median

FIGURE 3.16: Image smoothing (b) Gaussian $\sigma = 6$, (c) median 5×5 and GAIF $\epsilon = 0.04$ for all cases. The top part compares two results, to the left is a result of the indicated filter, to the right is a result of filtering with GAIF. The bottom part is zoomed-in versions of the top part, the red box is for the result of the indicated filter and the blue box is for the GAIF result.



(a) GIF (b) SWF (c) WLS

FIGURE 3.17: Image smoothing (a) GIF $r = 5, \epsilon = 0.01$, (b) SWF $r = 5$, (c) WLS $\lambda = 0.05, \alpha = 1$ and GAIF $\epsilon = 0.04$ for all cases. The top part compares two results, to the left is a result of the indicated filter, to the right is a result of filtering with GAIF. The bottom part is zoomed-in versions of the top part, the red box is for the result of the indicated filter and the blue box is for the GAIF result.



(a) SD (b) RGF (c) RTV

FIGURE 3.18: Image smoothing (a) SD $\lambda = 5, \mu = 50, \nu = 400, \text{iter} = 10$, (b) RG $\sigma_s = 3, \sigma_r = 0.01, \text{iter} = 4$, (c) RTV $\lambda = 0.005, \sigma = 3$ and GAIF $\epsilon = 0.04$ for all cases. The top part compares two results, to the left is a result of the indicated filter, to the right is a result of filtering with GAIF. The bottom part is zoomed-in versions of the top part, the red box is for the result of the indicated filter and the blue box is for the GAIF result.

3.5 Applications and Experimental Results

3.5.1 Single image haze removal

Tarel et al. [97, 98] proposed a method for single image dehazing which works in the four steps summarized in Figure 3.19 (black arrows path). To improve on the results of this technique, a veil refinement step is introduced which involves filtering the inferred veil image using an edge-aware filter. GAIF is used to perform the veil refinement step, more specifically, the image I is the inferred veil image while the image M is a mean or median filtered version of I. The authors in [97, 98] have considered both median and bilateral filters for veil refinement. Figures 3.20 and 3.21 are comparisons between three state-of-the-art techniques, including the techniques proposed in [97, 98], dark channel prior [71] and our technique. The top parts of the figures are the results of various techniques and the bottom parts are close-up views of two regions of the images. GAIF is resulting in the clearest and sharpest result among the four techniques.

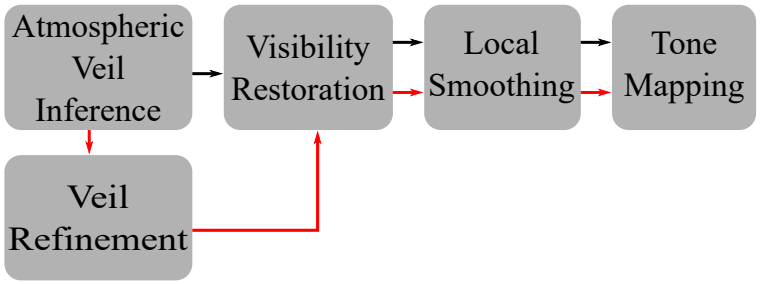
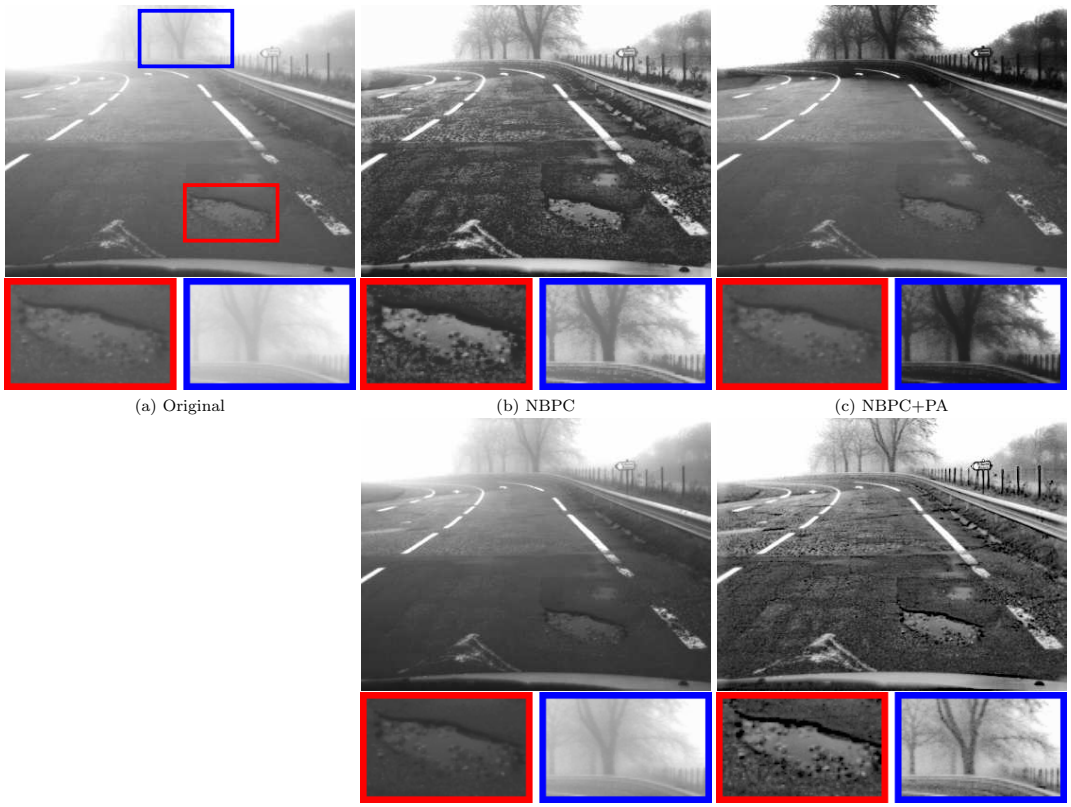


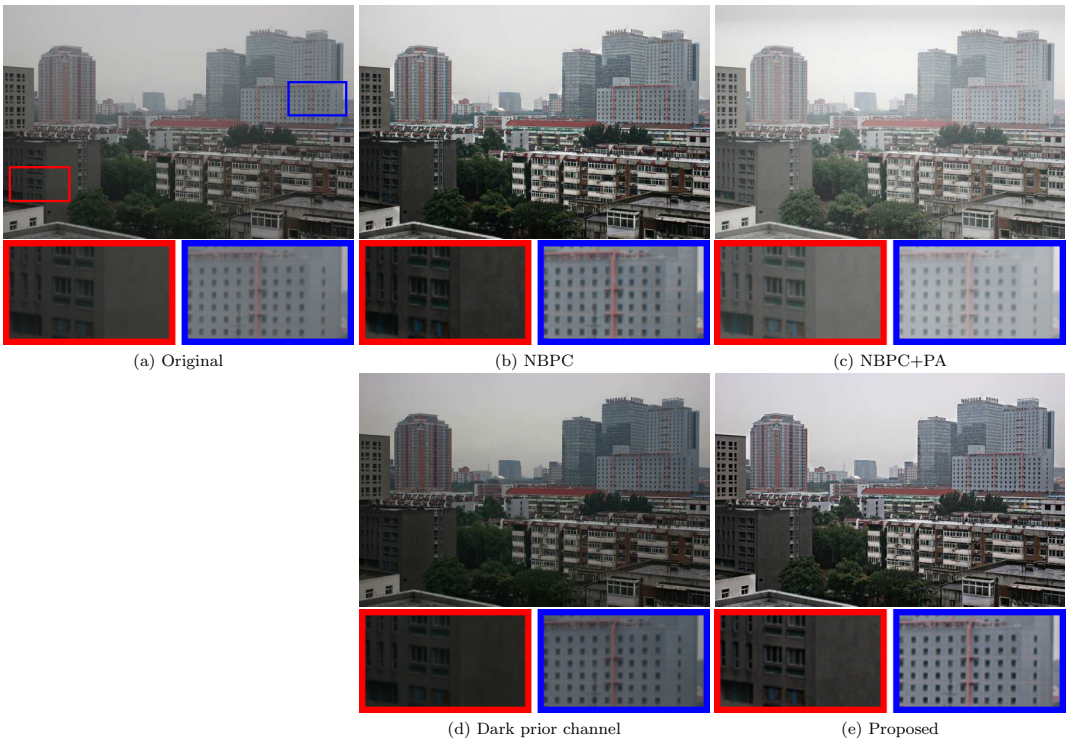
FIGURE 3.19: Single image de-hazing steps proposed in [97]. The black arrows refer to the original model and the red path includes the GAIF as the veil refinement step.



(a) Original (b) NBPC (c) NBPC+PA

(d) Dark prior channel (e) Proposed

FIGURE 3.20: A comparison of image haze removal on road image. GAIF is used to filter the atmospheric veil estimate in the no-black pixel constraint (NBPC) technique [97]. In comparison, dehazing using the original NBPC [97], NBPC+PA [98] and dark prior channel [71] techniques are evaluated.



(a) Original (b) NBPC (c) NBPC+PA

(d) Dark prior channel (e) Proposed

FIGURE 3.21: A comparison of image haze removal on city image. GAIF is used to filter the atmospheric veil estimate in the no-black pixel constraint (NBPC) technique [97]. In comparison, dehazing using the original NBPC [97], NBPC+PA [98] and dark prior channel [71] techniques are evaluated.

3.5.2 Flash/No-flash fusion denoising

Denoising an image taken without flash by utilizing another version of the same image with flash is a common digital photography problem [78]. In Figure 3.23, comparison between three representative filters is provided namely, guided filter (GIF) [49], joint bilateral filter (JBF) [78] and semi-guided bilateral filter (SGBF) [99]. GAIF is used in this application as a replacement to the bilateral filter in the scheme proposed in [78]. More specifically, GAIF is used to filter both images the flash and no-flash where I is the image to be filtered and $M = H * I$ where H is a linear average filter as shown in Figure 3.22. In Figure 3.23, not only did GAIF denoise the no-flash image, but it also filled the dark regions between the jugs with details from the flash image producing a significantly better result than JBF and GIF. However, comparable results to SGBF, but slightly sharper, especially the drawings on the jugs.

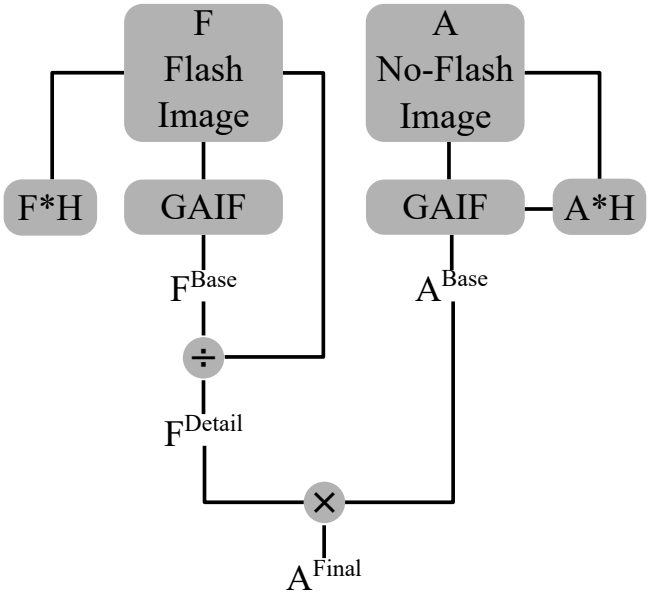
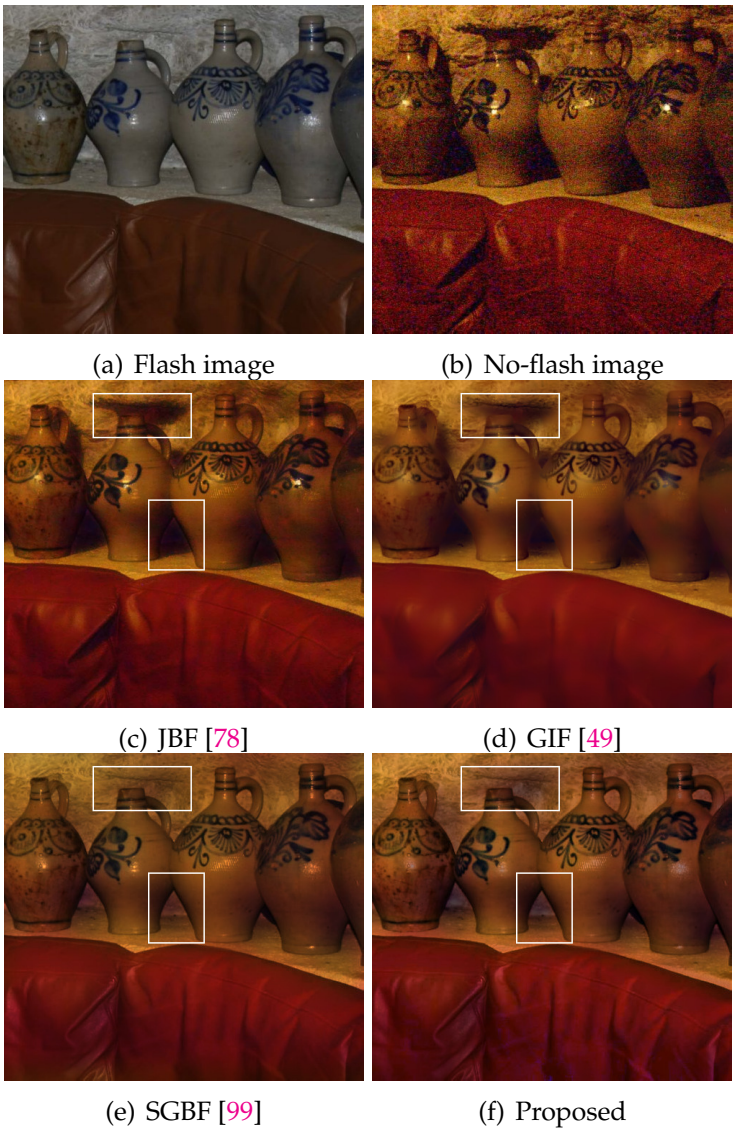


FIGURE 3.22: Flash/No flash denoising algorithm.



(a) Flash image (b) No-flash image

(c) JBF [78] (d) GIF [49]

(e) SGBF [99] (f) Proposed

FIGURE 3.23: A comparison of image flash/no-flash denoising between GAIF with (F_{base} : M = boxfilter$(I, w = 50), \Omega_k = 3, \epsilon = 1, A_{NR}$: M = boxfilter$(I, w = 50), \Omega_k = 5, \epsilon = 20$), the joint bilateral filter [78], guided image filter [49] with ($r = 9, \epsilon = 0.0004$) and Semi-guided filter [99] with (F_{base} : $\sigma_s = 8.5, \sigma_r = 0.5, N = 5, A_{NR}$: $\sigma_s = 8.5, \sigma_r = 0.35, N = 5$). The proposed filter is superior to the first two and is on par with (e) but at less computational complexity. In (f) GAIF manages to capture all the important details as annotated.

(a) Flash image (b) No-flash image

(c) JBF [78] (d) GIF [49]

(e) SGBF [99] (f) Proposed

FIGURE 3.24: A comparison of image flash/no-flash denoising between GAIF with ($F_{base} : M = \text{boxfilter}(I, w = 50), \Omega_k = 3, \epsilon = 1, A_{NR} : M = \text{boxfilter}(I, w = 50), \Omega_k = 5, \epsilon = 20$), the joint bilateral filter [78], guided image filter [49] with ($r = 9, \epsilon = 0.0004$) and Semi-guided filter [99] with ($F_{base} : \sigma_s = 8.5, \sigma_r = 0.5, N = 5, A_{NR} : \sigma_s = 8.5, \sigma_r = 0.35, N = 5$). The proposed filter is superior to the first two and is on par with (e) but at less computational complexity. In (f) GAIF manages to capture all the important details as annotated.

3.5.3 Image detail enhancement

Enhancing the details of an image starts with decomposing the image into base and details layers followed by details amplification. To avoid halo artefacts, edge-aware filters are used to produce the base layer as follows:

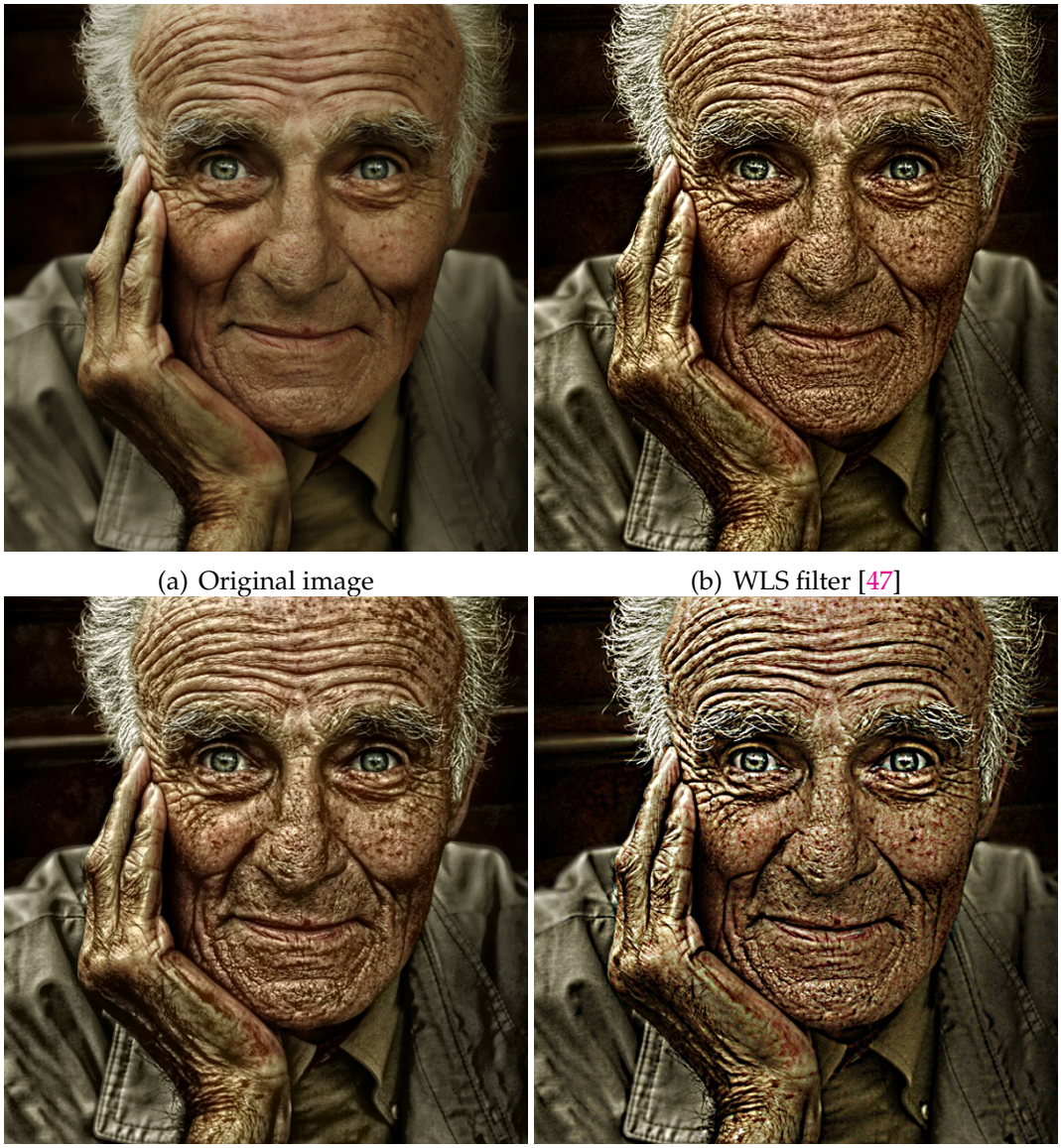
$$J = \text{GAIF}(I, M) + \gamma(I - \text{GAIF}(I, M)) \tag{3.22}$$

where γ is a magnification factor used to amplify the details, and M is median filtered version of I. Figures 3.25 and 3.26 compare the performance of GAIF to two representative techniques; weighted least-squares (WLS) [47] and semi-guided filter [99]. It can be observed that the proposed filter preserves the original edges intact, in other words, the filtered image has the edges of the input image as can be seen in Figure 3.25.

(a) Original image (b) WLS filter [47]

(c) Semi-GF [99] (d) Proposed

FIGURE 3.25: A comparison of Image details enhancement performance between GAIF ($M = \text{MEDFILT}(I, w = 13), \Omega_k = 3, \epsilon = 0.1$), weighted least-squares filter [47] ($\lambda = 0.125, \alpha = 1.2$) and the semi-guided filter [99] ($\sigma_s = 3.5, \sigma_r = 0.05, N = 5$). As annotated, the proposed filter excels at preserving the true edges of the input image while achieving comparable smoothing performance in other regions.



(a) Original image (b) WLS filter [47]

(c) Semi-GF [99] (d) Proposed

FIGURE 3.26: A comparison of Image details enhancement performance between GAIF ($M = \text{MEDFILT}(I, w = 13), \Omega_k = 3, \epsilon = 0.00001$), weighted least-squares filter [47] ($\lambda = 0.125, \alpha = 1.12$) and the semi-guided filter [99] ($\sigma_s = 3.5, \sigma_r = 0.05, N = 5$).

3.5.4 Edge detection

Another application of edge-aware filters is edge-detection. Images are pre-processed using edge-aware filtering followed by an edge-detection algorithm. To demonstrate the potential of GAIF at improving edge-detection, in Figure 3.27, an experiment on a synthetically generated image with known

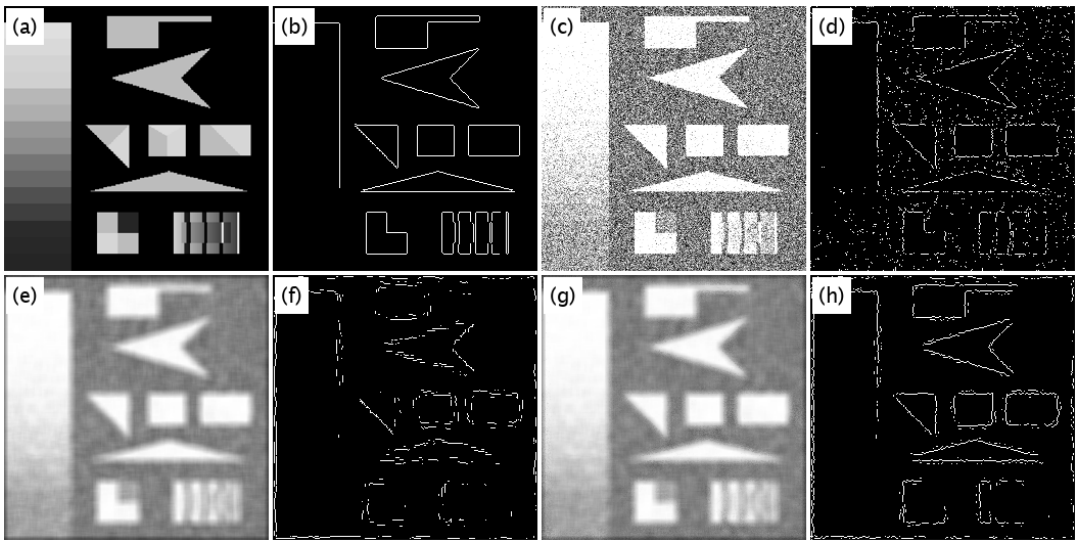


FIGURE 3.27: Edge detection by preprocessing image using GAIF. (a) is input image, (b) is the edge map of the input image, (c) is the input image plus a Gaussian noise with $\mu = 0.5$ and $\sigma^2 = 0.05$, (d) is the edge map of the image in (c), (e) is average filtered version of the image in (c) with window size 9, (f) edge map of the image in (e), (g) is GAIF filtered image with I is the image in (c) and M is the image in (e), finally (h) is the edge map of the image in (g).

edges is conducted, followed by adding noise to it. To detect the edges of the synthetically generated noisy image, it is initially preprocessed with GAIF followed by edge-detection. The ***edge*** command in MATLAB is used for edge detection which is an implementation of the Canny edge-detection algorithm [100].

3.6 Chapter summary

In this chapter, a novel filtering technique is presented with a number of applications in image processing and computer vision. GAIF achieves edge-preservation by interpolating between two patches. As a result, the filter can improve the results of linear and non-linear filters.

GAIF is a computationally efficient edge-preserving filter with a computational complexity of $O(N)$ where N is the number of pixels in the image. The efficiency of GAIF is demonstrated on a number of problems including single image haze-removal, flash/no-flash image fusion, image detail enhancement and edge detection.

Finally, it is important to note that GAIF models the image as an interpolation at the patch level between the input image and a smoothed version of which. This means that; the filter is constructed with two images, I and M, produced using the same type of sensor in mind. Hence, it is not straight forward to utilize the filter for tasks that require cross-domain fusion such as matting/feathering [101] and super-resolution.

Chapter 4

High-pass filter generalization of the TV model

4.1 Introduction

Denoising is one of the centre problems in signal processing, especially for non-stationary signals. The TV model [12] aims at reducing the ℓ_1 norm of the high-frequency component of the signal which is measured by the first derivative of the signal. The TV model has received a great deal of attention by researchers and was utilized to solve inverse problems such as denoising, deconvolution, inpainting as well as other general signal and image processing tasks: zooming, mean curvature motion of interfaces, segmentation and texture extraction [102] and references therein. A limitation of the TV model that was realized early on is what's known as staircase effect [103]. In other words, the total variation model prefers piece-wise constant solutions, which represents a very limited type of signals. Many attempts have been dedicated to tackle the staircasing problem [104] and to generalize the filter to other types of signals [105]

The model for observed signal y is as follows:

$$y = x + n, \tag{4.1}$$

where x is the desired signal and n is the zero-mean Gaussian white noise with variance σ_n. In the discrete case, the TV model for recovering signal x in (4.1) is expressed as following optimization problem:

$$x^* = \underset{x}{\operatorname{argmin}} \frac{1}{2}\|y - x\|_2^2 + \lambda \|Dx\|_1, \tag{4.2}$$

where D is a banded matrix with the elements [1, -1] along the main diagonal and the superdiagonal respectively. An example of the difference operator D is defined as follows:

$$D = \begin{bmatrix} 1 & -1 & & \\ & \ddots & & \\ & & 1 & -1 \end{bmatrix} \tag{4.3}$$

The motivation of this chapter is to explore new ways to solve the staircasing problem and to extend the model such that it can deal with wider classes of signals. A new generalization of the TV model is proposed. The main idea is that the difference operator used in the original TV model can be regarded as the simplest half band high-pass filter with fixed amplitude response. A natural generalization is to replace it with a high-pass filter with user controllable bandwidth and amplitude response.

4.2 The high-pass filter generalization

The regularization term $\|Dx\|_1$ is the ℓ_1 norm of high-pass filtered version of x, where the finite difference operator is a half-band high-pass filter, this is clear in Figure 4.1. By interpreting the operator from this point of view, a natural question is: why not use other high-pass filters? In other word, can better denoising performance be attained by minimizing the ℓ_1 norm of a different band other than the highest 50% e.g., the highest $10-90\%$? To answer

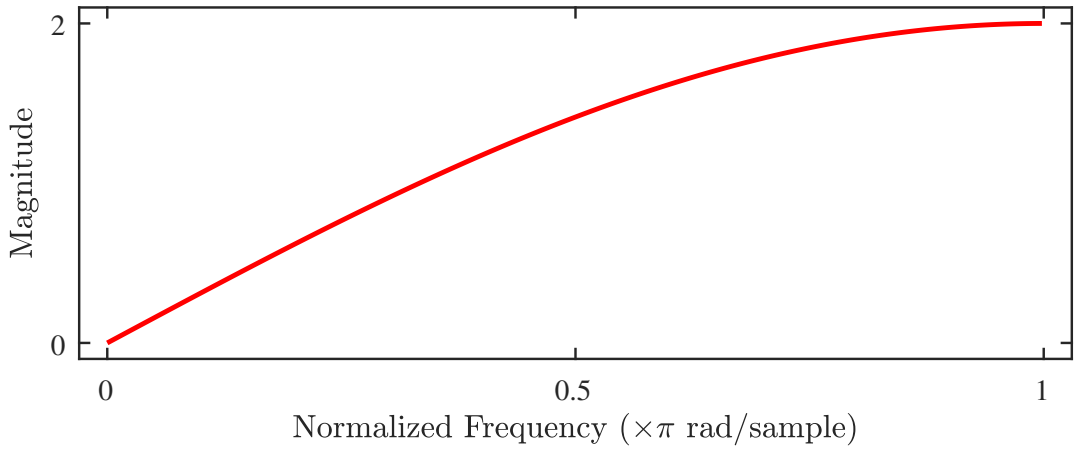


FIGURE 4.1: Amplitude response of the filter [1 -1]

this question, a generalized TV model of the following form is considered:

$$x^* = \underset{x}{\operatorname{argmin}} \frac{1}{2}\|y - x\|_2^2 + \lambda\|Hx\|_1, \tag{4.4}$$

where H is a banded matrix. Its rows are coefficients of a designed FIR high-pass filter. The original TV model is a special case when $H = D$. The FIR high-pass filters used in this work are described in next section. To solve this problem using the ADMM algorithm [17]. Model (4.4) is reformulated as a constrained optimization problem:

$$x^* = \underset{x}{\operatorname{argmin}} \quad \tfrac{1}{2}\|y - x\|_2^2 + \lambda\|z\|_1 \tag{4.5}$$

$$s.t \qquad\qquad Hx = z. \tag{4.6}$$

The augmented Lagrangian can be written as:

$$x^* = \underset{x}{\operatorname{argmin}} \frac{1}{2}\|y - x\|_2^2 + \lambda\|z\|_1 + \frac{\rho}{2}\|Hx - z + u^k\|_2^2, \tag{4.7}$$

which can be solved as follows:

$$x^{k+1} = \underset{x}{\operatorname{argmin}} \left(\frac{1}{2}\|y - x\|_2^2 + \frac{\rho}{2}\|Hx - z^k + u^k\|_2^2\right), \tag{4.8}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} \left(\lambda\|z\|_1 + \frac{\rho}{2}\|Hx^{k+1} - z + u^k\|_2^2\right), \tag{4.9}$$

$$u^{k+1} = u^k + Hx^{k+1} - z^{k+1}. \tag{4.10}$$

The solutions for equations (4.8) to (4.10) are:

$$x^{k+1} = (I + \rho H^T H)^{-1}(y - \rho H^T(z^k - u^k)), \tag{4.11}$$

$$z^{k+1} = \mathcal{S}_{\lambda/\rho}(Hx^{k+1} + u^k), \text{ where } \mathcal{S}_k(a) := a(1 - k/|a|)_+, \tag{4.12}$$

$$u^{k+1} = u^k + Hx^{k+1} - z^{k+1}. \tag{4.13}$$

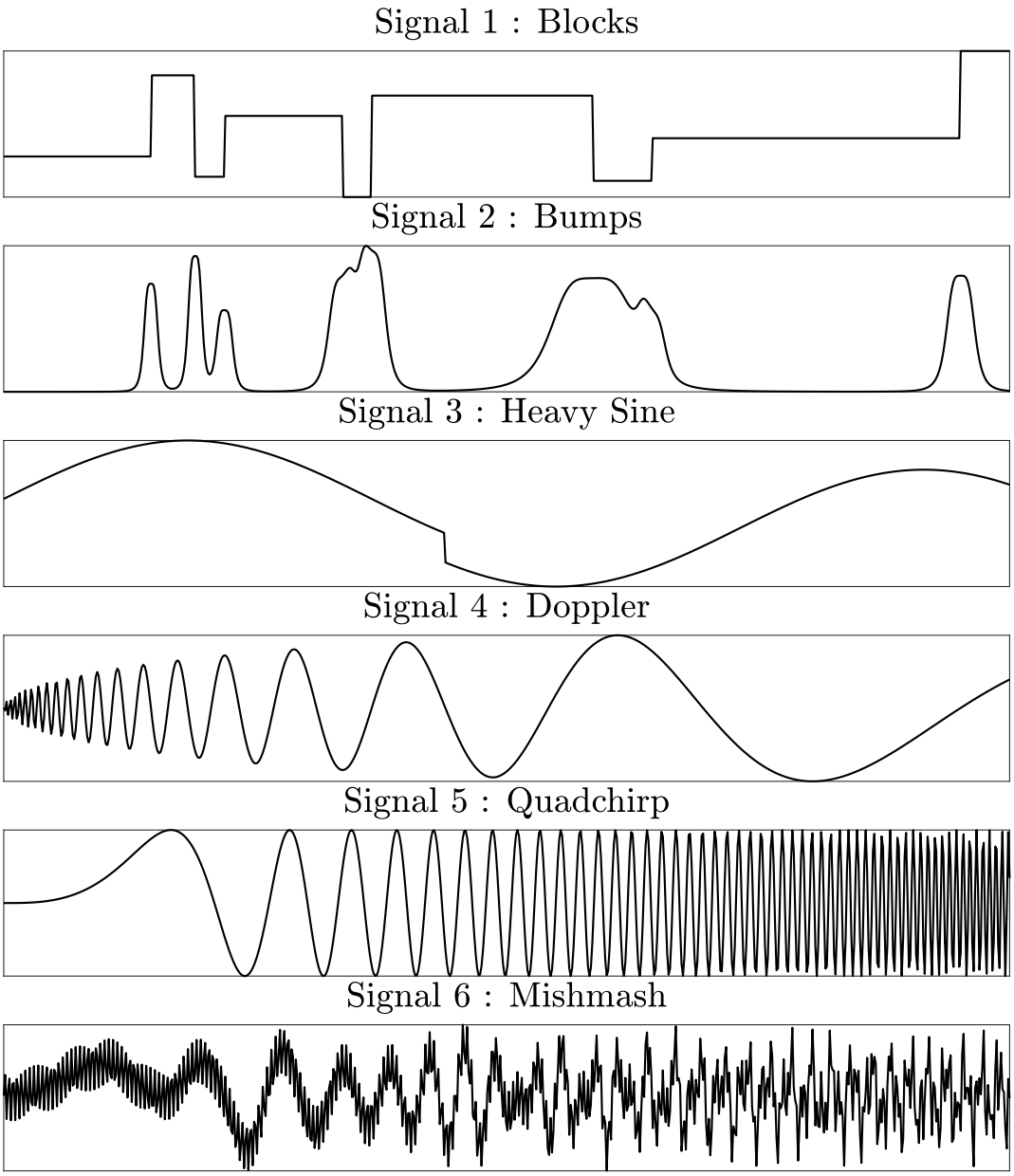


FIGURE 4.2: Six types of non-stationary test signals.

4.3 Results

To demonstrate the efficiency of the proposed model, the algorithm was run with a number of different configurations. Each configuration was solved 50 times by running the iterative equations in (4.11) to (4.13). Each of the 50 runs was done on a new instance of random noise n in (4.1). The reported result is the average mean absolute error (MAE) of 50 runs to come up with a good estimate of the true mean of MAE. The parameter ρ in Section 4.2 is fixed to 1 in all experiments. To get a better idea about the denoising capability

of the proposed model, its performance is compared to that of the original model (4.1) as well as wavelet denoising using a range of wavelets. Settings of configurations are summarized as follows:

- High pass filters H: linear phase symmetric (type I) was considered, with cut-off frequencies f_c from the interval $(0.1, 0.9)$ with orders N from the range $\{4, 8, 12, 16, 20, 24\}$. In addition to the symmetric filters, the high-pass Gaussian filters which are generated from low-pass Gaussian filter frequency response $H_{high}(\omega) = 1 - H_{low}(\omega)$ were considered. The bandwidth of $H_{low}(\omega)$ is controlled by the parameter σ of the Gaussian function. In the experiments, σ is from the list $\{1, 2, 3\}$ with the respective length of the filter being N in the list $\{4, 8, 12\}$.

- Signal classes x: for each configuration, filtered noisy versions of the 6 typical types of signals were considered as shown in Figure 4.2 which first appeared in [106] and are widely used as test signals to compare performance of signal denoising algorithms. These signals are generated using MATLAB's ***wnoise*** command with signal-to-noise ratio (SNR) from the list $\{1, 2, 5, 10\}$.

- Tuning parameter λ: a range of $\lambda's$ between $(1, 10)$ have been tested.

- For wavelet denoising, empirical Bayes-based method was used [107] accessible in MATLAB via the ***wdenoise*** command and experiments using all the wavelets from the list {db1, db3, db5, sym2, sym3, sym5, bior1.1, bior2.2, bior2.8} with 9 levels of decomposition were conducted.

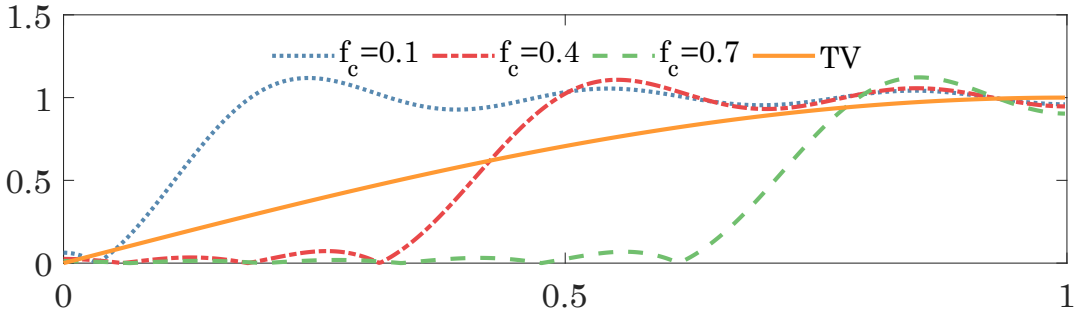


FIGURE 4.3: Amplitude response of a sample of filters $H(\omega)$ used in this work of order $N = 12$.

In Table 4.1, the reported results represent the minimum value of MAE found by solving (4.4) using different values of the tuning parameter λ for

a particular signal and SNR. The GTV column reports the best performance achieved using a filter H among different filters. Name of filter H is reported below MAE either as 1) H_{N,f_c}, where N is the filter order and f_c is the cut-off frequency, or 2) G_σ, where G: a high-pass of a Gaussian filter, σ is the Gaussian function width. The TV column reports the minimum MAE achieved using TV and the Wavelet column reports the minimum MAE achieved among various wavelets. The name of the wavelet is reported below MAE. The observations can be summarized as follows:

- From Table 4.1, it can be observed that for the Blocks signal, the TV filter is the best performing filter for all SNR values. In other words, TV is the best prior for piece-wise constant class of signals.
- For Bumps signal, in the case of low signal-to-noise ratio, total-variations yields the best results. However, in high SNR case, the high-pass equivalent of Gaussian filter with $\sigma = 1$ can perform slightly better.
- For Heavy Sine signal, the proposed model always performs better than both TV and wavelet-based denoising at all SNR levels and this can be attributed to the mismatch between the signal family (mostly smooth) and the assumption of the TV model. Regularizers based on Gaussian filters, which have $\sigma = \{2, 3\}$ with cut-off frequencies $f_c = \{0.26, 0.17\}$ respectively, favour solutions with lower frequency contents by penalizing wider parts of the high side of the spectrum.
- For Doppler signal, the proposed model achieves results better than TV, but both fall behind the wavelet's performance.
- For Quadchirp signal, the proposed filter achieves better denoising performance than TV. By moving from the lower to higher SNR, the cut-off frequency of the penalty filter H moves from $(0.26 \rightarrow 0.5 \rightarrow 0.6)$.
- For Mishmash signal, the proposed filter achieves better denoising performance than TV at all SNR levels by penalizing using a filter with cut-off frequency $f_c = 0.8$.

To demonstrate that the proposed model (4.4) doesn't suffer from the staircasing phenomenon, a comparison of the denoising capacity is conducted between the proposed model and TV in Figure 4.4, the bottom right part of which, is achieved by solving equation (4.4) with $\lambda = 5$ and H is a Gaussian high-pass filter with $\sigma = 3$ which has a cut-off frequency $f_c \approx 0.17$

TABLE 4.1: MAE of the six signals at various SNR levels.

	SNR = 1			SNR = 2		
Signal	**GTV**	**TV**	**Wavelet**	**GTV**	**TV**	**Wavelet**
Blocks	0.235	**0.174**	0.217	0.289	**0.179**	0.219
	G_2		db1	G_2		db1
Bumps	0.215	**0.214**	0.251	0.257	**0.253**	0.286
	$H_{4,\,0.2}$		db5	G_1		db5
Heavy Sine	0.130	0.131	**0.128**	**0.142**	0.175	0.152
	G_3		db3	G_3		db3
Doppler	0.205	0.227	**0.188**	0.251	0.292	**0.214**
	G_3		sym5	G_2		sym5
Quadchirp	**0.597**	0.611	0.643	**0.686**	0.710	0.915
	G_2		bior2.8	G_1		bior2.8
Mishmash	**0.596**	0.633	0.697	**0.708**	0.801	1.297
	$H_{12,\,0.8}$		bior2.8	$H_{8,\,0.8}$		bio2.8
	SNR = 5			**SNR = 10**		
Signal	**GTV**	**TV**	**Wavelet**	**GTV**	**TV**	**Wavelet**
Blocks	0.336	**0.190**	0.198	0.349	0.208	**0.189**
	G_2		db1	G_2		db1
Bumps	**0.295**	0.309	0.317	**0.332**	0.354	0.352
	G_1		sym5	G_1		sym5
Heavy Sine	**0.164**	0.240	0.190	**0.190**	0.300	0.197
	G_3		sym5	G_2		sym5
Doppler	0.318	0.395	**0.255**	0.368	0.476	**0.276**
	G_1		db5	G_1		sym5
Quadchirp	**0.723**	0.789	0.926	**0.734**	0.828	0.919
	$H_{4,\,0.6}$		sym5	$H_{24,\,0.6}$		sym5
Mishmash	**0.741**	0.956	3.199	**0.753**	1.005	6.425
	$H_{8,\,0.8}$		bior2.8	$H_{8,\,0.8}$		bior2.8

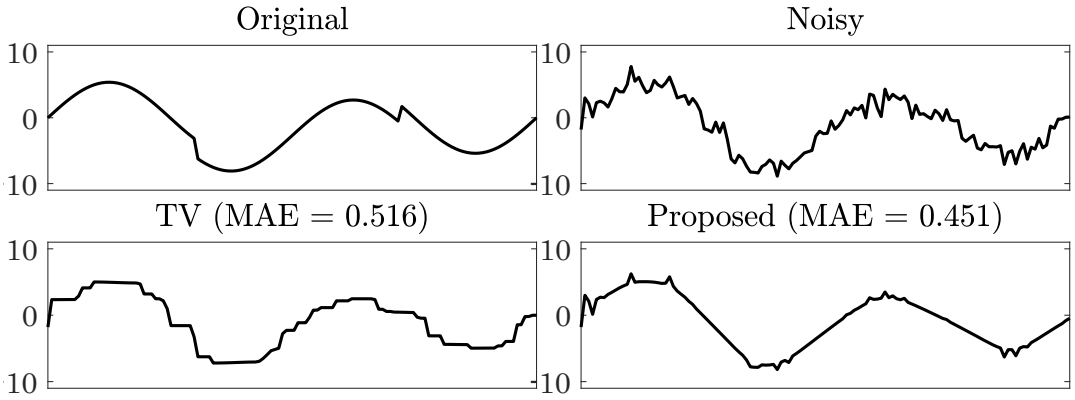


FIGURE 4.4: Comparison between filtering a noisy signal using TV (4.2) vs using the proposed model (4.4); the proposed model doesn't suffer from the staircasing phenomenon.

4.4 Chapter summary

In this chapter, one of the main drawbacks of the total variation model was addressed, namely the staircasing phenomenon by using a simple generalization of the model. It was demonstrated that based on the signal characteristics, better denoising performance is achievable by interpreting the finite difference operator D as a half-band high-pass filter, which made it possible to replace it with another banded matrix H of a high-pass filter with prescribed bandwidth. Extensive simulations on six types of test signals and comparison with the original TV model and wavelet transform based denoising have been conducted to evaluate the performance of the proposed generalization.

Chapter 5

Discrete Laplacian operator and its applications

5.1 Introduction

Two main operations and building blocks in many engineering disciplines in general [108, 109], image processing and computer vision tasks in particular [110], are first and second order derivatives, otherwise known as the *Gradient* and *Laplacian*. The ubiquity of these operators comes from the way systems and problems are modelled. Mathematical models relying on the language of calculus are at the centre. Some of these modelling paradigms include variational methods, partial differential equations (PDE), statistical and linear/non-linear optimization models.

Over the years, many attempts have been dedicated towards the generalization of those operators to different settings. Among the generalizations is the graph Laplacian [20] which found numerous applications in signal and image processing [111].

Of relevance to this work, is the generalization of integer order differential and integral operators to fractional orders. *Fractional Calculus* (FC) is a 300 years old concept dating back to the days of l'Hôpital and Leibniz [112]. FC has received increased interests over the last 30 years mainly due to their long memory property [109]. For a more recent historical survey, the reader is referred to [113].

Fractional-order derivatives have found numerous applications in electronic circuits and control systems [108, 114], signal processing [115], image processing, computer vision and pattern recognition [116, 117], biological systems and economics [118].

5.1.1 Fractional derivatives

The integer differential and integral operators are defined uniquely, and they are local. In other words, they consider the values of very close neighbouring points to the point of interest. On the other hand, fractional-order differential operators are non-local; larger neighbourhoods are considered in the computation resulting in long-term memory effect. This is one of the main reasons behind their appeal. There is a multitude of definitions of the fractional order derivatives [119, 120]. The notation D^ν is adopted to denote the derivative of a fractional order ν. The following is a list of the most common definitions:

- Forward Grunwald-Letnikov (GL)

$$D^\nu f(z) = e^{-j\theta\nu} \lim_{|h|\to 0} \frac{\sum_{k=0}^{\infty} (-1)^k \binom{\nu}{k} f(z-kh)}{|h|^\nu} \tag{5.1}$$

  where $\binom{\nu}{k}$ is the binomial coefficient and $h = |h|e^{j\theta}$ is a complex number with $\theta \in (-\pi, \pi]$.

- Riemann-Liouville (RL) fractional derivatives

$$D^\nu f(t) = \frac{1}{\Gamma(n-\nu)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\nu-n+1}} d\tau \tag{5.2}$$

  where $n-1 < \nu < n$ and $\Gamma(.)$ is Euler's Gamma function.

- Fourier domain fractional derivatives

$$D^\nu f(t) = \mathcal{F}^{-1}\left[(j\omega)^\nu \mathcal{F}\{f\}(\omega)\right], \quad \operatorname{Re}\nu > 0 \tag{5.3}$$

  where ω is the Fourier variable, $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier and the inverse Fourier transform respectively.

Most of the fractional derivative formulations start from a continuous formula which requires discretization to facilitate implementation.

5.1.2 Fractional Laplacian (FL)

The idea of extending the standard Laplacian operator to fractional order is an old idea (c.f [121] and references therein). The standard Laplacian is a local

operator. Local operators utilize the immediate neighbourhood only in the calculation where outliers can have big impact on the result. Many attempts have been made to create non-local operators that can alleviate the shortcomings of local operators in image processing applications [122]. Utilizing fractional-order Laplacian is such an attempt.

In a similar vein to the fractional derivatives, a multitude of fractional Laplacian (FL) definitions have been proposed over the past few decades but, a consensus on the most appropriate definition for an application is yet to be reached [123].

Of special interest to our work are two variants: the spectral fractional Laplacian [123] which is defined as follows:

$$-(-\Delta)^{\alpha/2} f(x) := \sum_{i \in \mathbb{N}} f_i \lambda_i^{\alpha/2} \phi_i(x) \tag{5.4}$$

where Δ is the continuous Laplacian operator applied on the function $f(x)$. ϕ_i and λ_i are the eigenfunctions and eigenvalues of the continuous Laplacian Δ respectively.

Secondly, the Fourier-transform based definition (pseudo-differential) [123]

$$\mathcal{F}\left\{(-\Delta)^{\alpha/2} f\right\}(\omega) = |\omega|^{\alpha} \mathcal{F}\{f\}(\omega) \tag{5.5}$$

It is important to note that both definitions are continuous, and a discretization step is required to make them applicable to digital data. For a comprehensive list of the different formulations, the reader is referred to [118, 121, 123] and references therein.

5.1.3 Related works

Fractional derivative operators in image processing

Fractional-order derivatives have found numerous applications in image processing. These applications can be categorized into three main categories based on the framework under which the derivative is used. The first category is *linear filtering*. In this category, a fractional-order derivative of an image is calculated through a linear filtering process. In some applications, the gradient image is the goal such as in edge detection [124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137] which is the earliest image processing application of fractional calculus. Another application is to use the

gradient image to enhance the input image through, for example an un-sharp masking scheme [138, 139, 140, 141, 132, 142, 143]. The third application in this category is contrast enhancement [144, 135, 145, 146].

The second category is referred to as *PDE-based* models. Modelling of image restoration problems using integer derivatives dates back to the late 80s and early 90s but utilizing fractional-order derivatives was first presented in 2007 [147] for the image denoising application and was later refined, improved and adopted for a number of applications. Among these applications are image denoising [148, 149, 150, 151, 152, 153, 154, 155], contrast enhancement [156, 157], image deblurring [158] and image super-resolution [159, 160].

The third category is related to the second and it is here referred to as *variational* models for image restoration problems. Similar to PDE(s), variational models with integer derivatives in the context of image processing date back to the 80s, but the introduction of fractional-order variational models is recent. Applications in this category include image denoising [161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172], in-painting [164, 173], fusion [162, 170], non-rigid registration [174], super-resolution [162] and optical flow estimation [175].

For the sake of completeness, a fourth category is covered that is based on a global optimization technique namely: fractional-order Darwinian particle swarm optimization (FODPSO). The reason for distinguishing this category is that the fractional-order derivative is not applied to the image but to a parameter of the model at hand such as the optimal threshold. The main application in this category is image segmentation [176, 177, 178].

Fractional Laplacian operators in image processing

The fractional Laplacian has not seen the same amount of adoption as is the case with the fractional derivative in signal and image processing. In [179], the authors introduced a scale-space model. In [180], the authors proposed a quadratic optimization model for blind image deconvolution involving the fractional Laplacian. In [181], the authors proposed a PDE model for vector field estimation flow-sensitive MRI imaging. In [182], the authors have demonstrated image denoising by solving a fractional diffusion equation (PDE). In [183], the authors proposed a variational model for image denoising based on the fractional Laplacian and later was extended to tomographic

reconstruction involving the fractional Laplacian as a regularizer [184].

5.1.4 Contributions

The contributions in this work can be summarized in the following:

I. Motivated by the construction of the spectral Laplacian (5.4), a discrete fractional Laplacian is proposed in the form of a matrix operator using the DCT transform. The discrete construction avoids the need for discretization which is typically required by the other fractional Laplacian construction techniques. The discretization step in the case of the pseudo-differential formulation in (5.5) usually involves solving a filter design problem [185, 186, 187].

II. Motivated by the reliance of the trend filter on the Laplacian operator, a computationally efficient implementation of traditional trend filtering is developed in the DCT transform domain.

III. Utilizing the proposed discrete fractional Laplacian, both the traditional and the ℓ_1 trend filters are extended to fractional order and their effectiveness at the image denoising task for higher levels of noise is demonstrated.

IV. Finally, applications of the proposed fractional Laplacian on a number of image processing tasks are demonstrated. However, it is important to stress that state-of-the-art performance in any of these applications is not claimed. The goal is to demonstrate the potential of the proposed operator.

The remainder of this chapter is organized as follows. Section 5.2 introduces the fractional Laplacian in the 1D and 2D cases. In Section 5.3, two groups of applications for the proposed operator are provided. In the first group namely: fast trend filters, trend filtering is first introduced followed by a DCT-based implementation of the ℓ_2 trend filter. The trend filtering is then generalized by proposing a fractional ℓ_1 and a fractional ℓ_2 trend filters. In the second group, a number of image processing applications are presented demonstrating the effectiveness of the proposed fractional Laplacian. Lastly, a discussion and conclusions are presented in Section 5.4.

5.2 A Discrete fractional Laplacian operator

5.2.1 Definition for 1D case

The Laplacian operator of single variable continuous function $f(x)$ is defined as follows:

$$\Delta f(x) = \frac{\partial^2 f(x)}{\partial x^2} \tag{5.6}$$

To implement the Laplacian operator, an approximate discretization is used as follows:

$$L\{f(x)\} = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} \tag{5.7}$$

where L is the discrete Laplacian operator and h is a small constant. This discretization amounts to uniformly sampling the function $f(x)$ with distance h. This operation can be implemented as a convolution as follows:

$$L\{f(x)\} = l * f \tag{5.8}$$

where $l = [1, -2, 1]$ is an FIR filter. The Laplacian in (5.8) can be formulated as a matrix operator in few different ways based on the boundary condition assumed. In this work a specific formulation with desirable properties is chosen as will be demonstrated later. Specifically, the Laplacian operator $L \in \mathbb{R}^{N \times N}$ is reformulated as follows:

$$L = \begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & & \cdot & \cdot & \cdot & \\ & & & -1 & 2 & -1 & \\ & & & & 1 & -1 & \end{bmatrix} \tag{5.9}$$

where, the rows between $[1, N-2]$ [1] are shifts of the FIR filter discussed earlier. The signals of interest are finite, and as can be seen in (5.7), the Laplacian filter is centred around x. In other words, to perform the convolution in (5.8), $f(x)$ needs to exist before and after x which, is not true at the boundaries. As a solution, the signal is extended on both sides [188].

Thus, the first and last rows in (5.9) are different from the middle rows. This signal extension near the boundary can be performed in several ways [189].

[1] Numbering starts from 0

In (5.9), a symmetric signal extension is assumed as will be discussed next.

The operator in (5.9) has the same first and last rows, this means equal boundary condition on both sides. For a signal $u \in \mathbb{R}^{N\times 1}$, the boundary condition in (5.9) assumes the signal u has zero slope at the boundary ($\dot{u}(0) = 0$, $\dot{u}(N-1) = 0$) and is symmetrically extended around the midpoint ($u(-1) = u(0), u(N) = u(N-1)$). For more details about the boundary conditions, the reader is referred to [189].

It is important to note that this particular operator in (5.9) is diagonalizable by the DCT type-II matrix [189] denoted by M

$$L = M^T E M \tag{5.10}$$

where E is a diagonal matrix of which the diagonal elements denoted $\{e(k)\}$ are the eigenvalues of L. This spectral decomposition is real which results in efficient computation.

Motivated by the spectral Laplacian in (5.4) and the diagonalization property of the discrete Laplacian in (5.10), the fractional Laplacian L_α [2] is defined as follows

$$L_\alpha = M^T E^\alpha M \tag{5.11}$$

where E^α is diagonal matrix with the diagonal elements raised to the power α.

To consolidate, we started from the discrete Laplacian filter $l = [1, -2, 1]$ which we used to formulate the matrix L in (5.9). Then we performed the eigendecomposition in (5.10) and we raised the diagonal eigenvalues matrix E to the power α to finally get the fractional Laplacian matrix L_α in (5.11).

Recall that when $\alpha = 1$ in (5.11) the standard Laplacian in (5.9) is recovered but, for α in the open set $\alpha \in (0, 1)$ different fractional Laplacian filters are constructed. The fractional Laplacians constructed are still diagonalizable by the DCT matrix as in (5.11) and share the same structure as the standard Laplacian in (5.9) with the difference that smaller values of α result in less sparse operator L_α. In other words, smaller α results in Laplacians with more non-zero coefficients hence the memory effect.

As a result of the increase in the number of coefficients, two things become clear: firstly, the dimensions of the fractional Laplacian need to be larger for smaller α to reduce the numerical inaccuracy due to truncation. Secondly, the

[2] The subscript notation is adopted for α to avoid the confusion with the exponent as L is a matrix operator

number of rows in L_α that are involved in the boundary condition increases. One way to recover a good filter l_α from the matrix L_α is to choose the middle row.

It is important to note that the eigenvalues of $L_\alpha \in \mathbb{R}^{N\times N}$ can be obtained by the relation $e(k) = \left[2 - 2\cos\left(k\frac{\pi}{N}\right)\right]$ [190]. Thus, the eigenvalues of the fractional-order Laplacian L_α are defined as follows:

$$e(k;\alpha) = \left[2 - 2\cos\left(k\frac{\pi}{N}\right)\right]^{\alpha} \qquad k = 0, 1, \ldots, N-1. \tag{5.12}$$

and the coefficients of the eigenvectors matrix are the same as DCT-II matrix which are as follows:

$$M(i,k) = \sqrt{\frac{2}{N}}\cos\left[\left(k+\frac{1}{2}\right)\frac{i\pi}{N}\right], \quad k,i = 0, 1, \ldots, N-1 \tag{5.13}$$

To get the FIR filter l_α from the operator L_α, the $\frac{N-1}{2}$th row of L_α is chosen based on the earlier discussion about avoiding the boundary condition. It can be readily demonstrated that the coefficients of the l_α have the following form:

$$l_\alpha(k) = \sum_{i=0}^{N-1} e\left(i;\alpha\right) M\left((N-1)/2, i\right) M\left(k,i\right) \tag{5.14}$$

5.2.2 Extension to the 2D case

Generalization to the 2D case is straight-forward as the 2D-DCT is usually performed as 1D-DCT calculated once on the rows and once on the columns [189]. The benefits of this formulation of the fractional Laplacian in the DCT domain over the traditional Fourier transform formulation are twofold. First, the fractional Laplacian filter l_α is easily constructed allowing for flexibility in applications. Second, it is computationally more efficient as filtering in the DCT domain avoids the need for the image extension as required by the DFT [191].

5.2.3 Computational complexity analysis

From the earlier development, it can be noticed that the proposed fractional Laplacian has two equivalent forms. The first being a matrix operator $L_\alpha \in R^{N\times N}$ which is constructed using (5.11). To calculate the fractional Laplacian

of a signal using L_α, a matrix-vector multiplication is performed, which has $O(N^2)$ computational complexity.

The second form is a linear FIR filter $l_\alpha \in R^N$ which can be constructed directly using (5.14). Calculating the fractional Laplacian of a signal of length M using l_α is a convolution operation with a computational complexity of $O(NM)$.

5.2.4 Discussion

The discussion about the other formulations of the matrix operator L have been delayed for a reason that will become clear soon. The alternative to the proposed Laplacian is to formulate it as a circulant matrix, assuming that the signal has a periodic extension as follows [189]:

$$L = \begin{bmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & & \cdot & \cdot & \cdot \\ & & & -1 & 2 & -1 \\ -1 & & & & -1 & 2 \end{bmatrix} \tag{5.15}$$

The L matrix in this formulation is diagonalizable by the discrete Fourier matrix (DFT) [189] which could be generalized the same way done in (5.11). However, the proposed Laplacian is computationally more efficient. This computational efficiency comes from two aspects. Firstly, the DFT matrix is complex while, the DCT is real. Secondly, the boundary condition in (5.15) assumes that the signal is periodic, in other words, the signal needs to be padded with a copy of itself on both sides creating discontinuities along the boundaries. On the other hand, the Laplacian in (5.9) assumes that the signal is padded with a mirror-image of itself along the boundary. This is a natural boundary condition and it is what MATLAB uses in the ***imfilter*** command with the `symmetric` option.

In [192], the authors have proposed to compute the fractional derivatives of images implicitly by utilizing a discretization of the following result [193]:

$$D^\nu \cos(\omega t) = \omega^\nu \cos\left(\omega t + \frac{\nu\pi}{2}\right) \tag{5.16}$$

where ν is the fractional exponent and ω is the continuous frequency variable. In other words, in contrast to what is done in this work, the authors do not provide a direct method for constructing the fractional differential operator (D^{ν}) in [192]; rather a method to compute its effect on a function. Being able to construct the operators affords more flexibility and applicability in linear algebraic settings.

In [194], the authors propose to approximate the function (5.16) in the DCT domain by treating it as a FIR filter design problem. This technique produces coefficients of a FIR filter which approximates the ideal fractional derivative operator. The main difference between this technique and ours is that our technique does not use an approximation and it generalizes the Laplacian operator rather than the first order derivative.

5.3 Applications

The Laplacian operator is very ubiquitous in applications and generalizations of which can be examined on such applications. The discussion about the applications have been split into two groups. The first group deals with a specific trend estimation technique which relies on the standard Laplacian, it is briefly introduced, followed by a new implementation technique, and finally, a generalization is demonstrated. In the second group, five image processing applications are considered.

5.3.1 Fast trend filters in the DCT domain

Definition

Estimating the trend of a signal or time-series is a common problem in many disciplines [195]. More specifically, given a signal $b \in \mathbb{R}^{N\times 1}$, it is assumed that the signal is composed of two components: a slowly varying component u known as the trend and a rapidly varying white Gaussian noise component denoted ϵ as follows:

$$b = u + \epsilon \tag{5.17}$$

Trend estimation is the process of producing an estimate $\hat{u}$ for the underlying trend u. The literature on trend estimation is very rich with various parametric and non-parametric techniques [196]. In this chapter however,

the interest is in a specific trend estimation technique which will be referred to as *trend filter*.

Concretely, capital and small letters represent matrices and column vectors respectively. For example, the nth column vector of the matrix M is denoted M_n, while the nth element of a vector u is denoted u_n. The ℓ_p-norm of u is $\|u\|_p$. The trend filter [197, 195] is formally defined as follows:

$$\underset{u}{\operatorname{argmin}} \|u-b\|_2^2 + \lambda \|Lu\|_p^p \tag{5.18}$$

where λ is a regularization parameter controlling the emphasis placed on the regularizer $\|Lu\|_p^p$. A special case of the model (5.18) is the generalized Wiener filter [198, 199] when $p = 2$, which is otherwise known as the Hodrick-Prescott trend filter in the statistics community [197]. By utilizing the fact that the Laplacian matrix L is diagonalized by the DCT matrix (5.10), the cost function can equivalently be defined in the DCT domain:

$$\underset{u}{\operatorname{argmin}} \|u-b\|_2^2 + \lambda \|Lu\|_2^2 = \|\bar{u}-\bar{b}\|_2^2 + \lambda \|M^T E \bar{u}\|_2^2 \tag{5.19}$$

where $\bar{u} = Mu$ and $\bar{b} = Mb$. Minimizing the cost function yields

$$\bar{u}(k) = \frac{\bar{b}(k)}{1+\lambda e(k)^2} \tag{5.20}$$

This result shows that the trend filter can be very efficiently implemented in the DCT domain as an element-wise division since the eigenvalues $e(k)$ can be pre-calculated.

This result also allows us to interpret the trend filter as a low-pass filter in the DCT domain. To this end, let $G = M^T$, the trend filter can thus be represented as follows:

$$u = G\bar{u} \tag{5.21}$$

$$= \sum_{k=1}^{N} \bar{u}(k) G(k) \tag{5.22}$$

$$= \sum_{k=1}^{N} \frac{\bar{b}(k)}{1+\lambda e(k)^2} G(k) \tag{5.23}$$

On the other hand, the observed vector b can be represented as

$$b = G\bar{b} \tag{5.24}$$

$$= \sum_{k=1}^{N} \bar{b}(k)G(k) \tag{5.25}$$

Comparing (5.23) with (5.25), Upon brief inspection, an interpretation of the trend filter as a shrinkage operation in the DCT domain can be gleaned. The level of shrinkage is controlled by the regularization parameter λ. Since each column vector $G(k)$ can be regarded as a frequency component with $k = 0$ corresponding to the DC and $k = N - 1$ corresponding to the highest frequency, the shrinkage is also frequency dependent. It can be shown that the eigenvalues have the property $e(0) = 0$ and $e(i) < e(k)$ for $i < k$. As a result, a higher frequency component will be shrunk more. Similar to a linear low-pass filter which attenuates the high frequency components in the Fourier transform domain, the trend filter is a low-pass filter which attenuates the high frequency components in the DCT domain.

Fractional ℓ_2 trend filter

As an application of the proposed operator in (5.11), the trend filter (5.19) is generalized by replacing the L operator with L_α operator leading to a new model. The new model is called: *fractional trend filter* which is the minimization of the following cost function:

$$\|u - b\|_2^2 + \lambda \|L_\alpha u\|_2^2 \tag{5.26}$$

where λ, similar to (5.18), is the regularization parameter which controls the emphasis placed on the regularizer. The solution to this cost function, can be performed in DCT domain as follows:

$$\bar{u}(k) = \frac{1}{1 + \lambda e(k)^{2\alpha}} . \bar{b}(k) \tag{5.27}$$

Compared with (5.20), more control over the magnitude response is achieved by changing α. To demonstrate the impact of α on the shape of the filter, the filter function (5.27) is plotted at various values of α in Figure 5.1. To facilitate the comparison, the x-axis is normalized because, at each α a new Laplacian L_α is generated that has its diagonal elements $e(k) \in [0, 4^\alpha]$. It is important to

note that a similar model was proposed in [180] with the difference that the solution was done in the Fourier domain.

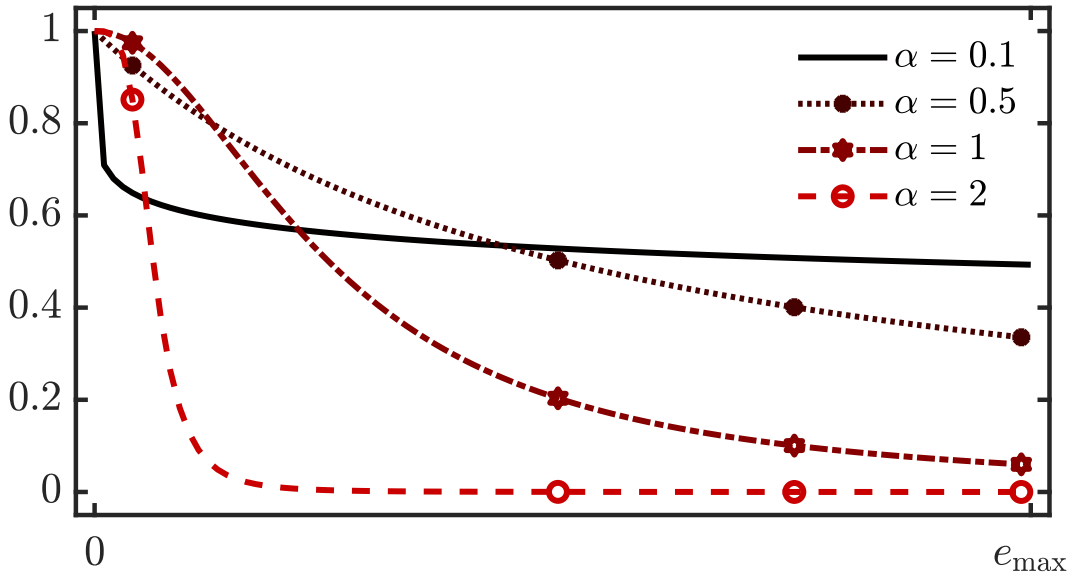


FIGURE 5.1: The shrinkage effect of the fractional ℓ_2 trend filter (first term in (5.27)) using various values of α.

From Figure 5.1, it is noticed that lower values of α allow more high frequencies to pass than the higher values of α.

Fractional ℓ_1 trend filtering

The ℓ_1 trend filtering [195] is extended to fractional ℓ_1 trend filtering of the form:

$$\underset{u}{\operatorname{argmin}} \|u - b\|_2^2 + \lambda \|L_\alpha u\|_1 \tag{5.28}$$

To solve this problem, the ADMM algorithm [17] is adopted as follows:

$$\underset{u,z}{\operatorname{argmin}} \|u - b\|_2^2 + \lambda \|z\|_1 + \frac{\rho}{2}\|z - L_\alpha u + v\|_2^2 \tag{5.29}$$

In Algorithm 2, l_α is one of the middle rows of L_α. It is a FIR filter. In 1D, l_α and its transpose l_α^T are convolved with signals. In 2D, two kernels using l_α are generated for the x and y directions as shown Figure 5.2. l_α^T is identical to l_α as they are symmetric around the centre. For images, an an-isotropic extension of the 1D model is used as follows

Algorithm 1: ADMM for Fractional ℓ_1 trend filter using matrix form

Result: u^{k+1}

$u^0 = b, z^0, v^0 = 0;$

while $||u^{k+1} - u^k||_2^2 / u^{k+1} > \epsilon$ **do**

$u^{k+1} = \left(I + \rho L_\alpha^T L_\alpha\right)^{-1} \left(b + \rho L_\alpha^T \left(z^k - u^k\right)\right)$

$z^{k+1} = S_{\lambda/\rho}\left(L_\alpha u^{k+1} + v^k\right)$ where $S_\kappa(a) = (a - \kappa)_+ - (-a - \kappa)_+$

$v^{k+1} = v^k + L_\alpha u^{k+1} - z^{k+1}$

end

Algorithm 2: ADMM for Fractional ℓ_1 trend filter using convolutions

Result: u^{k+1}

$u^0 = b, z^0, v^0 = 0;$

while $||u^{k+1} - u^k||_2^2 / u^{k+1} > \epsilon$ **do**

$u^{k+1} = b - \rho\left(l_\alpha^T * (z^k - u^k) - l_\alpha^T * l_\alpha * u^k\right)$

$z^{k+1} = S_{\lambda/\rho}\left(l_\alpha * u^{k+1} + v^k\right)$

$v^{k+1} = v^k + l_\alpha * u^{k+1} - z^{k+1}$

end

$$\underset{u}{\operatorname{argmin}} \|u - b\|_2^2 + \lambda\left(\|H_1 * u\|_1 + \|H_2 * u\|_1\right) \tag{5.30}$$

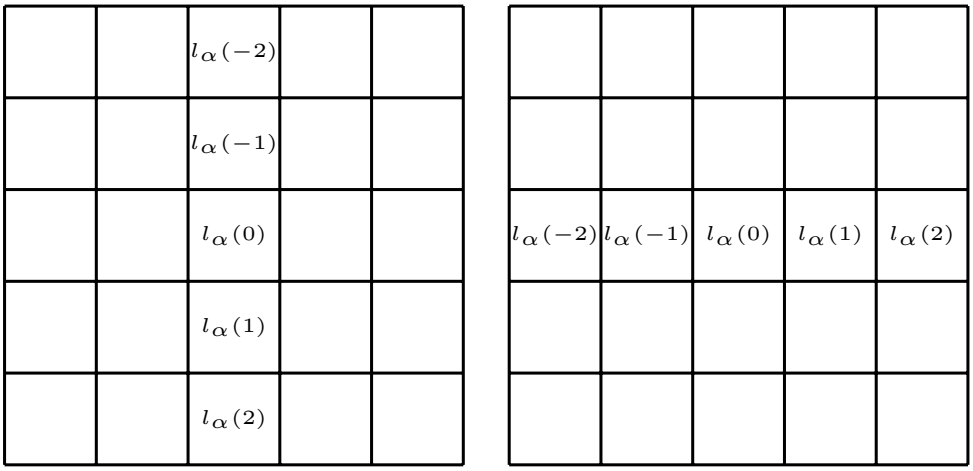


FIGURE 5.2: 5×5 2D masks for filtering images in (5.30) (H_1, H_2)

Algorithm 3: ADMM for Fractional ℓ_1 trend filter using FFT

Result: u^{k+1}

$u^0 = b, z^0, v^0 = 0$;

while $||u^{k+1} - u^k||_2^2 / u^{k+1} > \epsilon$ **do**

$u^{k+1} = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{b\}+\rho\mathcal{F}\{l_\alpha\}\odot\mathcal{F}\{z^k-v^k\}}{1+\rho\mathcal{F}\{l_\alpha\}^2}\right\}$

$z^{k+1} = S_{\lambda/\rho}\left(L_\alpha u^{k+1} + v^k\right)$

$v^{k+1} = v^k + L_\alpha u^{k+1} - z^{k+1}$

end

Discussion

It is notable that the fractional Laplacian in (5.14), being purely discrete, is a FIR filter of order N. To determine the order of the filter, the structure of the Laplacian should be exploited. The Laplacian filter is even symmetric and has a positive and relatively high middle coefficient surrounded on both sides by negative coefficients which decay towards zero approaching both ends. Smaller values of α correspond to filters with sharper roll-off in the frequency domain and as a result the filters are generally longer in the discrete domain.

To determine the length of the filter l_α, an empirical procedure is proposed that iteratively increases the length of the filter until the first and last coefficients become lower than a user specified threshold as can be seen in Algorithm 4.

Algorithm 4: Empirical procedure for determining the length N of the fractional-order Laplacian filter l_α

Input: α

Output: l_α, N

threshold $= 10^{-3}$;

$N = 3$;

$l_\alpha =$ use (5.14) ;

while $\sum \iota_{(|l_\alpha| > \text{threshold})} = N$ **do**

$N \leftarrow N + 1$

Calculate l_α using (5.14)

end

Frequency and impulse response of fractional Laplacian of various α's is presented in Figure 5.3

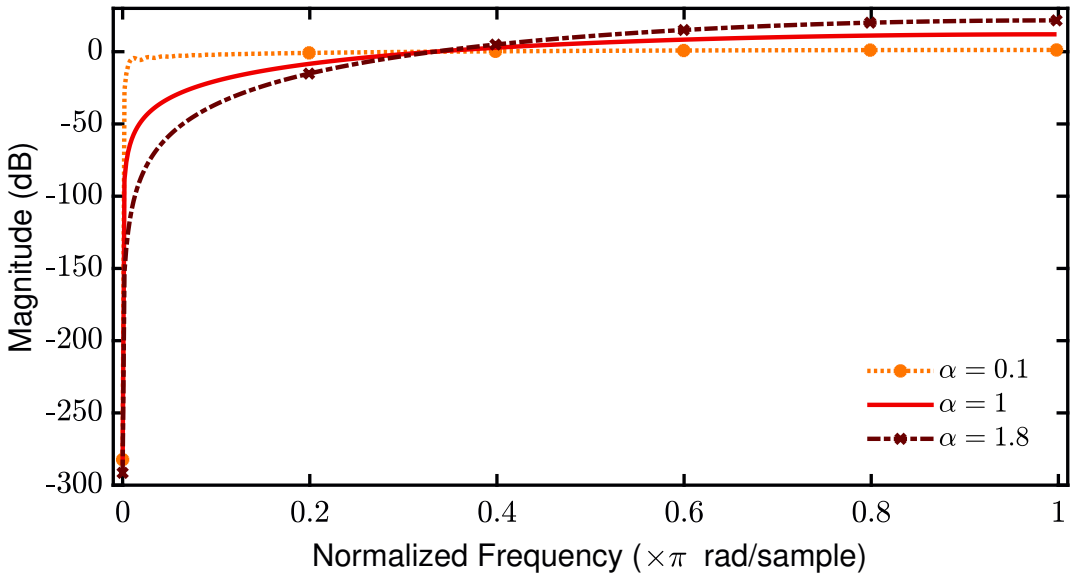


FIGURE 5.3: Frequency response of fractional Laplacian l_α.

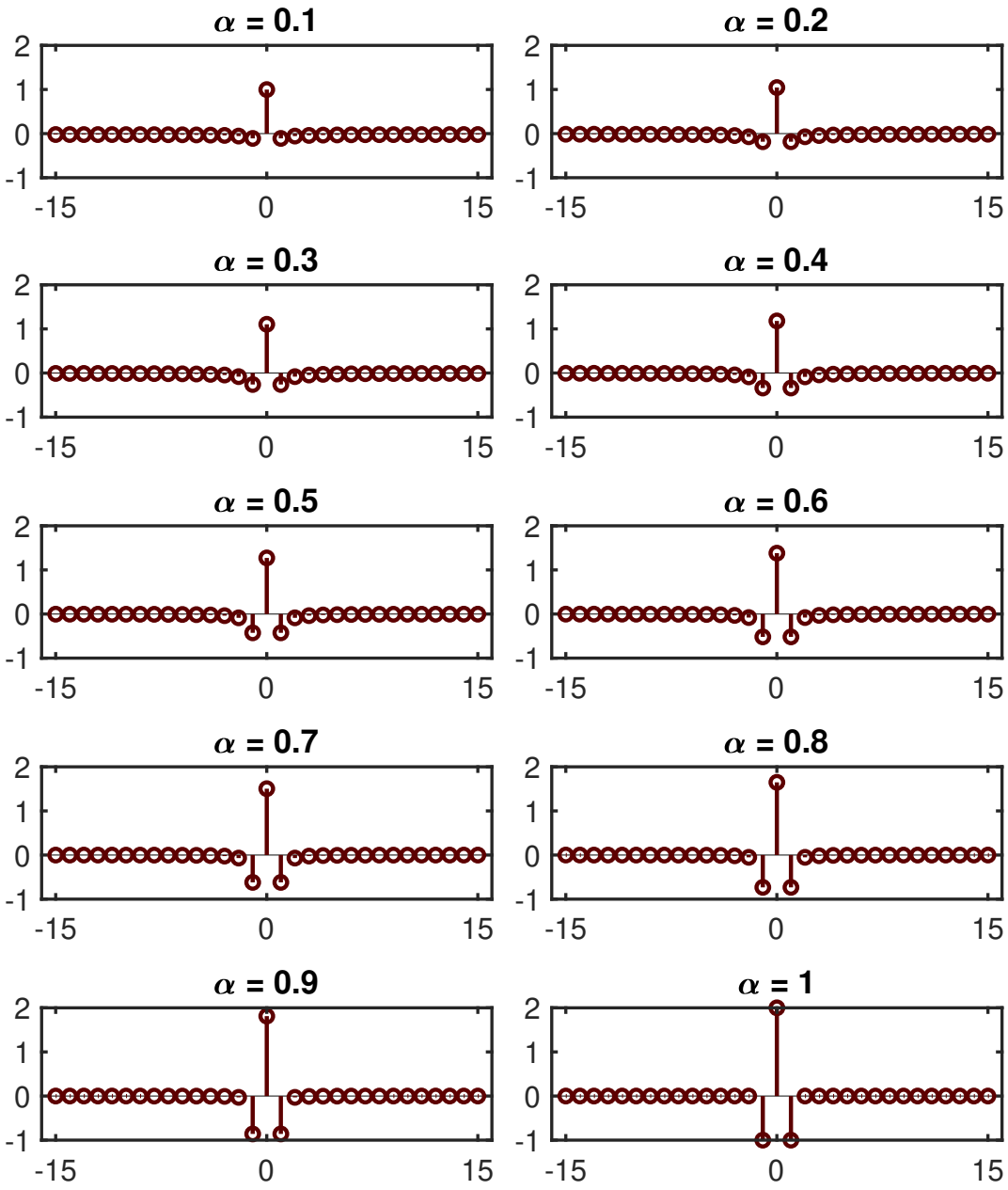


FIGURE 5.4: Impulse response of fractional Laplacian l_α.

Numerical examples

The use of the fractional trend filter in 1D and 2D settings is demonstrated. In the 1D case, a synthetic time-series data with a piece-wise linear trend and additive white Gaussian noise is generated. In Figures 5.5 and 5.6 synthesized data, synthesized trend and filters' results are reported for three values of $\alpha \in \{1, 0.6, 0.2\}$. From Figures 5.5 and 5.6 it is noticed that different fractional orders of trend filtering result in trend estimates that belong to different function families. In the case of $\alpha = 1$ and ℓ_1 fractional trend filter, which corresponds to the original ℓ_1 trend filter, the trend estimates are close to piece-wise linear but that is not the case for $\alpha = 0.6$ or 0.2.

In the 2D case, the fractional trend filter is tested on the image denoising task. An experiment is conducted on six greyscale images (cameraman, house, lena, peppers, pirate and blonde woman) shown in Figure 5.7. Starting with a ground truth image $\tilde{u}$ then noise is added to it with three noise levels $\sigma \in \{15, 25, 50\}$ forming an image b. The experiment is conducted at various values of $\alpha \in \{0.1 \dots 1\}$ with the goal of recovering an image $\hat{u}$ as close as possible to ground truth image $\tilde{u}$. Each experiment was run for 10 times (noise instances) and the average performance across the 10 runs is reported. Performance is measured in the form of mean squared error:

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[\tilde{u}\,(i,j) - \hat{u}\,(i,j)\right]^2. \tag{5.31}$$

In the case of fractional ℓ_1 trend filtering, the results can be found in Table 5.1. A pattern is noticed in the results that for higher noise levels, better reconstruction is achievable with smaller α. It is important to note here that the best regularization parameter λ was chosen using an exhaustive search.

In the case of fractional ℓ_2 trend filtering, experimental results can be found in Table 5.2. Similar procedure to ℓ_1 case was conducted. The results here again show that there is a benefit in using a fractional Laplacian for higher levels of noise.

In some cases, such as the "Cameraman" image and the "House" image, it turns out that the setting $\alpha = 1$ (corresponding to the Laplacian operator) leads to the best results. Such results should not be regarded as a weakness of the fractional Laplacian operator. Instead, this is an advantage of the fractional Laplacian operator which includes the Laplacian operator as a special

TABLE 5.1: Mean square error (MSE) of denoised image images for six images using fractional ℓ_1 trend filtering.

(A): Cameraman

	Noise σ		
α	**15**	**25**	**50**
0.1	10.87	19.95	34.14
0.2	11.79	18.56	37.11
0.3	13.69	19.40	44.17
0.4	13.78	23.01	34.85
0.5	11.24	16.70	**31.75**
0.6	9.12	**14.70**	32.82
0.7	9.61	15.09	32.54
0.8	**8.39**	15.79	32.96
0.9	8.49	15.06	33.36
1	8.75	15.23	33.84

(B): House

	Noise σ		
α	**15**	**25**	**50**
0.1	14.65	15.58	20.15
0.2	11.66	12.75	18.32
0.3	9.90	11.15	**18.16**
0.4	9.03	10.45	20.00
0.5	**8.73**	**10.40**	24.98
0.6	8.80	11.10	35.01
0.7	9.26	13.54	27.04
0.8	10.90	18.74	21.82
0.9	10.91	14.54	24.00
1	9.72	12.09	26.45

(C): Lena

	Noise σ		
α	**15**	**25**	**50**
0.1	11.56	16.86	21.17
0.2	13.56	14.47	19.51
0.3	12.23	13.20	**19.37**
0.4	11.66	12.68	21.04
0.5	**11.49**	**12.56**	25.63
0.6	11.53	12.68	35.54
0.7	11.64	13.03	28.89
0.8	11.78	14.13	23.85
0.9	11.93	16.11	26.24
1	12.09	16.81	25.17

(D): Peppers

	Noise σ		
α	**15**	**25**	**50**
0.1	11.15	17.34	**25.06**
0.2	12.94	14.55	25.33
0.3	11.12	13.63	28.64
0.4	10.47	14.59	36.11
0.5	10.88	18.17	35.43
0.6	13.07	16.87	29.12
0.7	9.64	13.38	25.33
0.8	9.90	14.52	26.24
0.9	**8.23**	**12.41**	27.65
1	8.69	12.49	29.50

(E): Pirate

	Noise σ		
α	**15**	**25**	**50**
0.1	11.61	16.22	21.49
0.2	12.99	13.94	**20.39**
0.3	11.69	12.71	21.09
0.4	11.10	12.18	24.28
0.5	**10.91**	**12.07**	31.71
0.6	10.94	12.25	36.36
0.7	11.04	12.88	24.59
0.8	11.19	15.26	27.61
0.9	11.35	16.90	22.56
1	11.52	15.65	23.68

(F): Blonde

	Noise σ		
α	**15**	**25**	**50**
0.1	14.43	15.37	19.86
0.2	12.02	13.15	**18.57**
0.3	10.74	12.11	18.81
0.4	10.28	**12.05**	20.90
0.5	10.46	13.07	25.75
0.6	11.63	16.24	35.17
0.7	12.36	20.73	28.09
0.8	**9.96**	14.25	29.38
0.9	10.17	15.35	23.53
1	10.48	13.33	25.39

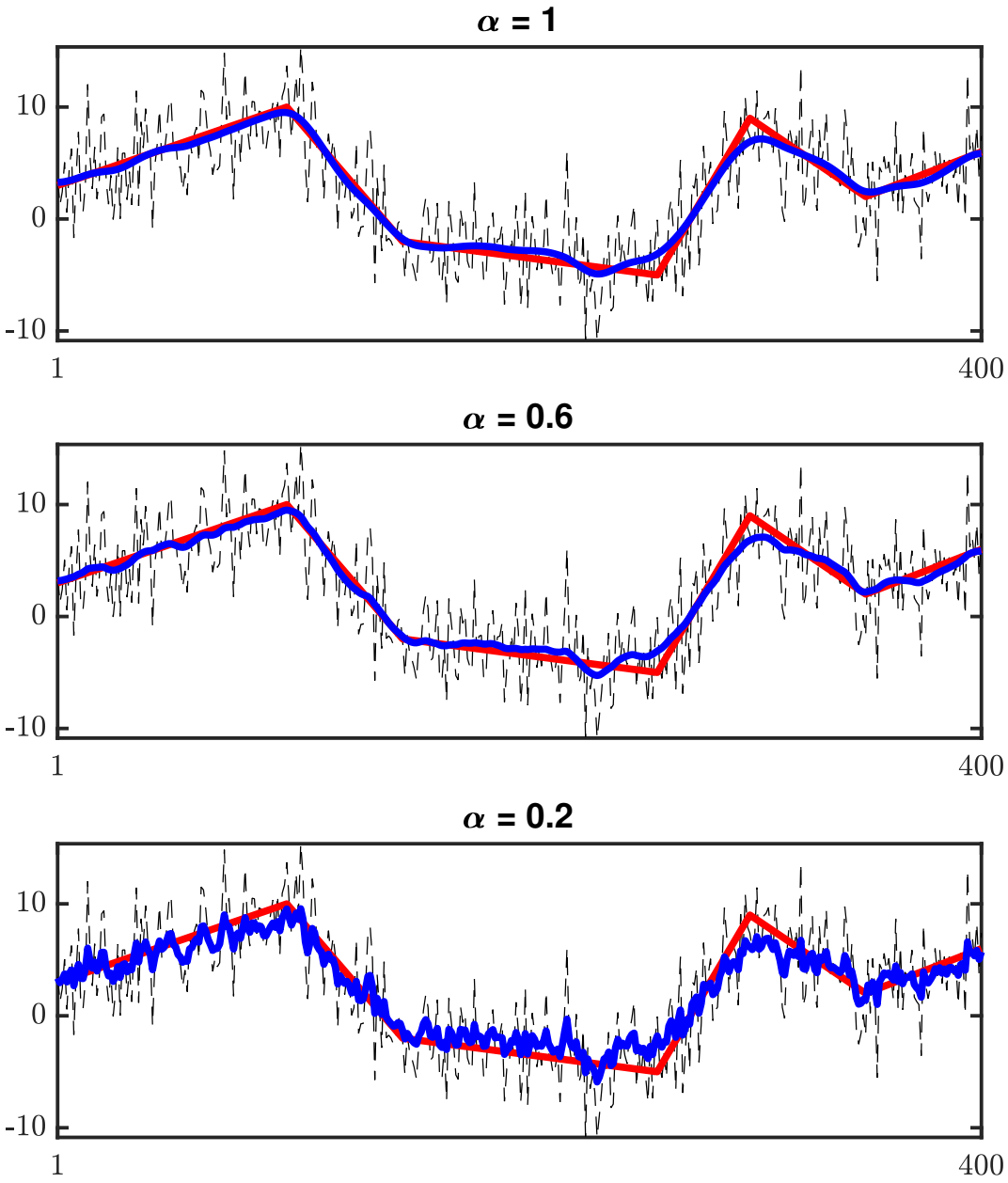


FIGURE 5.5: Fractional ℓ_2 trend filtering of synthesized data. Dashed black is the synthetic data. Solid red is the original trend. Solid blue is the filter result with λ chosen to minimize the mean squared error between the filter output and the original trend.

case. Compared with the Laplacian operator, the fractional Laplacian operator permits the user to "tune" the parameter α to achieve the desired result. As such, in other cases, better results were achieved by tuning α.

To get an idea about the difference in performance between the presented

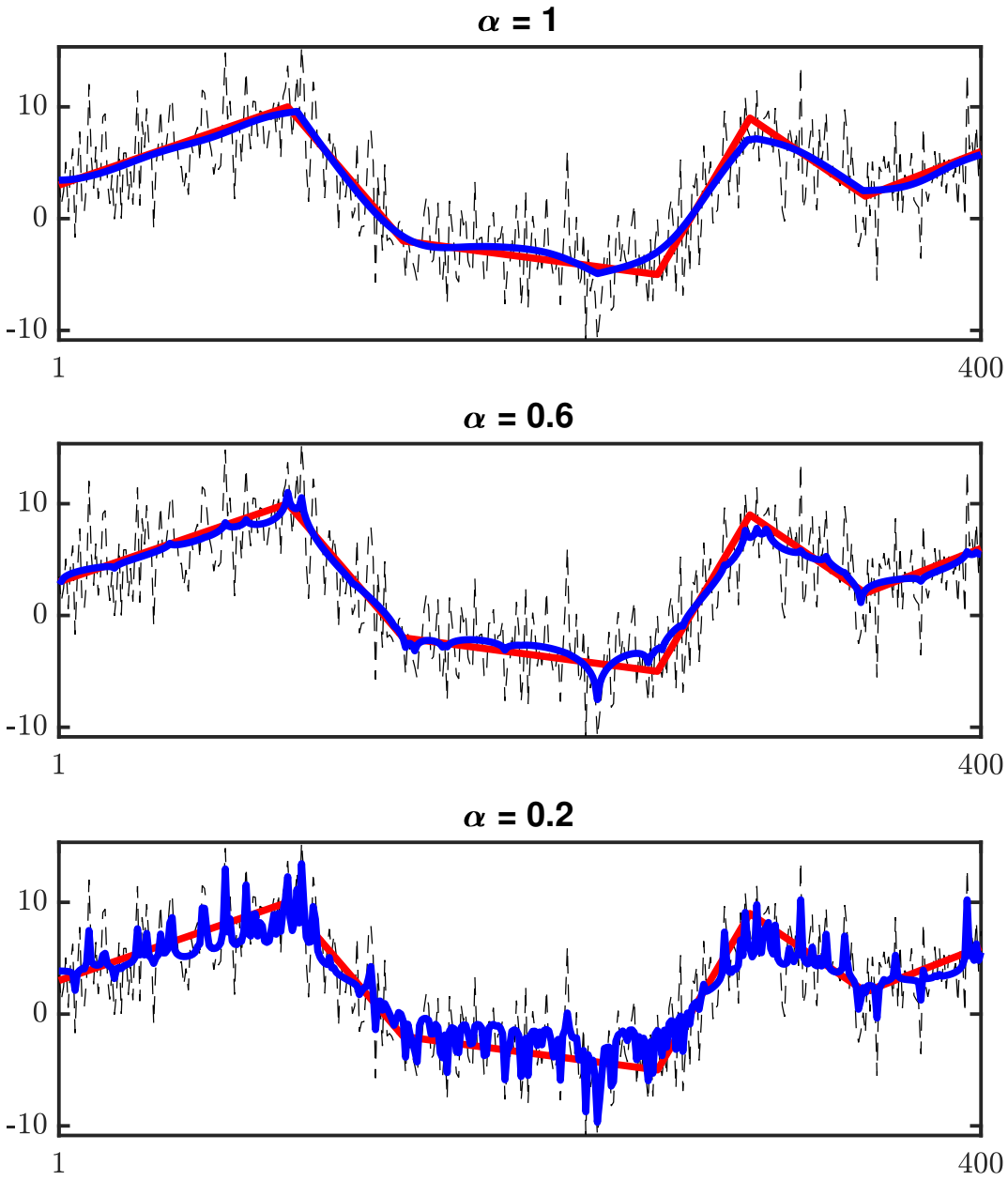


FIGURE 5.6: Fractional ℓ_1 trend filtering of synthesized data. Dashed black is the synthetic data. Solid red is the original trend. Solid blue is the filter result with λ chosen to minimize the mean squared error between the filter output and the original trend.

fractional trend filters and the state-of-the-art in image denoising, it is compared with BM3D [200] provided with the underlying noise standard deviation (not an estimate) in Table 5.3. BM3D is not the only state-of-the-art filter, but a representative example to demonstrate the performance gap between the fractional trend filter and the state-of-the-art in image denoising.

TABLE 5.2: Mean square error (MSE) of denoised image images for six images using fractional ℓ_2 trend filtering.

(A): Cameraman

	Noise σ		
α	**15**	**25**	**50**
0.1	12.75	18.68	31.44
0.2	11.38	18.70	21.83
0.3	10.82	14.58	19.10
0.4	11.12	13.10	17.80
0.5	10.06	12.59	17.15
0.6	9.50	12.31	**16.87**
0.7	9.21	12.15	17.03
0.8	9.08	12.07	17.51
0.9	9.01	12.04	18.13
1	**8.97**	**12.03**	18.83

(B): House

	Noise σ		
α	**15**	**25**	**50**
0.1	12.17	17.18	28.01
0.2	11.12	15.83	19.42
0.3	10.29	11.94	16.45
0.4	8.32	10.66	14.83
0.5	7.56	9.95	13.94
0.6	7.21	9.51	**13.87**
0.7	6.99	9.25	14.47
0.8	6.84	**9.13**	15.35
0.9	6.75	9.23	16.33
1	**6.69**	9.45	17.31

(C): Lena

	Noise σ		
α	**15**	**25**	**50**
0.1	12.61	17.88	28.38
0.2	11.09	17.53	20.79
0.3	10.43	13.74	18.07
0.4	10.51	12.59	16.70
0.5	9.75	12.17	16.03
0.6	9.42	11.95	**15.96**
0.7	9.30	11.86	16.43
0.8	9.27	**11.83**	17.14
0.9	**9.26**	11.85	17.94
1	9.28	11.91	18.77

(D): Peppers

	Noise σ		
α	**15**	**25**	**50**
0.1	12.69	18.59	31.35
0.2	10.86	18.12	21.37
0.3	9.68	13.10	17.81
0.4	9.30	11.54	15.97
0.5	8.38	10.86	15.01
0.6	8.05	10.47	**14.77**
0.7	7.87	10.25	15.24
0.8	7.76	10.14	16.01
0.9	7.69	**10.14**	16.89
1	**7.66**	10.27	17.80

(E): Pirate

	Noise σ		
α	**15**	**25**	**50**
0.1	12.47	17.62	27.68
0.2	10.88	16.99	20.33
0.3	10.11	13.37	17.67
0.4	10.13	12.27	16.36
0.5	9.34	11.82	15.73
0.6	8.99	11.58	**15.67**
0.7	8.85	11.46	16.13
0.8	8.79	11.41	16.83
0.9	8.76	**11.41**	17.62
1	**8.75**	11.45	18.45

(F): Blonde

	Noise σ		
α	**15**	**25**	**50**
0.1	12.01	18.33	24.64
0.2	11.46	15.58	19.16
0.3	10.98	12.55	16.71
0.4	9.42	11.60	15.41
0.5	8.78	11.14	**14.78**
0.6	8.55	10.91	14.91
0.7	8.45	10.80	15.53
0.8	8.41	**10.76**	16.35
0.9	**8.39**	10.81	17.24
1	8.40	10.95	18.14

TABLE 5.3: Mean square error (MSE) of denoised image images for six images using BM3D.

	Noise σ		
α	**15**	**25**	**50**
Cameraman	5.60	7.47	10.87
House	4.26	5.47	8.22
Lena	6.47	8.66	12.31
Peppers	5.38	7.29	11.05
Pirate	7.20	9.53	13.39
Blonde	5.88	7.90	11.66

FIGURE 5.7: Grayscale images (256×256) used in the denoising experiments. (a) cameraman, (b) house, (c) lena, (d) peppers, (e) pirate and (f) blonde woman.

5.3.2 Image processing applications

1D filtering

To get a better idea about the effect of using the proposed fractional Laplacian, a comparison on a test signal from the Wavelet Toolbox in MATLAB named blocks is conducted. The choice of this signal was made because; it is a piece-wise constant signal with sharp edges, allowing for clearer demonstration of the impact of the filters on edges. Results of linear filtering in Figure 5.8 demonstrate the long memory effect. The fractional Laplacian with smaller values of α have longer memory effect than the standard Laplacian.

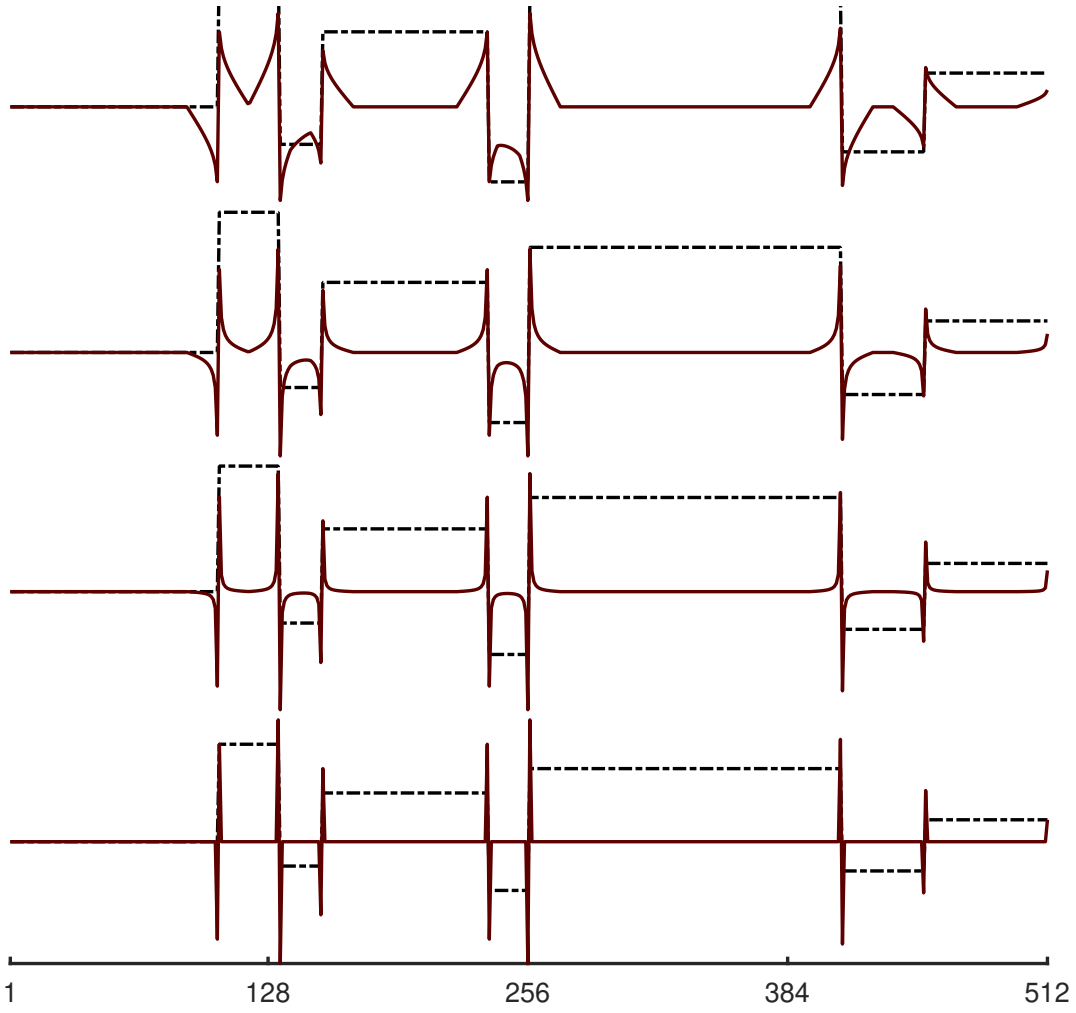


FIGURE 5.8: Blocks signal (dashed black) filtered using fractional Laplacian l_α (solid red) with different values of α. From top to bottom are filtered signals with $\alpha \in \{0.1, 0.4, 0.7, 1\}$.

Image sharpening

One use case for fractional Laplacian is to increase the image sharpness as follows:

$$J = I + \gamma(H_\alpha * I) \tag{5.32}$$

where γ is an amplification factor and H_α is the 2D isotropic fractional Laplacian kernel in Figure 5.9 formed by adding two kernels similar to the ones in Figure 5.2.

		$l_\alpha(-2)$		
		$l_\alpha(-1)$		
$l_\alpha(-2)$	$l_\alpha(-1)$	$2l_\alpha(0)$	$l_\alpha(1)$	$l_\alpha(2)$
		$l_\alpha(1)$		
		$l_\alpha(2)$		

FIGURE 5.9: 5×5 2D isotropic fractional Laplacian (H_α).

Figure 5.10 is a comparison of image sharpening performance between the standard Laplacian ($\alpha = 1$), the fractional Laplacian ($\alpha = 0.5$) and the guided filter [201]. The results in Figure 5.10 clearly demonstrate that a sharper image was achieved using a fractional Laplacian with $\alpha = 0.5$ than the standard Laplacian ($\alpha = 1$), this results is expected as the fractional-order Laplacian captures more information than the standard Laplacian as can be seen in the impulse responses of Laplacian filters in Figure 5.3. Sharpening using the guided filter is presented as a comparison with a state-of-the-art edge-aware filter. Recall that the fractional Laplacian is a linear operator while the guided filter is non-linear.

FIGURE 5.10: Image sharpening application. (a) is the original input image. (b) is sharpened image using the guided filter. (c) is sharpened image with $\alpha = 1$. (d) is sharpened image with $\alpha = 0.5$. Sharpened images are produced according to (5.32) with $\gamma = 2$. In the case of the guided filter, the residual of filtering is boosted ($J = I + \gamma(I - \mathrm{GF}(I))$.

Edge detection

The literature on edge detection is rich and it is beyond the scope of this study to list and compare with all techniques in the literature however, the potential use of the proposed fractional Laplacian in this task is being examined. Starting with the traditional Marr-Hildreth edge detector [202]. A greyscale image is first smoothed with a Gaussian filter to reduce the impact of noise and make the detection more robust. This is followed by Laplacian

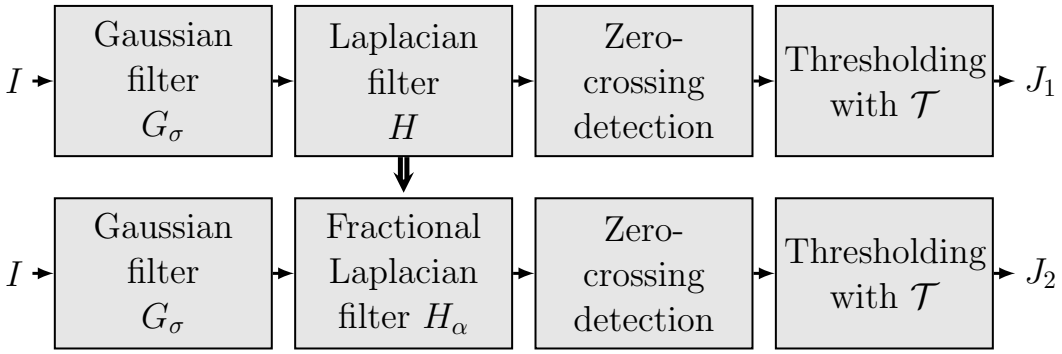


FIGURE 5.11: Block diagram of the Marr-Hildreth edge detector (top) and the extended version with our fractional Laplacian (bottom).

filtering step to find the edges. Filtering an image with the Laplacian kernel results in zero-crossings where edges are potentially located. Next, slopes at the zero-crossings are computed and finally a threshold $\mathcal{T}$ is applied to keep significant edges only. The algorithm is summarized in Figure 5.11.

Marr-Hildreth edge detector is generalized by replacing the Laplacian by its fractional-order generalization.

Figure 5.12 illustrates the results that can be achieved using various values of α. In Figure 5.12, it is clear that more robust edge detection in terms of edge lines can be achieved with values of α other than 1.

Edge detectors based on the second order derivative are known to be sensitive to noise [203] which begs the question: does the fractional Laplacian suffer from sensitivity to noise as well? To this end, a synthetic image with various forms of edges is used, these are edges that commonly exist in natural images, then Gaussian noise is added to it and finally processed it with the fractional Marr-Hildreth edge detector in Figure 5.13. From Figure 5.13 it is clear that the fractional Laplacian is less sensitive to noise. This result is expected because the fractional Laplacian has more coefficients (memory property) than the standard Laplacian.

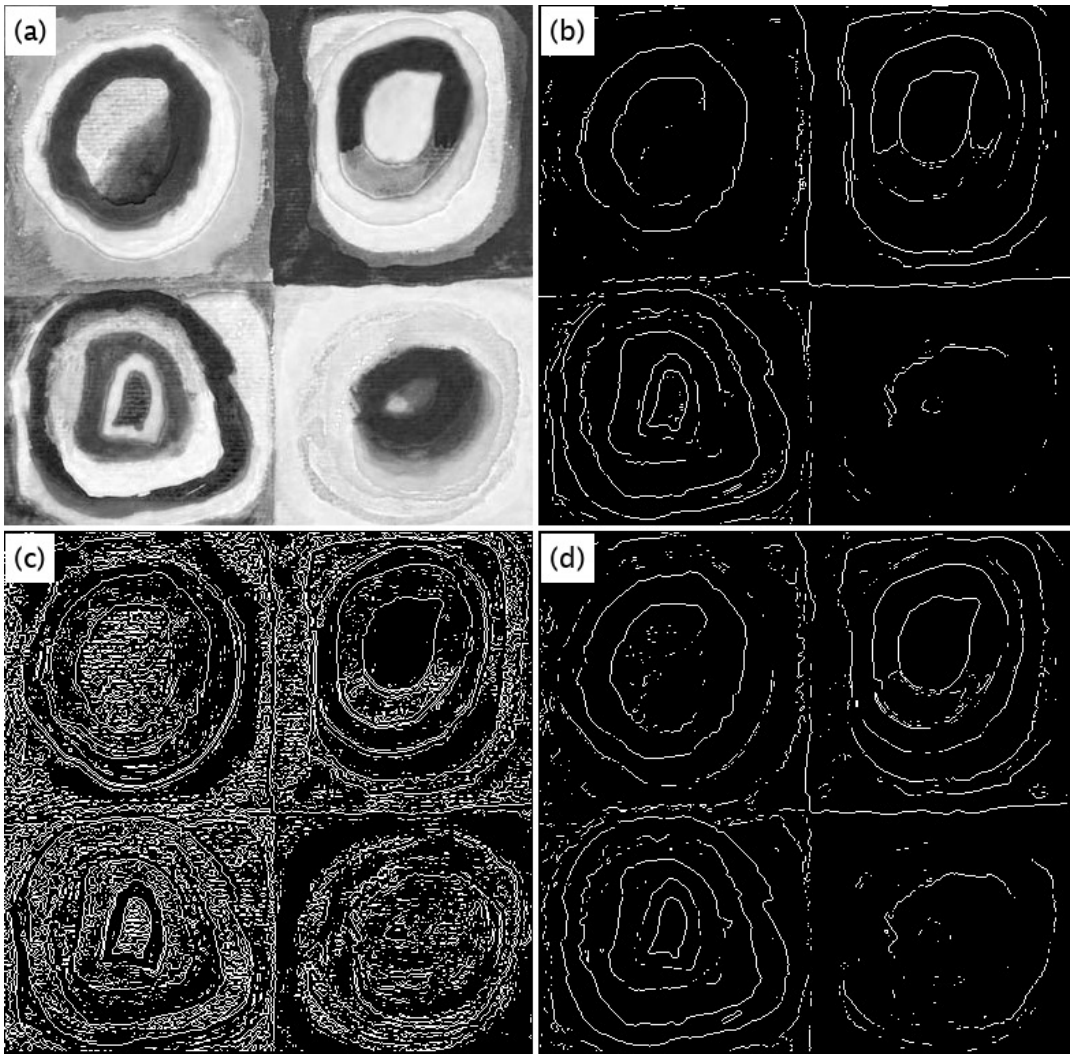


FIGURE 5.12: Fractional Marr-Hildreth edge detection. (a) is the original image. (b) is edge map produced using MATLAB's *edge* command (based on first order derivatives) using default values. (c) is edge map produced using the standard Marr-Hildreth. (d) is edge map produced using Fractional Marr-Hildreth with $\alpha = 0.2$. Gaussian smoothing with $\sigma = 1$ was used in this experiment.

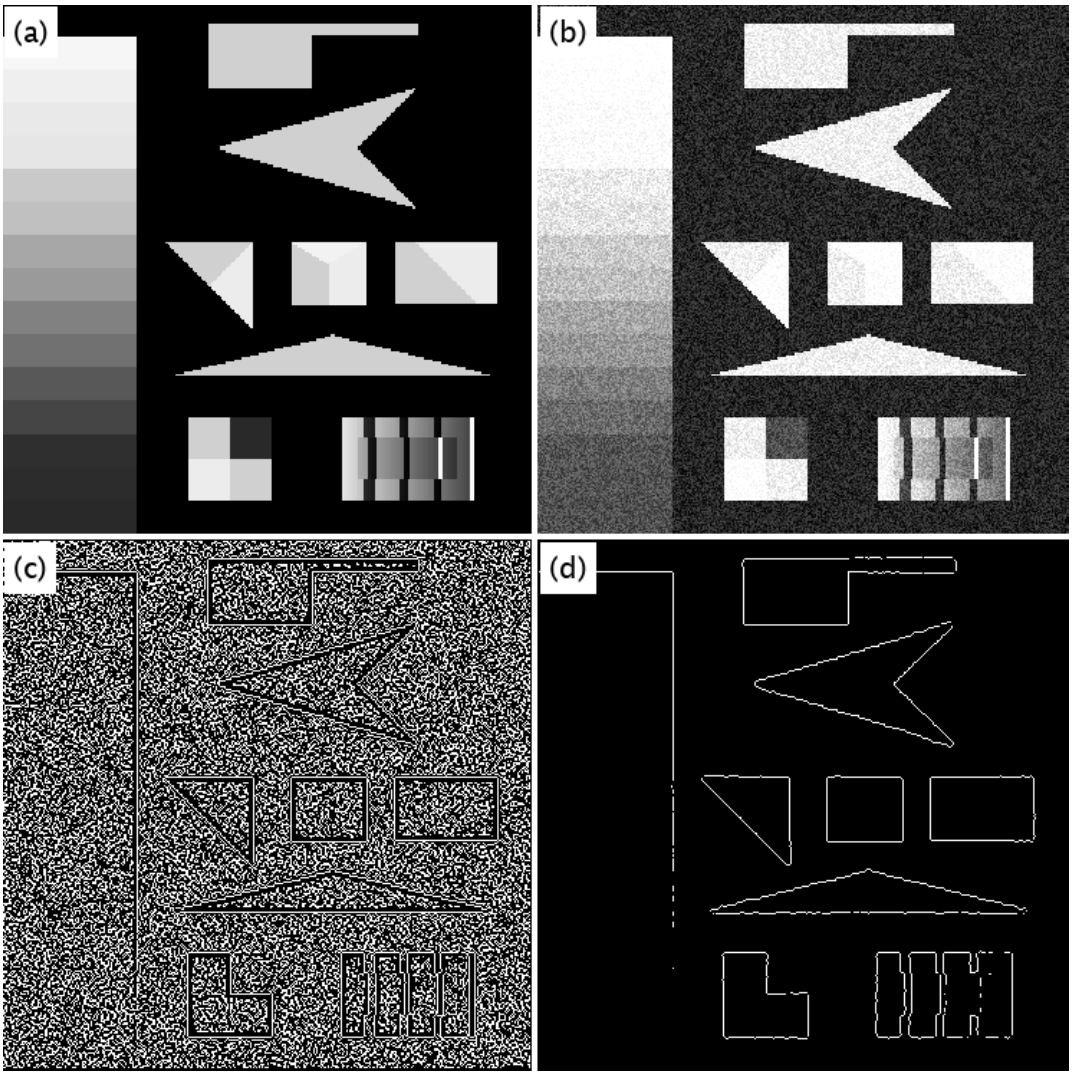


FIGURE 5.13: Noise robustness of the fractional Marr-Hildreth edge detector. (a) is the original image. (b) is a noisy image formed by adding noise with $\sigma = 0.3$ to the original image. (c) is edge map produced using the standard Marr-Hildreth. (d) is edge map produced using fractional Marr-Hildreth with $\alpha = 0.1$. Gaussian smoothing with $\sigma = 1$ was used in this experiment.

Shock filtering

Shock filtering was initially proposed by Osher and Rudin [204] for image enhancement but, the technique received a lot of interest from researchers. Shock filters are formulated as PDEs that are evolved over time to come up with the filtered image which is characterized to be piece-wise constant. The basic formulation of a shock filter is as follows:

$$\frac{\partial u}{\partial t} = -\operatorname{sign}(\Delta u)|\nabla u| \tag{5.33}$$

A more robust version utilizes a smoothing operator such as a Gaussian kernel, and is formulated as follows:

$$\frac{\partial u}{\partial t} = -\operatorname{sign}(\Delta(G_\sigma * u))|\nabla(G_\sigma * u)| \tag{5.34}$$

G_σ represents a two-dimensional Gaussian filter with σ being a smoothness parameter and is set to 1 in all experiments in this work.

The proceeding model is extended to fractional order

$$\frac{\partial u}{\partial t} = -\operatorname{sign}(L_\alpha(G_\sigma * u))|\nabla(G_\sigma * u)| \tag{5.35}$$

which can be solved using the explicit scheme [110, Appendix A]

$$u^{n+1} = u^n - \lambda \operatorname{sign}(L_\alpha(G_\sigma * u^n))|\nabla(G_\sigma * u^n)| \tag{5.36}$$

This extra parameter α gives more control over the filtering effect. In Figure 5.14, a comparison between the fractional ($\alpha = 0.6$) and the standard Laplacian ($\alpha = 1$) is presented. The fractional Laplacian produces better object segmentation effect with sharper edges than the case with the standard Laplacian.

Stopping criterion: For values of $0 < \alpha < 1$, time stepping (5.36) for a large amount of time results in images that are not pleasant. To give a sense of what is happening, and because the shock filter results in piece-wise constant images, a normalized anisotropic TV-L2 cost at every iteration is computed. The TV-L2 is:

$$\text{TV-L2} : \left(\|u^n - u^0\|_2^2 + \|\nabla_x u^n\|_1 + \|\nabla_y u^n\|_1\right)/N \tag{5.37}$$

FIGURE 5.14: α Shock filter of the top input image. Columns correspond to values of $\alpha \in \{0.6, 1\}$ from left to right. Rows correspond to values of $N \in \{1, 25, 100\}$ from left to right. The parameter λ was set to 0.1 for all results.

where N is the number of pixels in the image u. Figure 5.15 demonstrates the measure TV-L2 for different runs of the shock filter at different values of α.

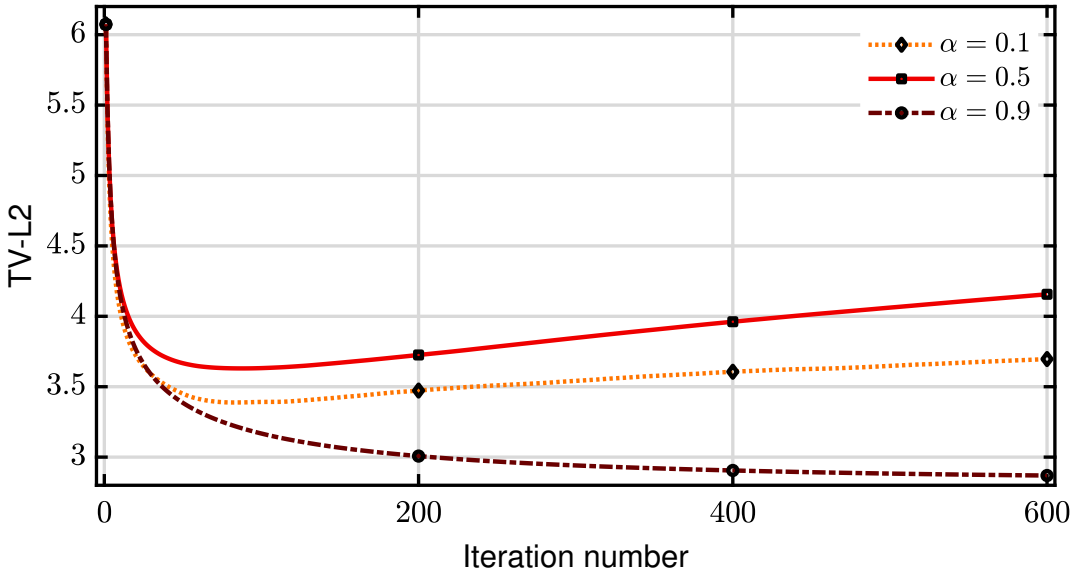


FIGURE 5.15: TV-L2 calculated for 600 iterations of the model in (5.36). Each curve corresponds to one value of $\alpha \in \{0.1, 0.5, 0.9\}$.

It is clear from Figure 5.15 that the shock filter might not converge. To avoid the shock filtered result from diverging, TV-L2 is adopted as a stopping criterion as in Algorithm 5. In Figure 5.16, the impact of using differ-

Algorithm 5: Fractional shock filter with stopping criterion

Input: α, I_{input}
Output: u^{n+1}
threshold $= 10^{-3}$;
$u^0 = I_{input}$;
while *TV-L2* (5.37) $>$ *threshold* **do**
  u^{n+1} using (5.36)
  $n \leftarrow n + 1$
end

ent values of α in the proposed shock filter is presented for different images where, the number of iterations is determined by the stopping criterion.

One thing to notice in Figure 5.16 is that the fractional shock filtering results in more abstract images and sharper edges. To further validate this observation, a scanline from the three channels (RGB) is plotted in Figure 5.17 for the four images in Figure 5.16. In each subplot of Figure 5.17, scanlines of two cases are presented to facilitate visual comparison. It can be observed in Figure 5.17 that the scanlines for $\alpha = 1$ are between the lower and upper envelopes of the ones with $\alpha = 0.1$, to understand the reason behind

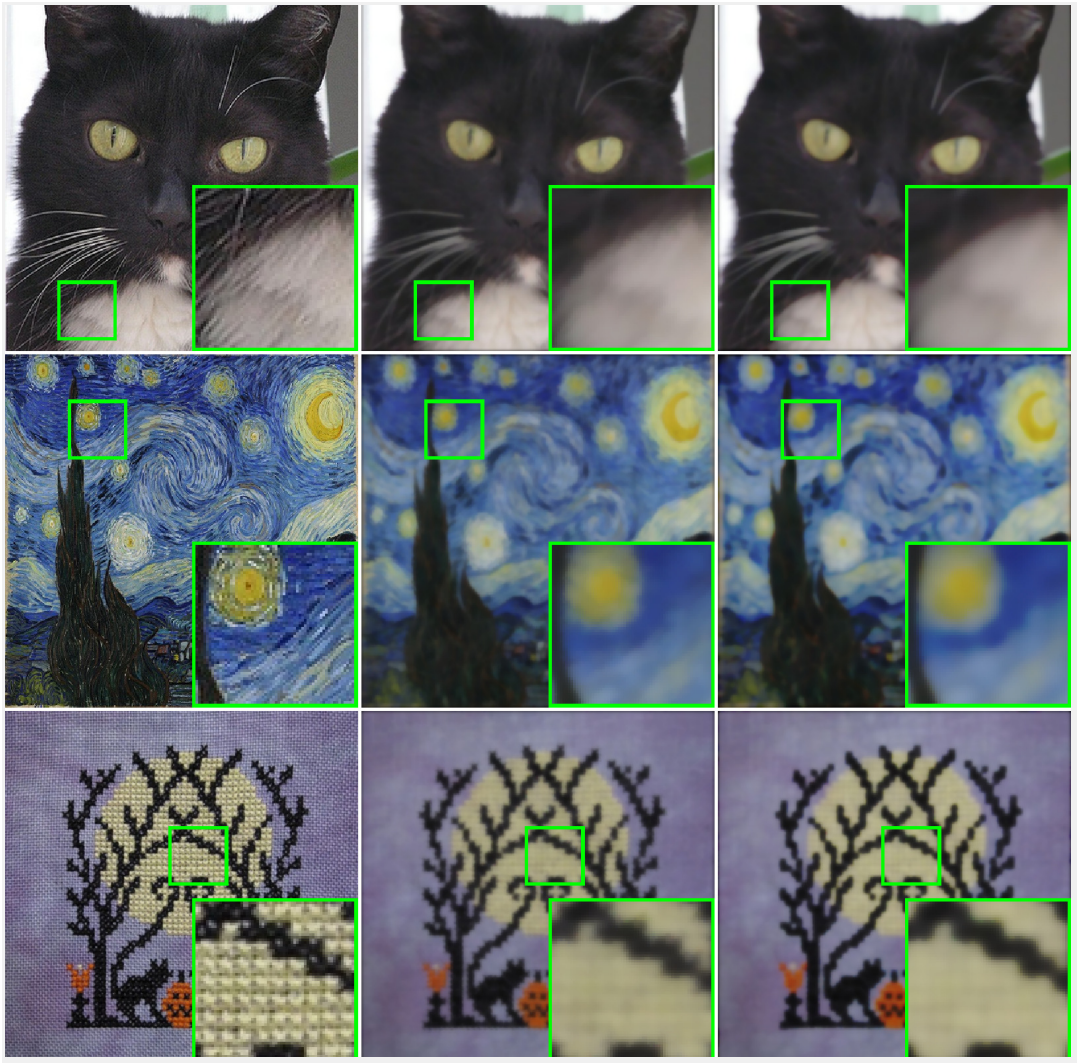


FIGURE 5.16: α Shock filter with stopping criterion Algorithm 5. Left column are input image. Middle column is the output images of shock filter with $\alpha = 1$ (Standard shock filter). Right column is the output of shock filtering with $\alpha = 0.1$. The parameter h was set to 0.05 in this experiment.

this result, we should consider the operation of the shock filter in (5.36). The fractional Laplacian (L_α) filters a smoothed version of u^n at every iteration n and the sign of the resulting signal is used. In Figure 5.18, a scanline is filtered using the standard Laplacian and the fractional Laplacian with $\alpha = 0.1$ for comparison, which demonstrates that the fractional Laplacian ($\alpha = 0.1$) results in a signal with more negative and positive parts and less zeros compared to the result of the standard Laplacian. This explains the reason behind the more abstract images achieved using the fractional Laplacian compared to the standard Laplacian.

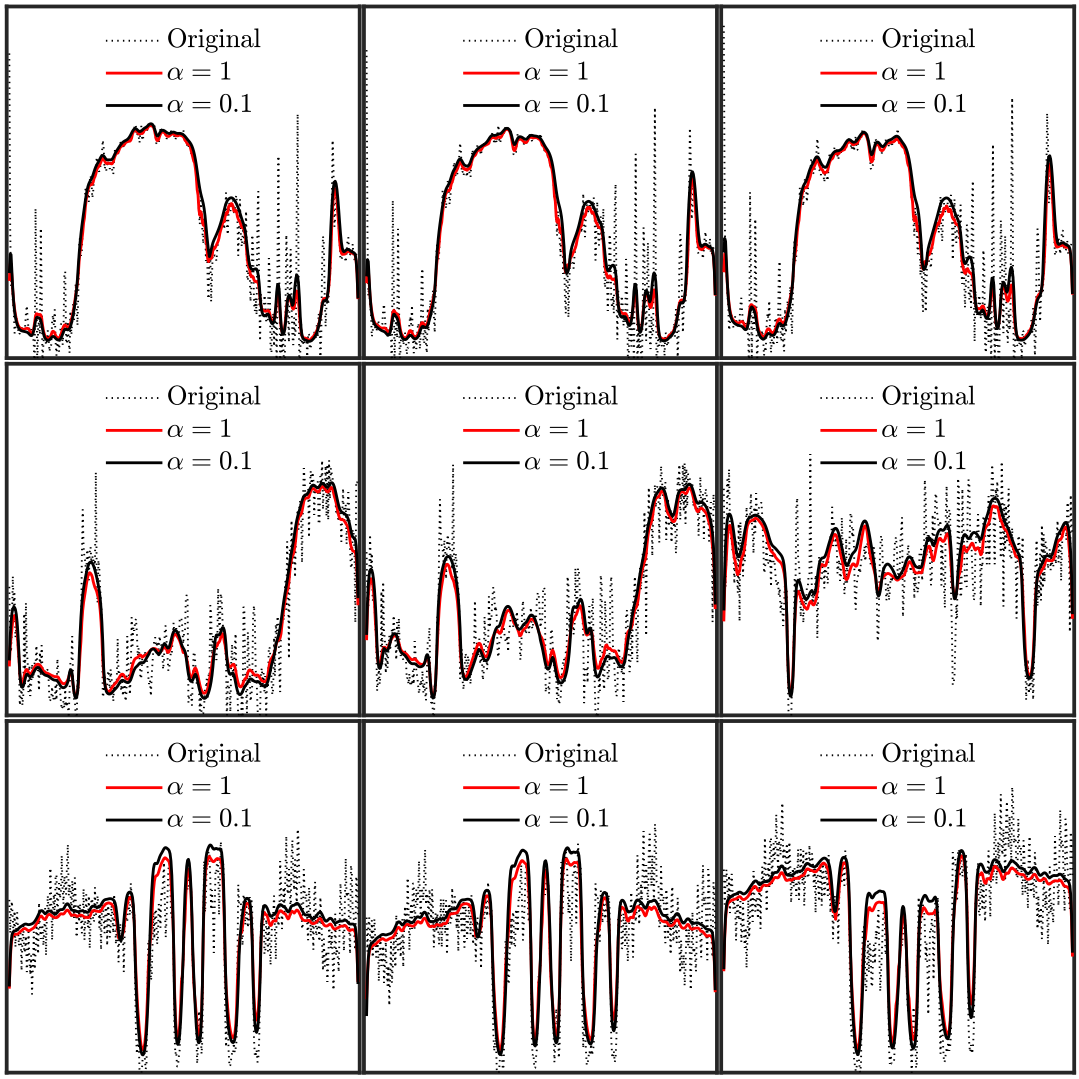


FIGURE 5.17: Scan-lines captured from the RGB (left to right) channels of the images shown in Figure 5.16. Using the fractional Laplacian results in more abstract lines.

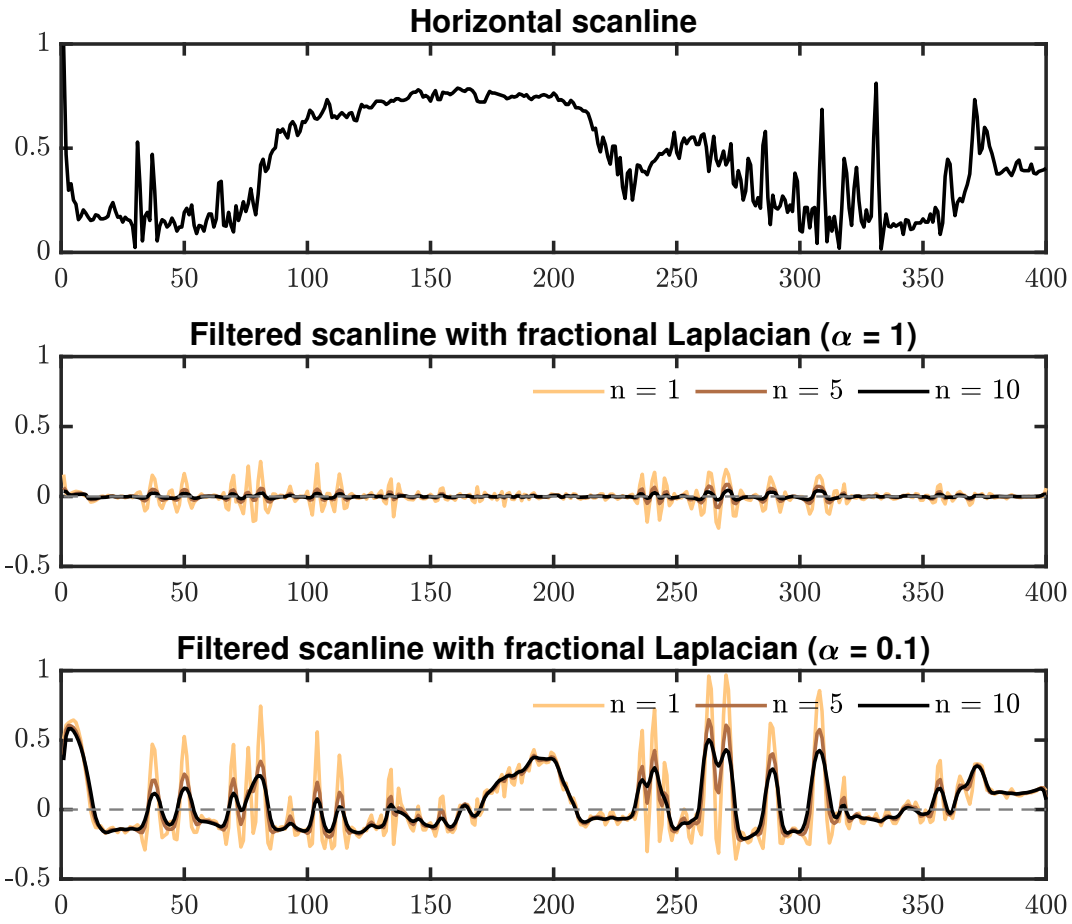


FIGURE 5.18: Scan-line (line 350) captured from the red channel of the cat image shown in Figure 5.16. The fractional Laplacian with $\alpha = 0.1$ results in more positive and negative signals and less zeros compared to the standard Laplacian ($\alpha = 1$)

α scale-space

The fractional Laplacian could be used as replacement for the standard Laplacian in the heat equation, this is known as α scale-space model [205]:

$$\frac{\partial u}{\partial t} = -L_\alpha u \tag{5.38}$$

This leads to different diffusion effects and rates. The implementation of this diffusion can be carried out efficiently in the DCT domain. First, the explicit scheme [110] is written:

$$u_{t+1} = u_t \quad -L_\alpha u_t \tag{5.39}$$

$$u_{t+1} = u_t \quad -M^T E^\alpha M u_t \tag{5.40}$$

$$\underbrace{Mu_{t+1}}_{U_{t+1}} = \underbrace{Mu_t}_{U_t} - \underbrace{MM^T}_{I} E^\alpha \underbrace{Mu_t}_{U_t} \tag{5.41}$$

Consequently, the filtered signal at iteration n becomes:

$$U_n = (I - E^\alpha)^n U_0 \tag{5.42}$$

A demonstration of the effect of α scale-space is presented in Figure 5.19. It is important to stress that this is an iterative linear filtering process. The filters are characterized as having low-pass response. The authors of [179] have explored combining multiple fractions of the Laplacian implemented in the Fourier transform domain.

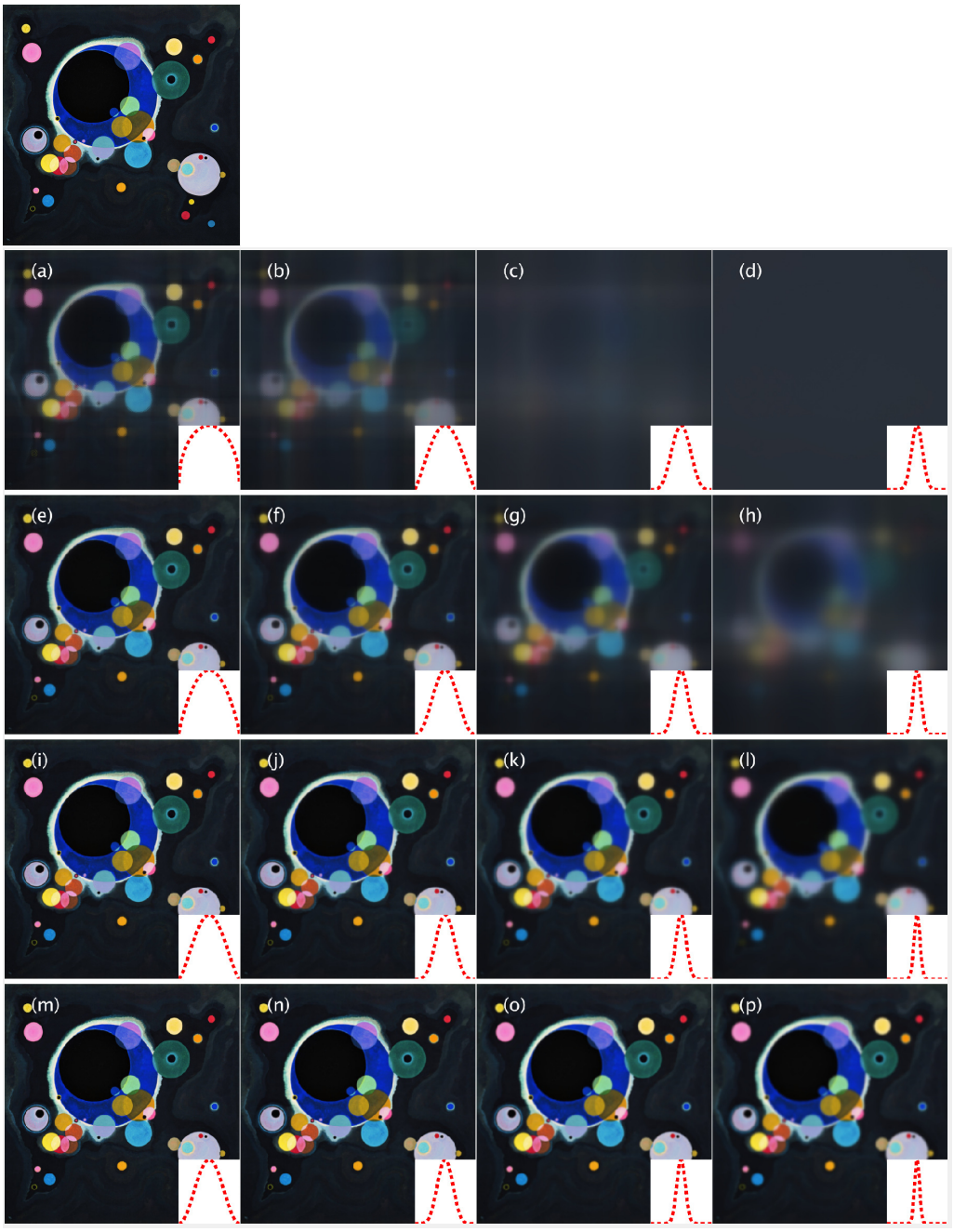


FIGURE 5.19: α scale-space filtered image on the top. Rows correspond to values of $\alpha \in \{0.2, 0.4, 0.7, 1\}$ from top to bottom. Columns correspond to values of $N \in \{1, 3, 10, 30\}$ from left to right. In the bottom right corner of each image is a plot of the α scale-space function in (5.42) with the corresponding values of α and N.

5.4 Chapter summary

In this chapter, a new technique for constructing a fractional Laplacian was presented using the DCT transform. The proposed operator is a matrix operator which avoids the need for discretization, a necessary step for implementation, typically done in DSP-based constructions. The trend filter was studied and a new DCT-based implementation was provided for it, which was used later in generalizing trend filters to the fractional-order. The proposed fractional Laplacian allowed us to make the generalization of another version of the trend filter namely the ℓ_1 trend filter to fractional-order.

To test the efficiency of the new operator, it was incorporated in five applications that traditionally relied on the Laplacian operator. Firstly, the proposed fractional trend filters were used for image denoising, and at higher noise levels it was shown that the fractional-order Laplacians tends to produce better results than the standard Laplacian. Secondly, the proposed operator was used in image sharpening and stronger sharpening effect was achieved compared to the standard Laplacian. Thirdly, Marr-Hildreth scheme for edge detection was generalized and more robust edge detection was demonstrated. Fourthly, it was demonstrated that the use of fractional-order Laplacian in shock filtering resulted in better segmented images. Finally, the proposed fractional Laplacian was incorporated in the α scale-space scheme and it was shown to result in faster diffusion.

Finally, it is worth reiterating; the aim of this chapter is to present an alternative way to define the fractional Laplacian operator, which, to the author's best knowledge, has not been done before. A demonstration of its potential application in solving a wide range of problems from data modelling to image processing was provided. It was observed from the simulations that the performance is sometimes sub-optimal. This is a result of applying the fractional Laplacian operator in a direct and non-sophisticated fashion. Further improvement can be achieved by using the fractional Laplacian operator as a building block in some successful algorithms.

Chapter 6

Conclusion and future directions

Mathematical optimization is a versatile tool in signal and image processing. Developing algorithms in the form of optimization models makes for a disciplined approach. In this thesis, four optimization based algorithms were presented.

In Chapter 2, graph signal denoising was formulated as a quadratic objective function. The objective function admitted an exact solution but, this solution involved the calculation of a matrix inverse, a prohibitively expensive task for many applications. Chapter 2 presented a solution based on using a matrix (graph operator) polynomial filter to approximate the inverse. The main idea behind the technique is to approximate the eigenvalues of the inverse using a least squares criterion avoiding the need for the expensive calculation of the eigendecomposition. The result is a polynomial filter of low degree, hence computationally efficient.

Chapter 3 presented GAIF, a novel local edge-preserving image filtering technique. GAIF achieved edge-preservation by interpolating between two patches. More specifically, GAIF can improve the edge-preservation smoothing ability of linear and non-linear filters. GAIF is computationally efficient and its edge-preserving smoothing performance was demonstrated on a number of image processing applications including single image haze-removal, flash/no-flash image fusion, image detail enhancement and edge detection.

In Chapter 4, a solution was proposed for one of the main drawbacks of the total variation model, namely the staircasing phenomenon. The proposed solution is a simple generalization of the TV model. It was demonstrated that based on the signal characteristics, better denoising performance is achievable by interpreting the finite difference operator D as a half-band high-pass filter, which made it possible to replace it with another banded matrix H of a high-pass filter with prescribed bandwidth.

Finally in Chapter 5, a new technique for constructing the fractional Laplacian was presented using the DCT transform. The proposed operator is a matrix operator, discrete by construction avoiding the need for discretization, a necessary step for implementation, typically done in DSP-based constructions. The trend filter was studied and a new DCT-based implementation was provided for it, which was used later in generalizing trend filters to the fractional-order. The proposed fractional Laplacian made it possible to make the generalization of another version of the trend filter namely the ℓ_1 trend filter to fractional-order.

The efficiency of the proposed operator was demonstrated by incorporating it in five applications that traditionally relied on the Laplacian operator. Firstly, in image denoising, it was shown that at higher noise levels the fractional-order Laplacian tends to produce better results than the standard Laplacian. Secondly, in image sharpening, stronger sharpening effect was achieved using the fractional Laplacian. Thirdly, in edge detection, more robust edge detection was demonstrated using the fractional Laplacian. Fourthly, in shock filtering, better segmented images were achieved using the fractional-order Laplacian compared to the standard Laplacian.

Research, by nature, is a never ending endeavour and the future directions from this work can be summarized in the following:

- It was noted in Section 2.7 that the GAIF filter is constructed from two images, I and M, produced using the same modality. This is a limiting factor for many applications that require cross-domain fusion such as matting/feathering [101] and super-resolution. It is of interest to enhance the GAIF by enabling it to handle images from different modalities to test its performance on more applications including multi-sensor fusion, image vectorization, colourization, non photo-realistic rendering and low-light image enhancement. This is one of the future directions.

- Another direction is based on a noticed departure from convex optimization towards non-convex models or convex models with non-convex terms in the signal and image processing communities [206, 77, 207, 208, 209] due to their flexibility and expressiveness. It is interesting to formulate non-convex image filters and design solvers for them.

- Another direction in optimization models is towards data-driven models [210, 211, 212], where the models are learnable either fully or partially.

Appendix A

Explicit kernel proof

In the special case of M is a box filter (linear), GAIF is equivalent to the self-guided GIF. Specifically, the model of the self-guided GIF is:

$$\arg\min_{a_k,b_k} C(a_k,b_k) = \sum_{p\in\Omega_k} (a_k I_p + b_k - I_p)^2 + \epsilon a_k^2 \tag{A.1}$$

optimizing for b_k results in the following:

$$b_k = \mu_k - a_k\mu_k = (1-\alpha_k)\mu_k \tag{A.2}$$

where

$$\mu_k = \frac{1}{N}\sum_{p\in\Omega_k} I_p$$

substituting b_k in the GIF model yields:

$$\arg\min_{a_k,b_k} C(a_k) = \sum_{p\in\Omega_k} (a_k I_p + (1-\alpha_k)\mu_k - I_p)^2 + \epsilon a_k^2 \tag{A.3}$$

Equation (A.3) is equivalent to GAIF with $M = \mu_k$.

Bibliography

[1] S. Boyd, C. Crusius, and A. Hansson, "Control applications of nonlinear convex programming," *J. Process Control*, vol. 8, no. 5-6, pp. 313–324, Oct. 1998.

[2] H. Lebret and S. Boyd, "Antenna array pattern synthesis via convex optimization," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 526–532, Mar. 1997.

[3] S. Sra, S. Nowoz, and S. J. Wright, *Optimization for machine learning*. Cambridge, Mass: MIT Press, 2011.

[4] R. Szeliski, *Computer vision : algorithms and applications*. London New York: Springer, 2011.

[5] S. J. D. Prince, *Computer Vision*. Cambridge University Press, 2012.

[6] D. Palomar, *Convex optimization in signal processing and communications*. Cambridge New York: Cambridge University Press, 2010.

[7] R. C. Gonzalez and R. E. Woods, *Digital image processing*. New York, NY: Pearson, 2018.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004.

[9] M. D. Intriligator, "Chapter 2 Mathematical programming with applications to economics," in *Handbook of Mathematical Economics Volume 1*. Elsevier, 1981, pp. 53–91.

[10] Hunt, "Bayesian Methods in Nonlinear Digital Image Restoration," *IEEE Trans. Comput.*, vol. C-26, no. 3, pp. 219–229, Mar. 1977.

[11] A. Beck, *First-order methods in optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2017.

[12] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, no. 1-4, pp. 259–268, Nov. 1992.

[13] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images," *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, Feb. 2009.

[14] J. Mairal, "Sparse Modeling for Image and Vision Processing," *Found. Trends Comput. Graph. Vis.*, vol. 8, no. 2-3, pp. 85–283, 2014.

[15] P. L. Combettes and J.-C. Pesquet, "Proximal Splitting Methods in Signal Processing," in *Springer Optimization and Its Applications*. Springer New York, 2011, pp. 185–212.

[16] N. Parikh, "Proximal Algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[18] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic, "Signal denoising on graphs via graph filtering," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2014, pp. 872–876.

[19] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 106–128, 2013.

[20] O. Lézoray and L. Grady, *Image Processing and Analysis with Graphs: Theory and Practice*. CRC Press, 2012.

[21] F. R. K. Chung, *Spectral Graph Theory*, ser. CBMS Regional Conference Series. Conference Board of the Mathematical Sciences, 1997, no. 92.

[22] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, 2013.

[23] ——, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, 2014.

[24] ——, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, 2014.

[25] A. Gadde, S. K. Narang, and A. Ortega, "Bilateral filter: Graph spectral interpretation and extensions," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 1222–1226.

[26] M. Onuki and Y. Tanaka, "Trilateral filter on graph spectral domain," in *IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 2046–2050.

[27] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-Frequency Analysis on Graphs," *Appl. Comput. Harmon. Anal.*, vol. 40, no. 2, pp. 260–291, Mar. 2016.

[28] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, 2011.

[29] S. K. Narang and A. Ortega, "Perfect Reconstruction Two-Channel Wavelet Filter Banks for Graph Structured Data," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2786–2799, Jun. 2012.

[30] ——, "Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4673–4685, 2013.

[31] D. B. H. Tay and J. Zhang, "Techniques for Constructing Biorthogonal Bipartite Graph Filter Banks," *IEEE Trans. Signal Process.*, vol. 63, no. 21, pp. 5772–5783, 2015.

[32] O. Teke and P. P. Vaidyanathan, "Extending Classical Multirate Signal Processing Theory to Graphs;Part I: Fundamentals," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 409–422, Jan. 2017.

[33] ——, "Extending Classical Multirate Signal Processing Theory to Graphs;Part II: M-Channel Filter Banks," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 423–437, Jan. 2017.

[34] A. Sakiyama and Y. Tanaka, "Oversampled Graph Laplacian Matrix for Graph Filter Banks," *IEEE Trans. Signal Process.*, vol. 62, no. 24, pp. 6425–6437, Dec. 2014.

[35] N. Tremblay and P. Borgnat, "Subgraph-Based Filterbanks for Graph Signals," *IEEE Trans. Signal Process.*, vol. 64, no. 15, pp. 3827–3840, Aug. 2016.

[36] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning theory and kernel machines*. Springer, 2003, pp. 144–158.

[37] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.

[38] G. H. Golub and C. F. Van Loan, *Matrix Computations*, ser. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[39] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal representations on graphs: Tools and applications," 2015.

[40] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[41] N. J. Higham, *Functions of matrices: theory and computation*. Siam, 2008.

[42] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[43] N. C. D. Center, "Global surface summary of day data," ftp://ftp.ncdc.noaa.gov/pub/data/gsod/, 2003.

[44] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Simul.*, vol. 4, no. 2, pp. 490–530, 2005.

[45] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," Aug. 2014.

[46] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *6th International Conference on Computer Vision (ICCV)*. Narosa Publishing House, 1998.

[47] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graphics*, vol. 27, no. 3, pp. 1–10, Aug. 2008.

[48] R. Fattal, "Edge-avoiding wavelets and their applications," *ACM Trans. Graphics*, vol. 28, no. 3, Jul. 2009.

[49] K. He, J. Sun, and X. Tang, "Guided Image Filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.

[50] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L0 gradient minimization," *ACM Trans. Graphics*, vol. 30, no. 6, pp. 1–12, Dec. 2011.

[51] S. Bi, X. Han, and Y. Yu, "An L1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition," *ACM Trans. Graphics*, vol. 34, no. 4, pp. 1–12, Jul. 2015.

[52] L. Karacan, E. Erdem, and A. Erdem, "Structure-preserving image smoothing via region covariances," *ACM Trans. Graphics*, vol. 32, no. 6, pp. 1–11, Nov. 2013.

[53] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Trans. Graphics*, vol. 30, no. 4, Jul. 2011.

[54] S. Paris, S. W. Hasinoff, and J. Kautz, "Local Laplacian filters," *Commun. ACM*, vol. 58, no. 3, pp. 81–91, Feb. 2015.

[55] Q. Zhang, L. Xu, and J. Jia, "100+ Times Faster Weighted Median Filter (WMF)," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[56] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M. N. Do, "Fast Global Image Smoothing Based on Weighted Least Squares," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5638–5653, Dec. 2014.

[57] Y. Kim, D. Min, B. Ham, and K. Sohn, "Fast Domain Decomposition for Global Image Smoothing," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 4079–4091, Aug. 2017.

[58] J. T. Barron and B. Poole, "The Fast Bilateral Solver," in *Computer Vision – ECCV*. Springer International Publishing, 2016, pp. 617–632.

[59] S. Ono, "L0 gradient projection," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1554–1564, Apr. 2017.

[60] H. Yin, Y. Gong, and G. Qiu, "Side window guided filtering," *Signal Process.*, vol. 165, pp. 315–330, Dec. 2019.

[61] G. Deng, "Guided Wavelet Shrinkage for Edge-Aware Smoothing," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 900–914, Feb. 2017.

[62] V. Ćurić, A. Landström, M. J. Thurley, and C. L. L. Hendriks, "Adaptive mathematical morphology – A survey of the field," *Pattern Recognit. Lett.*, vol. 47, pp. 18–28, oct 2014.

[63] J. Debayle and J.-C. Pinoli, "General Adaptive Neighborhood Image Processing: Part I: Introduction and Theoretical Aspects," *J. Math. Imaging Vision*, vol. 25, no. 2, pp. 245–266, aug 2006.

[64] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud, "Deterministic edge-preserving regularization in computed imaging," *IEEE Trans. Image Process.*, vol. 6, no. 2, pp. 298–311, 1997.

[65] R. Fattal, M. Agrawala, and S. Rusinkiewicz, "Multiscale shape and detail enhancement from multi-light image collections," *ACM Trans. Graphics*, vol. 26, no. 3, Jul. 2007.

[66] Z. G. Li, J. H. Zheng, and S. Rahardja, "Detail-Enhanced Exposure Fusion," *IEEE Trans. Image Process.*, vol. 21, no. 11, pp. 4672–4676, Nov. 2012.

[67] J. Guan and W.-K. Cham, "Quality estimation based multi-focus image fusion," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017.

[68] M. Al-nasrawi, G. Deng, and B. Thai, "Edge-aware smoothing through adaptive interpolation," *Signal Image Video Process.*, vol. 12, no. 2, pp. 347–354, Aug. 2017.

[69] Z. Zhou, B. Wang, and J. Ma, "Scale-Aware Edge-Preserving Image Filtering via Iterative Global Optimization," *IEEE Trans. Multimedia*, vol. 20, no. 6, pp. 1392–1405, Jun. 2018.

[70] B. Cai, X. Xing, and X. Xu, "Edge/structure preserving smoothing via relativity-of-Gaussian," in *IEEE International Conference on Image Processing (ICIP)*, Sep. 2017.

[71] K. He, J. Sun, and X. Tang, "Single Image Haze Removal Using Dark Channel Prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2341–2353, Dec. 2011.

[72] H. Badri, H. Yahia, and D. Aboutajdine, "Fast Edge-Aware Processing via First Order Proximal Approximation," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 6, pp. 743–755, Jun. 2015.

[73] S. Li, K. Zhang, Q. Hao, P. Duan, and X. Kang, "Hyperspectral Anomaly Detection With Multiscale Attribute and Edge-Preserving Filters," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 10, pp. 1605–1609, Oct. 2018.

[74] J. Xia, L. Bombrun, T. Adali, Y. Berthoumieu, and C. Germain, "Spectral–Spatial Classification of Hyperspectral Images Using ICA and Edge-Preserving Filter via an Ensemble Strategy," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4971–4982, Aug. 2016.

[75] P. Milanfar, "A Tour of Modern Image Filtering: New Insights and Methods, Both Practical and Theoretical," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 106–128, Jan. 2013.

[76] H. Talebi and P. Milanfar, "Global Image Denoising," *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 755–768, Feb. 2014.

[77] B. Ham, M. Cho, and J. Ponce, "Robust Guided Image Filtering Using Nonconvex Potentials," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 1, pp. 192–207, Jan. 2018.

[78] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," *ACM Trans. Graphics*, vol. 23, no. 3, p. 664–672, Aug. 2004.

[79] Z. Li, J. Zheng, Z. Zhu, W. Yao, and S. Wu, "Weighted Guided Image Filtering," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 120–129, Jan. 2015.

[80] Z. Lu, B. Long, K. Li, and F. Lu, "Effective Guided Image Filtering for Contrast Enhancement," *IEEE Signal Process. Lett.*, vol. 25, no. 10, pp. 1585–1589, Oct. 2018.

[81] G. Deng, "Edge-Aware BMA Filters," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 439–454, Jan. 2016.

[82] E. Eisemann and F. Durand, "Flash photography enhancement via intrinsic relighting," *ACM Trans. Graphics*, vol. 23, no. 3, Aug. 2004.

[83] T. Qiu, A. Wang, N. Yu, and A. Song, "LLSURE: Local linear SURE-based edge-preserving image filtering," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 80–90, Jan. 2013.

[84] K. Lu, S. You, and N. Barnes, "Double-Guided Filtering: Image Smoothing with Structure and Texture Guidance," in *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, Nov. 2017.

[85] Z. Li and J. Zheng, "Single Image De-Hazing Using Globally Guided Image Filtering," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 442–450, Jan. 2018.

[86] L. Xu, Q. Yan, Y. Xia, and J. Jia, "Structure extraction from texture via relative total variation," *ACM Trans. Graphics*, vol. 31, no. 6, Nov. 2012.

[87] W. Liu, P. Zhang, Y. Lei, X. Huang, J. Yang, and I. Reid, "A Generalized Framework for Edge-preserving and Structure-preserving Image Smoothing."

[88] D. Krishnan, R. Fattal, and R. Szeliski, "Efficient Preconditioning of Laplacian Matrices for Computer Graphics," *ACM Trans. Graphics*, vol. 32, no. 4, Jul. 2013.

[89] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 257–266, Jul. 2002.

[90] J.-W. Han, J.-H. Kim, S.-H. Cheon, J.-O. Kim, and S.-J. Ko, "A novel image interpolation method using the bilateral filter," *IEEE Trans. Consum. Electron.*, vol. 56, no. 1, pp. 175–181, Feb. 2010.

[91] Q. Yang, S. Wang, and N. Ahuja, "Real-Time Specular Highlight Removal Using Bilateral Filtering," in *Computer Vision – ECCV*. Berlin, Heidelberg: Springer, 2010, pp. 87–100.

[92] B. Ham, M. Cho, and J. Ponce, "Robust image filtering using joint static and dynamic guidance," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[93] A. Buades, B. Coll, and J.-M. Morel, "A Non-Local Algorithm for Image Denoising," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[94] W. Wang, F. Li, and M. K. Ng, "Structural Similarity-Based Nonlocal Variational Models for Image Restoration," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4260–4272, Sep. 2019.

[95] Y. Gong, B. Liu, X. Hou, and G. Qiu, "Sub-window Box Filter," in *IEEE Visual Communications and Image Processing (VCIP)*, Dec. 2018.

[96] Q. Zhang, X. Shen, L. Xu, and J. Jia, "Rolling Guidance Filter," in *Computer Vision – ECCV*. Springer International Publishing, 2014, pp. 815–830.

[97] J.-P. Tarel and N. Hautiere, "Fast visibility restoration from a single color or gray level image," in *IEEE 12th International Conference on Computer Vision (ICCV)*, Sep. 2009.

[98] J.-P. Tarel, N. Hautiere, A. Cord, D. Gruyer, and H. Halmaoui, "Improved visibility of road scene images under heterogeneous fog," in *IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2010, pp. 478–485.

[99] B. Thai, M. Al-nasrawi, G. Deng, and Z. Su, "Semi-guided bilateral filter," *IET Image Proc.*, vol. 11, no. 7, pp. 512–521, Jul. 2017.

[100] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[101] K. He, J. Sun, and X. Tang, "Fast matting using large kernel matting Laplacian matrices," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 2165–2172.

[102] M. Burger and S. Osher, "A Guide to the TV Zoo," in *Level Set and PDE Based Reconstruction Methods in Imaging*. Springer International Publishing, 2013, pp. 1–70.

[103] K. Jalalzai, "Some Remarks on the Staircasing Phenomenon in Total Variation-Based Image Denoising," *J. Math. Imaging Vision*, vol. 54, no. 2, pp. 256–268, Oct. 2015.

[104] K. Bredies, K. Kunisch, and T. Pock, "Total Generalized Variation," *SIAM J. Imag. Sci.*, vol. 3, no. 3, pp. 492–526, Jan. 2010.

[105] I. W. Selesnick, H. L. Graber, D. S. Pfeil, and R. L. Barbour, "Simultaneous Low-Pass Filtering and Total Variation Denoising," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1109–1124, Mar. 2014.

[106] D. L. Donoho and I. M. Johnstone, "Adapting to Unknown Smoothness via Wavelet Shrinkage," *J. Am. Stat. Assoc.*, vol. 90, no. 432, pp. 1200–1224, Dec. 1995.

[107] I. M. Johnstone and B. W. Silverman, "Needles and straw in haystacks: Empirical Bayes estimates of possibly sparse sequences," *Ann. Stat.*, vol. 32, no. 4, pp. 1594–1649, Aug. 2004.

[108] S. Das, *Functional Fractional Calculus for System Identification and Controls*. Springer-Verlag GmbH, Sep. 2007.

[109] M. D. Ortigueira, *Fractional Calculus for Scientists and Engineers*. Springer-Verlag GmbH, Jun. 2011.

[110] P. K. Gilles Aubert, *Mathematical Problems in Image Processing*. Springer-Verlag GmbH, Nov. 2006.

[111] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, "Graph Spectral Image Processing," *Proc. IEEE*, vol. 106, no. 5, pp. 907–930, May 2018.

[112] S. G. Samko, A. A. Kilbas, and O. I. Marichev, *Fractional integrals and derivatives*. Taylor & Francis, 1993.

[113] J. T. Machado, V. Kiryakova, and F. Mainardi, "Recent history of fractional calculus," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 16, no. 3, pp. 1140–1153, 2011.

[114] A. Elwakil, "Fractional-Order Circuits and Systems: An Emerging Interdisciplinary Research Area," *IEEE Circuits Syst. Mag.*, vol. 10, no. 4, pp. 40–50, 2010.

[115] R. Magin, M. D. Ortigueira, I. Podlubny, and J. Trujillo, "On the fractional signals and systems," *Signal Process.*, vol. 91, no. 3, pp. 350–371, 2011.

[116] Q. Yang, D. Chen, T. Zhao, and Y. Chen, "Fractional calculus in image processing: a review," *Fract. Calc. Appl. Anal*, vol. 19, no. 5, pp. 1222–1249, Jan. 2016.

[117] S. Samko, "Fractional integration and differentiation of variable order: an overview," *Nonlinear Dyn.*, vol. 71, no. 4, pp. 653–662, Mar. 2013.

[118] C. Pozrikidis, *The Fractional Laplacian*. Chapman and Hall/CRC, 2016.

[119] M. Ortigueira, "An introduction to the fractional continuous-time linear systems: the 21st century systems," *IEEE Circuits Syst. Mag.*, vol. 8, no. 3, pp. 19–26, 2008.

[120] E. C. de Oliveira and J. A. T. Machado, "A Review of Definitions for Fractional Derivatives and Integral," *Math. Probl. Eng.*, vol. 2014, pp. 1–6, 2014.

[121] J. L. Vázquez, "The Mathematical Theories of Diffusion: Nonlinear and Fractional Diffusion," in *Nonlocal and Nonlinear Diffusions and Interactions: New Methods and Directions*. Springer International Publishing, 2017, pp. 205–278.

[122] G. Gilboa and S. Osher, "Nonlocal Operators with Applications to Image Processing," *Multiscale Model. Simul.*, vol. 7, no. 3, pp. 1005–1028, Jan. 2009.

[123] A. Lischke, G. Pang, M. Gulian, F. Song, C. Glusa, X. Zheng, Z. Mao, W. Cai, M. M. Meerschaert, M. Ainsworth, and G. E. Karniadakis, "What is the fractional Laplacian? A comparative review with new results," *J. Comput. Phys.*, vol. 404, 2020.

[124] B. Mathieu, P. Melchior, A. Oustaloup, and C. Ceyral, "Fractional differentiation for edge detection," *Signal Process.*, vol. 83, no. 11, pp. 2421–2432, 2003.

[125] Y. Pu, W. Wang, J. Zhou, Y. Wang, and H. Jia, "Fractional differential approach to detecting textural features of digital image and its fractional differential filter implementation," *Science in China Series F: Information Sciences*, vol. 51, no. 9, pp. 1319–1339, Aug. 2008.

[126] C. B. Gao, J. L. Zhou, J. R. Hu, and F. N. Lang, "Edge detection of colour image based on quaternion fractional differential," *IET Image Proc.*, vol. 5, no. 3, pp. 261–272, Apr. 2011.

[127] W. X. Wang, W. S. Li, and X. Yu, "Fractional differential algorithms for rock fracture images," *Imaging Sci. J.*, vol. 60, no. 2, pp. 103–111, Apr. 2012.

[128] J. Chen, C. Huang, Y. Du, and C. Lin, "Combining fractional-order edge detection and chaos synchronisation classifier for fingerprint identification," *IET Image Proc.*, vol. 8, no. 6, pp. 354–362, Jun. 2014.

[129] C. Chi and F. Gao, "Palm Print Edge Extraction Using Fractional Differential Algorithm," *J. Appl. Math.*, vol. 2014, no. 2014, 2014.

[130] M. Ö. Arısoy and Ü. Dikmen, "Edge enhancement of magnetic data using fractional-order-derivative filters," *Geophysics*, vol. 80, no. 1, pp. J7–J17, Jan. 2015.

[131] A. Acharya, U. Mukherjee, and C. Fowlkes, "On Image segmentation using Fractional Gradients-Learning Model Parameters using Approximate Marginal Inference," May 2016.

[132] P. Amoako-Yirenkyi, J. K. Appati, and I. K. Dontwi, "A new construction of a fractional derivative mask for image edge analysis based on Riemann-Liouville fractional derivative," *Adv. Differ. Equ.*, vol. 2016, no. 1, p. 238, Sep. 2016.

[133] S. Kumar, R. Saxena, and K. Singh, "Fractional Fourier Transform and Fractional-Order Calculus-Based Image Edge Detection," *Circuits, Syst. Signal Process.*, vol. 36, no. 4, pp. 1493–1513, Jul. 2016.

[134] C. Georgescu, "Improved Edge Detection Algorithms Based on a Riesz Fractional Derivative," in *Image Analysis and Recognition (ICIAR)*. Cham: Springer International Publishing, 2018, pp. 201–209.

[135] A. Nandal, H. Gamboa-Rosales, A. Dhaka, J. M. Celaya-Padilla, J. I. Galvan-Tejada, C. E. Galvan-Tejada, F. J. Martinez-Ruiz, and C. Guzman-Valdivia, "Image Edge Detection Using Fractional Calculus with Feature and Contrast Enhancement," *Circuits, Syst. Signal Process.*, vol. 37, no. 9, pp. 3946–3972, Jan. 2018.

[136] J. E. Lavín-Delgado, J. E. Solís-Pérez, J. F. Gómez-Aguilar, and R. F. Escobar-Jiménez, "A New Fractional-Order Mask for Image Edge Detection Based on Caputo–Fabrizio Fractional-Order Derivative Without Singular Kernel," *Circuits, Syst. Signal Process.*, Jul. 2019.

[137] R. Abi Zeid Daou, F. El Samarani, C. Yaacoub, and X. Moreau, *Fractional Derivatives for Edge Detection: Application to Road Obstacles*. Cham: Springer International Publishing, 2020, pp. 115–137.

[138] J. You, S. Hungnahally, and A. Sattar, "Fractional discrimination for texture image segmentation," in *Proceedings of International Conference on Image Processing (ICIP)*, vol. 1, Oct. 1997, pp. 220–223.

[139] Y. Pu, "Fractional Calculus Approach to Texture of Digital Image," in *8th International Conference on Signal Processing (ICSP)*, vol. 2, Nov. 2006.

[140] Y. Pu, J. Zhou, and X. Yuan, "Fractional Differential Mask: A Fractional Differential-Based Approach for Multiscale Texture Enhancement," *IEEE Trans. Image Process.*, vol. 19, no. 2, pp. 491–511, Feb. 2010.

[141] M. Xu, J. Yang, D. Zhao, and H. Zhao, "An image-enhancement method based on variable-order fractional differential operators," *Biomed. Mater. Eng.*, vol. 26, no. s1, pp. S1325–S1333, Aug. 2015.

[142] S. Hemalatha and S. M. Anouncia, "G-L fractional differential operator modified using auto-correlation function: Texture enhancement in images," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 1689–1704, 2018.

[143] X. Luo, T. Zeng, W. Zeng, and J. Huang, "Comparative analysis on landsat image enhancement using fractional and integral differential operators," *Computing*, vol. 102, no. 1, pp. 247–261, Aug. 2019.

[144] S. Khanna and V. Chandrasekaran, "Fractional derivative filter for image contrast enhancement with order prediction," in *IET Conference on Image Processing (IPR 2012)*, Jul. 2012, pp. 1–6.

[145] K. Kaur, N. Jindal, and K. Singh, "Improved homomorphic filtering using fractional derivatives for enhancement of low contrast and non-uniformly illuminated images," *Multimed. Tools. Appl.*, vol. 78, no. 19, pp. 27 891–27 914, Jun. 2019.

[146] Q. Dai, Y.-F. Pu, Z. Rahman, and M. Aamir, "Fractional-Order Fusion Model for Low-Light Image Enhancement," *Symmetry*, vol. 11, no. 4, p. 574, Apr. 2019.

[147] J. Bai and X. Feng, "Fractional-Order Anisotropic Diffusion for Image Denoising," *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2492–2502, Oct. 2007.

[148] M. Janev, S. Pilipović, T. Atanacković, R. Obradović, and N. Ralević, "Fully fractional anisotropic diffusion for image denoising," *Math. Comput. Modell.*, vol. 54, no. 1-2, pp. 729–741, Jul. 2011.

[149] E. Cuesta, M. Kirane, and S. A. Malik, "Image structure preserving denoising using generalized fractional time integrals," *Signal Process.*, vol. 92, no. 2, pp. 553–563, 2012.

[150] S. Larnier and R. Mecca, "Fractional-order diffusion for image reconstruction," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2012.

[151] Y.-F. Pu, P. Siarry, J.-L. Zhou, and N. Zhang, "A fractional partial differential equation based multiscale denoising model for texture image," *Math. Methods Appl. Sci.*, vol. 37, no. 12, pp. 1784–1806, Sep. 2014.

[152] Y.-S. Zhang, F. Zhang, B.-Z. Li, and R. Tao, "Fractional domain varying-order differential denoising method," *Opt. Eng.*, vol. 53, no. 10, pp. 1–8, Apr. 2014.

[153] X. Yin, S. Zhou, and M. A. Siddique, "Fractional nonlinear anisotropic diffusion with p-Laplace variation method for image restoration," *Multimed. Tools. Appl.*, vol. 75, no. 8, pp. 4505–4526, Feb. 2015.

[154] W. Zhang, J. Li, and Y. Yang, "Spatial fractional telegraph equation for image structure preserving denoising," *Signal Process.*, vol. 107, pp. 368–377, Feb. 2015.

[155] Y.-F. Pu, N. Zhang, Y. Zhang, and J.-L. Zhou, "A texture image denoising approach based on fractional developmental mathematics," *Pattern Anal. Appl.*, vol. 19, no. 2, pp. 427–445, May 2015.

[156] Y. Pu, P. Siarry, A. Chatterjee, Z. Wang, Z. Yi, Y. Liu, J. Zhou, and Y. Wang, "A Fractional-Order Variational Framework for Retinex: Fractional-Order Partial Differential Equation-Based Formulation for Multi-Scale Nonlocal Contrast Enhancement with Texture Preserving," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1214–1229, Mar. 2018.

[157] Y. Pu, N. Zhang, Z. Wang, J. Wang, Z. Yi, Y. Wang, and J. Zhou, "Fractional-Order Retinex for Adaptive Contrast Enhancement of Under-Exposed Traffic Images," *IEEE Intell. Transp. Syst. Mag.*, pp. 1–1, 2019.

[158] Z. Guo, W. Yao, J. Sun, and B. Wu, "Nonlinear fractional diffusion model for deblurring images with textures," *Inverse Probl. Imaging*, vol. 13, no. 6, pp. 1161–1188, 2019.

[159] Z. Ren, C. He, and M. Li, "Fractional-order bidirectional diffusion for image up-sampling," *J. Electron. Imaging*, vol. 21, no. 2, pp. 1–13, 2012.

[160] X. Yin, S. Chen, L. Wang, and S. Zhou, "Fractional-order difference curvature-driven fractional anisotropic diffusion equation for image super-resolution," *Int. J. Model. Simul. Sci. Comput.*, vol. 10, no. 01, p. 1941012, Feb. 2019.

[161] J. Zhang and K. Chen, "A Total Fractional-Order Variation Model for Image Restoration with Nonhomogeneous Boundary Conditions and Its Numerical Solution," *SIAM J. Imag. Sci.*, vol. 8, no. 4, pp. 2487–2518, 2015.

[162] H. Li, Z. Yu, and C. Mao, "Fractional differential and variational method for image fusion and super-resolution," *Neurocomputing*, vol. 171, pp. 138–148, Jan. 2016.

[163] A. Ullah, W. Chen, and M. A. Khan, "A new variational approach for restoring images with multiplicative noise," *Comput. Math. Appl.*, vol. 71, no. 10, pp. 2034–2050, May 2016.

[164] Y. Pu, "Fractional-Order Euler-Lagrange Equation for Fractional-Order Variational Method: A Necessary Condition for Fractional-Order Fixed Boundary Optimization Problems in Signal Processing and Image Processing," *IEEE Access*, vol. 4, pp. 10 110–10 135, 2016.

[165] A. Abirami, P. Prakash, and K. Thangavel, "Fractional diffusion equation-based image denoising model using CN–GL scheme," *Int. J. Comput. Math.*, vol. 95, no. 6-7, pp. 1222–1239, Nov. 2017.

[166] M. Chen, Y.-F. Pu, and Y.-C. Bai, "A Fractional-Order Variational Residual CNN for Low Dose CT Image Denoising," in *Intelligent Computing Theories and Application (ICIC)*, D.-S. Huang, V. Bevilacqua, and P. Premaratne, Eds. Cham: Springer International Publishing, Jul. 2019, pp. 238–249.

[167] X. Jia, S. Liu, X. Feng, and L. Zhang, "FOCNet: A Fractional Optimal Control Network for Image Denoising," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern. Recognit. (CVPR)*, Jun. 2019, pp. 6047–6056.

[168] R. H. Chan and H.-X. Liang, "Truncated Fractional-Order Total Variation Model for Image Restoration," *J. Oper. Res. Soc. China*, vol. 7, no. 4, pp. 561–578, Dec. 2019.

[169] A. Kumar, M. O. Ahmad, and M. N. S. Swamy, "A Framework for Image Denoising Using First and Second Order Fractional Overlapping Group Sparsity (HF-OLGS) Regularizer," *IEEE Access*, vol. 7, pp. 26 200–26 217, 2019.

[170] J.-J. Mei, Y. Dong, and T.-Z. Huang, "Simultaneous image fusion and denoising by using fractional-order gradient information," *J. Comput. Appl. Math.*, vol. 351, pp. 212–227, 2019.

[171] Q. Wang, J. Ma, S. Yu, and L. Tan, "Noise detection and image denoising based on fractional calculus," *Chaos Soliton. Fract.*, p. 109463, Oct. 2019.

[172] M. R. Chowdhury, J. Zhang, J. Qin, and Y. Lou, "Poisson image denoising based on fractional-order total variation," *Inverse Probl. Imaging*, vol. 14, no. 1, pp. 77–96, 2020.

[173] X. Yang and B. Guo, "Fractional-order tensor regularisation for image inpainting," *IET Image Proc.*, vol. 11, no. 9, pp. 734–745, Sep. 2017.

[174] J. Zhang and K. Chen, "Variational image registration by a total fractional-order variation model," *J. Comput. Phys.*, vol. 293, pp. 442–461, 2015.

[175] S. G. Bardeji, I. N. Figueiredo, and E. Sousa, "Optical flow with fractional order regularization: Variational model and solution method," *Appl. Numer. Math.*, vol. 114, pp. 188–200, Apr. 2017.

[176] P. Ghamisi, M. S. Couceiro, J. A. Benediktsson, and N. M. F. Ferreira, "An efficient method for segmentation of images based on fractional calculus and natural selection," *Expert Syst. Appl.*, vol. 39, no. 16, pp. 12 407–12 417, Nov. 2012.

[177] P. Ghamisi, M. S. Couceiro, F. M. L. Martins, and J. A. Benediktsson, "Multilevel Image Segmentation Based on Fractional-Order Darwinian Particle Swarm Optimization," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 5, pp. 2382–2394, May 2014.

[178] F. Guo, H. Peng, B. Zou, R. Zhao, and X. Liu, "Localisation and segmentation of optic disc with the fractional-order Darwinian particle swarm optimisation algorithm," *IET Image Proc.*, vol. 12, no. 8, pp. 1303–1312, Aug. 2018.

[179] S. Didas, B. Burgeth, A. Imiya, and J. Weickert, "Regularity and Scale-Space Properties of Fractional High Order Linear Filtering," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 13–25.

[180] P. D. Romero and V. F. Candela, "Blind Deconvolution Models Regularized by Fractional Powers of the Laplacian," *J. Math. Imaging Vision*, vol. 32, no. 2, pp. 181–191, May 2008.

[181] P. D. Tafti, R. Delgado-Gonzalo, A. F. Stalder, and M. Unser, "Fractal modelling and analysis of flow-field images," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, Apr. 2010.

[182] P. Gatto and J. S. Hesthaven, "Numerical Approximation of the Fractional Laplacian via hp-finite Elements, with an Application to Image Denoising," *J. Sci. Comput.*, vol. 65, no. 1, pp. 249–270, Oct. 2015.

[183] H. Antil and S. Bartels, "Spectral Approximation of Fractional PDEs in Image Processing and Phase Field Modeling," *Comput. Meth. Appl. Mat.*, vol. 17, no. 4, Jan. 2017.

[184] H. Antil, Z. Di, and R. Khatri, "Bilevel Optimization, Deep Learning and Fractional Laplacian Regularization with Applications in Tomography," *Inverse Prob.*, Mar. 2020.

[185] B. Maundy, A. S. Elwakil, and T. J. Freeborn, "On the practical realization of higher-order filters with fractional stepping," *Signal Process.*, vol. 91, no. 3, pp. 484–491, Mar. 2011.

[186] T. J. Freeborn, B. Maundy, and A. Elwakil, "Second order approximation of the fractional Laplacian operator for equal-ripple response," in *53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2010.

[187] A. M. AbdelAty, A. S. Elwakil, A. G. Radwan, C. Psychalinos, and B. J. Maundy, "Approximation of the Fractional-Order Laplacian S^{α} As a Weighted Sum of First-Order High-Pass Filters," *IEEE Trans. Circuits Syst. II*, vol. 65, no. 8, pp. 1114–1118, Aug. 2018.

[188] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge Press,U.S., 1996.

[189] G. Strang, "The Discrete Cosine Transform," *SIAM Rev.*, vol. 41, no. 1, pp. 135–147, 1999.

[190] V. Britanak, P. C. Yip, and K. R. Rao, "Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations," in *Discrete Cosine and Sine Transforms*, V. Britanak, P. C. Yip, and K. R. Rao, Eds. Academic Press, 2006, pp. 16–50.

[191] G. Deng, "Fast algorithm for zero-phase linear filter using discrete cosine transform," *Electron. Lett.*, vol. 55, no. 10, pp. 621–623, May 2019.

[192] C.-C. Tseng and S.-L. Lee, "Computation of partial fractional derivative of digital image using discrete cosine transform," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2013, pp. 2828–2831.

[193] C.-C. Tseng, S.-C. Pei, and S.-C. Hsia, "Computation of fractional derivatives using Fourier transform and digital FIR differentiator," *Signal Process.*, vol. 80, no. 1, pp. 151–159, Jan. 2000.

[194] M. Kumar and T. K. Rawat, "Design of fractional order differentiator using type-III and type-IV discrete cosine transform," *Eng. Sci. Technol.*, vol. 20, no. 1, pp. 51–58, Feb. 2017.

[195] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "ℓ_1 Trend Filtering," *SIAM Rev.*, vol. 51, no. 2, pp. 339–360, 2009.

[196] B. Bruder, T.-L. Dao, J.-C. Richard, and T. Roncalli, "Trend Filtering Methods for Momentum Strategies," *SSRN Electronic Journal*, 2011.

[197] R. J. Hodrick and E. C. Prescott, "Postwar U.S. Business Cycles: An Empirical Investigation," *J. Money Credit Bank.*, vol. 29, no. 1, pp. 1–16, Feb. 1997.

[198] W. K. Pratt, "Generalized Wiener Filtering Computation Techniques," *IEEE Trans. Comput.*, vol. C-21, no. 7, pp. 636–641, Jul. 1972.

[199] B. Fischer and J. Modersitzki, "Curvature Based Image Registration," *J. Math. Imaging Vision*, vol. 18, no. 1, pp. 81–85, 2003.

[200] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[201] K. He, J. Sun, and X. Tang, "Guided Image Filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.

[202] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Soc. B*, vol. 207, no. 1167, pp. 187–217, Feb. 1980.

[203] R. Gonzalez, *Digital image processing*. New York, NY: Pearson, 2018.

[204] S. Osher and L. I. Rudin, "Feature-Oriented Image Enhancement Using Shock Filters," *SIAM J. Numer. Anal.*, vol. 27, no. 4, pp. 919–940, 1990.

[205] R. Duits, L. Florack, J. de Graaf, and B. ter Haar Romeny, "On the Axioms of Scale Space Theory," *J. Math. Imaging Vision*, vol. 20, no. 3, pp. 267–298, May 2004.

[206] A. Lanza, S. Morigi, I. Selesnick, and F. Sgallari, "Nonconvex nonsmooth optimization via convex–nonconvex majorization–minimization," *Numer. Math.*, vol. 136, no. 2, pp. 343–381, Oct. 2016.

[207] A. Lanza, S. Morigi, I. W. Selesnick, and F. Sgallari, "Sparsity-Inducing Nonconvex Nonseparable Regularization for Convex Image Processing," *SIAM J. Imag. Sci.*, vol. 12, no. 2, pp. 1099–1134, Jan. 2019.

[208] X. Guo, Y. Li, J. Ma, and H. Ling, "Mutually Guided Image Filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 694–707, Mar. 2020.

[209] I. Selesnick, A. Lanza, S. Morigi, and F. Sgallari, "Non-convex Total Variation Regularization for Convex Denoising of Signals," *J. Math. Imaging Vision*, Jan. 2020.

[210] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep Convolutional Neural Network for Inverse Problems in Imaging," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4509–4522, Sep. 2017.

[211] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep Image Prior," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[212] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, "Solving inverse problems using data-driven models," *Acta Numerica*, vol. 28, pp. 1–174, May 2019.