UNIVERSITY OF CALIFORNIA, SAN DIEGO

Efficient Methods for Image Denoising using Learned Patch Priors

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Signal and Image Processing)

by

Shibin Parameswaran

Committee in charge:

Professor Truong Nguyen, Chair
Professor Pamela Cosman
Professor Lawrence Saul
Professor Mohan Trivedi
Professor Zhuowen Tu

2018

The dissertation of Shibin Parameswaran is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2018

DEDICATION

To my brother - for giving new meaning to my life.

To my Bee and my Bean - who are my world.

To Dad, Mom, Nani and Bruzzer - whose love is my strength.

To Achan and Amma - for all their sacrifices in making me who I am.

EPIGRAPH

The solutions all are simple — after you have arrived at them.

— Robert M. Pirsig, Zen and the Art of Motorcycle Maintenance

TABLE OF CONTENTS

Signature Page . . . iii

Dedication . . . iv

Epigraph . . . v

Table of Contents . . . vi

List of Figures . . . ix

List of Tables . . . xi

Acknowledgements . . . xii

Vita . . . xvi

Abstract of the Dissertation . . . xviii

Chapter 1 Introduction . . . 1
1.1 Patch-based image denoising . . . 1
1.2 Role of priors in denoising . . . 3
1.3 Thesis overview . . . 4

Chapter 2 Background . . . 7
2.1 Image denoising . . . 8
2.2 From patch-wise to whole image denoising . . . 9
2.3 Overview of patch priors . . . 12

Chapter 3 Patch Matching for Image Denoising Using Neighborhood-based Collaborative Filtering . . . 17
3.1 Introduction . . . 17
3.2 Background . . . 19
3.2.1 Targeted Database and Targeted Image Denoising . . . 19
3.2.2 BM3D and Its Collaborative Filtering . . . 20
3.2.3 External BM3D . . . 21
3.2.4 External LPG-PCA . . . 21
3.2.5 Patch Matching . . . 22
3.2.6 Neighborhood-based Collaborative Filtering . . . 23
3.3 Proposed Method . . . 24
3.3.1 Patch Matching Criterion . . . 25
3.3.2 Finding the Set of *Close* Neighbors . . . 25
3.3.3 Choice of Weights . . . 27

3.4 Experimental Results . . . 28
3.4.1 Text Denoising . . . 29
3.4.2 Face Image Denoising . . . 32
3.5 Discussion . . . 33
3.6 Conclusion . . . 37

Chapter 4 Targeted Video Denoising for Decompressed Videos . . . 38
4.1 Introduction . . . 38
4.2 Related Work . . . 40
4.2.1 Establishing temporal coherence in video denoising . . . 40
4.2.2 Targeted Image Denoising . . . 40
4.3 Targeted Video Denoising . . . 41
4.4 Experimental Results . . . 44
4.5 Conclusion . . . 47

Chapter 5 Fast external denoising using pre-learned transformations . . . 48
5.1 Introduction . . . 48
5.2 Background and Related Work . . . 49
5.2.1 Targeted Image Denoising Filter . . . 49
5.2.2 Expected Patch Log Likelihood . . . 49
5.2.3 Generic vs. targeted database of patches . . . 50
5.3 Proposed Method . . . 51
5.3.1 Whole image denoising . . . 51
5.3.2 Patch denoising . . . 52
5.3.3 Offline training and iterations . . . 54
5.4 Experimental Results . . . 56
5.4.1 Text denoising . . . 59
5.4.2 Face image denoising . . . 63
5.4.3 License plate denoising . . . 63
5.4.4 Generic image denoising on BSDS dataset . . . 64
5.5 Conclusion . . . 64

Chapter 6 Accelerating GMM-based patch priors for image restoration . . . 66
6.1 Introduction . . . 66
6.2 Expected Patch Log-Likelihood (EPLL) . . . 69
6.2.1 Complexity via eigenspace implementation . . . 72
6.2.2 Computation time analysis . . . 74
6.3 Fast EPLL: the three key ingredients . . . 74
6.3.1 Speed-up via flat tail spectrum approximation . . . 75
6.3.2 Speed-up via a balanced search tree . . . 77
6.3.3 Speed-up via the restriction to a random subset of patches . . . 79
6.3.4 Performance analysis . . . 81
6.4 Related methods . . . 83

6.5 Numerical experiments . . . 86
6.6 Conclusion . . . 92

Chapter 7 Image denoising with generalized Gaussian mixture model patch priors . 94
7.1 Introduction . . . 94
7.2 Background . . . 99
7.2.1 Image restoration with patch based priors . . . 99
7.2.2 Patch denoising with GMM priors . . . 102
7.3 Generalized Gaussian Mixture Models . . . 104
7.3.1 Learning GGMMs . . . 106
7.3.2 Patch denoising with GGMM priors . . . 109
7.4 Discrepancy function: analysis and approximations . . . 112
7.4.1 Theoretical analysis . . . 114
7.4.2 Numerical approximation . . . 120
7.5 Shrinkage functions: analysis and approximations . . . 125
7.5.1 Theoretical analysis . . . 125
7.5.2 Numerical approximations . . . 126
7.6 Experimental evaluation . . . 128
7.7 Conclusions and Discussion . . . 138

Chapter 8 Conclusion . . . 141
8.1 Summary of contributions . . . 141
8.2 Future work . . . 142

Appendix A Additional analysis and results of FEPLL algorithm . . . 144
A.1 Flat tail approximation based acceleration . . . 144
A.2 Search tree acceleration . . . 145
A.3 Stochastic patch sub-sampling . . . 146
A.4 Additional results . . . 148

Appendix B Proofs for Theorems in Chapter 7 . . . 153
B.1 Proof of equation (7.23) . . . 153
B.2 Proof of Proposition 7.4.2 . . . 154
B.3 Proof of Theorem 7.4.3 . . . 154
B.4 Proof of Theorem 7.4.4 . . . 157
B.5 Proof of Theorem 7.4.5 . . . 158
B.6 Proof of Theorem 7.4.6 . . . 159
B.7 Proof of Proposition 7.5.1 . . . 162

Bibliography . . . 166

LIST OF FIGURES

Figure 2.1: Illustrations of zero-mean generalized Gaussian distributions for different values of the shape parameter . . . 13
Figure 2.2: Illustrations of the iso-lines of four two-dimensional priors based on sparse analysis regularization . . . 14
Figure 2.3: Illustrations of the iso-lines of three two-dimensional zero-mean Gaussian distributions and their mixture . . . 15

Figure 3.1: Sub-optimal patch matching leads to sub-optimal results . . . 18
Figure 3.2: Collaborative filtering using close neighbors . . . 26
Figure 3.3: Sample images from text image dataset . . . 30
Figure 3.4: Visual and objective comparison of denoising performance of a text image 31
Figure 3.5: Sample images from FEI face image dataset . . . 33
Figure 3.6: Visual and objective comparison of denoising performance of a face image 35

Figure 4.1: Performance comparison on **Miss America** sequence (First 60 frames) . . 42
Figure 4.2: Frame 5 of the **Miss America** sequence . . . 42
Figure 4.3: Performance comparison on **Coastguard** sequence (First 60 frames) . . . 43
Figure 4.4: Performance comparison on **Diving** sequence (55 frames) . . . 43

Figure 5.1: The effect of size of patch matrix, number of anchors, and weighting of the patch matrix . . . 57
Figure 5.2: Sample images from text image dataset . . . 58
Figure 5.3: Sample images from face image dataset . . . 58
Figure 5.4: Visual and objective comparison of denoising performance of a face image 60
Figure 5.5: Sample images from the license plate dataset . . . 60
Figure 5.6: Visual and objective comparison of denoising performance of the same license image under different noise levels . . . 61
Figure 5.7: Visual and objective comparison of denoising performance of one of the BSDS test images using a ***generic*** database created from BSDS training set 62

Figure 6.1: Flat tail approximation: (a) with eigenvalues display on linear and (b) logarithmic scale . . . 75
Figure 6.2: Balanced search tree: (left) Our obtained search tree (with numbers of nodes for each level) . . . 77
Figure 6.3: Illustration of patch subsampling . . . 80
Figure 6.4: (top) Average speed-up and (bottom) average PSNR for our three accelerations, and all possible combinations of them, on the 40 images of the BSDS validation set. . . . 82
Figure 6.5: Illustration of a denoising problem with noise standard deviation $\sigma = 20$ . . 88
Figure 6.6: (a) PSNR and (b) SSIM versus time for different restoration methods in a denoising problem with noise standard deviation $\sigma = 20$ . . . 89
Figure 6.7: Illustration of a deblurring problem with noise standard deviation $\sigma = 0.5$. 90

Figure 6.8: FEPLL on various inverse problems 93

Figure 7.1: Histograms of the projection of $200,000$ clean patches on 6 eigenvectors $j = 1,2,3,4,62$ and 63 of the covariance matrix of one component k of the mixture 97
Figure 7.2: Illustration of EPLL framework for image denoising with a GMM prior . . 104
Figure 7.3: Illustration of our extension of EPLL to GGMM priors 111
Figure 7.4: Convolution of two Gaussian distributions 115
Figure 7.5: Convolution of Laplacian with Gaussian 116
Figure 7.6: Convolution of generalized Gaussian with Gaussian 118
Figure 7.7: Illustrations of the log-discrepancy function 120
Figure 7.8: Illustrations of our approximations of φ_λ^ν and the corresponding underlying posterior distribution $\mathcal{N}(0,\nu,\lambda) * \mathcal{G}(0,1)$ (where $\nu = .8$ and $\lambda = 4$) 122
Figure 7.9: Lookup tables used to store the values of the parameters γ_λ^ν, β_1, β_2 and h . . 123
Figure 7.10: Illustrations of the shrinkage function 124
Figure 7.11: Set of 100 patches, sorted by the norm of their gradient, and generated to be independently distributed according to (from left to right) a GMM, GGMM, LMM and HLMM 129
Figure 7.12: Average log-likelihood of all non-overlapping patches (with subtracted mean) of each of the 40 images of our validation subset of the testing BSDS dataset for the GMM, GGMM, LMM and HLMM 130
Figure 7.13: Qualitative comparison on *Castle* 133
Figure 7.14: Qualitative comparison on *Cameraman* 134
Figure 7.15: Qualitative comparison on *Barbara* 135
Figure 7.16: Evolution of performance of EPLL with a GGMM, LMM ($\nu = 1$) and a GMM ($\nu = 2$) under misspecification of the noise standard deviation σ . . 136
Figure 7.17: Influence of our approximations on denoising performance 138

Figure A.1: Naïve approach *vs.* the proposed flat-tail approximation 145
Figure A.2: Influence of the parameter ρ on denoising performance 146
Figure A.3: Influence of clustering techniques on the search tree and the results 147
Figure A.4: Timing scatter obtained over 5 runs for 40 images using Gaussian trees built using different clustering methods we tested 148
Figure A.5: Influence of random patch subsampling parameter s on performance 149
Figure A.6: Denoising results where $\sigma = 20$ obtained on five standard images, from top to bottom: Barbara, Boat, Couple, Fingerprint, Lena and Mandrill. 150
Figure A.7: Illustration of a deblurring problem of a Gaussian convolution of width 3 pixels with a noise of standard deviation $\sigma = 2$ 151
Figure A.8: Illustration of a super-resolution by a factor $\times 3$ with a noise of standard deviation $\sigma = 2$ 152

LIST OF TABLES

Table 3.1: Text image results from TID 30
Table 3.2: Text image results from external BM3D and external LPG-PCA 32
Table 3.3: FEI face dataset results from TID 34
Table 3.4: FEI face dataset results from external-BM3D and external-LPG-PCA 34

Table 5.1: Comparison of average PSNR, SSIM and time taken by each of the algorithms to denoise a text image of size 107×104 pixels 59
Table 5.2: Same as Table 5.1 but for 10 face images of size 90×65 from FEI face dataset 59
Table 5.3: Same as Table 5.1 but for 10 license images of average size 44×92 cropped from the Caltech cars (1999) dataset 61
Table 5.4: Same as Table 5.1 but for 100 test images of size 321×481 of the BSDS dataset 62

Table 6.1: Comparison of the execution time of our implementation of EPLL with and without proposed accelerations 71
Table 6.2: Denoing performance and timing comparison 84
Table 6.3: Deblurring performance and timing comparison 92

Table 7.1: Shrinkage function under generalized Gaussian priors 127
Table 7.2: Image denoising performance comparison of EPLL algorithm with GMM and GGMM priors 131
Table 7.3: Image denoising performance comparison of EPLL algorithm with different priors 132
Table 7.4: Runtime profiles (averaged over ten runs) of GGMM-EPLL corresponding to the denoising experiment shown in Figure 7.17 139

ACKNOWLEDGEMENTS

I thank my thesis advisor, Professor Truong Nguyen, for his support and guidance during the course of this journey. Professor Nguyen afforded me the space and freedom to pursue my goals while stepping in at the right times. I admire his student-centered approach to training and mentorship which I immensely benefited from. Under his watchful direction, I was able to identify and hone in on a thesis topic and contribute to the field to my fullest ability. Professor Nguyen's respect for a student's circumstances is one of the main reasons for my success in this pursuit while effectively balancing work commitments and a young family. I am grateful to Professor Nguyen for giving me a place in his lab and calmly leading me to my academic goal.

The last half of this journey was made even more intellectually stimulating and enjoyable through my collaboration with Dr. Charles-Alban Deledalle. I am grateful for the countless hours we spent together discussing and working on ideas. During the course of our collaboration, along with gaining technical knowledge and getting inspired to make beautiful illustrations, I also learned important lessons on work-life balance. I thank Charles for his friendship and advice.

I am thankful to my committee members Professor Pamela Cosman, Professor Lawrence Saul, Professor Mohan Trivedi and Professor Zhouwen Tu for their guidance and support over the years. I am grateful to Professor Tu's insightful comments during my qualification exam and our discussions thereafter. His suggestions will be especially helpful to navigate the ever-changing topology of machine learning with success and continue impactful research in the future. Before Professor Cosman was my thesis committee member, I worked with her as her teaching assistant. I learned finer points of teaching and mentoring while working with her. I have had the pleasure of interacting with Professor Trivedi both in professional and personal capacities. Through our interactions, I found a role model for academic success and family.

Having completed my Master's degree under Professor Lawrence Saul, it is safe to say he taught me a major portion of what I know about machine learning and conducting research. It is under him that I learned to interpret results, organize the findings and communicate clearly. He

made me read the article “The science of scientific writing” [1] which helped and continues to help my technical writing immensely. I am thankful to Lawrence and the then Saul-lab members Diane Hu, Chih-Chieh (Jie) Cheng, Laurens van der Maaten, Justin Ma and Youngmin Cho for their support during the first part of my graduate training. Importantly, it is through Lawrence that I met Dr. Kilian Weinberger with whom I published my first major paper. This immensely productive collaboration was instrumental to my development as a researcher. I am ever-so grateful to Kilian for providing me with this mentorship and self-confidence during the most critical point of my graduate tenure.

Thanks to the members of the Video Processing Lab for providing numerous suggestions that have made my thesis research stronger. A special thanks goes to Enming Luo, who took me under his wing when I first joined the lab, and to Subarna Tripathi, Byeong Kang, Yeejin Lee and Ji Dai for their support and company. I appreciate them for being there to offer advice, to agree with me about the "absurdity" of the reviewer comments that led to my paper rejections and to share the many cups of coffee during my time in the lab.

I am grateful for the SMART Scholarship program that enabled my return to graduate school following an extended leave of absence. A special thanks to the Director of the SMART program, Ms. Chris Deckard, for her mentorship, direction and support over the course of my studies. I am thankful for the support of my colleagues and mentors at SSC Pacific – especially my SMART mentor, Dr. Sunny Fugate.

Of all the people I met in UCSD over the years, Robert Rome has had a significant impact in my life. First as the ECE department administrator and later as a friend, he provided constant and unwavering support during the ebb and flow of my unique graduate school journey. Rob has gone above and beyond his call of duty to help me and my family, especially with my brother. I am grateful to his friendship.

Savita Geuther, Mary Alice Kiisel, Ali Irturk and Caitlin Stallings are special individuals in my life who have constituted my support structure over different periods of my graduate school

tenure. I thank them from the bottom of my heart for their warmth and kindness to me and my family. I thank my Tae Kwon Do instructors, Dr. Jack Nitschke and Dr. Carmen Alonso, for their generosity and mentorship over the years. It is under them that I learned and practiced the art form that has given me the strength to persevere.

The generosity and encouragement of my extended family in facilitating my education cannot be overstated. I cannot possibly list all the family members and their contributions because I am truly a product of their love and support. However, my acknowledgements would be incomplete without mentioning my extended siblings who took care of me and with whom I spent the most time growing up – Vani chechi, Sajitha chechi, Santhoshettan, Pradiyettan, Prasannettan, Bhabhettan, Bindu chechi and my ever-loving well-wisher Lakshmi chechi. I also thank my honorary parents Valliamma and Valliachan, Ahila sithi, Sivadas maama, Sridhara maama, my undergraduate room-mate/uncle Viji maama, my childhood inspiration Bhaskara maama, and their families.

Last but not least, there are no words to express my gratitude to my immediate family. The least I could do is to dedicate this thesis to them, as I did. I thank Mom and Dad for the unbelievable support that they offered to us, especially in the past two years. I am fortunate for having the love and support of Nani and Bruzzer who have embraced me as a brother. I am grateful to my brother, Shiji, for grounding me and helping to keep perspective in life. The foundation for everything I accomplish, all of my successes and growth are the sacrifices and love of my parents, Achan and Amma. The self-sacrifices they have made in their lives for me and my brother, and their unconditional love inspire me to be a better person and a better parent every single day. Their unwavering trust and belief in me gives me the courage to face any challenge in life. Above all, this achievement belongs to my best friend and the love of my life, Bee and our little son. You fill my life with boundless joy and happiness. Your love is my fuel and without you, none of this has any meaning. Everything I do and achieve in life is for you and because of you. Thank you!

Parts of this dissertation are based on published or submitted papers.

Chapter 3, in part, is a reprint of materials in the journal paper by "Patch Matching for Image Denoising Using Neighborhood-based Collaborative Filtering", S. Parameswaran, E. Luo and T. Q. Nguyen [2]. The dissertation author is the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of materials in the conference paper "Targeted video denoising for decompressed videos", S. Parameswaran, E. Luo and T. Q. Nguyen [3]. The dissertation author is the primary investigator and author of this paper.

Chapter 5, in part, is a reprint of materials in the conference paper "Fast External Denoising Using Pre-Learned Transformations", S. Parameswaran, E. Luo, C-A. Deledalle and T. Q. Nguyen [4].The dissertation author is the primary investigator and author of this paper.

Chapter 6, in part, is a reprint of materials in the journal paper "Accelerating GMM-based patch priors for image restoration: Three ingredients for a 100$\times$ speed-up", S. Parameswaran, C-A. Deledalle, L. Denis and T. Q. Nguyen [5].The dissertation author is the primary investigator and author of this paper.

Chapter 7, in part, is a reprint of materials in the journal paper "Image restoration with generalized Gaussian mixture model patch priors", C-A. Deledalle, S. Parameswaran, and T. Q. Nguyen [6]. The dissertation author is the secondary investigator and author of this paper.

VITA

2004	B. S. in Electrical Engineering, University of Wisconsin, Madison B. S. in Mathematics, University of Wisconsin, Madison
2009	M. S. in Electrical Engineering, University of California, San Diego
2011-2018	Engineer, SSC Pacific
2018	Ph. D. in Electrical Engineering, University of California, San Diego

PUBLICATIONS

During doctoral studies:

C-A. Deledalle, **S. Parameswaran** and T. Q. Nguyen, "Image denoising with generalized Gaussian mixture model patch priors", *SIAM Journal on Imaging Sciences*, 2018 (In press).

S. Parameswaran, C-A. Deledalle, L. Denis and T. Q. Nguyen, "Accelerating GMM-based patch priors for image restoration: Three ingredients for a 100$\times$ speed-up", *IEEE Transactions on Image Processing*, 2018 (In press).

S. Parameswaran, E. Luo and T. Q. Nguyen, "Patch matching for image denoising using neighborhood-based collaborative filtering", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.28, Issue 2, pp. 392-401, February 2018.

S. Parameswaran, E. Luo, C-A. Deledalle and T. Q. Nguyen, "Fast external denoising using pre-learned transformations", *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1025-1033, 2017.

S. Parameswaran, E. Luo and T. Q. Nguyen, "Targeted Video Denoising for Decompressed Videos", *Proceedings of 2017 IEEE International Conference on Image Processing (ICIP)*, pp. 2981-2985, 2017.

Other publications:

S. Parameswaran and K. Q. Weinberger, "Large margin multi-task metric learning", *Advances in neural information processing systems (NIPS)*, 2010.

S. Parameswaran, J. Harguess, C. Barngrover, S. Shafer and M. Reese, "Evaluation schemes for video and image anomaly detection algorithms", *SPIE Defense+ Security*, 2016. **(Best paper award)**

S. Parameswaran, K. Rainey, "Vessel classification in overhead satellite imagery using weighted bag of visual words", *SPIE Defense+ Security*, 2015. **(Best paper award)**

M. Jaszewski, **S. Parameswaran**, E. Hallenborg and B. Bagnall, "Evaluation of maritime object detection methods for full motion video applications using the PASCAL VOC Challenge framework", *SPIE/IS&T Electronic Imaging*, 2015.

S. Parameswaran, S. Melvin, Lt. M. Crewes and E. Dorman, "To Catch A Botnet", *MILCOM (ITAR/FOUO)* 2014.

S. Parameswaran, J. Ellen, "Identifying Outliers in Human Movement Trajectories Clustered By Hausdorff Distance", *Proceedings of the International Conference on Data Mining (DMIN)*, 2014.

S. Parameswaran, C. Lane, B. Bagnall and H. Buck, "Marine object detection in UAV full-motion video", *SPIE Defense+ Security*, 2014.

K. Rainey, **S. Parameswaran** and J. Harguess, "Maritime vessel recognition in degraded satellite imagery", *SPIE Defense+ Security*, 2014.

G. Young, C. Lane, B. Bagnall and **S. Parameswaran**, "Robust real-time horizon detection in full-motion video", *SPIE Defense+ Security*, 2014.

K. Rainey, **S. Parameswaran**, J. Harguess and J. Stastny, "Vessel classification in overhead satellite imagery using learned dictionaries", *SPIE Optical Engineering+ Applications*, 2012.

J. Ellen, J. Kaina and **S. Parameswaran**, "Implicit group membership detection in online text: analysis and applications", *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction (SBP)*, 2012.

J. Ellen and **S. Parameswaran**, "Machine Learning for Author Affiliation within Web Forums–Using Statistical Techniques on NLP Features for Online Group Identification", *International Conference on Machine Learning and Applications (ICMLA)*, 2011.

A. Gupta, **S. Parameswaran** and C-H. Lee, "Classification of electroencephalography (EEG) signals for different mental activities using Kullback Leibler (KL) divergence", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009.

Patent items:

Systems and methods for real-time horizon detection in images (US Patent 9,305,214), 2016.

Apparatus and Method for Using a Support Vector Machine and Flow-Based Features to Detect Peer-to-Peer Botnet Traffic (US Patent App. 15/362,60), 2018.

ABSTRACT OF THE DISSERTATION

Efficient Methods for Image Denoising using Learned Patch Priors

by

Shibin Parameswaran

Doctor of Philosophy in Electrical Engineering (Signal and Image Processing)

University of California, San Diego, 2018

Professor Truong Nguyen, Chair

Cameras have become ubiquitous leading to an increase in the amount of video and image data captured by amateurs and professionals alike. Their ease of deployability makes them a great sensor for security applications as well. Hence, there is an ever-growing need to *efficiently* process and enhance captured image and videos for improving the performance of subsequent computer vision algorithms or simply for aesthetic reasons. To address this need, we focus on creating efficient techniques for large scale image and video denoising with varying degrees of genericity.

We start by introducing a robust patch matching technique that increases the efficacy of denoising algorithms that build patch-specific filters. We show that using our matching

criterion in multiple leading denoising algorithms provides additional performance gains over using default distance metrics. Next, we present a strategy to extend patch-based image denoising algorithms into a decompressed video denoising paradigm without increasing computational complexity. We leverage pre-calculated motion vectors present in a compressed video's bitstream to establish temporal correspondences, thus keeping the per-frame complexity of the video denoising algorithm equivalent to that of the corresponding image denoising method. Following this, we relax the patch-specific constraint on design of denoising filters leading to one of the fastest algorithms that uses *targeted* local patch prior.

Recognizing that a *targeted* patch prior could be a limiting factor for a wide variety of natural images, we develop an efficient denoising algorithm that uses a Gaussian Mixture Model (GMM) to model a *generic* patch prior for image restoration. It is two orders of magnitude faster than similar methods while providing a very competitive quality-vs-speed operating curve. The final work presented in this thesis improves upon GMM priors by proposing a more expressive distribution using Generalized Gaussian Mixture Models (GGMM) patch priors. We circumvent the prohibitive computational complexity of using GGMM patch priors for image restoration by introducing asymptotically accurate but computationally efficient approximations to the bottlenecks encountered in this formulation. Our evaluations indicate that the GGMM prior is consistently a better fit for modeling image patch distribution and performs better on average in image denoising task.

Chapter 1

Introduction

The advances in camera technology and storage has led to an explosive growth in the amount of images and videos captured for personal and professional uses such as for sharing on social media sites or security purposes. Quite often, data is captured in poor conditions leading to degraded images (e.g. blurry, noisy and foggy). The process of recovering the underlying clean image from its degraded counterpart is collectively known as *image restoration*. In this thesis, we will primarily focus on one particular type of restoration called *denoising*. Denoising algorithms aim to reconstruct underlying clean signals (images or videos) from observed noisy signals.

1.1 Patch-based image denoising

In the last decade, the field of image denoising has seen tremendous advancements. One of the innovations that enabled these developments is patch-based denoising. These methods divide an image into overlapping patches (small subimages of, say, 8×8 pixels) and remove noise by applying a denoising filter on a patch-by-patch basis. Patch-based denoising methods can be broadly divided into two classes – *internal* and *external* methods. Internal methods are those algorithms that rely exclusively on the information contained within an image when designing the denoising filter. Popular methods that fall into this category are non-local means (NLM)

[7], BM3D [8], k-SVD [9], SAFIR [10], NL-Bayes [11], DDID [12] and LPG-PCA [13]. To denoise a patch in the noisy image (commonly referred to as a *query* patch), these methods rely on information contained in *reference* patches (i.e. other patches that are similar to the query patch) within the noisy image itself. Due to this reason, internal methods often suffer when the noise level in an image is high and also when denoising rare patches due to the lack of an adequate number of reference patches [14]. To ease these problems, external denoising methods were developed in order to restore noisy patches using information from outside the given noisy image. Methods in this category have taken many different approaches for leveraging external information. For instance, the Expected Patch Log Likelihood (EPLL) algorithm [15] denoises patches using a Gaussian Mixture Model (GMM) prior learned on an external database of clean patches. Other methods include learning mapping functions to map noisy input patches to clean ones using multi-layer perceptrons (MLP) [16], piecewise linear estimators [17, 18], Field of Experts [19], iPiano [20, 21] and straightforward adaptations of NLM, BM3D and k-SVD [9, 22, 23].

Recently, Luo *et al.* [24, 23] proposed a variant of external denoising called *targeted* image denoising (TID) that utilizes category-specific external databases. This follows from the observation that a large *generic* database does not necessarily contain enough useful information (and may often contain irrelevant information) needed to denoise the noisy image of a specific category. Hence, instead of using a generic database, TID obtains reference patches from a targeted or category-specific database that contains images *relevant* to the noisy image. For example, while denoising a face image, a targeted database is made up of patches from other face images whereas while processing a license plate image a targeted database will be constructed using license plate images only. This approach of using targeted databases tailored to particular domains and tasks can be viewed in a similar category as a straightforward adaptation of task-driven image processing methods [25].

In this thesis, I will mainly focus on external denoising algorithms and its targeted database

variant. In the first part of my work, I will consider algorithms that perform the best with targeted databases. In the later part, the focus will be shifted to those that are effective even with generic databases. Please note that *generic* and *targeted* are relative terms – a category-specific database may not be considered targeted if the category exhibits a wide variety (e.g. types of domestic animals). For the purposes of this thesis, a database is called targeted when the images share very similar content with only subtle changes in scale and pose (e.g. close-up images of U.S. automobile license plates). Also, external denoising algorithms that use generic databases can be easily adapted to targeted paradigm by replacing generic database of patches with targeted databases.

1.2 Role of priors in denoising

A *prior* is the knowledge that one has about the problem that is being solved, *a priori* or beforehand. Priors on natural images play an important role in image denoising (and in general image restoration, as well). They have a direct effect on the existence and the quality of the solutions in denoising and other ill-posed restoration scenarios. Early image priors were designed to preserve local statistics that are commonly found in natural images. Total variation constraint is one such prior that encourages solutions consisting of piece-wise constant regions in images [26]. Buades et al. [27] demonstrated the superiority of a non-local prior using the observation that images often contain repeating patterns, albeit at a distance. This property commonly referred to as *non-local self-similarity*, is the prior used in the famous non-local means algorithm (NLM) [27]. Non-local self-similarity combined with sparse or low-rank constraint is the basis for successful algorithms such as BM3D [28], LPG-PCA [29], etc. These priors are effective but they do not use any side-information other than the content of a single image, i.e., the given noisy image.

In contrast, recent research demonstrated that powerful priors can be realized by learning a model for image patch distributions [19, 30, 15, 17, 31, 32, 33, 34, 35]. These *learned priors*

provided excellent performance improvements by allowing a way to incorporate information contained in large external datasets. Just like any other machine learning applications, the field of learned priors consists of a vast array of different strategies. It ranges from *lazy learning* methods where there is no training involved (similar to k-nearest neighbors), to learning a generative model or a discriminative model and, more recently, to deep learning-based approaches. These innovations have inadvertently led to improvements in denoising quality yielding state-of-the-art performances. However, their improvements are obtained at the expense of computational complexity and general algorithmic efficiency leading to algorithms that are impractical for large scale image restoration applications. In this thesis, we systematically address the shortcomings of the existing patch-based denoising algorithms that use *learned* patch priors in an effort to improve their efficacy and computational efficiency.

1.3 Thesis overview

Chapter 2 provides a brief background on the concepts covered in this dissertation such as image degradation and restoration, patch-based denoising and patch priors.

Chapter 3 introduces a robust patch matching strategy that can be used to gain more from algorithms that use database of patches for designing denoising filters (lazy learning). The performance of these algorithms are closely tied to the selection of reference patches. Hence, we present a new matching criterion that lends robustness to the patch matching stage of image denoising algorithms. We show that using our matching criterion in multiple leading denoising algorithms provides additional performance gains over using default distance metrics. This work is not constrained by demands of computational complexity. The sole focus of this effort is to explore the power of patch matching on algorithms that use external domain specific databases.

Chapter 4 demonstrates that motion vectors contained in compressed video streams can be utilized to extend patch-based image processing algorithms to video denoising without

adding computational complexity. We leverage pre-calculated motion vectors present in a compressed video's bitstream to establish temporal correspondences, thus keeping the per-frame complexity of the video denoising algorithm equivalent to that of the corresponding image denoising method. The main lesson learned over the course of this work is that although optical flow was the gold standard for finding temporal correspondences, a coarser but readily available approximation obtained from motion vectors provided incredible computational savings while providing comparable quality.

It was also clear that the speed and efficiency of the underlying image processing algorithm is paramount. A powerful but slow denoising algorithm can never be used for large scale applications or video denoising. For instance, in Chapter 4 we extend TID [24, 23] to video denoising. The new video denoising algorithm provided very good performance, but was not practically viable due to its high computational demands during runtime. We needed algorithms that can effectively find high quality solutions and are extremely efficient during runtime.

To this end, in Chapter 5, Chapter 6 and Chapter 7, we depart from lazy learning paradigm to using progressively powerful priors that are trained using patch datasets. These approaches yearn to minimize the complexity of calculations during runtime by pre-calculating (when possible) and/or introducing novel approximations in place of time-consuming steps.

Specifically, in Chapter 5 we introduce a new external denoising algorithm that utilizes pre-learned transformations to accelerate filter calculations during runtime. By moving computationally demanding steps to an offline learning stage, the proposed approach finds a balance between processing speed and obtaining high quality denoising estimates. We show that the proposed approach is extremely effective when the transformations are learned using a targeted database. It provides comparable denoising performance while being orders of magnitude faster than TID [24, 23], the state-of-the-art targeted denoising method.

Targeted denoising works extremely well when denoising images that have a clear category or domain, for e.g. face images, license plates etc. However, if a targeted database is unavailable

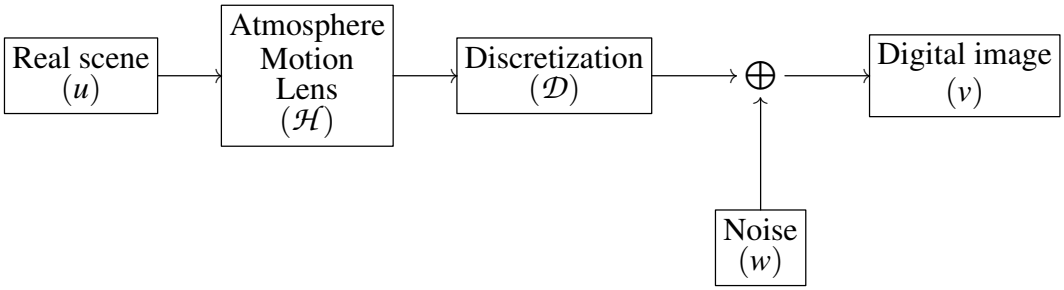
due to wide variations in a category or domain mismatch, targeted denoising can lead to sub-optimal over-smoothed results. To alleviate this issue, it is also necessary to have denoising algorithms that can work well with generic databases. For this purpose, we conducted an exhaustive complexity analysis of EPLL algorithm which is the current state-of-the-art algorithm. As is the case with TID, although very effective in denoising, its high runtime complexity makes EPLL ill-suited for most practical applications. In Chapter 6, we propose three approximations to the original EPLL algorithm and achieve dramatic speed-up of two orders of magnitude over it while incurring a negligible drop in the restored image quality.

Further analysis of GMM patch prior, revealed that patch distributions are not necessarily Gaussian in all directions and therefore a GMM is not an optimal choice. To address this issue, in Chapter 7 we introduce a new algorithm in the EPLL framework that uses a generalized Gaussian mixture model (GGMM) patch prior. We show that the proposed fully flexible GGMM prior captures the underlying distribution of patches better than a GMM. However, the non-Gaussianity of its components presents major challenges in its application to the computationally intensive process of image restoration. In this chapter, we provide asymptotically accurate approximations and computational recipes for fast evaluation of bottleneck steps, so that using EPLL with a GGMM prior on an image with more than tens of thousands of patches is computationally viable.

Chapter 2

Background

Image processing is a subset of signal processing where the input and output are digital images. The type of image processing that takes a corrupted image as input and outputs an estimate of the underlying clean image is called image restoration. The corruptions in image can be caused during the image formation process due to a myriad of factors such as the blur introduced by atmospheric condition, camera lens, motion of the object, sensor limitations and noise. This degradation process can be simplified as follows:



or mathematically,

$$v = \mathcal{D}\mathcal{H}u + w \tag{2.1}$$

where u and v are the original scene and uncorrupted image captured by the imaging system, $\mathcal{H}$ represents the blur, $\mathcal{D}$ models the discretization operation and w the additive noise. The additive

noise w constitutes different types of disturbances that can be attributed to lighting conditions (shot noise), sensor limitations (electronic noise) and storage procedure (quantization noise), to name a few. Admittedly, all of these disturbances may not be additive in nature, however they can be reliably approximated by additive Gaussian noise using variance stabilization transforms [36]. Therefore, in majority of the image processing literature, w is modeled as an additive white Gaussian noise (AWGN). Additionally, for ease of notation, the degradation operations are usually combined and denoted by a single linear operator $\mathcal{A}$, *i.e.*, $\mathcal{D}\mathcal{H}u = \mathcal{A}u$. In this thesis, we will follow this convention along with the AWGN assumption for w. Accordingly, image restoration can be defined as the problem of estimating an image $u \in \mathbb{R}^N$ (N is the number of pixels) from noisy linear observations $v \in \mathbb{R}^M$ given by:

$$v = \mathcal{A}u + w \tag{2.2}$$

where $\mathcal{A} : \mathbb{R}^N \to \mathbb{R}^M$ is the linear degradation operator and $w \in \mathbb{R}^M$ is a noise component assumed to be white and Gaussian. Typical examples for operator $\mathcal{A}$ are: identity matrix (for *denoising*), a low pass filter (for *deconvolution*), a masking operator (for *inpainting*), or a projection on a random subspace (for *compressive sensing*).

The rest of this chapter provides short overview of the topics that are pertinent to problems investigated in this dissertation.

2.1 Image denoising

In this thesis, we will focus on one particular type of restoration called *image denoising* where $\mathcal{A}$ is the identity matrix. We will also assume that the variance of the AWGN is known *a priori*. Hence, the degradation model can be written as:

$$v = u + w \tag{2.3}$$

where u, v and w belongs to $\mathbb{R}^N$ and w is white Gaussian noise with known variance σ^2.

The reasons for this simplification are multi-fold. In image restoration, in addition to the challenges of a standard denoising problem, there is an additional challenge presented by the need for inversion of the degradation operator. This additional step makes the problem heavily *ill-posed* which requires additional constraints or assumptions which can be collectively called as the *prior*. As explained earlier, image denoising problems also benefit from better *prior* models. For this reason, designing better prior models for improving denoising solutions will lead to better solutions for image restoration as well. Hence, we use the simplified denoising framework as the testbed to develop better general image restoration methods.

Below, we discuss the two denoising strategies utilized by the algorithms in this thesis.

2.2 From patch-wise to whole image denoising

There are different strategies for performing patch-based image denoising. One of the main approaches can be termed *patch-wise* denoising, where each patch of a given noisy image v is restored independently. A *patch-wise* denoising approach can be summarized as follows:

1. Extract overlapping patches,$\{y_1, ..., y_N\}$, from a given noisy image v,

$$y_k = \mathcal{P}_k v \tag{2.4}$$

   where $\mathcal{P}_k : \mathbb{R}^N \to \mathbb{R}^P$ is a linear operator extracting a patch with P pixels centered at the pixel with location k.

2. After patch extraction, apply denoising procedure on each patch by solving the following optimization problem for each patch.

$$\left\{ \hat{x_k} \longleftarrow \underset{x_k \in \mathbb{R}^P}{\operatorname{argmin}} \frac{1}{2}\|x_k - y_k\|^2 + \lambda f(x_k) \right\}_{k=1,...,N} \tag{2.5}$$

where $y_k = \mathcal{P}_k v$ and $x_k = \mathcal{P}_k u$ are the pixel vectors formed from the k-th patch of size $\sqrt{P} \times \sqrt{P}$ extracted from the noisy image and latent clean image, respectively, $\hat{x_k}$ is the denoised estimate, $f(\cdot)$ is a regularizer and λ is the regularization constant.

3. After denoising each patch, the whole image estimate is obtained by replacing the patch estimates back in the image grid (reverse of patch extraction) and averaging the overlapping areas,

$$\hat{u} = \sum_k w_k \mathcal{P}_k^t \hat{x_k} \tag{2.6}$$

where $\hat{u}$ is the denoised image estimate, $\mathcal{P}_k^t$ performs the complementary operation to patch extraction by placing the input as the k-th patch in the image grid and w_k is the weight of the k-th patch during averaging (*i.e.*, $w_k = \frac{1}{P}$ for uniform weighting).

4. The above steps can iterated, if necessary. That is, after obtaining an estimate for the underlying clean image, the above steps can be repeated by setting $v = \hat{u}$.

This procedure simply focuses on denoising individual patches with the underlying assumption that the aggregation of the resulting denoised estimates of all patches will lead to a global estimate (whole image) that is close to the underlying clean image. Patch-wise denoising strategy is utilized by a number of successful algorithms such as BM3D [28], LPG-PCA [13], and TID [24, 23].

However, without a constraint on the appearance of the global estimate (whole image), extending the patch-wise strategy to general restoration problems is non-trivial. Also, in this setting, the patch aggregation step with or without uniform weights is a heuristic that needs special attention for each iteration of the algorithm.

An alternative approach is to include a term in the optimization problem to ensure that the

resulting global estimate is close to the noisy input image.

$$\underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\|u-v\|^2 + \lambda \sum_{i=1}^{N} f\left(\mathcal{P}_i u\right) \tag{2.7}$$

where x is the clean image, y is the noisy image, $\mathcal{P}_i$ is the patch extractor operator that extracts the i-th patch and $f(\cdot)$ is a patch-wise function. This formulation is what we call *whole-image* denoising. The first term in Equation (2.7) ensures that the estimated image as a whole closely resembles the original noisy image. The second term is a patch-wise denoising function and is usually interpreted as a prior on underlying clean patches. Under a Bayesian formulation, the first term is the negative log-likelihood assuming Gaussian noise, and therefore, the patch-wise function can be expressed as $f\left(\mathcal{P}_i u\right) = -\log p\left(\mathcal{P}_i u\right)$, where $p(.)$ is the *a priori* probability density function.

Unlike the patch-wise strategy (Equation (2.5)), the *whole-image* denoising formulation can be easily extended to general restoration problems by simply incorporating $\mathcal{A}$ in the first term.

$$\underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\|\mathcal{A}u-v\|^2 + \lambda \sum_{i=1}^{N} f\left(\mathcal{P}_i u\right) \tag{2.8}$$

Also, the whole image formulation problems of eq. (2.7) and eq. (2.8) are usually solved using alternating minimization strategies. Under these schemes, the process of iterating for progressively better estimates and patch aggregation become natural steps in the minimization procedure rather than heuristic strategies as in the patch-wise formulation.

In this thesis, we will start with the state-of-the-art patch-wise algorithm (chapters 3-4), then proceed to whole-image formulation in favor of its genericity (chapters 5-7).

2.3 Overview of patch priors

In this section, we will provide a quick overview of patch priors that have been used in image restoration. For brevity of expressions, we will drop the patch subscript and denote $\mathcal{P}_i u = x$ for the rest of this section.

Whitening priors

A popular attempt in designing patch priors considers a whitening transform $W \in \mathbb{R}^{P\times P}$, *i.e.*, an orthogonal transform that decorrelates entries of x. Typically, the transform W can be chosen by principal component analysis on a dataset of clean patches [37], or based on some prior knowledge, *e.g.*, by assuming decorrelations in the Fourier or wavelet domain [38]. Considering decorrelated and independent transform coefficients allows for the prior distribution to be separable in the transformed domain (*i.e.*, with independent marginals). Early works on wavelet representations of images (*e.g.*, Mallat, 1989 [38] and Moulin and Liu, 1999 [39]) suggested modeling such coefficients by a zero-mean generalized Gaussian distribution (GGD) of the form

$$p(x) \propto \exp\left(-\rho^{\nu}\|Wx\|_{\nu}^{\nu}\right) \tag{2.9}$$

where $\rho > 0$ and $\nu \geqslant 0$. Here, choosing $1 < \nu \leqslant 2$ models each coefficient by a bell shape distribution with small tails (the Gaussian distribution for $\nu = 2$), see first row on Figure 2.1(a-b). Image patches are thus assumed to be concentrated on a convex cluster (an ellipsoid for $\nu = 2$), see second row on Figure 2.1(a-b). If this prior is used in eq. (2.7), the regularization parameter λ is equivalent to ρ^{ν} and as ρ increases the coefficients tend to get closer and closer to zero. This leads to diminishing edge features which yields over-smoothed images.

To circumvent this issue and preserve edges, $\nu \leqslant 1$ is considered. It corresponds to modeling each coefficient by a distribution with a peak at zero (non-differentiability) and large

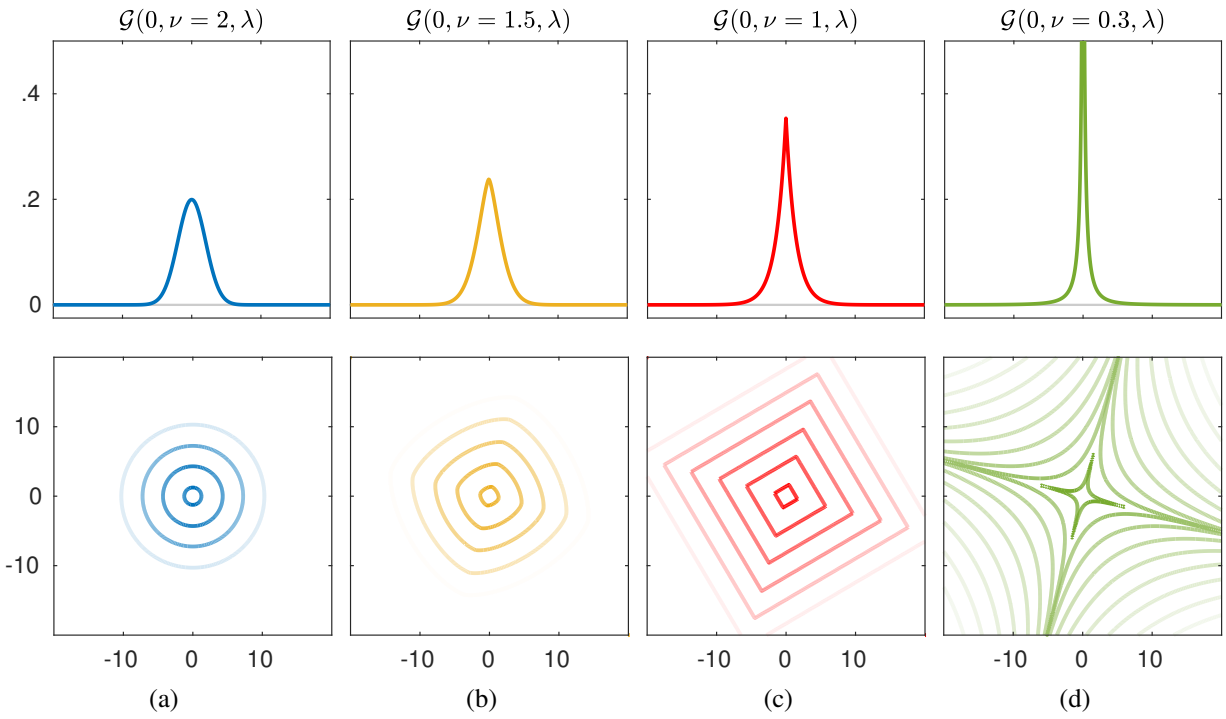


Figure 2.1: Illustrations of zero-mean generalized Gaussian distributions for different values of the shape parameter ν. From left to right: $\nu = 2, 1.5, 1, .3$. Top: one-dimensional versions with variance $\lambda = 2$. Bottom: iso-contours of the two-dimensional versions with W being a rotation matrix .

tails, see first row on Figure 2.1(c-d). Image patches are thus assumed to be spread on an union of orthogonal subspaces aligned with the directions of the rows of W, see second row on Figure 2.1(c-d). Under this scheme, most coefficients are set to zero (*sparsity*) with a few non-zero coefficients, preserving edges and resulting in sharper images.

Synthesis sparse priors

More recent works have shown that the distribution of clean image patches can be better modeled by assuming they are sparse combinations of the columns of a redundant and overcomplete dictionary $D \in \mathbb{R}^{P \times Q}$, $Q \geqslant P$ (*i.e.*, composed of linearly dependent atoms). This is typically achieved with undecimated (*a.k.a,* shift-invariant) wavelets [40, 41] or, more generally, frames [42, 43]. A dictionary can also be learned on a large dataset of clean patches with the

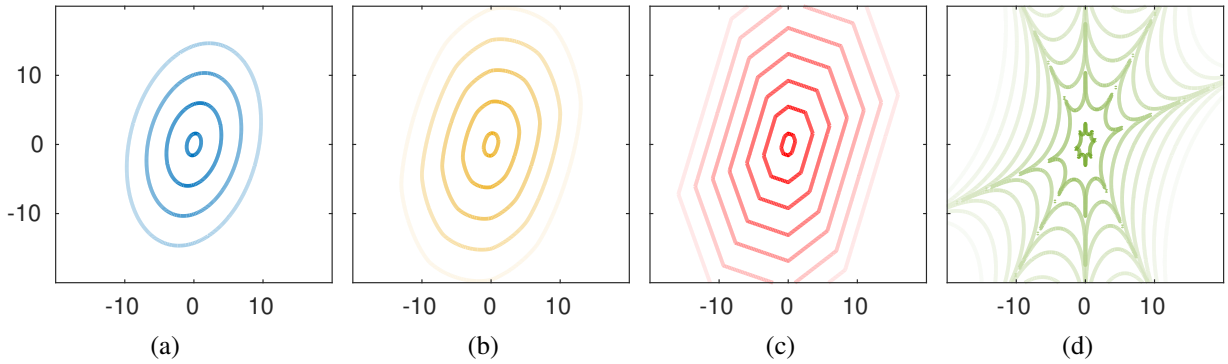


Figure 2.2: Illustrations of the iso-lines of (a-d) four two-dimensional priors based on sparse analysis regularization with $0 < \nu < 2$ and a dictionary $\Omega \in \mathbb{R}^{4\times 2}$ with four directions. From left to right: the shape parameter varies respectively as $\nu = 2$, 1.5, 1, .3.

k-SVD algorithm [30]. This is to suggest that the coefficients of an image are spread on a union of *non-orthogonal* subspaces aligned with the columns of D, see Figure 2.2. The so-called synthesis regularization framework [44], applied to image patches in [45, 46], corresponds to the following choice of prior distribution

$$p(x) \propto \exp\left(-\rho^{\nu}\|\hat{\alpha}\|_{\nu}^{\nu}\right) \quad \text{where} \quad \{\hat{\alpha} = \operatorname*{argmin}_{\alpha} \|x - D\alpha\|_{\nu}^{\nu}\}, \tag{2.10}$$

where $\rho > 0$ and $\nu \geqslant 0$. Similar to the whitening framework, choosing $\nu \leqslant 1$ enforces sparsity without penalizing large non-zero coefficients.

Analysis sparse priors

Alternatively, the distribution of clean images can also be captured by modeling the correlations of its patches with the rows of a redundant dictionary $\Omega \in \mathbb{R}^{Q\times P}$ that does not require to span the set of clean images (Ω can be rank deficient). This is the case with the Total-Variation model [26] that considers the gradient of an image $\Omega = \nabla$, and any type of filter bank analysis. Similar to the synthesis framework, the dictionary can also be learned on a large dataset of clean patches with the analysis k-SVD algorithm [47]. This suggests that the correlations of an image

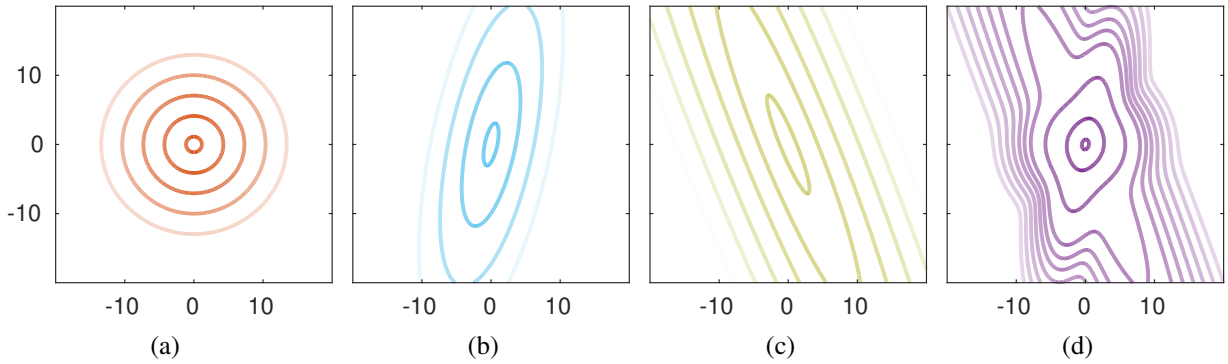


Figure 2.3: Illustrations of the iso-lines of (a-c) three two-dimensional zero-mean Gaussian distributions and (d) their mixture with weights $1/2$, $1/3$ and $1/6$ respectively.

are spread on a union of non-orthogonal subspaces aligned with the rows of Ω. The analysis regularization framework [44], corresponds to the following choice of prior

$$p(x) \propto \exp\left(-\rho^{\nu}\|\Omega x\|_{\nu}^{\nu}\right) \tag{2.11}$$

where $\rho > 0$ and $\nu > 0$. Again, choosing $\nu \leqslant 1$ enforces correlations to be mostly zeros – *co-sparsity* – and a few large non-zero correlations. A difficulty with such an approach is to cope with the non-injectivity of Ω.

GMM priors

Rather than modeling clean patches as spread on a union of non-orthogonal sub-spaces, an alternative is to consider a union of ellipsoids (called clusters). To this end, the authors of [17, 15] suggested using a zero-mean Gaussian mixture model (GMM) prior, that, for any patch $x \in \mathbb{R}^P$, is given by

$$p(x) = \sum_{k=1}^{K} w_k \mathcal{N}_P(x; 0_P, \Sigma_k) \tag{2.12}$$

where K is the number of components, $w_k > 0$ are weights such that $\sum_k w_k = 1$, and $\mathcal{N}_P(0_P, \Sigma_k)$ denotes the multi-variate Gaussian distribution with zero-mean and covariance $\Sigma_k \in \mathbb{R}^{P \times P}$. An illustration of a GMM and its components is given in Figure 2.3. The hyper-parameters w_k and Σ_k can be learned using the Expectation Maximization algorithm [48] on a large dataset of clean patches [15]. Last but not least, when $K = 1$, this approach is equivalent to the whitening approach with $\rho W = \Sigma_1^{-1/2}$ and $\nu = 2$.

Note that the overview given here is mainly to provide a background for the methods covered in this thesis and is far from being exhaustive. Beyond the four aforementioned priors, many other image and patch priors have been proposed in the literature, including Gaussian scale mixtures [49], Fields-of-Experts [19], Total Generalized Variation [50] and Student mixture models [34] to just cite a few that we have omitted.

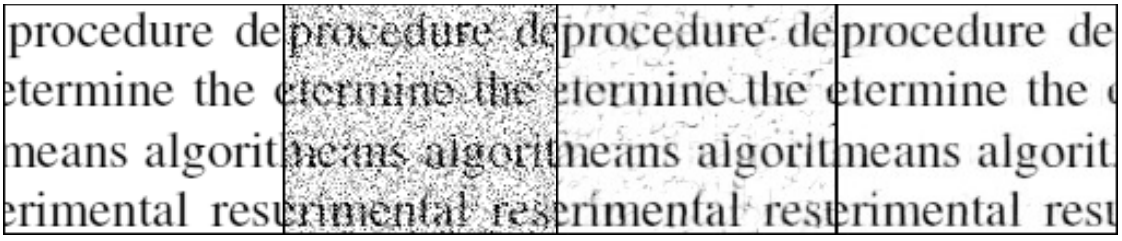
In the following chapters, we will use patch priors that are extensions and constrained versions of analysis priors and GMM prior before introducing a new generalized Gaussian mixture model (GGMM) patch prior. First, in Chapter 3, we introduce a noise-robust patch matching strategy taking inspiration from recommendation systems. We demonstrate that by improving patch matching, we can extract better performance from the state-of-the-art patch-wise algorithms that use patch-specific (*localized*) whitening and analysis priors.

Chapter 3

Patch Matching for Image Denoising Using Neighborhood-based Collaborative Filtering

3.1 Introduction

In patch-based denoising, the overall denoising performance is significantly influenced by how similar the reference patches are to the underlying clean patch – i.e. the patch from the clean image corresponding to the patch being denoised, prior to noise corruption. The main difficulty of this scenario is that the clean image is not known *a priori*. Therefore, patch matching has to be carried out using query patches from the noisy image or an imperfect estimate of the clean image. This usually results in the retrieval of a non-ideal set of reference patches, which, in turn, leads to subpar denoising results. We illustrate this problem by comparing the denoising performances obtained by conducting patch matching using query patches from the noisy image versus its clean version. The results of this simple experiment are shown in Figure 3.1. This example shows the effect of patch matching on denoising performance. In this example, we attempt to



(a) original image (b) noisy image (c) TID (d) TID-ideal

Figure 3.1: Sub-optimal patch matching leads to sub-optimal results: Original clean image (a) is corrupted with Gaussian noise ($\sigma = 80/255$). The noisy image (b) is then denoised using the Targeted Image Denoising [23] algorithm with reference patches found from an external text database. Figures (c) and (d) show the denoising results when the query patches are taken from the noisy image and the clean image, respectively. Please note that in practical situations, the clean image is not known *a priori*.

denoise a text image corrupted by heavy Gaussian noise ($\sigma = 80/255$) using the state-of-the-art patch-based denoising algorithm called Targeted Image Denoising (TID) algorithm [24, 23]. In the first case, reference patches are found using query patches from the noisy image whereas in the second case, patches from the underlying clean image are used during patch matching. The rest of the denoising algorithm is kept unchanged in both cases, and the corresponding denoising results are shown in Figures 3.1(c) and 3.1(d), respectively. It can be observed that the denoising performance achieved in the second case is superior when compared to the first case.

In practical situations, the clean image is not known *a priori*, and the second case shown in our toy example is not plausible. We show the result of this ideal scenario for demonstration purposes only. It points to an opportunity to improve the performance of patch-based denoising algorithms by improving the robustness of the patch matching step.

In this chapter, we propose a new way to find better reference patches based on k-nearest neighbor collaborative filtering (kNN-CF), which enables robust patch matching. The novelty of our work is twofold. First, we approach the problem of finding similar patches as a collaborative filtering (recommendation system) problem in order to provide robustness to noise. Second, we show how our method is an effective framework to seamlessly combine information from patches internal to the noisy image (internal denoising) and those from an external database of clean

patches (external denoising).

This chapter is organized as follows: In Section 3.2, we briefly review popular patch-based denoising algorithms, including the recently introduced Targeted Image Denoising (TID) [23] algorithm, and provide a quick background on nearest neighbor collaborative filtering, which is the basis of the proposed method. We present the proposed patch matching scheme in Section 3.3. Our experimental results are detailed in Section 3.4, followed by a discussion in Section 3.5 and conclusion in Section 3.6.

3.2 Background

3.2.1 Targeted Database and Targeted Image Denoising

TID is an external denoising algorithm that utilizes a *targeted* database for denoising an image. A targeted database, as mentioned earlier, contains patches that are relevant to the noisy image of interest and is shown to be better than a generic database [23]. In addition to providing more relevant reference patches, using a class- or domain-specific database allows the size of the database to remain smaller and leads to better computational speeds during denoising.

Algorithm 1 provides an overview of the TID algorithm. The reference patches are concatenated to form the data matrix $\mathbf{P}$, which is then used to calculate a denoising filter by solving a group sparsity minimization problem and using a localized Bayesian prior. Please refer to [23] for a detailed explanation and derivation. Note that the TID filter is strongly influenced by the reference patches since they are used both in TID's transformation ($\mathbf{U}$) and its shrinkage (Λ) operations.

Algorithm 1 Targeted Image Denoising [23]

INPUT: Query patch (q)
Database of clean patches ($\mathbf{D}$)
Noise variance (σ^2)
OUTPUT: Denoised patch ($\hat{p}$)

1. Find n patches p_1, p_2, $\dots$ p_n from the database that are similar to q
2. Form data matrix:

$$\mathbf{P} = [p_1, p_2, \dots p_n]$$

3. Form weight matrix:

$$\mathbf{W} = \frac{1}{\alpha} \operatorname{diag}\{w_1, w_2, \dots, w_n\}$$

where $w_i = \exp\left(-\frac{\|q-p_i\|^2}{h^2}\right)$ for some user-tunable bandwidth parameter h, and α is a normalization parameter so that the weights add up to 1.

4. Compute the eigen-decomposition:

$$[\mathbf{U}, \mathbf{S}] = \operatorname{eig}\left(\mathbf{PWP}^T\right)$$

5. Compute the shrinkage matrix:

$$\Lambda = \left(\operatorname{diag}\left(\mathbf{S} + \sigma^2\mathbf{I}\right)\right)^{-1} \operatorname{diag}(\mathbf{S})$$

6. Denoise q: $\hat{p} = \mathbf{U}\Lambda\mathbf{U}^T\, q$

3.2.2 BM3D and Its Collaborative Filtering

BM3D [8] is a transform-domain denoising procedure that attenuates noise by enhancing group sparsity through *collaboratively* filtering patch groups. Patch groups are formed by finding similar patches (reference patches) and stacking them to produce three dimensional stacks. This 3D stack of similar patches (group) is filtered *jointly* to produce individual denoised patch estimates of each member of the group. The joint filtering, which involves 3D group linear transformation, coefficient shrinkage, and 3D inverse transformation is called collaborative filtering in the BM3D procedure. This is a fundamentally different approach than the procedure termed collaborative filtering in recommendation systems, which is the basis for our approach. We provide a more detailed explanation of recommendation system collaborative filtering technique in Section 3.2.6.

Algorithm 2 BM3D [8]

INPUT: Query patch (q)
Database of clean patches (**D**)
Noise variance (σ^2)
OUTPUT: Denoised patch ($\hat{p}$)
1. Find n patches $p_1, p_2, \ldots p_n$ from the database that are similar to q
2. Stack them together to form 3D matrix
3. Apply a 3D transform on the 3D patch stack (e.g. 3D-DCT)
4. Perform coefficient shrinkage in the 3D transform domain (e.g. thresholding/wiener filtering)
5. Apply inverse 3D transform to obtain $\hat{p}$

3.2.3 External BM3D

External BM3D [24, 23] is a variation of BM3D where the reference patches to produce 3D stacks are obtained from an external database. BM3D is a two-stage procedure where the algorithms used in both the stages are almost identical to each other except for one step. The general outline of the core algorithm constituting external BM3D is shown in Algorithm 2. The only difference between the first and second stages of BM3D is in the coefficient shrinkage step (step 4 in Algorithm 2). In stage one, coefficient shrinkage is carried out via hard thresholding, whereas in stage two, Wiener filtering is applied to accomplish the same. The output of stage one is used as an oracle for Wiener filtering in the second stage.

Unlike TID, the impact of reference patches has no effect on determining the transformation or the shrinkage procedure. That is, BM3D uses a predetermined transformation and shrinkage procedure. However, the reference patches constituting the 3D stack of patches undergoing 3D transformation do have an impact on the values of the transform domain coefficients, hence, they play a role in overall denoising.

3.2.4 External LPG-PCA

LPG-PCA [13] was originally introduced as an internal denoising algorithm that uses principal component analysis and local pixel grouping to efficiently remove noise from a noise-corrupted image. In LPG-PCA, local pixel grouping is attained by finding patches that are similar

Algorithm 3 LPG-PCA [13]

INPUT: Query patch (q)
Database of clean patches (**D**)
Noise variance (σ^2)
OUTPUT: Denoised patch ($\hat{p}$)
1. Find patches $p_1, p_2, \ldots p_n$ from the database that are similar to q
2. Apply PCA transform and perform shrinkage
3. Inverse PCA transform to obtain denoised patch $\hat{p}$

to the query patch via a patch matching procedure within a local window of the noisy image. By obtaining patches from an external database instead of the given noisy image in its local pixel grouping stage, LPG-PCA can be easily modified to become an external denoising algorithm [23]. The procedural outline for external-LPG-PCA is shown in Algorithm 3. Like BM3D, this algorithm is also applied twice with an updated noise variance estimated from the first stage output.

Due to its reliance on principal component analysis, reference patches have a more direct impact on the denoising filter than the BM3D scenario. The quality of the reference patches affect the transformation matrix obtained during PCA, which is applied to the pixel groups during denoising. However, the patches do not have as much of an impact on the shrinkage step as they have in the TID filter.

3.2.5 Patch Matching

As observed, the first step of all of these patch-based denoising algorithms is to find reference patches given a query patch. In general, the reference patches are obtained by selecting the nearest neighbors of a query patch q using the Euclidean distance.

Given that the denoising filters are modeled using the reference patches (in varying degrees), it is imperative that the performance of these algorithms relies on the selection of good reference patches. However, finding reference patches using a noisy query patch leads to less than ideal matches, which in turn degrades the denoising performance.

Hence, the proposed approach aims to add robustness to the patch matching stage and retrieve a better set of reference patches, even when the query patches are not clean. Instead of solely trusting the matching patches retrieved using a noisy query patch, we pose patch matching as a recommendation system problem and incorporate suggestions from a set of patches *known* to be similar to the query patch (selection of this set of close neighbors is explained in detail in Section 3.3.2). By substituting the proposed CF-based patch matching in place of the standard patch matching scheme (Step 1 in Algorithms 1, 2, and 3), we will demonstrate that CF-based methods provide better reference patches, leading to improved denoising results.

We will demonstrate the effectiveness of our new CF-based patch matching scheme by using it in conjunction with the TID algorithm and external versions of BM3D and LPG-PCA. We will also compare our results with conventional internal denoising algorithms, such as BM3D and NLM, since they are widely accepted benchmarks for patch-based denoising algorithms.

3.2.6 Neighborhood-based Collaborative Filtering

Here, we provide a brief review of neighborhood-based collaborative filtering from which we derive the proposed patch matching criterion. Collaborative filtering (CF) is a popular recommender system approach that predicts a given user's preference by aggregating other users' preferences [51, 52, 53, 54, 55]. In particular, nearest neighbor-based collaborative filtering (NN-CF) method predicts the unknown rating of an active user a by combining the ratings from users most similar to a. More formally, user a's rating of an item i, p_{ai}, is predicted using the ratings from user a's nearest neighbors as shown below:

$$p_{ai} = \bar{r}_a + \alpha \sum_{u \in \mathcal{N}(a)} w(a,u)\,(r_{ui} - \bar{r}_u) \tag{3.1}$$

where $\bar{r}_a$ and $\bar{r}_u$ are the mean ratings given by users a and u respectively, r_{ui} is the rating given by user u to item i, $\mathcal{N}(a)$ is the set of neighbors of user a, $w(a,u)$ is the weight given to user u by

user a based on their similarity, and α is a normalization factor so that the weights sum to 1, i.e. $\alpha = 1/\left[\sum_u w(a,u)\right]$.

In the above definition, the terms *user* and *item* are used in the most general sense. For instance, in social media settings, a *user* could be a member of a social media platform and the *items* recommended could be other members who share similar interests. Similarly, in a patch matching scenario, we propose to use this approach to *recommend* reference patches to a noisy query patch as detailed in the following section.

3.3 Proposed Method

The matching criterion we propose is largely inspired by the recommendation system algorithms, specifically the neighborhood-based collaborative filtering. In order to use collaborative filtering for patch matching, we make the following substitutions to the NN-CF formulation presented in Section 3.2.6:

- Active user $\rightarrow$ Query patch
- Other users $\rightarrow$ Other patches of the noisy image (internal patches)
- Items $\rightarrow$ Clean patches in the external database (external patches)
- Rating given by user u to item i $\rightarrow$ Distance between patch u and patch i

In addition, we will substitute the set of users similar to the active user [$\mathcal{N}(a)$ in eq.(3.1)] with a set of patches that we will refer to as the *close* neighbors of the query patch.

3.3.1 Patch Matching Criterion

Incorporating the substitutions detailed above, the matching criterion for a query patch, q, to a patch in the database, j, is given by

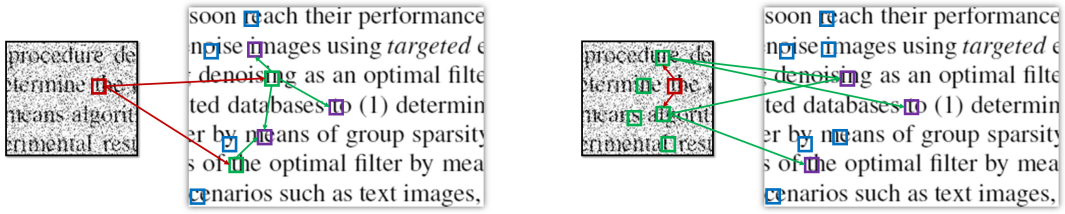
$$\hat{d}_{qj} = d_{qj} + \alpha \sum_{u \in \mathcal{N}(q)} w(q,u)\, d_{uj}, \tag{3.2}$$

where d_{*j} is the distance between patches $*$ and j, $\mathcal{N}(q)$ is the set of *close* neighbors of q, $w(q,u)$ is the weight based on the *closeness* of patch q and patch u, and $\hat{d}_{qj}$ is the corrected dissimilarity measure that is used to compare patches q and j. In the above equation, we have set the bias of the query patch (similar to the $\bar{r}_a$ in eq. (3.1)) to be equal to the distance between the query patch q and patch j, d_{qj}. The bias term for close neighbors is set to zero, leading to the following criterion: for a patch to be selected as a reference patch, it not only has to be similar to the query patch but also to the close neighbors of the query patch. This added constraint gives more control and robustness in the matching stage.

3.3.2 Finding the Set of *Close* Neighbors

We define *close* neighbors as the patches that share close similarities (domain or content) with the query patch. Ideally, these are the patches that we *know* are *similar* to the query (e.g. sparse neighborhood graph). In most practical cases, this prior knowledge may not be available. Hence, in this work, we consider the following two ways of choosing the set of close neighbors that are *expected* to be most similar to the query patch:

1. From external database (Ext-CF): This approach is illustrated in Figure 3.2(a) and outlined in Algorithm 4. Given a noisy query patch, we use its nearest neighbors from the external clean database as close neighbors. This is the same as in the case of external denoising approaches but differs in the number of neighbors selected. For a regular patch-based



(a) CF using external close neighbors

(b) CF using internal close neighbors

Figure 3.2: Collaborative filtering using close neighbors: Query patch is indicated with a red box. Its close neighbors are denoted by red arrows, and reference patches recommended by close neighbors are shown in purple.

Algorithm 4 Patch Matching using Ext-CF

INPUT: Query patch (q)
Database of clean patches (**D**)
Number of reference patches (n)
Number of close neighbors ($m << n$)
OUTPUT: Set of reference patches ($\{p_1, ...p_n\}$)
1. Find the set of close neighbors $\mathcal{N}(q)$: The m nearest neighbors of q from the ***external database*** using the default distance measure (e.g. Euclidean)
2. Get the set of recommended patches from close neighbors
3. Obtain n patches from the set of recommendations that are closest to q based on the modified distance measure defined in eq. (3.2)

external denoising scenario, the number of reference patches chosen is an order of magnitude higher than the the number of close neighbors selected in our approach (i.e. < 10). Our assumption is that the set of immediate neighbors (say, 3 nearest neighbors) of the query patch is not significantly altered by noise and will stay the same in both clean and noisy situations. Based on the substitutions detailed earlier in this section, this method can be thought of as an example of *item-oriented* collaborative filtering [55], i.e. making recommendations based on other items known to be rated highly by the active user.

2. From noisy image (Int-CF): In this approach, as illustrated in Figure 3.2(b) and Algorithm 5, the close neighbors of the query patch are selected from the noisy image itself. Here, the closeness comes from the domain similarity since the patches are all from the same image, therefore sharing the same noise characteristics, image scale, lighting, etc. When the close neighbors are selected from the noisy image itself, the reference patches from the

Algorithm 5 Patch Matching using Int-CF

INPUT: Query patch (q)
Database of clean patches (**D**)
Number of reference patches (n)
Number of close neighbors ($m << n$)
Noisy image (I)
OUTPUT: Set of reference patches ($\{p_1, ...p_n\}$)
1. Find the set of close neighbors $\mathcal{N}(q)$: The m nearest neighbors of q from the ***noisy image*** using the default distance measure (e.g. Euclidean)
2. Get the set of recommended patches from close neighbors
3. Obtain n patches from the set of recommendations that are closest to q based on the modified distance measure defined in eq. (3.2)

external database are forced to be close to both the query patch and other patches in the noisy image that look similar to the query patch. This allows for seamlessly combining the strengths of both internal and external denoising. In terms of CF analogy, this is similar to the *user-oriented* version of neighborhood-based collaborative filtering [51] where recommendations are made based on other users similar to the active user.

3.3.3 Choice of Weights

In collaborative filtering, the ratings of the neighbors are weighted according to their similarities to the active user. Similarly, the term $w(q,u)$ in equation 3.2 weighs the distance between patch j and close neighbor u to control u's contribution to the matching criterion. For this work, we have opted for a simple weighting scheme where the weight $w(q,u)$ is set to 1 if and only if u is a close neighbor of q [54]. That is,

$$w(q,u) = \begin{cases} 1 & \text{if } u \text{ is a close neighbor of } q \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Although we have used the 0-1 weighting scheme for simplicity, it is also possible to use other weighting schemes, such as Pearson Correlation Coefficient, or to learn the weights from the data itself [56].

3.4 Experimental Results

The denoising experiments are conducted on two scenarios: text image denoising and face image denoising. As noted earlier, we use external denoising algorithms, such as TID, external-BM3D, and external-LPG-PCA, that use default patch matching using only the query patch as our baselines and evaluate the improvement that CF-based patch matching approaches provide. The baselines using the default patch matching is denoted as "Reg" in the results. For comparison, we also provide the best-case denoising results of these approaches – i.e. when the underlying clean patch is used for patch matching (assuming the ground truth image is known). This is referred to as "Ideal" in our results and uses the regular patch matching as well. In addition to the external denoising methods, we also compare our results to the internal denoising algorithms BM3D and NLM. The internal denoising methods are indicated as iBM3D and iNLM in our results summary.

The results obtained from using our proposed patch matching criteria are listed as "Ext-CF" and "Int-CF" in the result tables, where "Ext" and "Int" represent the source from which the close neighbors of the query patch were chosen. Specifically, "Int-CF" uses patches from the noisy image as its close neighbors during the collaborative filtering step while "Ext-CF" selects the close neighbors from the external database.

The TID algorithm can be used iteratively by using the output of the previous iteration during the patch matching stage. That is, patches from the denoised image obtained after the first iteration of TID are used as the query patches in the second iteration. In our results, the first and second iterations of TID are denoted as "Stage 1" and "Stage 2," respectively. The same procedure is followed for external-BM3D and external-LPG-PCA. For brevity, we directly show the results obtained at the end of the second stage of these algorithms.

Algorithm parameters

In all experiments, the size of the patches used is set to 8×8; the number of reference patches (n) is fixed at 40, and Euclidean metric is used to measure patch-wise (dis)similarity. For the CF-based methods, the number of close neighbors is set to 3. From the set of patches suggested by the close neighbors (which includes the close neighbors themselves and has no duplicates), the dissimilarity measure given in eq. (3.2) is used to find the final set of 40 nearest neighbors of the query patch. These parameters can be easily tailored based on the noise level and image categories. However, we have kept them constant throughout all our experiments for ease of comparison.

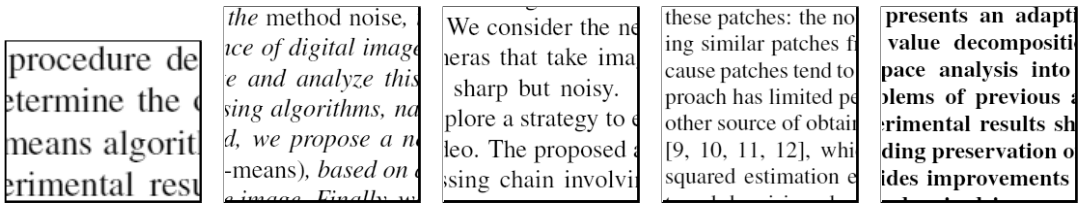
We used the default parameters for iBM3D as suggested in the implementation provided by its authors[1]. For iNLM and Int-CF, internal reference patches are selected from local search windows of size 101×101. The details of the targeted external databases consisting of clean text and face patches will be discussed in respective sections below.

3.4.1 Text Denoising

In the first set of experiments, a text image is denoised using an external database containing other similar but not identical text images. The images in this case are simple binary images with a clean white background and black text. The database images are from completely unrelated sources and vary in font styles and sizes. Some examples from the text image dataset are shown in Figure 3.3. Tables 3.1 and 3.2 summarize the results of the text denoising experiments. Table 3.1 shows that Ext-CF and Int-CF patch matching schemes improve the performance of both stages of the TID algorithm. Ext-CF leads to the best PSNR values under high noise scenarios by leveraging the external close neighbors, while Int-CF gives the highest SSIM measures since it uses information internal to the given image as well as information from the external database.

Table 3.2 indicates that Ext-CF patch matching improves the denoising performance of

[1] http://www.cs.tut.fi/~foi/GCF-BM3D/

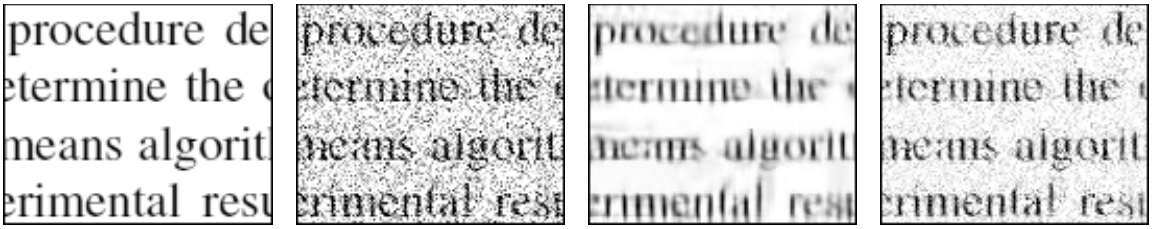


(a) (b) (c) (d) (e)

Figure 3.3: Sample images from text image dataset [23]. The denoising experiments are conducted on noise corrupted versions of image 5.2(a). Five different realizations of Gaussian noise are used for each noise variance. The images 5.2(b)–5.2(e) are four examples out of nine clean text images that constitute the external database.

Table 3.1: Text image results: Average PSNR (in DB) and SSIM measures obtained from TID algorithm using CF-based patch matching. Results that are better than the respective baselines are shown in bold. Ideal case results are not included in the comparison.

				TID: Stage One			**TID: Stage Two**			**With Gnd Truth**
				Baseline	**Proposed**		**Baseline**	**Proposed**		**Best case**
	$\sigma\times$ 255	iBM3D	iNLM	Reg	Ext-CF	Int-CF	Reg	Ext-CF	Int-CF	Ideal
PSNR:	20	28.09	24.24	31.71	31.51	**31.73**	32.13	31.75	32.09	32.26
	30	24.83	21.15	30.60	**30.68**	**30.72**	31.27	31.02	**31.29**	31.58
	40	22.37	19.26	28.55	**28.95**	**28.84**	29.58	**29.68**	**29.71**	30.19
	50	20.60	18.08	26.56	**27.24**	**26.98**	27.97	**28.29**	**28.14**	28.95
	60	19.32	17.29	24.78	**25.62**	**25.29**	26.56	**26.96**	**26.78**	27.99
	70	18.36	16.64	23.16	**24.04**	**23.76**	25.25	**25.72**	**25.46**	27.28
	80	17.60	16.01	21.74	**22.43**	**22.33**	23.98	**24.29**	**24.18**	26.73
SSIM:	20	0.9827	0.7777	0.9812	0.9806	**0.9850**	0.9934	0.9930	**0.9940**	0.9947
	30	0.9622	0.6968	0.9626	**0.9633**	**0.9707**	0.9896	**0.9897**	**0.9913**	0.9934
	40	0.9310	0.6411	0.9384	**0.9409**	**0.9510**	0.9825	**0.9830**	**0.9859**	0.9907
	50	0.8838	0.5996	0.9087	**0.9145**	**0.9259**	0.9717	**0.9733**	**0.9779**	0.9878
	60	0.8417	0.5643	0.8726	**0.8852**	**0.8979**	0.9575	**0.9605**	**0.9680**	0.9849
	70	0.8015	0.5309	0.8288	**0.8514**	**0.8623**	0.9368	**0.9422**	**0.9533**	0.9823
	80	0.7635	0.4980	0.7854	**0.8097**	**0.8210**	0.9110	0.9103	**0.9315**	0.9801



(a) Original (b) Noisy ($\sigma = \frac{80}{255}$) (c) iBM3D (17.52, 0.7571) (d) iNLM (16.00, 0.4944)

(e) eBM3D+baseline (21.80, 0.8812) (f) eBM3D+Ext-CF (22.21, 0.8752) (g) eBM3D+Int-CF (21.50, 0.8827) (h) eBM3D+Ideal (26.40, 0.9466)

(i) eLPG-PCA+baseline (22.17, 0.7944) (j) eLPG-PCA+Ext-CF (22.76, 0.8040) (k) eLPG-PCA+Int-CF (22.22, 0.7973) (l) eLPG-PCA+Ideal (24.17, 0.8304)

(m) TID+baseline (23.92, 0.9119) (n) TID+Ext-CF (24.38, 0.9135) (o) TID+Int-CF (23.95, 0.9284) (p) TID+Ideal (26.77, 0.9798)

Figure 3.4: Visual and objective comparison of denoising performance of a text image. Patch matching criterion used is indicated after the "+" sign. The objective evaluation metrics of each case is shown in parenthesis in (PSNR, SSIM) format.

Table 3.2: Text image results: Average PSNR (in DB) and SSIM measures obtained from external-BM3D and external-LPG-PCA using CF-based patch matching. Results that are better than the respective baselines are shown in bold. Ideal case results are not included in the comparison.

				External-BM3D				**External-LPG-PCA**			
				Baseline	**Proposed**		**Best case**	**Baseline**	**Proposed**		**Best case**
	$\sigma\times$ 255	iBM3D	iNLM	Reg	Ext-CF	Int-CF	Ideal	Reg	Ext-CF	Int-CF	Ideal
PSNR:	20	28.09	24.24	28.79	**28.85**	28.74	31.14	32.70	32.53	32.69	32.94
	30	24.83	21.15	27.00	**27.08**	26.78	29.51	30.27	30.23	30.27	30.61
	40	22.37	19.26	25.74	**25.87**	25.37	28.50	28.18	**28.30**	**28.25**	28.77
	50	20.60	18.08	24.71	**24.79**	24.17	27.78	26.40	**26.63**	**26.45**	27.27
	60	19.32	17.29	23.74	**23.90**	23.11	27.22	24.84	**25.18**	**24.91**	26.02
	70	18.36	16.64	22.90	**23.01**	22.25	26.75	23.48	**23.82**	**23.57**	25.03
	80	17.60	16.01	21.94	**22.07**	21.35	26.35	22.25	**22.58**	**22.26**	24.17
SSIM:	20	0.9827	0.7777	0.9563	**0.9567**	0.9572	0.9662	0.9689	**0.9718**	**0.9703**	0.9765
	30	0.9622	0.6968	0.9467	**0.9476**	0.9470	0.9588	0.9435	**0.9473**	**0.9438**	0.9553
	40	0.9310	0.6411	0.9384	**0.9397**	0.9379	0.9546	0.9153	**0.9186**	**0.9178**	0.9303
	50	0.8838	0.5996	0.9294	**0.9297**	0.9270	0.9520	0.8845	**0.8923**	**0.8883**	0.9083
	60	0.8417	0.5643	0.9175	**0.9195**	0.9149	0.9501	0.8538	**0.8644**	**0.8555**	0.8814
	70	0.8015	0.5309	0.9049	0.9036	0.9016	0.9486	0.8208	**0.8319**	**0.8268**	0.8570
	80	0.7635	0.4980	0.8841	0.8780	**0.8846**	0.9473	0.7903	**0.8010**	**0.7956**	0.8359

external-LPG-PCA and external-BM3D. The other proposed approach, Int-CF, improves the baseline version of LPG-PCA consistently as well.

In Figure 3.4, we display the denoised images from one of the high noise settings for visual comparison. Visually and objectively, the methods that use the proposed patch matching schemes yield better denoising results compared to their respective baseline methods.

3.4.2 Face Image Denoising

Face image denoising is an example of a setting where the database images and the image to be denoised are not a perfect match. In this setting, we denoise a face image with a database of face images of other people. This setting also displays variations in lighting, contrast, gender, etc. Some examples demonstrating the diversity of the image dataset are shown in Figure 3.5. For our experiments, we randomly picked five images and added Gaussian noise. These noisy images were then denoised using an external database consisting of ninety face images from different individuals.

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

Figure 3.5: Sample images from FEI face image dataset [57]. The image 5.3(a) is one of five test images used in our experiments. The rest of the images, 5.3(b)–5.3(e), are four examples out of ninety clean images that constitute the clean external database. All face images in our experiments have a size of 90×65.

The denoising results obtained from this dataset are presented in Tables 3.3 and 3.4. In this experiment, since the image to be denoised does not closely match the database images, the Int-CF results are improved compared to other methods, including those obtained from Ext-CF. This can be attributed to the power of combining both internal and external information while denoising an image. Since the external database is different from the image to be denoised, it is important to leverage as much information as possible from the internal image to find the best matching patches. Figure 3.6 shows the denoising results of one of our test images. Our patch matching schemes yield comparable visual quality to the baseline algorithms while giving higher PSNR and SSIM values.

3.5 Discussion

Currently, patch-based denoising algorithms use standard metrics for patch matching that are not robust to noise. In the absence of a robust matching scheme, the patches retrieved using the noise-corrupted query patch hinders the overall denoising process. We have introduced a patch matching scheme where a query patch takes suggestions from other close neighbors. This collaborative approach to finding similar patches lends a degree of robustness to noise that a single pairwise distance fails to provide.

Table 3.3: FEI face dataset results: Average PSNR (in DB) and SSIM measures obtained from TID algorithm using CF-based patch matching. Results that are better than the respective baselines are shown in bold. Ideal case results are not included in the comparison.

				TID: Stage One			**TID: Stage Two**			**With Gnd Truth**
				Baseline	**Proposed**		**Baseline**	**Proposed**		**Best case**
	$\sigma\times$ 255	iBM3D	iNLM	Reg	Ext-CF	Int-CF	Reg	Ext-CF	Int-CF	Ideal
PSNR:	20	31.38	26.67	31.65	31.62	**31.75**	32.06	31.80	31.99	34.16
	30	29.45	24.98	29.64	29.54	**29.79**	30.40	30.22	**30.42**	33.74
	40	27.53	23.26	27.37	27.27	**27.57**	28.52	28.44	**28.61**	33.46
	50	26.88	21.94	25.89	25.75	**26.06**	27.30	27.14	**27.36**	33.46
	60	25.80	20.62	24.59	24.36	**24.76**	25.94	25.85	**26.06**	33.33
	70	25.23	19.53	23.52	23.30	**23.70**	25.08	24.94	**25.23**	33.41
	80	24.43	18.59	22.73	22.48	**22.88**	24.30	24.11	**24.39**	33.42
SSIM:	20	0.9028	0.7258	0.8976	**0.9003**	**0.9028**	0.9143	0.9099	**0.9142**	0.9495
	30	0.8601	0.6305	0.8412	**0.8440**	**0.8506**	0.8772	0.8738	**0.8796**	0.9448
	40	0.8118	0.5239	0.7622	0.7615	**0.7719**	0.8264	0.8261	**0.8310**	0.9433
	50	0.7921	0.4464	0.6955	0.6941	**0.7072**	0.7818	0.7755	**0.7889**	0.9428
	60	0.7599	0.3780	0.6470	0.6387	**0.6582**	0.7407	0.7386	**0.7483**	0.9431
	70	0.7217	0.3177	0.5821	0.5750	**0.5935**	0.6852	0.6795	**0.6964**	0.9423
	80	0.7123	0.2751	0.5555	0.5464	**0.5666**	0.6716	0.6639	**0.6816**	0.9428

Table 3.4: FEI face dataset results: Average PSNR (in DB) and SSIM measures obtained from external-BM3D and external-LPG-PCA using CF-based patch matching. Results that are better than the respective baselines are shown in bold. Ideal case results are not included in the comparison.

				External-BM3D				**External-LPG-PCA**			
				Baseline	**Proposed**		**Best case**	**Baseline**	**Proposed**		**Best case**
	$\sigma\times$ 255	iBM3D	iNLM	Reg	Ext-CF	Int-CF	Ideal	Reg	Ext-CF	Int-CF	Ideal
PSNR:	20	31.38	26.67	31.62	31.46	31.44	35.86	31.70	31.49	31.63	33.39
	30	29.45	24.98	30.00	29.81	29.88	34.75	29.80	29.61	29.79	31.81
	40	27.53	23.26	28.36	28.17	28.34	34.05	27.70	27.47	**27.77**	30.04
	50	26.88	21.94	27.16	27.03	27.14	33.56	26.23	25.87	**26.25**	28.69
	60	25.80	20.62	25.98	25.81	**25.98**	33.17	24.73	24.47	**24.87**	27.47
	70	25.23	19.53	25.14	24.97	**25.21**	32.90	23.57	23.10	**23.71**	26.28
	80	24.43	18.59	24.40	24.20	**24.45**	32.56	22.76	22.23	**22.88**	25.20
SSIM:	20	0.9028	0.7258	0.9060	0.9026	0.9025	0.9595	0.9006	0.8945	0.9003	0.9265
	30	0.8601	0.6305	0.8693	0.8655	0.8679	0.9497	0.8492	0.8441	**0.8503**	0.8898
	40	0.8118	0.5239	0.8241	0.8197	**0.8250**	0.9421	0.7813	0.7670	**0.7836**	0.8434
	50	0.7921	0.4464	0.7796	0.7734	**0.7843**	0.9365	0.7196	0.6973	**0.7201**	0.7994
	60	0.7599	0.3780	0.7479	0.7415	**0.7531**	0.9321	0.6651	0.6402	**0.6691**	0.7637
	70	0.7217	0.3177	0.6952	0.6852	**0.7048**	0.9272	0.5919	0.5709	**0.5981**	0.7077
	80	0.7123	0.2751	0.6811	0.6740	**0.6916**	0.9240	0.5605	0.5303	**0.5709**	0.6720

(a) Original

(b) Noisy ($\sigma = \frac{40}{255}$)

(c) iBM3D (27.79, 0.8130)

(d) iNLM (23.22, 0.5194)

(e) eBM3D+baseline (29.37, 0.8477)

(f) eBM3D+Ext-CF (29.17, 0.8422)

(g) eBM3D+Int-CF (29.44, 0.8451)

(h) eBM3D+Ideal (35.41, 0.9496)

(i) eLPG-PCA+baseline (28.57, 0.8117)

(j) eLPG-PCA+Ext-CF (28.38, 0.7992)

(k) eLPG-PCA+Int-CF (28.77, 0.8251)

(l) eLPG-PCA+Ideal (31.25, 0.8741)

(m) TID+baseline (29.49, 0.8526)

(n) TID+Ext-CF (29.46, 0.8539)

(o) TID+Int-CF (29.68, 0.8579)

(p) TID+Ideal (35.10, 0.9541)

Figure 3.6: Visual and objective comparison of denoising performance of a face image. Patch matching criterion used is indicated after the "+" sign. The objective evaluation metrics of each case is shown in parenthesis in (PSNR, SSIM) format.

Our results indicate that using CF-based patch matching schemes are effective in noisy scenarios. The most convincing evidence is the improvement they provide in the first stage of TID when using the noisy query patch for patch matching.

The performance boosts of LPG-PCA and BM3D under the new patch matching schemes are lower than TID. This is due to the difference in roles of the reference patches in the respective denoising filters. In TID, reference patches contribute to both the transformation and shrinkage filters. Whereas, in LPG-PCA reference patches contribute to the transformation matrix, and in BM3D, their effect is more indirect (via the transform domain coefficients). The more dependent a denoising filter is on the reference patches, the more pronounced is the effect of patch matching on its performance.

Int-CF, the approach of consulting internal close neighbors before conducting a patch matching on the external database, provides a novel way to leverage both internal and external patch information, thus, seamlessly combining the internal and external denoising. It could be argued that, in contrast to Int-CF, a straightforward way to combine internal and external denoising is to directly combine patches from the noisy image and the external database while designing denoising filters. However, this strategy will not be successful. The algorithms presented in this work, such as TID and LPG-PCA, design their denoising filters in a data-adaptive manner. Therefore, any noise in the set of reference patches can be detrimental to their performance. Thus, using purely external patches is a much better solution than directly combining external and internal patches. The addition of noisy patches in the external BM3D will also degrade the quality of the denoising filter but to a lesser degree. In short, the performance improvements obtained by Int-CF patch matching over the default matching strategy demonstrates its ability to successfully fuse information from both internal and external patches.

Finally, Ext-CF can be leveraged to quickly narrow down the search space in a large external database. By using an appropriately indexed large database with all the pairwise distances precalculated, Ext-CF allows us to focus on only those patches in the database that are nearest

neighbors of the close neighbors. A thorough evaluation of the speed-up versus performance obtained through this approach is one of the directions of future research.

3.6 Conclusion

In this chapter, we present a novel patch matching criterion taking inspiration from neighborhood-based collaborative filtering approaches. We demonstrate the effectiveness of the new patch matching scheme by applying it to denoising images from two different categories displaying different levels of ease and complexity. Our experiments show that the collaborative filtering-based patch matching improves the performance of state-of-the-art patch-based denoising algorithms by picking better reference patches.

In the next chapter, we will demonstrate a strategy to efficiently leverage temporal correspondences between patches to extend patch-based TID to video denoising.

Acknowledgments

Parts of this chapter is a reprint of materials in the journal paper by “Patch Matching for Image Denoising Using Neighborhood-based Collaborative Filtering”, S. Parameswaran, E. Luo and T. Q. Nguyen [2]. The dissertation author is the primary investigator and author of this paper.

Chapter 4

Targeted Video Denoising for Decompressed Videos

4.1 Introduction

Patch-based denoising algorithms such as non-local means (NLM) [58] and BM3D [59], originally introduced to denoise images, have been successfully adapted for video denoising [60, 61, 62]. These approaches assume that images (or videos) contain repeated structures which promises the existence of mutually similar patches within an image/video. For image denoising, similar patches are obtained by searching a 2D neighborhood around the pixel (or patch) being processed. However, for videos, a 3D region around a patch involving its spatial and temporal neighborhoods is searched to find similar patches [60, 61, 62, 63]. Since the similar patches used during denoising are obtained from within an image (or video), these methods fall into a category known as *internal denoising* algorithms. Although very efffective, internal denoising approaches do have some limitations when denoising rare patches [64, 65] and when operating under high noise scenarios. Rare patch effect is nominally alleviated in videos by using patches from multiple frames. However, since similar patches are selected from within the noisy video,

the limitations that arise under high noise scenarios still remain a challenge.

In the case of image denoising, researchers have shown that these issues can be ameliorated by using *external* denoising algorithms where similar patches are obtained from a large external database of clean patches [66, 67, 68, 69]. Recently, Luo *et al.* [70, 71] argued for the use of a *targeted* database that incorporated prior knowledge about the scene in an image instead of using large *generic* databases. Their method, known as targeted image denoising (TID), achieved impressive gains in denoising performance by using targeted databases over state-of-the-art internal denoising algorithms like NLM and BM3D, and other external denoising algorithms that use generic databases [70, 71].

In this chapter, our focus is on denoising decompressed videos corrupted during transmission. To this end, we adapt TID algorithm to perform video denoising, thus incorporating the advantages of the state-of-the-art external denoising into the video denoising domain. Temporal information contained in a video sequence is leveraged by re-purposing the motion vectors available in a compressed video to establish correspondences between patches in neighboring frames. We evaluate the extended TID algorithm on multiple decompressed video sequences and show that our adaptation makes the already powerful denoising filter more suited for video denoising with virtually *no* added overhead.

Contributions: The contributions of this work are as follows. First, we extend the state-of-the-art TID algorithm to video denoising paradigm. Second, we avoid introducing additional complexity per frame while finding the temporal neighborhood by using the sparse motion vectors extracted from the bitstream of the compressed video. Use of sparse motion vectors instead of dense optical flow or predictive block matching keeps the per-frame complexity of the video denoising algorithm the *same* as that of the underlying image denoising algorithm with only minor reduction in denoising performance.

4.2 Related Work

4.2.1 Establishing temporal coherence in video denoising

A critical step in adapting an image denoising algorithm to videos is exploiting the temporal coherency among video frames while searching for similar patches. Taking advantage of this similarity between consecutive frames has been key to the success of existing patch-based video denoising algorithms [60, 61, 63, 62, 72, 73, 74]. Temporal coherence was loosely enforced in the original NLM extension [60] by simply searching for patches in a 3D spatio-temporal volume centered around the query patch. Some researchers have argued that explicit motion estimation using optical flow to incorporate temporal coherence provides better results for real-world videos [61, 74]. Video extension of BM3D [62] avoids explicit motion estimation. It uses a predictive-search block matching scheme that searches for similar patches in spatio-temporal volumes that are adapted based on the query patch (data-adaptive). All these approaches of involving the temporal domain adds more complexity to the denoising process. For compressed videos, we show that this added complexity can be avoided by re-using the already available motion vectors present in the bitstream of the compressed video.

4.2.2 Targeted Image Denoising

As explained in earlier chapters, TID designs data-adaptive optimal denoising filters maximally utilizing the information contained in patches in the given targeted database. First, TID retrieves clean patches similar to the noisy patch under consideration (query patch). Then, the retrieved similar patches are used to learn an optimal denoising filter by solving a group sparsity minimization problem and using a localized Bayesian prior. Please refer to [71] for a detailed explanation and derivation of the TID filter.

The data-adaptive nature of the TID denoising filter makes selection of similar patches from the database very important. We improve the efficacy of TID on videos by leveraging

Algorithm 6 Targeted Video Denoising

INPUT: Query patch (q), Motion vectors (v_X, v_y), Database of clean patches (**D**), Noise variance (σ^2)

OUTPUT: Denoised patch ($\hat{p}$)

1. Obtain the temporal neighbors of the query patch q based on motion vectors (e.g. $q^{t\pm1}$)
2. Find n reference patches p_1, p_2, ... p_n from the database using q and its temporal neighbors
3. Form data matrix by concatenating patches retrieved using q and using its temporal neighbors ($q^{t\pm1}$)

$$\mathbf{P} = \left[\mathbf{P^{t-1}}, \mathbf{P^{t}}, \mathbf{P^{t+1}}\right] = [p_1, p_2, \dots p_n]$$

4. Form weight matrix: $\mathbf{W} = \frac{1}{\alpha}\,\mathrm{diag}\{w_1, w_2, \dots, w_n\}$

where $w_i = \exp\left(-\frac{\|q-p_i\|^2}{h^2}\right)$, h is user-tunable bandwidth parameter and α is normalization parameter so that the weights add up to 1.

5. Perform eigen-decomposition: $[\mathbf{U}, \mathbf{S}] = \mathrm{eig}\left(\mathbf{PWP}^T\right)$
6. Perform shrinkage: $\Lambda = \left(\mathrm{diag}\left(\mathbf{S} + \sigma^2\mathbf{I}\right)\right)^{-1}\mathrm{diag}(\mathbf{S})$
7. Denoise q: $\hat{p} = \mathbf{U}\Lambda\mathbf{U}^T\, q$

temporal coherence between video frames during patch matching.

As assumed by Luo et al. [70, 71], we also assume that a targeted database is given. Choosing an appropriate database can be a challenging problem in itself and is out of the scope of this study. However, please note that in the absence of a targeted database, a very large generic database of clean patches can be used without any change to the underlying algorithm.

4.3 Targeted Video Denoising

Let $v(i)$ and $u(i)$ be the observed video signal and original signal, respectively, at the spatio-temporal index $i = (x, y, t)$. Here, we consider the case where the noise signal $\eta(i)$ is i.i.d. Gaussian with zero mean and variance σ^2. That is, $v(i) = u(i) + \eta(i)$, where $\eta(i) \sim \mathcal{N}(0, \sigma^2)$. The variance σ^2 is assumed to be known *a priori*.

An outline of the extension of TID for videos, Targeted Video Denoising (TVD) algorithm, is shown in Algorithm 6. For the remainder of this section we provide detailed explanation of the modifications made to the original TID algorithm.

We start our adaptation of TID by establishing and identifying temporal correspondences

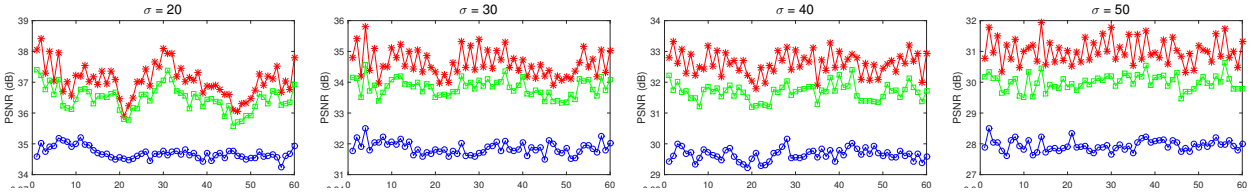


Figure 4.1: Performance comparison on **Miss America** sequence (First 60 frames): Per-frame PSNR comparison of the proposed TVD algorithm ('$*$' marker) with the VBM3D algorithm ('$\circ$' marker) for different noise levels. For reference, results obtained using the original TID applied on a per-frame basis with no temporal assistance is also shown ('$\square$' marker).

Figure 4.2: Frame 5 of the **Miss America** sequence: Visual comparison of a frame corrupted by AWGN of $\sigma = 30$ (top left), the original frame (top right), the VBM3D result (bottom left) and the output of the proposed method (bottom right)

between patches in consecutive frames (**step 1** of algorithm 6). Previously, temporal correspondence between patches of a frame and its neighboring frames were calculated using optical flow [61] or data-adaptive predictive-search block matching [62] or block matching in a non-adaptive fixed window [60]. These methods add considerable overhead since these operations are carried out for each video frame. To circumvent this problem we propose using motion vectors that are available in a compressed video stream, which are generated during video compression.

Video compression schemes such as MPEG-4 rely heavily on motion estimation to exploit the high spatial and temporal redundancy present in consecutive frames of a video. The estimated motion vectors are used to perform motion-compensated frame differencing to obtain better compression of information contained in videos. Compressed video contains frames that fall

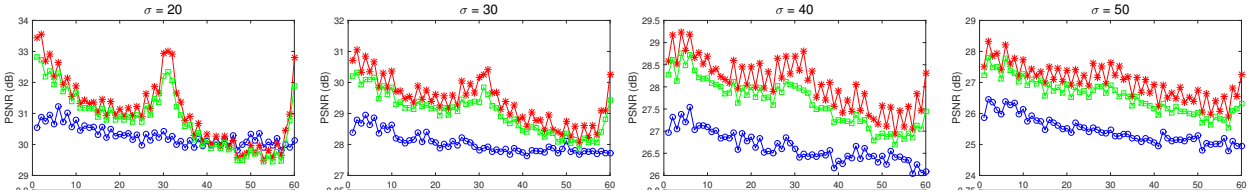


Figure 4.3: Performance comparison on **Coastguard** sequence (First 60 frames): Per-frame PSNR comparison of the proposed TVD algorithm ('$*$' marker) with the VBM3D algorithm ('$\circ$' marker) for different noise levels. For reference, results obtained using the original TID applied on a per-frame basis with no temporal assistance is also shown ('$\square$' marker).

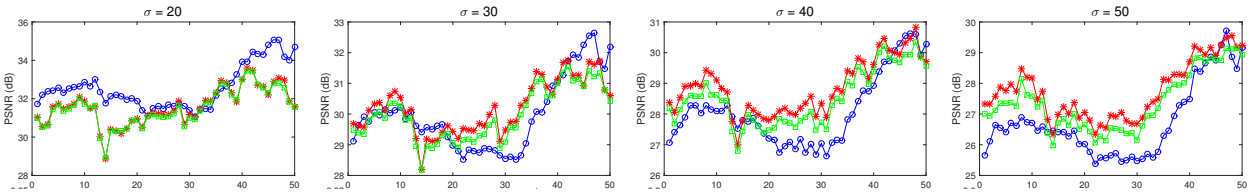


Figure 4.4: Performance comparison on **Diving** sequence (55 frames): Per-frame PSNR comparison of the proposed TVD algorithm ('$*$' marker) with the VBM3D algorithm ('$\circ$' marker) for different noise levels. For reference, results obtained using the original TID applied on a per-frame basis with no temporal assistance is also shown ('$\square$' marker).

into three different picture types: P, B and I frames. The P frames use data from previous frame and have motion vectors that relate the current frame to its previous frame. Likewise, the B frames use data from both its previous and next frames and hence have motion vectors in both temporal directions. Since these motion vectors are encoded in compressed video representations, they can be accessed during decompression with no additional cost. We propose using these motion vectors to identify the temporal neighbors of patches in P and B frames. The I frames are intra-coded and not dependent on past or future frames. They are similar to static images, with no motion vector information. These frames can be either denoised as an image without using temporal information or subjected to optical flow calculation.

Unlike the dense optical flow, which is defined for all pixels, motion vectors used in a video compression scheme are usually sparse and are defined per-macroblock instead of per-pixel – e.g. a single motion vector for each 16x16 block. Therefore, every pixel in a macroblock has the same motion vector. Although these coarse motion fields may not capture the true motion of every single pixel of a frame, they provide a good approximation and can be easily incorporated

into the denoising pipeline with no additional complexity.

After establishing a temporal correspondence between pixels of consecutive frames, each patch can be associated to its counterpart in temporal directions using the motion vectors of its central pixel. This patch correspondence is then used during patch matching as follows (**steps 2-3** of algorithm 6): let the set of patches retrieved from the targeted database using the query patch q of frame t be $\mathbf{P^t}$ and the sets retrieved using q's temporal neighbors ($q^{t\pm1}$) at time $t-1$ and $t+1$ be $\mathbf{P^{t-1}}$ and $\mathbf{P^{t+1}}$ respectively. Then, the data matrix $\mathbf{P} = \left[\mathbf{P^{t-1}}, \mathbf{P^t}, \mathbf{P^{t+1}}\right]$; that is, the data matrix used to design the denoising filter is formed by taking the union of the sets of similar patches retrieved using q and all of its temporal neighbors. In addition to providing a certain degree of temporal consistency, this augmentation utilizing patch correspondence also provides robustness to random noise.

The rest of the denoising algorithm (**steps 4-7** of algorithm 6) identical to the original TID algorithm [70, 71].

4.4 Experimental Results

We tested our algorithm on three grayscale videos with different characteristics. For our first set of experiments we use the well-known “Miss America" and “Coastguard" benchmark sequences. Miss America is a fairly static scene featuring small movements, whereas Coastguard features object movement and tracking. The third video sequence tested is footage of a diver (video # 007) taken from the UCF sports action dataset [75, 76]. The test sequences were created by adding varying amounts of Gaussian noise to the clean video frames.

Targeted database: We assume that either three noise-free images (frames) from the long video or video of the same setting captured on a different occasion are available. For example, three images shot while a surveillance video is being recorded which are sent or saved for offline enhancement of the video in the future. With this application scenario in mind, we use the Miss

America and Coastguard sequences to demonstrate the efficacy of TVD under the more *ideal* of these two scenarios. That is, for these videos, the targeted database was created using three random frames from their respective clean videos. We use the diving sequence to demonstrate a more realistic scenario where a targeted database is created using an earlier video shot in a similar setting. In particular, for the diving video (video # 007), we use a *different* video featuring a *different* diver as the targeted database (video # 005 from UCF sports). This is a more realistic setting where we have access to clean videos that are *similar* but *different* from the noisy video. To minimize the database size for the diving sequence, we pick a moving window of three frames from video # 005 to be the targeted database.

Parameter settings: We use the same parameter settings for all the experiments. The targeted video denoising algorithm is run twice where the result from the first iteration is used for finding similar patches in the second iteration. The patch size is set to 8x8 pixels. We retrieve a total of 120 similar patches from the database for every noisy patch ($n = 120$). Therefore, if a query patch has only one temporal neighbor then both of them will be used to retrieve 60 reference patches each. We compare our results with the state-of-the-art VBM3D algorithm [62]. We use the VBM3D implementation from the BM3D website (http://www.cs.tut.fi/ foi/GCF-BM3D/). For all the experiments, we used the default parameters except for the temporal window size which is set to 3 (by default this is set to 9 in the VBM3D package). To demonstrate the improvement obtained using temporal correspondences, we also compare our results obtained from TID algorithm, with $n = 120$, applied on a per-frame basis without using any temporal knowledge.

The **Miss America** sequence is of QCIF resolution (144x180). The per-frame denoising results obtained on this video sequence are shown in Figure 4.1. Qualitatively and quantitatively, the proposed algorithm clearly outperforms VBM3D and TID on enhancing this video in all the noise settings we tested. TVD achieves a global PSNR (calculated on the whole sequence) improvement of around 2.4-3.1 dB and global SSIM improvement of 0.04-0.1 over VBM3D. For

visual comparison, one of the frames from the $\sigma = 30$ noise setting is shown in Figure 4.2.

The **Coastguard** video contains moving objects and also features egomotion due to the camera tracking a moving boat. The resolution of this video is 144x176 pixels. The per-frame comparison of the PSNR values obtained after denoising this sequence with TVD, TID and VBM3D are presented in Figure 4.3. The spikes in the performance curves of the proposed algorithm and TID are due to the location of the frames used as the targeted database. However, this advantage is canceled out in the high noise scenarios (e.g. $\sigma = 50$) because of the difficulty in finding true matches using severely corrupted patches. The global PSNR improvements shown by TVD over VBM3D range from 0.8-1.6 dB (Δglobal-SSIM = 0.04-0.09) as the noise is increased from $\sigma = 20$ to 50.

We downsampled the **diving** sequence from the UCF sports action dataset [75, 76] from a spatial resolution of 404x720 to 101x180. Both the testing sequence and the database sequence have 55 frames each. The per-frame denoising results obtained on this video sequence are presented in Figure 4.4. Under the low noise setting ($\sigma = 20$), although TVD has a slightly higher global SSIM than VBM3D (0.003), global PSNR of VBM3D is better than TVD by 1.1 dB. More importantly, the proposed approach outperforms VBM3D for higher noise settings with $\sigma \geq 30$ yielding global PSNR improvements of 0.4-1.2 dB (Δglobal-SSIM = 0.02-0.07).

Runtime: The proposed TVD algorithm has the same runtime complexity as TID since the extraction of motion vectors does not add any noticeable overhead. Our current implementation of TID denoising algorithm takes approximately 60s per frame in the diving sequence (101x180). The runtimes of TID and TVD can be optimized by using large-scale approximate nearest neighbor algorithms for patch matching such as the RIANN [77] or PatchTable [78] algorithms.

4.5 Conclusion

We have introduced a patch-based video denoising algorithm by extending the TID algorithm [71, 70]. In order to take advantage of the similarity among video frames in a temporal neighborhood, we augmented the localized Bayesian prior of TID with temporal coherency prior. Using the motion vectors present in the compressed bitstream, we denoise decompressed videos with no added complexity. Our results demonstrate that targeted video denoising consistently outperforms its image processing counterpart, TID, and the state-of-the-art internal video denoising algorithm VBM3D in mid- and high-noise conditions.

Even though the video extension does not overburden the runtime complexity, the underlying image processing algorithm, TID, takes around one minute per frame of size 101×180. This is prohibitively slow and cannot be used for large scale applications such as video processing. Therefore, next chapter tackles the problem of designing a fast but competent denoising algorithm.

Acknowledgments

Parts of this chapter is a reprint of materials in the conference paper "Targeted video denoising for decompressed videos", S. Parameswaran, E. Luo and T. Q. Nguyen [3]. The dissertation author is the primary investigator and author of this paper.

Chapter 5

Fast external denoising using pre-learned transformations

5.1 Introduction

External denoising algorithms alleviate the shortcomings of internal methods such as difficulty in handling high noise level and rare patches [14]. However, external (both generic and targeted) methods are either slower to converge to an acceptable solution or are computationally expensive than internal methods. EPLL [15] is one of the most popular external denoising algorithms but it is slow to produce a good quality solution requiring many iterations with heavily overlapped patches. Additionally, in each iteration of EPLL, one has to calculate the patch log-likelihood of the patch under the learned GMM prior. This involves calculating Mahalanobis distance using the covariance matrices of each mixture component which is computationally more complex than calculating Euclidean distances. The TID filter produces a higher quality image within 2-3 iterations but is computationally expensive. Depending on the size of the image and the database used, TID processing time is usually orders of magnitude slower than EPLL. These issues make these successful external denoising methods unsuitable for processing videos and

large images.

In this chapter, we propose a fast external denoising algorithm that we refer to as FED. This algorithm is efficient during runtime and can be trained with ease and flexibility.

5.2 Background and Related Work

The proposed approach falls under the external denoising category. An external denoising algorithm uses information from outside the noisy image when estimating the denoising filter. In this section we provide a brief overview of two external denoising algorithms that inspired the development of our method.

5.2.1 Targeted Image Denoising Filter

TID designs optimal denoising filters for each noisy patch in the image by utilizing the given targeted database maximally. To this end, for each noisy patch, TID retrieves clean patches similar to the noisy patch and obtains an optimal denoising filter for it by solving a group sparsity minimization problem and using a localized Bayesian prior [23].

The data-adaptive nature of TID makes it very effective. However, designing a new filter for each patch during runtime can lead to excessive computational complexity as the size of the noisy image increases. The complexity of TID is also dependent on the size of the database and the number of similar patches used for designing the filter. This makes TID unsuitable for large images and videos.

5.2.2 Expected Patch Log Likelihood

Expected Patch Log Likelihood (EPLL) [15] is another successful external denoising algorithm. EPLL models the patch priors using Gaussian Mixture Models from an external database of *generic* patches. These learned patch priors are used to denoise images efficiently.

This simple yet powerful algorithm has been shown to outperform both internal denoising algorithms such as BM3D [8] and external denoising algorithms such as external-NLM. The proposed method is similar to the EPLL algorithm, however the denoising filter proposed in this paper is more powerful when a targeted database is available.

5.2.3 Generic vs. targeted database of patches

As metioned above, TID uses an external database composed of patches from images that are visually similar to the noisy image that are being denoised. Such a database that closely resembles the characteristics of the test (noisy) image is referred to as the targeted database. Whereas, the Gaussian Mixture Models in the original EPLL study is learned from a large database of patches (2×10^6 patches) taken from images in the Berkeley Segmentation Database [79]. A patch database that contains patches from a wide range of images from various domains is termed a generic database. In general, using a targeted database is better than relying on a generic database.

Choosing an appropriate database with *relevant* images can be a challenging problem in itself and is an active research area. This direction of research requires a thorough evaluation of recent developments in image retrieval and grouping algorithms such as deep learning for image retrieval [80] and web scale photo clustering [81]. However, a careful evaluation of database construction methods is out of the scope of this study and we leave it for future work. Although our algorithm is designed to be used with a targeted database, we also include results from experiments using a generic database to give the reader an idea of the performance of our algorithm in this unideal setting.

5.3 Proposed Method

In this section, we describe the proposed denoising approach. It is based on a whole image denoising formulation that is regularized by targeted patch-based denoising estimate. First, let us define a model for the noisy image as $y = x + \eta \in \mathbb{R}^N$ where η is the i.i.d. Gaussian noise with zero-mean and variance σ^2, i.e., $\eta \sim \mathcal{N}(0, \sigma^2 I_N)$. Let us also define a patch extractor operator $P_i \in \mathbb{R}^{d \times N}$ that extracts i-th patch from x, i.e., $P_i x \in \mathbb{R}^d$ is a d-dimensional vector obtained by lexicographic ordering of $\sqrt{d} \times \sqrt{d}$ patch extracted from the image x.

5.3.1 Whole image denoising

Given a noisy image y, the noise variance σ^2, and a set of targeted image patches $A \in \mathbb{R}^{d \times k}$ of k patches, we propose to optimize the following cost function to estimate the underlying clean image, x.

$$\min_{x,\{z_i\}} \frac{1}{2\sigma^2}\|x-y\|_2^2 + \frac{\beta}{2}\left[\sum_{i=1}^{N} \|P_i x - A z_i\|_2^2 + \lambda \|z_i\|_2^2\right] \tag{5.1}$$

The above formulation ensures that the estimated image as a whole closely resembles the original noisy image y (first term), and the individual patches of the estimated image can be expressed as a linear combination of the patches in the patch-matrix A with minimum error. The optimization parameter β controls the relative contribution of the patch-by-patch reconstruction strategy of the second term to our overall goal of denoising the whole image, and the parameter λ ensures the existence of feasible solutions for the set of coefficient vectors $\{z_i | i = 1 \ldots N\}$.

The solution for the minimization in eq. (5.1) can be carried out by alternating between:

1. Fixing x and finding optimal set of $\{z_i\}$ using the assumption that z_i's are mutually independent.

2. Fixing $\{z_i\}$ and solving for x.

We defer the details of the first of these two steps (finding optimal $\{z_i\}$) to Section 5.3.2 below. Assuming we have the optimal solutions for all coefficient vectors $\{z_i | i = 1 \dots N\}$, we will first discuss solving for the optimal x.

Fixing all $\{z_i\}$ and solving for optimal x leads to

$$\widehat{x} = \left(\frac{1}{\sigma^2} I_N + \beta \sum_{i=1}^{N} P_i^T P_i \right)^{-1} \left(\frac{1}{\sigma^2} y + \beta \sum_{i=1}^{N} P_i^T A z_i \right) \tag{5.2}$$

where the matrix P_i^T performs the complementary operation of P_i of placing a patch back as the i-th patch of the image, and $\sum_{i=1}^{N} P_i^T P_i$ counts the number of estimates obtained for each pixel in the image. Therefore, the solution of the whole image denoising shown in eq. (5.2) is simply a weighted average of the given noisy image and the image estimate obtained by denoising individual patches independently.

5.3.2 Patch denoising

We now focus on the solution for optimal $\{z_i\}$ obtained by fixing x. The optimal z_i can be found by solving the following minimization problem for each i separately:

$$\min_{z_i} \|P_i x - A z_i\|_2^2 + \lambda \|z_i\|_2^2 \tag{5.3}$$

Here, the second term is added for regularization purposes and to ensure a unique solution. The solution of eq. (5.3) has a closed-form expression which leads to estimate $A\widehat{z_i}$ as:

$$A\widehat{z_i} = A \left(A^T A + \lambda I_k \right)^{-1} A^T P_i x \tag{5.4}$$

If we let (U, D, V) be the singular value decomposition (SVD) of $A = UDV^T$, then eq. (5.4) simplifies to:

$$A\widehat{z_i} = U\frac{D^2}{D^2 + \lambda I_d}U^T P_i x \tag{5.5}$$

where the division must be read as an element-wise division of diagonal elements. By noticing that $AA^T = UD^2U^T$, then we can rewrite eq. (5.5) as:

$$A\widehat{z_i} = \underbrace{U\frac{S}{S + \lambda I_d}U^T}_{\text{Denoising Filter}}\underbrace{P_i x}_{\text{Patch}} \tag{5.6}$$

where U and S are obtained via the eigen-decomposition of AA^T. Repeating the above steps for all $i = \{1 \dots N\}$ will provide denoised estimates for all of the patches in a given noisy image. Here, the parameter λ is chosen such that it is proportional to the noise variance in y (which is assumed to be known *a priori*.)

Proposed choice for A matrix

Note that in eq. (5.3), we have opted for an ℓ_2 constraint over an ℓ_1 constraint to facilitate a closed form solution to our optimization problem. This choice is driven to ensure computational efficiency. However, the ℓ_2-norm does not promote sparsity and hence the reconstruction accuracy depends heavily on the quality and relevance of patches in matrix A. Therefore, it is not ideal to use the entire targeted database in place of A. A valid alternate option is to set matrix $A = [p_1, \dots, p_m]$ where $p_{1:m}$ are the m closest neighbors of $P_i x$. According to Luo et al. [23], this option leads to the optimal usage of the provided external database and will yield a filter similar to the TID filter [24, 23] (the main difference being the absence of the whole image denoising part in TID). On the contrary, it adds an undesirable amount of computational complexity during runtime.

To alleviate these issues, we propose to find a set of anchor patches, $\{a_1 \dots a_k\}$, to

represent the entire targeted database. If the targeted database can be clustered in k groups, then each anchor patch can be thought of as the representative of one of these clusters. Then, a patch dictionary is created for each of the anchor patches using their respective m nearest neighbors. By taking the eigenvalue decompositions of these k patch matrices, we can calculate the corresponding U_i and S_i matrices, for all $i \in 1 \ldots k$. During runtime, a noisy query patch, q, can be denoised using U_i and S_i that correspond to the anchor patch, a_i, that is most similar to q. Since the anchor points are independent of the noisy image, finding anchor patches and calculating their corresponding denoising filters can be carried out off-line; thus avoiding delays during runtime.

In practice, the fidelity of the data matrix A can be improved by introducing weight matrix, i.e, by replacing A with $AW^{1/2}$ [24, 23]. Here the weight matrix $W = \frac{1}{\alpha}\,\mathrm{diag}\{w_1, w_2, \ldots, w_n\}$ where $w_i = \exp\left(-\frac{\|a-p_i\|^2}{h^2}\right)$ for some user-tunable bandwidth parameter h, and α is a normalization parameter so that the weights add up to 1. This modification also diminishes the adverse effect of irrelevant neighbors (that are far away from the anchor patches) on the learned denoising filters. Matrices U and S are obtained off-line from the eigen-decomposition of AWA^T.

5.3.3 Offline training and iterations

Thanks to anchor patches, the process of designing denoising filters is decoupled from the actual denoising step. This enables us to split the proposed algorithm into an off-line training stage and online denoising stage. The proposed denoising algorithm is summarized in Algorithm 7.

For best results, each query patch should be matched to an anchor patch that most closely resembles the underlying clean patch. Since at first, the matching is carried out with noisy patch itself, it is recommended that the runtime algorithm is repeated for more than one iteration. Each subsequent iteration uses patches from the previous image estimate to find anchor patches and performs the whole image denoising with progressively larger values of β in eq. (5.2).

Algorithm 7 Fast External Denoising

TRAINING PHASE (Offline)

Input:

D:	Patch database
k:	No. of anchors,
m:	Max no. of neighbors,
h:	bandwidth parameter

Output:

$a_1 \dots a_k$:	Anchor patches
$\{(U_1, S_1) \dots\}$:	Eigen-decomposition

1: Find k anchor patches from D ▷ *k-means clustering*

2: **for each** $a_i \in \{a_1 \dots a_k\}$ **do**

3: Find m nearest neighbors from $D \to (p_1 \dots p_m)$

4: Form matrix $A = [p_1 \dots p_m]$

5: Form weights: $W = \frac{1}{\alpha} \operatorname{diag}\{w_1, \dots, w_m\}$ where $w_j = \exp\left(-\frac{\|a_i - p_j\|^2}{h^2}\right)$ and $\alpha = \Sigma_j w_j$.

6: Compute eigen-decomposition of weighted matrix:

$$[U_i, S_i] = \operatorname{eig}\left(AWA^T\right)$$

7: **return** $\{a_1 \dots a_k\}$ and $\{(U_1, S_1) \dots (U_k, S_k)\}$

DENOISING PHASE (runtime)

Input:

y:	noisy image
σ^2:	noise variance
$a_1 \dots a_k$:	Anchor patches
$\{(U_1, S_1) \dots\}$:	Eigen-decomposition

Output:

$\hat{x}$:	Denoised image estimate

1: **for each** noisy patch, $q \in y$, **do**

2: Find the index i of the closest patch a_i to q

3: Compute the shrinkage matrix:

$$\Lambda = \left(\operatorname{diag}\left(S_i + \sigma^2 I_d\right)\right)^{-1} \operatorname{diag}(S_i)$$

4: Perform patch denoising $\hat{p} = U_i \Lambda U_i^T q$

5: Perform whole image denoising using eq. (5.2)

6: **return** denoised image $\hat{x}$

5.4 Experimental Results

We present denoising results on three grayscale image datasets containing text images, face images and license plate images, respectively. We compare the performance of the proposed approach against leading fast denoising algorithms in both internal and external denoising categories. To represent the internal denoising category, we choose BM3D [8] as it is the fastest and most popular internal denoising algorithm. For external denoising, we compare the performance with EPLL [15] trained on both generic and targeted priors. We also compare our results with TID to demonstrate the trade-off between speed-up vs. quality achieved by the proposed method. We tested these algorithms in low-, mid- and high-noise settings by varying the variance (σ^2) of the additive Gaussian noise from $\left(\frac{20}{255}\right)^2$ to $\left(\frac{80}{255}\right)^2$.

Algorithm settings and parameters

In all of our experiments, the size of the patches used is set to 8×8 and Euclidean metric is used to measure patch-wise (dis)similarity. For BM3D[1] and EPLL[2], we used the implementations provided by their respective authors.

The proposed algorithm is repeated for 3 iterations with a patch overlap of 4, 6 and 7 pixels in the first, second and third iterations, respectively (i.e. stride lengths $N_s = [4,2,1]$). For EPLL and BM3D, we use the default parameters and number of iterations prescribed by the original authors in their corresponding implementations. In the cases of EPLL and TID, we also show the results obtained and time taken by these algorithms when the patch overlap parameter and the number of iterations are matched to FED. For the three-iterations-EPLL, a 200 component GMM is learned on the targeted database (tar-EPLL3) and the parameters λ and β that gave the best performance on the validation sets were selected. These corresponded to $\lambda = \frac{N}{\sigma^2}$ and $\beta = \frac{1}{\sigma^2}[1,4,16]$.

[1] BM3D: http://www.cs.tut.fi/~foi/GCF-BM3D/

[2] EPLL: https://people.csail.mit.edu/danielzoran/epllcode.zip

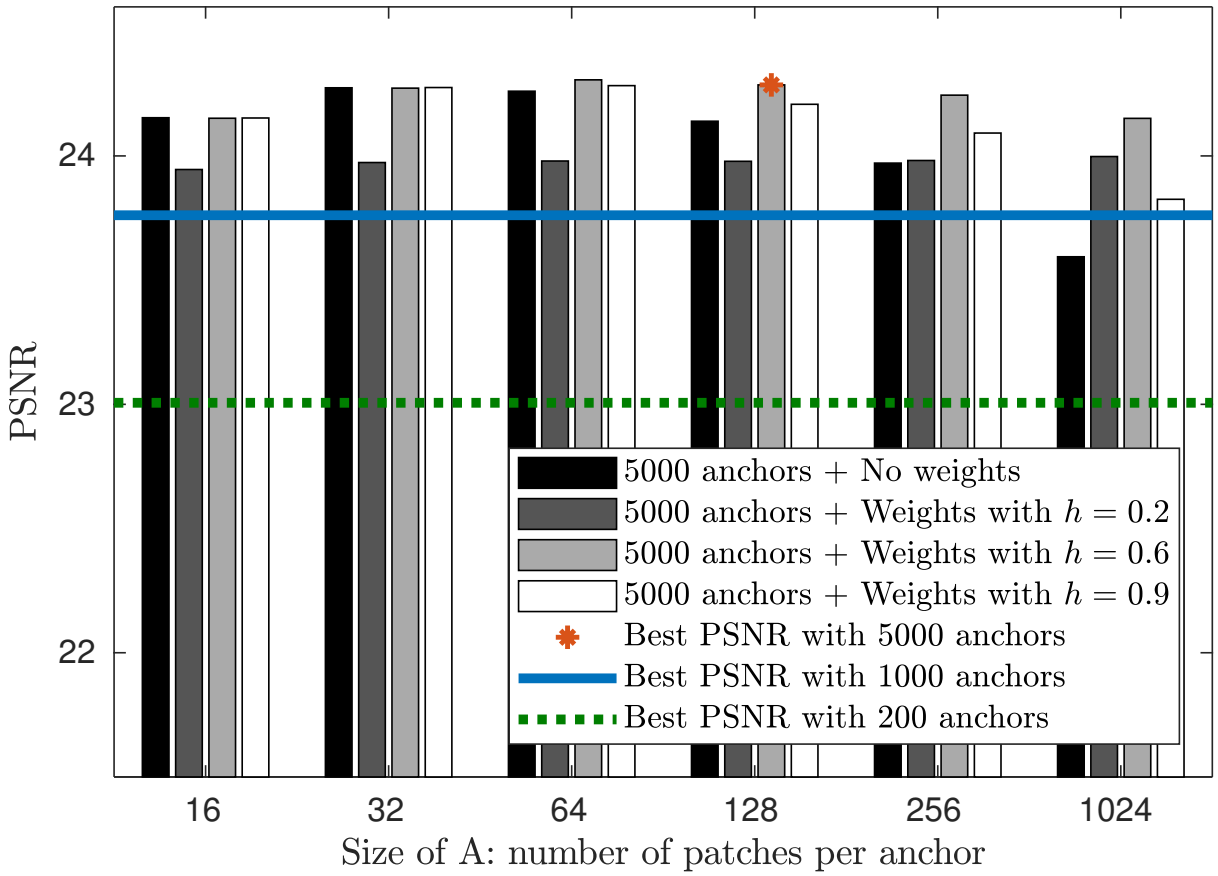
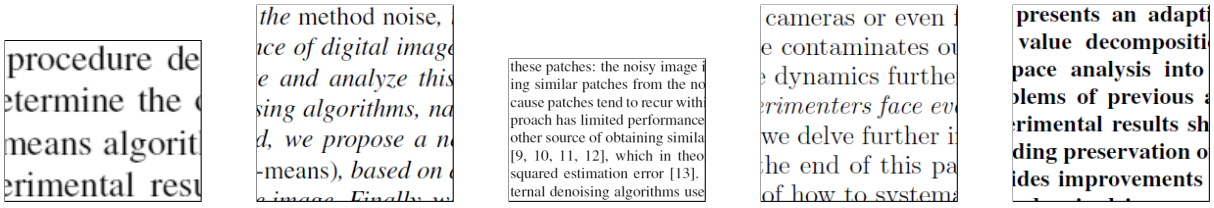


Figure 5.1: The effect of size of patch matrix, number of anchors, and weighting of the patch matrix

FED parameter selection: number of anchors, size of A matrix and weighting

We use k-means clustering algorithm to identify a pre-determined number (k) of anchor patches from the database. Other methods relying on dictionary learning [82, 83] or high dimensional regression trees [84] can also be used for finding anchor patches from the database. For simplicity, we chose to cluster the database into k clusters and use the means of each cluster as our anchor patches. The number of anchor patches k is a parameter that can be set via cross validation.

Once the anchor patches are identified, the next parameter that we have to set is the number of neighbors (m) for each anchor patch. The chosen neighbors form the respective data matrices A_i for the anchors a_i. As mentioned above, we can weigh each neighbor selected for



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

Figure 5.2: Sample images from text image dataset [23]. Experiments are conducted on noise corrupted versions of image 5.2(a). The images 5.2(b)–5.2(e) are four examples out of nine clean text images that are used as the targeted database.

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

Figure 5.3: Sample images from face image dataset [57]. Figure 5.3(a) shows one of ten test images that were used in our experiments. The images 5.3(b)–5.3(e) are four examples out of 80 face images that make up the targeted database.

forming the A matrix with a weight based on its similarity to its corresponding anchor patch.

Figure 5.1 illustrates the average denoising performance obtained on the validation datasets of text, face and license image datasets with varying number of anchor patches (k), the effect of the weighting and different number of neighbors (m) included in an anchor's A matrix. The bar graph shows the variations in performance when k is set to 5000 and m is varied from 16 to 1024 and the $h = [0.2, 0.6, 0.9]$ in the weighting matrix (used in AWA^T) or with no weighting. For brevity, we only display the best results obtained for $k = 200$ and $k = 1000$ with above mentioned values of m and h. For the three datasets with targeted databases, the best average PSNR is obtained when $k = 5000$, $m = 128$ and $h = 0.6$.

In the following, we will present the denoising results and discuss the characteristics of the datasets we used.

Table 5.1: Comparison of average PSNR, SSIM and time taken by each of the algorithms to denoise a text image of size 107×104 pixels. The speed-up is calculated with respect to the proposed FED algorithm.

	$\sigma\times$ 255	**BM3D** 2 iterations $N_s = [6,4]$	**EPLL** 5 iterations $N_s = [1]$	**tar-EPLL** 5 iterations $N_s = [1]$	**tar-EPLL3** 3 iterations $N_s = [4,2,1]$	**TID** 3 iterations $N_s = [4,2,1]$	**FED** 3 iterations $N_s = [4,2,1]$
PSNR:	20	28.64	28.04	30.38	29.92	39.37	35.65
	40	23.14	22.95	25.51	24.96	33.00	31.95
	60	19.27	20.07	22.67	22.09	29.31	28.80
	80	17.59	18.27	20.80	20.11	26.52	25.92
SSIM:	20	0.9856	0.9796	0.9910	0.9878	0.9966	0.9950
	40	0.9434	0.9303	0.9704	0.9533	0.9850	0.9843
	60	0.8392	0.8705	0.9368	0.9053	0.9645	0.9607
	80	0.7583	0.7920	0.8873	0.8282	0.9229	0.9053
Time (seconds):	-	0.12s	7.06s	6.83s	1.55s	2273.01s	1.39s
Speed up:	-	$\times$0.08	$\times$5.09	$\times$4.91	$\times$1.12	$\times$1635.26	$\times$1.00

Table 5.2: Same as Table 5.1 but for 10 face images of size 90×65 from FEI face dataset [57].

	$\sigma\times$ 255	**BM3D** 2 iterations $N_s = [6,4]$	**EPLL** 5 iterations $N_s = [1]$	**tar-EPLL** 5 iterations $N_s = [1]$	**tar-EPLL3** 3 iterations $N_s = [4,2,1]$	**TID** 3 iterations $N_s = [4,2,1]$	**FED** 3 iterations $N_s = [4,2,1]$
PSNR:	20	31.37	31.40	31.99	31.15	32.26	32.11
	40	27.63	27.86	28.32	27.09	28.51	28.20
	60	25.70	25.68	26.08	24.67	26.09	25.60
	80	24.37	24.29	24.56	23.00	24.27	23.73
SSIM:	20	0.9054	0.9048	0.9160	0.8860	0.9201	0.9164
	40	0.8176	0.8136	0.8283	0.7554	0.8273	0.8094
	60	0.7576	0.7477	0.7612	0.6381	0.7524	0.7193
	80	0.6973	0.6859	0.6964	0.5483	0.6741	0.6288
Time (seconds):	-	0.05	2.74	2.73	0.87	878.20	0.71
Speed up:	-	$\times$0.07	$\times$3.84	$\times$3.83	$\times$1.23	$\times$1236.90	$\times$1.00

5.4.1 Text denoising

Text image dataset [23] contains images cropped from documents. These images have a clean white background with black text and display simple edge structures and almost no texture. The test image to be denoised and database images vary in font styles and sizes. Some examples from the text image dataset are shown in Figure 5.2. Out of 14 images in our dataset, the targeted database is created using nine images, four are used as a validation set and one image is used for testing. Since the denoising experiments reported for this dataset are conducted on only one

(a) Original (b) Noisy ($\sigma = \frac{30}{255}$) (c) BM3D (28.96, 0.8572) (d) EPLL (28.82, 0.8518)

(e) tar-EPLL (29.18, 0.8652) (f) tar-EPLL3 (28.42, 0.8311) (g) TID (29.17, 0.8608) (h) FED (28.98, 0.8533)

Figure 5.4: Visual and objective comparison of denoising performance of a face image. The objective evaluation metrics of each case is shown in parenthesis in (PSNR, SSIM) format.

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

Figure 5.5: Sample images from the license plate dataset created from Caltech Cars (1999) [85]. Figure 5.5(a) shows one of ten test images that were used in our experiments. The images 5.5(b)–5.5(e) show four examples out of 90 images from the targeted database.

image, we average our results over 5 independent trials with different noise realizations for each noise level. This is a simple setting to show the effect of a near perfect targeted database.

Table 5.1 reports the quantitative results in terms of PSNR and SSIM [86] of the proposed FED algorithm compared to other competing fast denoising algorithms. FED algorithm trained on

Table 5.3: Same as Table 5.1 but for 10 license images of average size 44×92 cropped from the Caltech cars (1999) dataset [85].

	$\sigma\times$ 255	**BM3D** 2 iterations $N_s = [6,4]$	**EPLL** 5 iterations $N_s = [1]$	**tar-EPLL** 5 iterations $N_s = [1]$	**tar-EPLL3** 3 iterations $N_s = [4,2,1]$	**TID** 3 iterations $N_s = [4,2,1]$	**FED** 3 iterations $N_s = [4,2,1]$
PSNR:	20	26.15	25.99	26.87	26.55	25.27	25.87
	40	21.66	21.86	22.98	22.53	23.11	22.83
	60	19.16	19.52	20.70	20.13	21.02	20.89
	80	17.68	17.87	18.98	18.32	19.48	19.41
SSIM:	20	0.9330	0.9366	0.9476	0.9432	0.9223	0.9260
	40	0.8270	0.8452	0.8803	0.8706	0.8666	0.8640
	60	0.6984	0.7306	0.8009	0.7869	0.7885	0.7954
	80	0.6022	0.6322	0.7336	0.7102	0.7349	0.7480
Time (seconds):	-	0.03	1.74	1.75	0.46	311.97	0.45
Speed up:	-	$\times$0.07	$\times$3.90	$\times$3.91	$\times$1.02	$\times$693.27	$\times$1.00

(a) Original

(b) Noisy ($\sigma = \frac{20}{255}$) (c) BM3D (25.77, 0.9497) (d) EPLL (25.51, 0.9508) (e) tar-EPLL (26.84, 0.9624) (f) tar-EPLL3 (26.47, 0.9586) (g) TID (25.37, 0.9525) (h) FED (26.47, 0.9541)

(i) Noisy ($\sigma = \frac{50}{255}$) (j) BM3D (19.48, 0.7977) (k) EPLL (19.64, 0.8295) (l) tar-EPLL (21.36, 0.8801) (m) tar-EPLL3 (20.99, 0.8734) (n) TID (22.72, 0.9101) (o) FED (21.99, 0.8833)

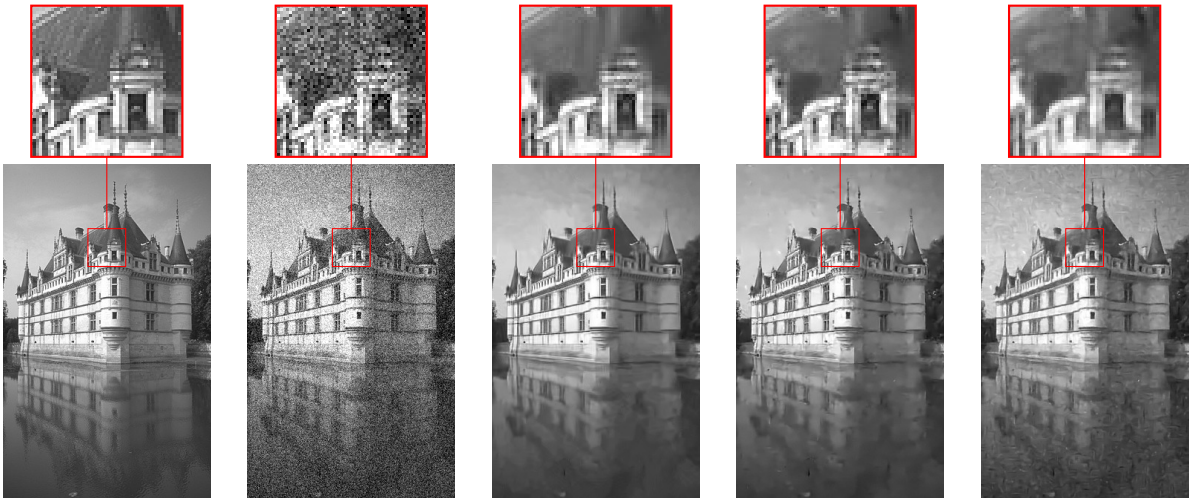
(p) Noisy ($\sigma = \frac{80}{255}$) (q) BM3D (17.07, 0.6766) (r) EPLL (16.97, 0.6862) (s) tar-EPLL (18.72, 0.7885) (t) tar-EPLL3 (18.26, 0.7826) (u) TID (19.43, 0.7840) (v) FED (19.81, 0.8183)

Figure 5.6: Visual and objective comparison of denoising performance of the same license image under different noise levels. The objective evaluation metrics of each case is shown in parenthesis in (PSNR, SSIM) format.

a targeted database provides much higher PSNR and SSIM than BM3D and EPLL in all the noise level settings tested. The EPLL algorithm trained on the targeted dataset and run with default parameters (5 iterations) performed better than generic EPLL. Still, the proposed algorithm

Table 5.4: Same as Table 5.1 but for 100 test images of size 321×481 of the BSDS dataset [79].

	σ $\times$ 255	**BM3D** 2 iters N_s=[6,4]	**EPLL** 5 iters [1]	**EPLL3** 3 iters [4,2,1]	**FED** 3 iters [4,2,1]
PSNR:	30	27.57	27.66	26.95	26.87
	40	26.30	26.43	25.56	25.30
	50	25.45	25.51	24.51	23.95
SSIM:	30	0.7607	0.7717	0.7257	0.7155
	40	0.7101	0.7175	0.6494	0.6308
	50	0.6704	0.6729	0.5843	0.5532
Time (s):	-	1.66s	83.38s	19.34s	8.99s
Speed up:	-	$\times$0.18	$\times$9.27	$\times$2.15	$\times$1.00



(a) Original (b) Noisy ($\sigma = \frac{30}{255}$) (c) BM3D (28.59, 0.8300) (d) EPLL (28.54, 0.8262) (e) Generic FED (27.25, 0.7323)

Figure 5.7: Visual and objective comparison of denoising performance of one of the BSDS test images using a ***generic*** database created from BSDS training set. The objective evaluation metrics of each case is shown in parenthesis in (PSNR, SSIM) format.

outperformed targeted EPLL consistently by a margin ranging from 5-7 dB in low-,mid- and high-noise settings. The best performing algorithm on this dataset, the TID algorithm, is three orders of magnitude slower. The PSNR gains of TID over the proposed FED drop quickly from 5dB for $\sigma = \frac{20}{255}$ to under 1dB for $\sigma = \frac{40}{255}$. We argue that compromising a small amount of quality to gain such an enormous speed-up is highly warranted. Note that such computational complexity

makes TID unsuitable for denoising large images/videos, whereas FED is a natural fit for these domains.

5.4.2 Face image denoising

Face image denoising experiments were conducted on a subset of images taken from FEI face dataset [57]. It contains one image per person and has no overlap between the set of images used as the noisy images (test set), validation set and the targeted database. Out of the 100 images of distinct individuals, we used 10 individuals for creating noisy images, 10 for validation purposes and the targeted database contained images of a different set of 80 individuals. Since there is no overlap between the individuals in these different partitions, the images to be denoised are not a perfect match to the ones used in the targeted database. This setting also displays variations in lighting, contrast, gender, etc. Some examples from this dataset are shown in Figure 5.3.

The results obtained on the face images are shown in Table 5.2. Quantitatively, FED obtains comparable results as those obtained from BM3D and different versions of EPLL. However, FED is almost 4 times faster than the better-performing EPLL variants. In addition, a visual comparison of results obtained on one of the images, shown in Figure 5.4, indicates that FED algorithm provides a more visually pleasing denoised estimate.

5.4.3 License plate denoising

License plate dataset was created by cropping license plates from the Caltech Cars (1999) dataset [85]. This dataset was originally collected for testing object category discovery algorithms and contains images of rear views of cars from Caltech parking lots. Therefore, the license plate images, which are only a small part of the car images, naturally contain small amounts of noise and display large variations in lighting and contrast. We include the results from this setting to

demonstrate the performance of the proposed FED algorithm on realistic settings with targeted datasets containing similar but not identical images.

Table 5.3 shows the quantitative results obtained on this dataset using the different denoising algorithms. The PSNR and SSIM values of BM3D and EPLL are better than FED for almost all noise settings. However, FED does better under the highest noise setting we tested. Although quantitatively slightly under par, Figure 5.6 indicates that FED results are *consistently* better visually than other estimates.

5.4.4 Generic image denoising on BSDS dataset

As a limiting case, we also include the results obtained by FED on a set of generic images with a generic database. We use the Berkeley Segmentation Dataset (BSDS) [79]. The database of patches was created by randomly sampling 2 million patches from BSDS training set images. The denoising experiments are conducted on the BSDS test set consisting of 100 images. The results reported in Table 5.4 are averaged over all 100 images for each noise setting.

Since FED is designed to work well with targeted databases, as expected, BM3D and EPLL achieve better PSNR and SSIM measures in this dataset. However, FED results are comparable and are obtained in one-ninth of the time taken by EPLL. For visual comparison, we have included the results of one of the test images in Figure 5.7.

5.5 Conclusion

In this chapter, we presented a new external denoising algorithm that is more efficient than the current state-of-the-art methods. The proposed algorithm is faster during runtime and achieves better performance when used with targeted databases than EPLL, the current state-of-the-art efficient external denoising algorithm. It is also orders of magnitude faster than the powerful state-of-the-art TID algorithm without compromising much in terms of quality. This balance

between speed and quality was achieved by transferring computationally demanding steps of designing optimal filters to an offline training step. Specifically, the information from a targeted database is extracted and stored in pre-learned transformations that are used directly during runtime. The proposed approach is extremely powerful when the transformation matrices are learned using a targeted database. However, when trained on a generic dataset the algorithm is unable to reconstruct texture details leading to over-smoothing, as can be observed in Figure 5.7. This can be avoided to a limited extent by increasing the number of anchor points so that detailed texture patches are properly represented. Another approach is using sophisticated algorithms (e.g. dictionary learning) in place of k-means to identify a more representative set of anchor patches.

Other than increasing the number and representativeness of anchor patches, a more principled extension that can counter database mismatch is to use a more powerful and expressive patch prior. To this end, in the next chapter, we seek to improve the runtime efficiency and accelerate the EPLL algorithm that uses a GMM prior.

Acknowledgments

Parts of this chapter is a reprint of materials in the conference paper "Fast External Denoising Using Pre-Learned Transformations", S. Parameswaran, E. Luo, C-A. Deledalle and T. Q. Nguyen [4]. The dissertation author is the primary investigator and author of this paper.

Chapter 6

Accelerating GMM-based patch priors for image restoration

6.1 Introduction

Patch-based methods form a very popular and successful class of image restoration techniques. These methods process an image on a patch-by-patch basis where a patch is a small sub-image (*e.g.*, of 8×8 pixels) that captures both geometric and textural information. Patch-based algorithms have been at the core of many state-of-the-art results obtained on various image restoration problems such as denoising, deblurring, super-resolution, defogging, or compression artifact removal to name a few. In image denoising, patch-based processing became popular after the success of the Non-Local Means algorithm [7]. Subsequently, continued research efforts have led to significant algorithmic advancements in this area [82, 8, 15, 37, 25, 17, 11]. Other inverse problems such as image super-resolution and image deblurring have also benefited from patch-based models [87, 88, 89, 90, 91, 92].

Among these various patch-based methods, the Expected Patch Log-Likelihood algorithm (EPLL) [15] deserves a special mention due to its restoration performance and versatility. The

EPLL introduced an innovative application of Gaussian Mixture Models (GMMs) to capture the prior distribution of patches in natural images. Note that a similar idea was introduced concurrently in [17]. The success of this method is evident from the large number of recent works that extend the original EPLL formulation [93, 46, 94, 95, 96, 97, 33]. However, a persistent problem of EPLL-based algorithms is their high runtime complexity. For instance, it is orders of magnitude slower than the well-engineered BM3D image denoising algorithm [8]. However, extensions of BM3D that perform super-resolution [98] and other inverse problems [99] require fundamental algorithmic changes, making BM3D far less adaptable than EPLL. Other approaches that are as versatile as EPLL [100, 20, 101] either lack the algorithmic efficiency of BM3D or the restoration efficacy of EPLL.

Another class of techniques that arguably offers better runtime performance than EPLL-based methods (but not BM3D) are those based on deep learning. With the advancements in computational resources, researchers have attempted to solve some classical inverse problems using multi-layer perceptrons [16] and deep networks [102, 103, 104]. These methods achieve very good restoration performance, but are heavily dependent on the amount of training data available for each degradation scenario. Most of these methods learn filters that are suited to restore a specific noise level (denoising), blur (deblurring) or upsampling factor (super-resolution), which makes them less attractive to serve as generic image restoration solutions. More recently, Zhang *et al.* [105] demonstrated the use of deep residual networks for general denoising problems,002 single-image super-resolution and compression artifact removal. Unlike earlier deep learning efforts, their approach can restore images with different noise levels using a single model which is learned by training on image patches containing a range of degradations. Even in this case, the underlying deep learning model requires retraining whenever a new degradation scenario different from those considered during the learning stage is encountered.

More recently, [106] proposed training a single deep network to solve many inverse problems. This work uses an iterative scheme that alternatively enforces a good fit to the learned

prior model of natural images (via a projection operator performed by a deep neural network) and a satisfying fidelity to the data (via the direct model). The projection operator takes the form of an auto-encoder, trained by an adversarial strategy, that processes 64×64 patches with 1024-dimensional latent space (*i.e.*, 4 times smaller). This framework shows promise but, in its current form, requires a very large image database for training. In addition, almost all of the training and testing in the original publication are conducted on small 64×64 images. Another limitation, as noted by the authors, is that the regularization parameter that controls the contribution of prior is fixed during training stage. In other words, regularization parameter cannot be changed during test time which can be an issue in certain situations. In contrast, our approach needs much less data during training stage and, during test time, the regularization parameter can be tuned according to the signal-to-noise ratio of the image being addressed.

Moreover, it is much harder to gain insight into the actual model learned by a deep architecture compared to a GMM. For this reason, even with the advent of deep learning methods, flexible algorithms like EPLL that have a transparent formulation remain relevant for image restoration.

Recently, researchers have tried to improve the speed of EPLL by replacing the most time-consuming operation in the EPLL algorithm with a machine learning-based technique of their choice [107, 108]. These methods were successful in accelerating EPLL to an extent but did not consider tackling all of its bottlenecks. In contrast, our focus is on accelerating EPLL by proposing algorithmic approximations to all the prospective bottlenecks present in the original algorithm proposed by Zoran *et al.* [15]. To this end, we first provide a complete computational and runtime analysis of EPLL, present a new and efficient implementation of original EPLL algorithm and then finally propose innovative approximations that lead to a novel algorithm that is more than $100\times$ faster compared to the efficiently implemented EPLL (and $350\times$ faster than the runtime obtained by using the original implementation [15]).

Contributions The main contributions of this work are the following. We introduce three

strategies to accelerate patch-based image restoration algorithms that use a GMM prior. We show that, when used jointly, they lead to a speed-up of the EPLL algorithm by two orders of magnitude. Compared to the popular BM3D algorithm, which represents the current state-of-the-art in terms of speed among CPU-based implementations, the proposed algorithm is almost an order of magnitude faster. The three strategies introduced in this work are general enough to be applied individually or in any combination to accelerate other related algorithms. For example, the random subsampling strategy is a general technique that could be reused in any algorithm that considers overlapping patches to process images; the flat tail spectrum approximation can accelerate any method that needs Gaussian log-likelihood or multiple Mahalanobis metric calculations; finally, the binary search tree for Gaussian matching can be included in any algorithm based on a GMM prior model and can be easily adapted for vector quantization techniques that use a dictionary.

For reproducibility purposes, we release our software on GitHub along with a few usage demonstrations (available at https://goo.gl/xjqKUA).

6.2 Expected Patch Log-Likelihood (EPLL)

We consider the problem of estimating an image $x \in \mathbb{R}^N$ (N is the number of pixels) from noisy linear observations $y = \mathcal{A}x + w$, where $\mathcal{A} : \mathbb{R}^N \to \mathbb{R}^M$ is a linear operator and $w \in \mathbb{R}^M$ is a noise component assumed to be white and Gaussian with variance σ^2. In a standard denoising problem $\mathcal{A}$ is the identity matrix, but in more general settings, it can account for loss of information or blurring. Typical examples for operator $\mathcal{A}$ are: a low pass filter (for *deconvolution*), a masking operator (for *inpainting*), or a projection on a random subspace (for *compressive sensing*). To reduce noise and stabilize the inversion of $\mathcal{A}$, some *prior* information is used for the estimation of x. The EPLL introduced by Zoran and Weiss [15] includes this *prior* information as a model for the distribution of patches found in natural images. Specifically, the EPLL defines the restored image as the maximum *a posteriori* estimate, corresponding to the following minimization

problem:

$$\underset{x}{\operatorname{argmin}} \frac{P}{2\sigma^2}\|\mathcal{A}x - y\|^2 - \sum_{i \in I} \log p\left(\mathcal{P}_i x\right) \tag{6.1}$$

where $I = \{1, \ldots, N\}$ is the set of pixel indices, $\mathcal{P}_i : \mathbb{R}^N \to \mathbb{R}^P$ is the linear operator extracting a patch with P pixels centered at the pixel with location i (typically, $P = 8 \times 8$), and $p(.)$ is the *a priori* probability density function (*i.e.*, the statistical model of noiseless patches in natural images). While the first term in eq. (6.1) ensures that $\mathcal{A}x$ is close to the observations y (this term is the negative log-likelihood under the white Gaussian noise assumption), the second term regularizes the solution x by favoring an image such that all its patches fit the *prior* model of patches in natural images. The authors of [15] showed that this *prior* can be well approximated (upon removal of the DC component of each patch) using a zero-mean Gaussian Mixture Model (GMM) with $K = 200$ components, that reads for any patch $z \in \mathbb{R}^P$, as

$$p\left(z\right) = \sum_{k=1}^{K} w_k \frac{1}{(2\pi)^{P/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2} z^t \Sigma_k^{-1} z\right), \tag{6.2}$$

where the weights w_k (such that $w_k > 0$ and $\sum_k w_k = 1$) and the covariance matrices $\Sigma_k \in \mathbb{R}^{P \times P}$ are estimated using the Expectation-Maximization algorithm [48] on a dataset consisting of 2 million "clean" patches extracted from the training set of the Berkeley Segmentation (BSDS) dataset [109].

Half-quadratic splitting

Problem (6.1) is a large non-convex problem where $\mathcal{A}$ couples all unknown pixel values x and the patch prior is highly non-convex. A classical workaround, known as half-quadratic splitting [110, 111], is to introduce N auxiliary unknown vectors $z_i \in \mathbb{R}^P$, and consider instead

Table 6.1: Comparison of the execution time of our implementation of EPLL with and without proposed accelerations. Experiment conducted on 40 images each of size 481×321 in denoising setting. Profiling was carried out using `MATLAB` (R2014b) on a PC with Intel(R) Core(TM) i7-4790K CPU @4.00GHz and 16 GB RAM. Execution times are reported as average number of seconds per image (s) and percentage of the total time (%).

Step	Without accelerations		With the proposed accelerations	
(Gaussian selection)	38.27s	95 %	0.21s	67 %
(Patch estimation)	0.95s	2 %	0.04s	13 %
(Patch extraction)	0.43s	1 %	0.02s	6 %
(Patch reprojection)	0.15s	0 %	0.01s	4 %
Others	0.52s	1 %	0.03s	10 %
Total	40.32s		0.31s	

Algorithm 8 The five steps of an EPLL iteration [15]

for all $i \in I$

$\tilde{z}_i \leftarrow \mathcal{P}_i \hat{x}$ (Patch extraction)

$k_i^\star \leftarrow \underset{1 \leqslant k_i \leqslant K}{\operatorname{argmin}} \log w_{k_i}^{-2} + \log \left| \Sigma_{k_i} + \frac{1}{\beta} \mathrm{Id}_P \right| + \tilde{z}_i^t \left(\Sigma_{k_i} + \frac{1}{\beta} \mathrm{Id}_P \right)^{-1} \tilde{z}_i$ (Gaussian selection)

$\hat{z}_i \leftarrow \left(\Sigma_{k_i^\star} + \frac{1}{\beta} \mathrm{Id}_P \right)^{-1} \Sigma_{k_i^\star} \tilde{z}_i$ (Patch estimation)

$\tilde{x} \leftarrow \left(\sum_{i \in I} \mathcal{P}_i^t \mathcal{P}_i \right)^{-1} \sum_{i \in I} \mathcal{P}_i^t \hat{z}_i$ (Patch reprojection)

$\hat{x} \leftarrow \left(\mathcal{A}^t \mathcal{A} + \beta \sigma^2 \mathrm{Id}_N \right)^{-1} \left(\mathcal{A}^t y + \beta \sigma^2 \tilde{x} \right)$ (Image estimation)

the penalized optimization problem that reads, for $\beta > 0$, as

$$\underset{x, z_1, \ldots, z_N}{\operatorname{argmin}} \frac{P}{2\sigma^2} \|\mathcal{A}x - y\|^2 + \frac{\beta}{2} \sum_{i \in I} \|\mathcal{P}_i x - z_i\|^2 - \sum_{i \in I} \log p(z_i). \tag{6.3}$$

When $\beta \to \infty$, the problem (6.3) becomes equivalent to the original problem (6.1). In practice, an increasing sequence of β is considered, and an alternating optimization scheme is used:

$$\left\{ \hat{z}_i \leftarrow \underset{z_i}{\operatorname{argmin}} \frac{\beta}{2} \|\mathcal{P}_i \hat{x} - z_i\|^2 - \log p(z_i) \right\}_{i=1..N} \tag{6.4}$$

$$\hat{x} \leftarrow \underset{x}{\operatorname{argmin}} \frac{P}{2\sigma^2} \|\mathcal{A}x - y\|^2 + \frac{\beta}{2} \sum_{i \in I} \|\mathcal{P}_i x - \hat{z}_i\|^2. \tag{6.5}$$

Algorithm

Subproblem (6.5) corresponds to solving a linear inverse problem with a Tikhonov regularization, and has an explicit solution often referred to as Wiener filtering:

$$\hat{x} = \left(\mathcal{A}^t \mathcal{A} + \frac{\beta\sigma^2}{P} \sum_{i \in I} \mathcal{P}_i^t \mathcal{P}_i \right)^{-1} \left(\mathcal{A}^t y + \frac{\beta\sigma^2}{P} \sum_{i \in I} \mathcal{P}_i^t \hat{z}_i \right), \tag{6.6}$$

where $\mathcal{P}_i^t \mathcal{P}_i$ is a diagonal matrix whose i-th diagonal element corresponds to the number of patches overlapping the pixel of index i. This number is a constant equal to P (assuming proper boundary conditions are used), which allows to split the computation into two steps *Patch reprojection* and *Image estimation* as shown in Alg. 8. Note that the step *Patch reprojection* is simply the average of all overlapping patches. In contrast, subproblem (6.4) cannot be obtained in closed form as it involves a term with the logarithm of a sum of exponentials. A practical solution proposed in [15] is to keep only the component $k_i^\star$ maximizing the likelihood for the given patch assuming it is a zero-mean Gaussian random vector with covariance matrix $\Sigma_{k_i} + \frac{1}{\beta}\mathrm{Id}_P$. With this approximation, the solution of (6.4) is also given by Wiener filtering, and the resulting algorithm iterates the steps described in Alg. 8. The authors of [15] found that using $T = 5$ iterations, with the sequence $\beta = \frac{1}{\sigma^2}\{1, 4, 8, 16, 32\}$, for the initialization $\hat{x} = y$, provides relevant solutions in denoising contexts for a wide range of noise level values σ^2.

6.2.1 Complexity via eigenspace implementation

The algorithm summarized in the previous section may reveal cumbersome computations as it requires performing numerous matrix multiplications and inversions. Nevertheless, as the matrices Σ_k are known *prior* to any calculation, their eigendecomposition can be computed offline to improve the runtime. If we denote the eigendecomposition (obtained offline) of $\Sigma_k = U_k S_k U_k^t$, such that $U_k \in \mathbb{R}^{P \times P}$ is unitary and S_k is diagonal with positive diagonal elements ordered in decreasing order, steps *Gaussian selection* and *Patch estimation* can be expressed in the space of

coefficients c as

$$\left\{\tilde{c}_i^k \leftarrow U_k^t \tilde{z}_i\right\}_{\substack{k=1..K \\ i=1..N}} \qquad O(NKP^2) \tag{6.7}$$

$$\left\{k_i^\star \leftarrow \underset{1\leqslant k\leqslant K}{\operatorname{argmin}}\, \iota^k + \sum_{j=1}^{P}\left(\log \nu_j^k + \frac{[\tilde{c}_i^k]_j^2}{\nu_j^k}\right)\right\}_{i=1..N} \qquad O(NKP) \tag{6.8}$$

$$\left\{[\hat{c}_i]_j \leftarrow \gamma_j^{k_i^\star}[\tilde{c}_i^{k_i^\star}]_j\right\}_{\substack{j=1..P \\ i=1..N}} \qquad O(NP) \tag{6.9}$$

$$\left\{\hat{z}_i \leftarrow U_{k_i^\star}\hat{c}_i\right\}_{i=1..N} \qquad O(NP^2) \tag{6.10}$$

where $[\tilde{c}]_j$ denotes the j-th entry of vector $\tilde{c}$, $\iota^k = -2\log w_k$, $\nu_j^k = [S_k]_{jj} + \frac{1}{\beta}$, and $\gamma_j^k = [S_k]_{jj}/\nu_j^k$ with $[S_k]_{jj}$ the j-th entry on the diagonal of matrix S_k. The complexity of each operation is indicated on its right and corresponds to the number of operations per iteration of the alternate optimization scheme. The steps *Patch extraction* and *Patch reprojection* share a complexity of $O(NP)$. Finally, the complexity of step *Image estimation* depends on the transform $\mathcal{A}$. In many scenarios of interest, $\mathcal{A}^t\mathcal{A}$ can be diagonalized using a fast transform and the inversion of $\mathcal{A}^t\mathcal{A} + \beta\sigma^2\mathrm{Id}_N$ can be performed efficiently in the transformed domain (since Id_N is diagonal in any orthonormal basis). For instance, it leads to $O(N)$ operations for denoising or inpainting, and $O(N\log N)$ for periodical deconvolutions or super-resolution problems, thanks to the fast Fourier transform (these are the settings we have adopted in this chapter). If $\mathcal{A}$ cannot be easily diagonalized, this step can be performed using conjugate gradient (CG) method, as done in [15], at a computational cost that depends on the number of CG iterations (*i.e.*, on the conditioning of $\mathcal{A}^t\mathcal{A} + \beta\sigma^2\mathrm{Id}_N$). In any case, as shown in the next section, this step has a complexity independent of P and K and is one of the faster operations in the image restoration problems considered in this chapter.

6.2.2 Computation time analysis

In order to uncover the practical computational bottlenecks of EPLL, we have performed the following computational analysis. To identify clearly which part is time consuming, it is important to make the algorithm implementation as optimal as possible. Therefore, we refrain from using the `MATLAB` code provided by the original authors [15] for speed comparisons. Instead, we use a `MATLAB/C` version of EPLL based on the eigenspace implementation described above, where some steps are written in `C` language and interfaced using `mex` functions. This version, which we refer to as EPLLc, provides results identical to the original implementation while being 2-3 times faster. The execution time of each step for a single run of EPLLc is reported in the second column of Table 6.1. Reported times fit our complexity analysis and clearly indicate that the Step *Gaussian selection* causes significant bottleneck due to $O(NP^2K)$ complexity.

In the next section, we propose three independent modifications leading to an algorithm with a complexity of $O(NP\bar{r}\log K/s^2)$ with two constants $1 \leqslant s^2 \leqslant P$ and $1 \leqslant \bar{r} \leqslant P$ that control the accuracy of the approximations introduced. The algorithm, in practice, is more than 100 times faster as shown by its runtimes reported in the third column.

6.3 Fast EPLL: the three key ingredients

We propose three accelerations based on (i) scanning only a (random) subset of the N patches, (ii) reducing the number of mixture components matched, and (iii) projecting on a smaller subspace of the covariance eigenspace. We begin by describing this latter acceleration strategy in the following paragraph.

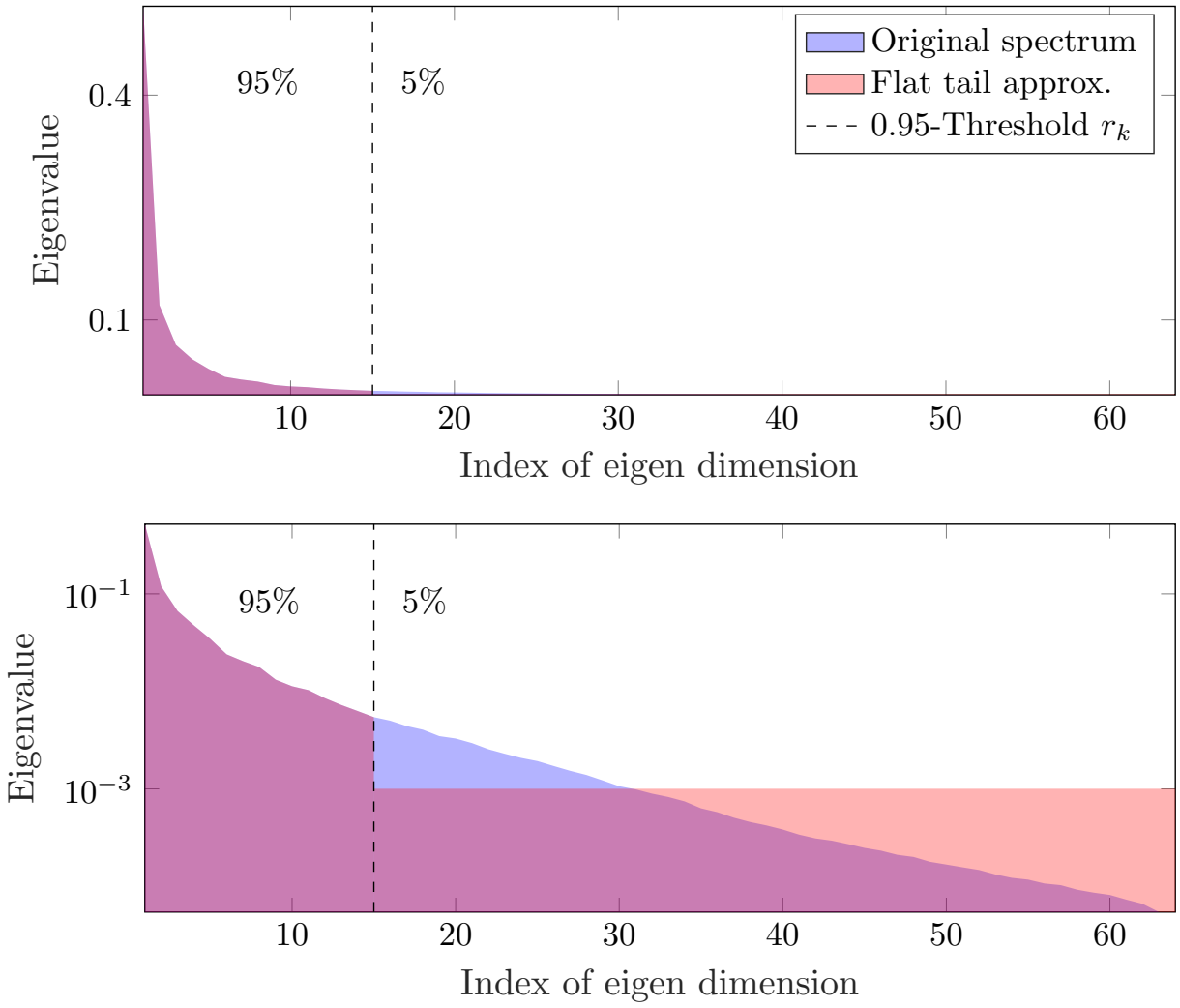


Figure 6.1: Flat tail approximation: (a) with eigenvalues display on linear and (b) logarithmic scale. The $r_k = 15$ first eigen components explains $\rho = 95\%$ of the variability.

6.3.1 Speed-up via flat tail spectrum approximation

To avoid computing the P coefficients of the vector $\tilde{c}_i^k$ in eq. (6.7), we rely on a flat-tail approximation. The k-th Gaussian model is said to have a flat tail if there exists a rank r_k such that for any $j > r_k$, the eigenvalues are constant: $[S_k]_{j,j} = \lambda_k$. Denoting by $\bar{U}_k \in \mathbb{R}^{P \times r_k}$ (resp. $\bar{U}_k^c \in \mathbb{R}^{P \times r_k^c}$) the matrix formed by the r_k first (resp. $r_k^c = P - r_k$ last) columns of U_k, we have

$\bar{U}_k^c(\bar{U}_k^c)^t = \mathrm{Id}_P - \bar{U}_k\bar{U}_k^t$. It follows

$$(\Sigma_k + \tfrac{1}{\beta}\mathrm{Id}_P)^{-1} = \bar{U}_k(\bar{S}_k + \tfrac{1}{\beta}\mathrm{Id}_{r_k})^{-1}\bar{U}_k^t + (\lambda_k + \tfrac{1}{\beta})^{-1}(\mathrm{Id}_P - \bar{U}_k\bar{U}_k^t), \tag{6.11}$$

$$(\Sigma_k + \tfrac{1}{\beta}\mathrm{Id}_P)^{-1}\Sigma_k = \bar{U}_k(\bar{S}_k + \tfrac{1}{\beta}\mathrm{Id}_{r_k})^{-1}\bar{S}_k\bar{U}_k^t + \lambda_k(\lambda_k + \tfrac{1}{\beta})^{-1}(\mathrm{Id}_P - \bar{U}_k\bar{U}_k^t), \tag{6.12}$$

where $\bar{S}_k \in \mathbb{R}^{r_k \times r_k}$ is the diagonal matrix formed by the r_k first rows and columns of S_k. Steps *Gaussian selection* and *Patch estimation* can thus be rewritten as

$$\left\{ \tilde{c}_i^k \leftarrow \bar{U}_k^t \tilde{z}_i \right\}_{\substack{k=1..K \\ i=1..N}} \qquad O(NKP\bar{r}) \tag{6.13}$$

$$\left\{ k_i^\star \leftarrow \operatorname*{argmin}_{1 \leqslant k \leqslant K} \iota_k + r_k^c \log \nu_P^k + \frac{\|\tilde{z}_i\|^2}{\nu_P^k} + \sum_{j=1}^{r_k} \left(\log \nu_j^k + \frac{[\tilde{c}_i^k]_j^2}{\nu_j^k} - \frac{[\tilde{c}_i^k]_j^2}{\nu_P^k} \right) \right\}_{i=1..N} \qquad O(NK\bar{r}) \tag{6.14}$$

$$\left\{ [\hat{c}_i]_j \leftarrow (\gamma_j^{k_i^\star} - \gamma_P^{k_i^\star}) \left[\tilde{c}_i^{k_i^\star}\right]_j \right\}_{\substack{j=1..r_{k_i^\star} \\ i=1..N}} \qquad O(NP\bar{r}) \tag{6.15}$$

$$\left\{ \hat{z}_i \leftarrow \bar{U}_{k_i^\star}\hat{c}_i + \gamma_P^{k_i^\star}\tilde{z}_i \right\}_{i=1..N} \qquad O(NP\bar{r}) \tag{6.16}$$

where $\nu_P^k = \lambda_k + \frac{1}{\beta}$, $\gamma_P^k = \lambda_k/\nu_P^k$. As $\|\tilde{z}_i\|^2$ can be computed once for all k, the complexity of each step is divided by $P/\bar{r}$, where $\bar{r} = \frac{1}{K}\sum_{k=1}^K r_k$ is the average rank after which eigenvalues are considered constant.

In practice, covariance matrices Σ_k are not flat-tail but can be approximated by a flat-tail matrix by replacing the lowest eigenvalues by a constant λ_k. To obtain a small value of $\bar{r}$ (hence a large speed-up), we preserve a fixed proportion $\rho \in (0,1]$ of the total variability and replace the smallest eigenvalues accounting for the remaining $1-\rho$ fraction of the variability by their average (see Fig. 6.1): r_k is the smallest integer such that $\mathrm{Tr}(\bar{S}_k) \geqslant \rho\,\mathrm{Tr}(S_k)$. Choosing $\rho = 0.95$ means that 5% of the variability, in the eigendirections associated to the smallest eigenvalues, is assumed to be evenly spread in these directions. In practice, the choice of $\rho = 0.95$ leads to an average rank of $\bar{r} = 19.6$ (for $P = 8 \times 8$) for a small drop of PSNR as shown in Fig. 6.4. Among several other covariance approximations that we tested, for instance, the one consisting in keeping only the

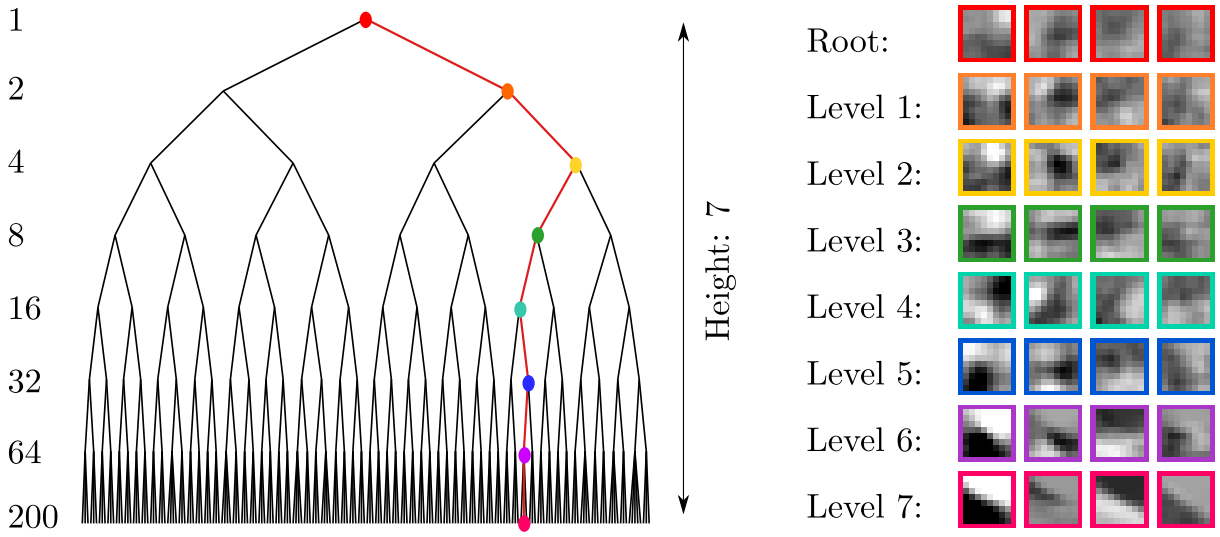


Figure 6.2: Balanced search tree: (left) Our obtained search tree (with numbers of nodes for each level). (right) Four patches well represented by each of the eight nodes along the branch highlighted in red in the tree. These patches are randomly sampled from the generative model encoded at each node of the branch (patches on the same column are generated from the same random seed).

r_k first directions, the flat tail approximation provided the best trade-off in terms of acceleration and restoration quality. The analyses showing the superiority of the proposed approximation over the more common approach of keeping only the first r_k directions, and the effect of ρ on image quality are included in Appendix A.

6.3.2 Speed-up via a balanced search tree

As shown in Table 6.1, the step *Gaussian selection* has a complexity of $O(NP^2K)$, reduced to $O(NPK\bar{r})$ using the flat tail spectrum approximation. This step remains the biggest bottleneck since each query patch has to be compared to all the K components of the GMM. To make this step even more efficient, we reduce its complexity using a balanced search tree. As described below, such a tree can be built offline by adopting a bottom-up strategy that repeatedly collapses the original GMM to models with fewer components, until the entire model is reduced to a single Gaussian model.

We progressively combine the GMM components from one level to form the level above, by clustering the K components into $L < K$ clusters of similar ones, until the entire model is reduced to a single component. The similarity between two zero-mean Gaussian models with covariance Σ_1 and Σ_2 is measured by the symmetric Kullback-Leibler (KL) divergence

$$\mathrm{KL}(\Sigma_1, \Sigma_2) = \tfrac{1}{2}\,\mathrm{Tr}(\Sigma_2^{-1}\Sigma_1 + \Sigma_1^{-1}\Sigma_2 - 2\mathrm{Id}_P). \tag{6.17}$$

Based on this divergence, at each level n, we look for a partition Ω^n of the K Gaussian components into L clusters (with about equal sizes) minimizing the following optimization problem

$$\underset{\Omega^n}{\operatorname{argmin}} \sum_{l=1}^{L} \sum_{k_1,k_2 \in \Omega_l^n} \mathrm{KL}(\Sigma_{k_1}, \Sigma_{k_1}), \tag{6.18}$$

such that $\bigcup_{l=1}^{L} \Omega_l^n = [K]$ and $\Omega_{l_1}^n \bigcap_{l_1 \neq l_2} \Omega_{l_2}^n = \emptyset$, where Ω_l^n is the l-th set of Gaussian components for the GMM at level n. This clustering problem can be approximately solved using the genetic algorithm of [112] for the Multiple Traveling Salesmen Problem (MTSP). MTSP is a variation of the classical Traveling Salesman Problem where several salesmen visit a unique set of cities and return to their origins, and each city is visited by exactly one salesman. This attempts to minimize the total distance traveled by all salesmen. Hence, it is similar to our original problem given in eq. (6.18) where the Gaussian components and the clusters correspond to K cities and L salesmen, respectively. Given the clustering at level n, the new GMM at level $n-1$ is obtained by combining the zero-mean Gaussian components such that, for all $1 \leqslant l \leqslant L$:

$$w_l^{n-1} = \sum_{k \in \Omega_l^n} w_k^n \quad \text{and} \quad \Sigma_l^{n-1} = \frac{1}{w_l^{n-1}} \sum_{k \in \Omega_l^n} w_k^n \Sigma_k^n, \tag{6.19}$$

where Σ_k^n and w_k^n are the corresponding covariance matrix and weight of the k-th Gaussian component at level n. Following this scheme, the original GMM of $K = 200$ components is collapsed into increasingly more compact GMMs with $K = 64$, 32, 16, 8, 4, 2 and 1 components.

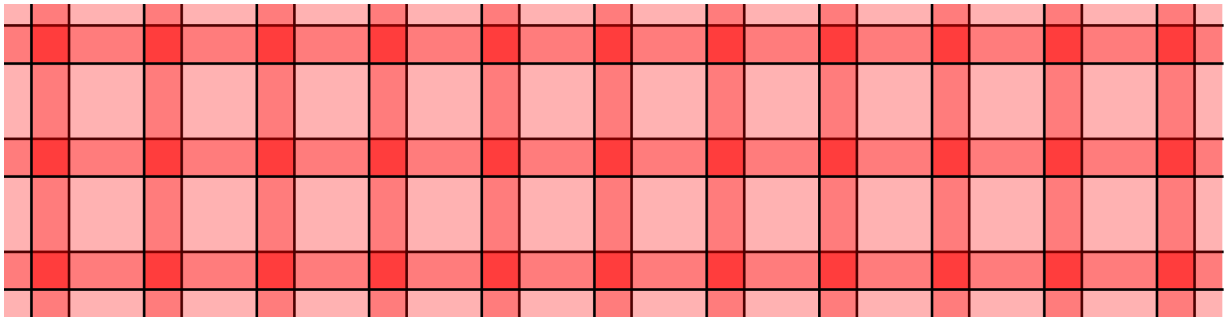
The main advantage of using MTSP compared to classical clustering approaches, is that this procedure can be adapted easily to enforce approximately equal sized clusters, simply by enforcing that each salesman visits at least 3 cities for the last level and 2 for the other ones.

We also experimented with other clustering strategies such as the hierarchical kmeans-like clustering in [113] and hierarchical agglomerative clustering. With no principled way to enforce even-sized clusters, these approaches, in general, lead to unbalanced trees (with comb structured branches) which result in large variations in computation times from one image to another. Although they all lead to similar denoising performances, we opted for MTSP based clustering to build our Gaussian tree in favor of obtaining a stable speed-up profile for our resulting algorithm. Please refer to Appendix A for timing comparisons of MTSP vs. other tree building strategies.

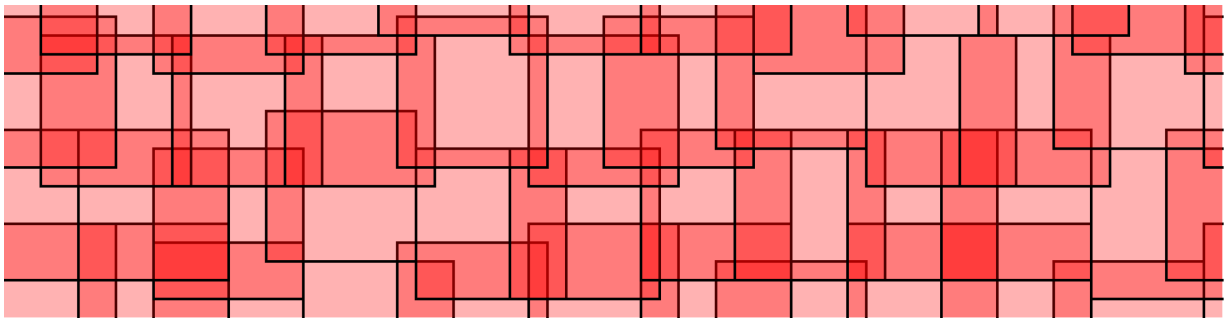
In Fig. 6.2 we show that the tree obtained using MTSP-based clustering is almost a binary tree (left) and also display the types of patches it encodes along a given path (right). Such a balanced tree structure lets one avoid testing each patch against all K components. Instead, a patch is first compared to the two first nodes in level 1 of the tree, then the branch providing the smallest cost is followed and the operation is repeated at higher levels until a leaf has been reached. Using this balanced search tree reduces the complexity of step *Gaussian selection* to $O(NP\bar{r}\log K)$.

6.3.3 Speed-up via the restriction to a random subset of patches

The simplest and most effective proposed acceleration consists of subsampling the set I of N patches to improve the complexity of the four most time-consuming steps, see Table 6.1. One approach, followed by BM3D [8], consists of restricting the set I to locations on a regular grid with spacing $s \in [1, \sqrt{P}]$ pixels in both directions, leading to a reduction of complexity by a factor s^2. We refer to this approach as the regular patch subsampling. A direct consequence is that $|I| = N/s^2$ and the complexity is divided by s^2. However, we observed that this strategy consistently creates blocky artifacts revealing the regularity of the extraction pattern. A random sampling approach,



(a) Regular patch subsampling at (i_0, j_0)



(b) Stochastic patch subsampling at (i, j)

Figure 6.3: Illustration of patch subsampling. Instead of extracting all patches, only a subset of patches is extracted either (a) regularly or (b) with some randomizations. Patches are represented by 8×8 squares and the red intensity represents the number of patches overlapping the corresponding pixel.

called "jittering", used in the computer graphics community [114] is preferable to limit this effect. This procedure ensures that each pixel is covered by at least one patch. The location (i_0, j_0) of a point of the grid undergoes a random perturbation, giving a new location (i, j) such that

$$\begin{aligned} & i_0 - \left\lfloor \tfrac{\sqrt{P}-s}{2} \right\rfloor \leqslant i \leqslant i_0 + \left\lfloor \tfrac{\sqrt{P}-s}{2} \right\rfloor \\ \text{and} \quad & j_0 - \left\lfloor \tfrac{\sqrt{P}-s}{2} \right\rfloor \leqslant j \leqslant j_0 + \left\lfloor \tfrac{\sqrt{P}-s}{2} \right\rfloor, \end{aligned} \tag{6.20}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation. We found experimentally that independent and uniform perturbations offered the best performance in terms of PSNR and visual quality against all other tested strategies. In addition, we also resample these positions at each of the T iterations and add a (random) global shift to ensure that all pixels have the same expected number of patches

covering them.

Figure 6.3 illustrates the difference between a regular grid and a jittered grid of period $s=6$ for patches of size $P=8\times 8$. In both cases, all pixels are covered by at least one patch, but the stochastic version reveals an irregular pattern.

Nevertheless, when using random subsampling, a major bottleneck occurs when $\mathcal{A}^t\mathcal{A}$ is not diagonal because the inversion involved in eq. (6.6) cannot be simplified as in Alg. 8. Using a conjugate gradient is a practical solution but will negate the reduction of complexity gained by using subsampling. To the best of our knowledge, this is the main reason why patch subsampling has not been utilized to speed up EPLL. Here, we follow a different path. We opt for approximating the solution of the original problem (involving all patches) rather than evaluating the solution of an approximate problem (involving random subsample of patches). More precisely, we speed up Alg. 8 by replacing the complete set of indices by the random subset of patches. In this case, step *Patch reprojection* consists of averaging only this subset of overlapping restored patches. This novel and nuanced idea avoids additional overhead and attains dramatic complexity improvements compared to the standard approach. Note that even in the case of some inverse problems, such as deblurring, super-resolution, inpainting and devignetting, this strategy can still be used in order to avoid conjugate gradient and maintain a large speed-up.

Experiments conducted on our validation dataset show that this strategy used with $s=6$ leads to an acceleration of about $36\times$ with less than a 0.2dB drop in PSNR. In comparison, for a similar drop of PSNR, the regular patch subsampling can only achieve a $9\times$ acceleration with $s=3$ (plots included in Appendix A).

6.3.4 Performance analysis

Figure 6.4 shows the image restoration performance and speed-up obtained when the three ingredients are applied separately or in combination. The results are averaged over 40 images from the test set of BSDS dataset [109] that is set aside for validation purposes. The

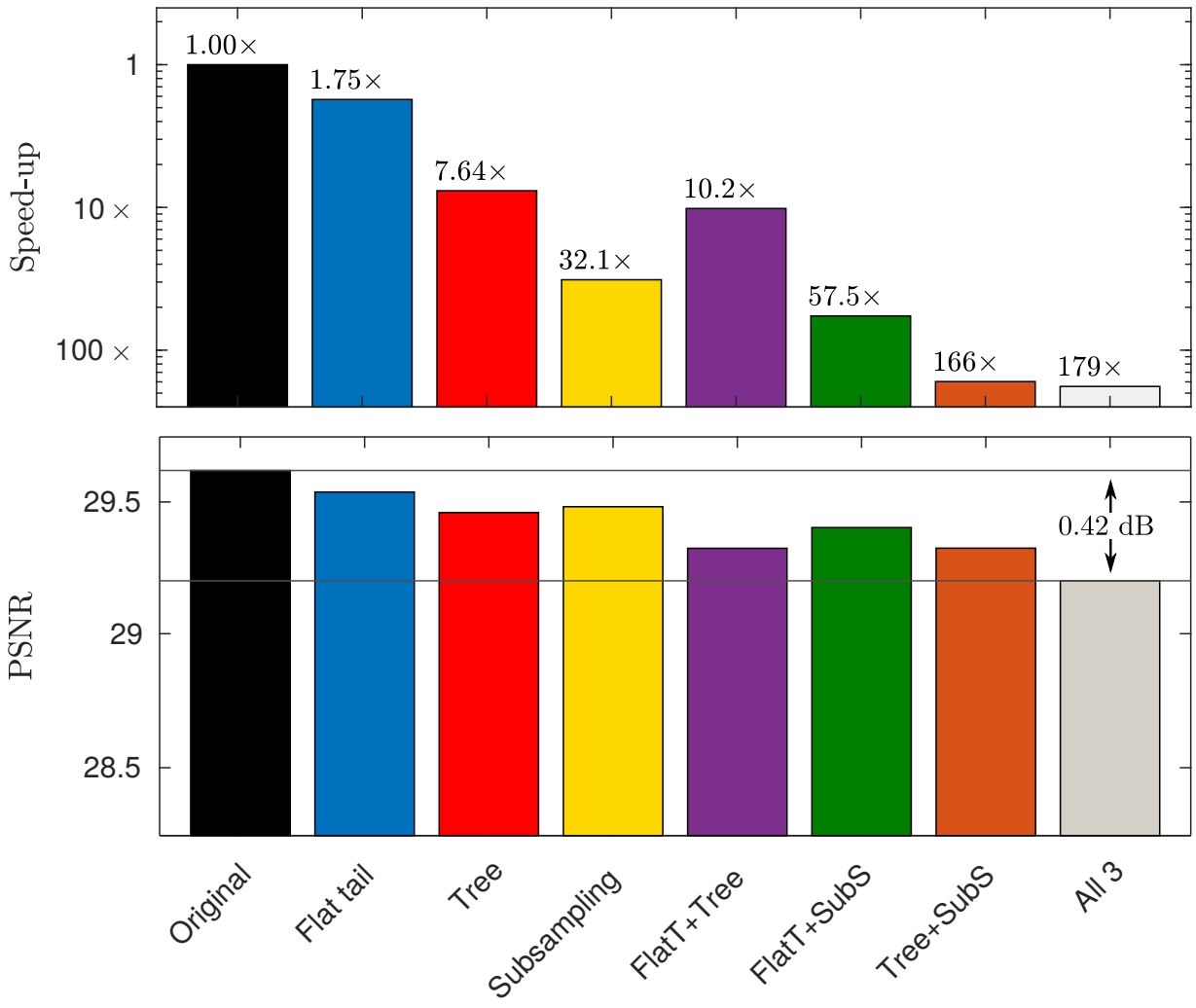


Figure 6.4: (top) Average speed-up and (bottom) average PSNR for our three accelerations, and all possible combinations of them, on the 40 images of the BSDS validation set.

speed-up is calculated with respect to the EPLLc implementation which is labeled "original" in Fig. 6.4. Among the three ingredients, random subsampling or jittering (labeled "subsampling") leads to the largest speed-up (32×), while the usage of the search tree provides more than 7× faster processing. The average speed-up obtained when combining all three ingredients is around 179× on our validation set consisting of images of size 481×321, for an average drop of PSNR less than 0.5dB. Note that most of the runtime gain is achieved by combining binary search tree and random subsampling (166×), while the flat tail acceleration leads to a modest acceleration of less than 10%. Nevertheless, we include it in our approach because it introduces another

parameter (ρ) that addresses the bottleneck presented by the dimensionality (P^2) in a unique way. Adjusting ρ lets a practitioner choose an appropriate operating point suitable for their need in the speed-vs-quality trade-off space without losing too much in terms of quality.

6.4 Related methods

To the best of our knowledge, there are only two other approaches [107, 108] that have attempted to accelerate EPLL. Unlike our approach, these methods focus on accelerating only one of the steps of EPLL namely the *Gaussian selection* step. Both use machine learning techniques to reduce its runtime.

In [107], the authors use a binary decision tree to approximate the mapping $\tilde{z}_i \mapsto k_i^*$ performed in step *Gaussian selection*. At each node k and level n of the tree, the patch $\tilde{z}_i$ is confronted with a linear separator in order to decide if the recursion should continue on the left or right child given by

$$\langle a_k^n, \tilde{z}_i \rangle + b_k^n \geqslant 0 \tag{6.21}$$

where (a_k^n, b_k^n) are the parameters of the hyperplane for the k-th node at level n. These separators are trained offline on all pairs of $(\tilde{z}_i, k_i^*)$ obtained after the first iteration of EPLL for a *given* β and noise level σ. Once a leaf has been reached, its index provides a first estimate for the index k_i^*. To reduce errors due to large variations among the neighboring pixels, this method further employs a Markov random fields on the resulting map of Gaussian components which runs in $O(NK)$ complexity. Hence, their overall approach reduces the complexity of step *Gaussian selection* from $O(NKP^2)$ to $O(N(PD+K))$, where $D = 12$ is the depth of the learned decision tree.

In [108], the authors approximate the *Gaussian selection* step, by using a gating (feed-

Table 6.2: Denoing performance and timing comparison: PSNR, SSIM and execution time on the BSDS test set (average on 60 images of size 481×321), and on six standard images (each of size 512×512 and results averaged over 10 independent noise realizations) for the proposed FEPLL and FEPLL′, EPLL (with timing given for both EPLLm [15] and our EPLLc), BM3D [8], CSF [100], RoG [108] and DnCNN [115] with 3 different levels of noise.

σ	Algo.	Berkeley	Barbara	Boat	Couple	Fingerprint	Lena	Mandrill
		PSNR/SSIM						
	FEPLL	36.8 /.959	36.9 /.958	36.6 /.930	36.6 /.944	35.4 /.984	38.2 /.941	35.1 /.959
	FEPLL′	37.1 /.962	37.1 /.959	36.7 /.933	36.8 /.946	35.5 /.984	38.3 /.943	35.2 /.960
	EPLL	37.3 /.963	37.6 /.962	36.8 /.933	37.3 /.950	36.4 /.987	38.6 /.944	35.2 /.960
5	RoG	37.1 /.959	36.7 /.954	36.5 /.920	37.2 /.946	36.4 /.987	38.3 /.939	35.1 /.956
	BM3D	37.3 /.962	38.3 /.964	37.3 /.939	37.4 /.949	36.5 /.987	38.7 /.944	35.3 /.959
	$CSF_{3\times3}$	36.8 /.952	37.0 /.955	36.7 /.929	37.1 /.945	36.2 /.986	38.2 /.938	34.8 /.953
	DnCNN	34.5 /.915	35.5 /.937	34.4 /.878	34.5 /.893	32.9 /.968	36.3 /.907	32.3 /.899
	FEPLL	29.1 /.812	29.0 /.852	30.2 /.802	29.9 /.813	27.6 /.908	32.3 /.863	26.4 /.784
	FEPLL′	29.3 /.831	29.5 /.866	30.3 /.814	30.1 /.825	27.8 /.916	32.4 /.864	26.5 /.802
	EPLL	29.5 /.836	29.8 /.872	30.6 /.821	30.4 /.834	28.3 /.924	32.6 /.869	26.7 /.807
20	RoG	29.4 /.828	28.4 /.838	30.5 /.815	30.3 /.827	28.3 /.922	32.5 /.865	26.5 /.794
	BM3D	29.4 /.824	31.7 /.904	30.8 /.824	30.7 /.842	28.8 /.928	33.0 /.876	26.6 /.794
	$CSF_{3\times3}$	29.0 /.805	28.3 /.821	30.2 /.802	29.9 /.812	28.0 /.916	31.8 /.837	26.1 /.778
	DnCNN	30.0 /.847	31.0 /.896	31.1 /.831	31.0 /.848	28.8 /.930	33.4 /.883	27.0 /.815
	FEPLL	24.5 /.614	23.6 /.636	25.5 /.644	25.1 /.629	22.2 /.722	27.4 /.742	21.4 /.468
	FEPLL′	24.5 /.620	23.8 /.648	25.5 /.646	25.1 /.636	22.4 /.745	27.3 /.733	21.6 /.499
	EPLL	24.8 /.631	24.0 /.660	25.8 /.659	25.4 /.649	22.6 /.755	27.6 /.747	21.7 /.506
60	RoG	24.6 /.623	23.3 /.626	25.6 /.654	25.2 /.640	22.4 /.739	27.4 /.747	21.5 /.482
	BM3D	24.8 /.637	26.3 /.757	25.9 /.671	25.6 /.666	23.8 /.801	28.2 /.778	21.7 /.500
	$CSF_{3\times3}$	22.0 /.489	21.4 /.490	22.7 /.502	22.5 /.505	21.2 /.727	23.4 /.511	20.2 /.466
	DnCNN	25.3 /.665	24.9 /.717	26.4 /.686	26.0 /.683	23.1 /.781	28.5 /.790	22.1 /.555
		Time (in seconds)						
	FEPLL	0.27	0.38	0.38	0.37	0.36	0.38	0.38
	FEPLL′	0.96	1.28	1.28	1.28	1.26	1.27	1.28
	EPLLc	43.82	71.14	71.24	71.31	71.28	71.23	71.31
	EPLLm	82.68	145.15	143.68	144.30	144.13	144.15	143.86
	RoG	1.16	1.92	1.92	1.93	1.91	1.93	1.92
	BM3D	1.60	2.52	2.68	2.59	2.17	2.61	2.64
	$CSF_{3\times3}$	0.87	1.09	1.09	1.13	1.09	1.09	1.08
	$CSF_{3\times3}^{gpu}$	0.38	0.39	0.39	0.39	0.39	0.39	0.39
	DnCNN	2.09	3.57	3.55	3.60	3.60	3.60	3.61
	$DnCNN^{gpu}$	0.28	0.63	0.63	0.64	0.63	0.64	0.64

forward) network with one hidden layer

$$\tilde{z}_i \mapsto \iota^k + \sum_{j=1}^{Q} \left(\log \nu_j^k + \omega_j^k (\tilde{d}_i)_j^2 \right) \quad \text{with} \quad \tilde{d}_i = V^t \tilde{z}_i \tag{6.22}$$

where Q is the size of the hidden layer. The matrix $V \in \mathbb{R}^{P\times Q}$ encodes the weights of the first layer, ω^k corresponds to the weights of the hidden layer and they are learned discriminatively to approximate the exact *posterior* probability:

$$\tilde{z}_i \mapsto \iota^k + \sum_{j=1}^{P} \left(\log \nu_j^k + \frac{(\tilde{c}_i^k)_j^2}{\nu_j^k} \right) \quad \text{with} \quad \tilde{c}_i^k = U_k^t \tilde{z}_i \tag{6.23}$$

that we encounter in eq. (6.7) and (6.8). Theoretically, a new network will need to be trained for each type of degradations, noise levels and choices of β (recall that $\nu_j^k = (S_k)_{jj} + \frac{1}{\beta}$). However, the findings of [108] indicate that applying a network learned on clean patches and with $\frac{1}{\beta} = 0$ is effective regardless of the type of degradation or the value of β. Their main advantage can be highlighted by comparing eq. (6.22) and (6.23) where complexity is reduced from $O(NKP^2)$ to $O(NQ(K+P))$. The authors utilize this benefit by choosing $Q = 100$.

Unlike these two approaches, our method does not try to learn the Gaussian selection rule directly (which depends on both the noise level through $1/\beta$ and the *prior* model through the GMM). Instead, we simply define a hierarchical organization of the covariance matrices Σ_k. In other words, while the two other approaches try to infer the *posterior* probabilities (or directly the maximum *a posteriori*), our approach provides an approximation to the *prior* model. During runtime, this approximation of the *prior* is used in the *posterior* for the Gaussian selection task. Please note that the value of β does not play a role in determining the *prior*. This allows us to use the same search tree independently of the noise level, degradations, *etc.* Given that the main advantage of EPLL is that the same model can be used for any type of degradations, it is important that this property remains true for the accelerated version. Last but not least, the training of our search tree takes a few minutes while the training steps for the above mentioned approach take from several hours to a few days [107].

Apart from methods that accelerate EPLL, [33] and [93] are two works that use strategies sharing similarities to our proposed flat-tail and binary search tree approximations, respectively.

In [33], the authors use a flat-tail GMM to model distribution of noisy patches. Since the GMM is learned directly on noisy patches, the constant value of the tail corresponds to the noise variance. This allows to improve inference when learning on noisy datasets, estimating the noise level, and retrieving the intrinsic dimension of each cluster. This is different from our approach which uses GMM priors learned on clean patch and provides flat-tail approximation using mean of least significant coefficients. In [93], the authors introduce a general data structure called *covariance tree* (CovTree). A CovTree is constructed by first building a binary space partition tree on patches (data points) and patches in each node are then modeled using a Gaussian distribution. While the resulting tree of Gaussians do share some similarities with our binary search tree, the learning process is very different. The covariance tree is built directly on data points, our approach is applied on an already learned GMM. In addition, unlike CovTree, our proposed strategy can handle any number of components (not just powers of two) and also encodes all parameters of mixture components including the mixing weights.

In the next section, we show that our proposed accelerations produce restoration results with comparable quality to competing methods while requiring a smaller amount of time.

6.5 Numerical experiments

In this section, we present the results obtained on various image restoration tasks. Our experiments were conducted on standard images of size 512×512 such as Barbara, Boat, Couple, Fingerprint, Lena, Mandrill and on 60 test images of size 481×321 from the Berkeley Segmentation Dataset (BSDS) [109] (the original BSDS test set contains 100 images, the other 40 was used for validation purposes while setting parameters ρ and s). For denoising, we compare the performance of our fast EPLL (FEPLL) to the original EPLL algorithm [15] and BM3D [8]. For the original EPLL, we have included timing results given by our own `MATLAB/C` implementation (EPLLc) and the `MATLAB` implementation provided by the authors (EPLLm). We also compare our

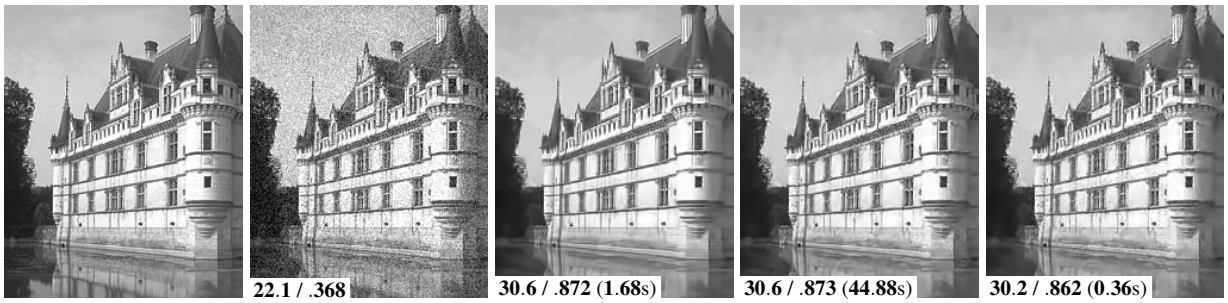
restoration performance and runtime against other fast restoration methods introduced to achieve competitive trade-off between runtime efficiency and image quality. These methods include RoG [108] (a method accelerating EPLL based on feedforward networks described in Sec. 6.4), and CSF [100] (a fast restoration technique using random field-based architecture).

For deblurring experiments, we additionally compare with field-of-experts (FoE)-based non-blind deconvolution [20] denoted as iPiano. We contacted the corresponding author of [107] and got confirmation that the implementation of their algorithm (briefly described in 6.4) is not publicly available. Due to certain missing technical details, we were unable to reimplement it faithfully. However, the results reported in [107] indicate that their algorithm performs in par with BM3D in terms of both PSNR and time. Hence, BM3D results can be used as a faithful proxy for the expected performance of Wang et al.'s algorithm [107].

To explicitly illustrate the quality vs. runtime tradeoff of FEPLL, we include results obtained using a slightly slower version of FEPLL referred to as FEPLL$'$, that does not use the balanced search tree and uses a flat tail spectrum approximation with $\rho = 0.98$. Please note that FEPLL$'$ is not meant to be better or worse than FEPLL, it is just another version running at a different PSNR/time tradeoff which allows us to compare our algorithm to others operating in different playing fields.

Finally, to illustrate the versatility of FEPLL, we also include results for other inverse problems such as devignetting, super-resolution, and inpainting. Additional results can be found in Appendix A.

Parameter settings In our experiments, we use patches of size $P = 8 \times 8$, and the GMM provided by Zoran *et al.* [15] with $K = 200$ components. The 200-components GMM is progressively collapsed into smaller GMMs with $K = 64, 32, 16, 8, 4, 2$ and 1, and then all Gaussians of the tree are modified offline by flat-tail approximations with $\rho = 0.95$. The final estimate for the restored image is obtained after 5 iterations of our algorithm with β set to $\lambda\sigma^{-2}\{1, 4, 8, 16, 32\}$ where $\lambda = \min\{N^{-1}\|\mathcal{A}^t\mathcal{A}\|_F^2\|\mathcal{A}\|_2^{-2}, 250\sigma^2\}$. For denoising, where $\mathcal{A}$ is identity, $\lambda = 1$ which boils



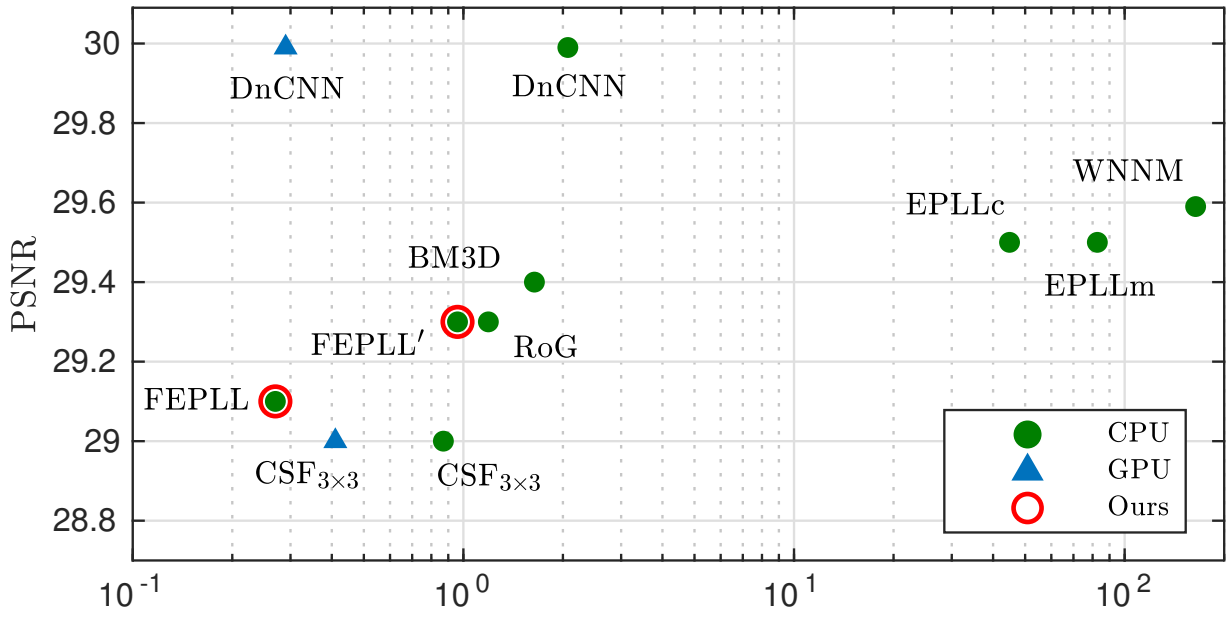
(a) Reference x (b) Noisy image y (c) BM3D result $\hat{x}$ (d) EPLLc result $\hat{x}$ (e) FEPLL result $\hat{x}$

Figure 6.5: Illustration of a denoising problem with noise standard deviation $\sigma = 20$. Part of: (a) the original image (b) its noisy version (c-e) denoised results of competitive methods with PSNR/SSIM and time (inset).

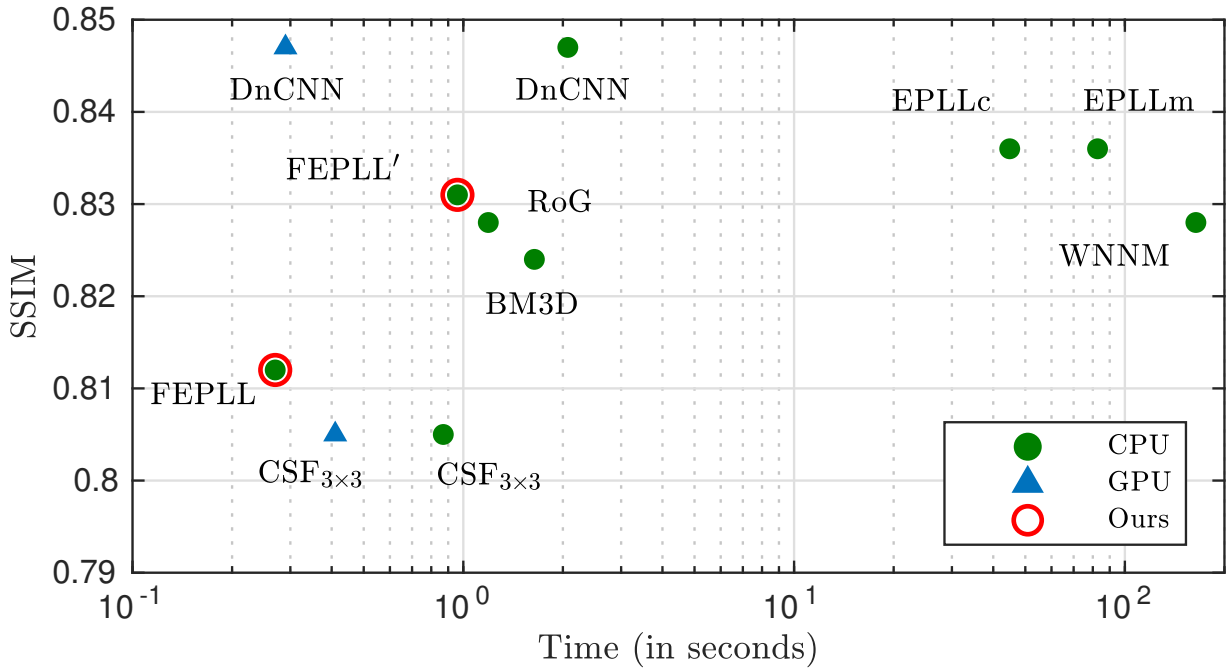
down to the setting used by Zoran *et al.* [15]. For inverse problems, we found that the initialization $\hat{x} = (\mathcal{A}^t\mathcal{A} + 0.2\sigma^2/\lambda\nabla)^{-1}\mathcal{A}^t y$, with ∇ the image Laplacian, provides relevant solutions whatever the linear operator $\mathcal{A}$ and the noise level σ^2. While the authors of [15] do not provide any further direction for setting β and the initialization in general inverse problems, our proposed setting leads to competitive solutions irrespective of $\mathcal{A}$ and σ^2. For BM3D [8], EPLLm [15], RoG [108], CSF [100] and iPiano [20] we use the implementations provided by the original authors and use the default parameters prescribed by them.

Denoising

Table 6.2 shows the quantitative performances of FEPLL on the denoising task compared to EPLLm [15], EPLLc (our own `MATLAB/C` implementation), RoG [108] BM3D [8], CSF [100] and DnCNN [105]. We evaluate the algorithms under low-, mid- and high-noise settings by using Gaussian noise of variance 5^2, 20^2 and 60^2, respectively. The result labeled "Berkeley" is an average over 60 images from the BSDS testing set [109]. All numbers are averages obtained on 10 independent noise realizations. Figures 6.6 provide graphical representations of these performances in terms of PSNR/SSIM versus computation time for the BSDS images for the noise variance setting $\sigma^2 = 20^2$. In this figure, we have also included a recent approach based

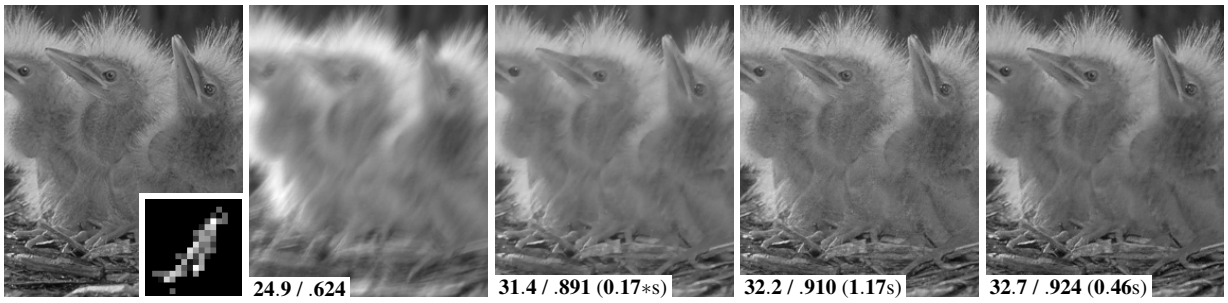


(a)



(b)

Figure 6.6: (a) PSNR and (b) SSIM versus time for different restoration methods in a denoising problem with noise standard deviation $\sigma = 20$. PSNR, SSIM and time are averaged on the 60 BSDS test images, each of size 481×321. Optimal methods tend to be in the top-left corner.



(a) Reference x / Blur kernel (b) Blurry image y (c) CSF_{pw} result $\hat{x}$ (d) RoG result $\hat{x}$ (e) FEPLL result $\hat{x}$

Figure 6.7: Illustration of a deblurring problem with noise standard deviation $\sigma = 0.5$. Part of: (a) the original image and the blur kernel (inset), (b) blurry version (c-e) deblurred results of competitive methods with PSNR/SSIM and time (inset). The '*' indicates runtime on GPU while others are CPU times.

on weighted nuclear norm minimization (WNNM) [116]. On average, FEPLL results are 0.5dB below regular EPLL and BM3D; however, FEPLL is approximately 7 times faster than BM3D, 170-200 times faster than EPLLc and over 350 times faster than EPLLm. FEPLL outperforms the faster CSF algorithm in terms of both PSNR and time. In this case, FEPLL is even faster than the GPU accelerated version of CSF (CSF_{gpu}). Our approach is 4 times faster than RoG with a PSNR drop of 0.1-0.3dB. Nevertheless, if we slow down FEPLL to FEPLL$'$, we can easily neutralize this quality deficit while still being faster than RoG. While DnCNN offers better results, DnCNN (CPU) is about $10\times$ slower than FEPLL (also CPU). Our FEPLL (CPU) is also slightly faster than DnCNN (GPU). WNNM offers comparable results to FEPLL' but is about $500\times$ slower. Note that these accelerations are obtained purely based on the approximations and *no* parallel processing is used. Also, in most cases, a loss of 0.5dB may not affect the visual quality of the image. To illustrate this, we show a sample image denoised by BM3D, EPLL and FEPLL in Fig. 6.5.

Deblurring

Table 6.3 shows the performance of FEPLL when used for deblurring as compared to RoG [108], iPiano [20] and CSF [100]. Once again, FEPLL uses Alg. 8 with operator $\mathcal{A}$ in the *Image estimation* step defined by $\mathcal{A}x = \mathcal{F}^{-1}[\mathcal{F}[h] \odot \mathcal{F}[x]]$, where $\mathcal{F}$ is the fast Fourier transform and $\mathcal{F}^{-1}$ its inverse, $\odot$ indicates element-wise product and h is the blur kernel. For these experiments, we use the blur kernel provided by Chen et al. [20] along with their algorithm implementation. The results under the label "Berkeley" are averaged over 60 images from the BSDS test dataset [109]. The results labeled "Classic" is averaged over the six standard images (Barbara, Boat, Couple, Fingerprint, Lena and Mandrill). FEPLL consistently outperforms its efficient competitors both in terms of quality and runtime. Although the GPU version of CSF is faster, the restoration quality obtained by CSF is 2-3dB lower than FEPLL. The proposed algorithm outperforms RoG by 1-1.8dB while running 3 and 5 times faster on "Berkeley" and "Classic" datasets, respectively.

The superior qualitative performance of FEPLL is demonstrated in Fig. 6.7. For brevity, we only include the deblurring results obtained from the top competitors of FEPLL algorithm in terms of both quality and runtime. As observed, FEPLL provides the best quality vs. runtime efficiency trade-off. In contrast, a deblurring procedure using the regular EPLL is around 350 times slower than FEPLL with the original implementation [15]. Specifically, on the sample image shown in Fig. 6.7, EPLL provides a qualitatively similar result (not shown in the figure) with a PSNR of 32.7 dB and SSIM of 0.922 in 142 seconds.

Other inverse problems

Unlike BM3D, EPLL and FEPLL are more versatile and handle a wide range of inverse problems without any change in formulation. In Fig. 6.8, we show the results obtained by FEPLL on problems such as (a) devignetting, which involves a progressive loss of intensity, (b) super-resolution and (c) inpainting. To show the robustness of our method, the input images of size

Table 6.3: Deblurring performance and timing comparison: PSNR, SSIM and execution time on the BSDS test set (average of 60 images of size 481×321), and on standard images (average of 6 images of size 512×512) for the proposed FEPLL and FEPLL$'$, CSF [100], RoG [108] and Chen et al.'s [20] method that is called *iPiano* in their implementation. The blur kernel used is the one provided in along with iPiano implementation and noise is set to $\sigma = 0.5$.

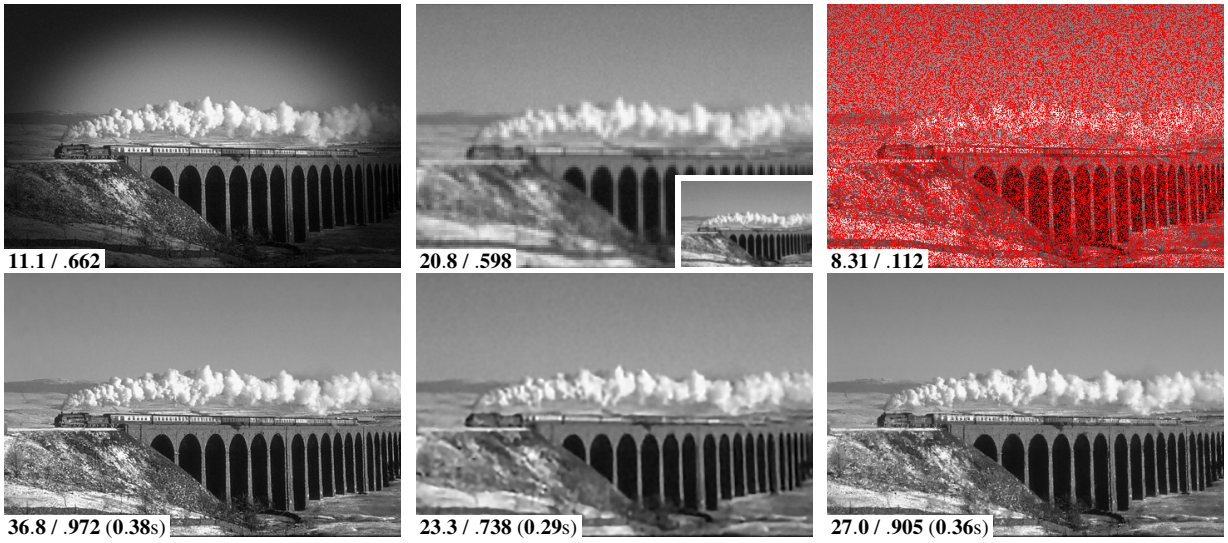
Algo.	Berkeley		Classic	
	PSNR/SSIM	Time (s)	PSNR/SSIM	Time (s)
iPiano	29.5 / .824	29.53	29.9 / .848	59.10
CSF_{pw}	30.2 / .875	0.50 (0.14*)	30.5 / 0.870	0.47 (0.14*)
RoG	31.3 / .897	1.19	31.8 / .915	2.07
FEPLL	33.1 / .928	0.40	32.8 / .931	0.46
FEPLL'	33.2 / .930	1.01	33.0 / .933	1.82

481×321 were degraded with zero-mean Gaussian noise with $\sigma=2$. All of the restoration results were obtained within/under 0.4 seconds and with the same set of parameters explained above (cf. *Parameter settings*).

6.6 Conclusion

In this chapter, we accelerate EPLL by a factor greater than 100 with negligible loss of image quality (less than 0.5dB). This is achieved by combining three independent strategies: a flat tail approximation, matching via a balanced search tree, and stochastic patch sampling. We show that the proposed accelerations are effective in denoising and deblurring problems, as well as in other inverse problems such as super-resolution and devignetting. An important distinction of the proposed accelerations is their genericity: the accelerated EPLL prior can be applied to many restoration tasks and various signal-to-noise ratios, in contrast to existing accelerations based on learning techniques applied to specific conditions (such as image size, noise level, blur kernel, etc.) and that require an expensive re-training to address a different problem.

Since the speed-up is achieved solely by reducing the algorithmic complexity, we believe that further inclusion of accelerations based on parallelization and/or GPU implementations will allow for real-time video processing. Moreover, the acceleration techniques introduced in this



(a) devignetting (b) $\times 3$ super-resolution (c) 50% inpainting

Figure 6.8: FEPLL on various inverse problems. All inputs contain Gaussian noise with $\sigma = 2$. Top row: (a) the observation in a devignetting problem, (b) the bi-cubic interpolation and the actual low-resolution size image (inset) in a $\times 3$ super-resolution problem and (c) the observation in an inpainting problem with 50% of missing pixels shown in red. Bottom row: respective FEPLL results all obtained in less than 0.4s.

work are general strategies that can be used to speed up other image restoration and/or related machine learning algorithms.

In the next chapter, we investigate the suitability and usability of a class of priors that are more general than GMM.

Acknowledgments

Parts of this chapter is a reprint of materials in the journal paper "Accelerating GMM-based patch priors for image restoration: Three ingredients for a 100$\times$ speed-up", S. Parameswaran, C-A. Deledalle, L. Denis and T. Q. Nguyen [5]. The dissertation author is the primary investigator and author of this paper.

Chapter 7

Image denoising with generalized Gaussian mixture model patch priors

7.1 Introduction

As we saw in the earlier chapters, priors on natural images play an important role in image restoration algorithms. Image priors are used to denoise or regularize ill-posed restoration problems such as deblurring and super-resolution, to name just a few. Early attempts in designing image priors relied on modeling local pixel gradients with Gibbs distributions [117], Laplacian distributions (total variation) [26, 118], hyper-Laplacian distribution [111], generalized Gaussian distribution [119], or Gaussian mixture models [120]. Concurrently, priors have also been designed by modeling coefficients of an image in a transformed domain using generalized Gaussian [38, 39, 121, 122] or scaled mixture of Gaussian [49] priors for wavelet or curvelet coefficients [123]. Alternatively, modeling the distribution of patches of an image (*i.e.*, small windows usually of size 8×8) has proven to be a powerful solution. In particular, popular patch techniques rely on non-local self-similarity [27], fields of experts [19], learned patch dictionaries [30, 45, 47], sparse or low-rank properties of stacks of similar patches [28, 37, 11, 124], patch re-

occurrence priors [125], or more recently mixture models patch priors [15, 17, 31, 32, 33, 34, 35].

Of these approaches, a successful approach introduced by Zoran and Weiss [15] is to model patches of clean natural images using Gaussian Mixture Model (GMM) priors. The agility of this model lies in the fact that a prior learned on clean image patches can be effectively employed to restore a wide range of inverse problems. It is also easily extendable to include other constraints such as sparsity or multi-resolution patches [46, 96]. The use of GMMs for patch priors make these methods computationally tractable and flexible. Although GMM patch prior is effective and popular, in this article, we argue that a generalized Gaussian mixture model (GGMM) is a better fit for image patch prior modeling. Compared to a Gaussian model, a generalized Gaussian distribution (GGD) has an extra degree of freedom controlling the shape of the distribution and it encompasses Gaussian and Laplacian models.

Beyond image restoration tasks, GGDs have been used in several different fields of image and signal processing, including watermark detection [126], texture retrieval [122], voice activity detection [127] and MP3 audio encoding [128], to cite just a few. In these tasks, GGDs are used to characterize or model the prior distribution of clean signals, for instance, from their DCT coefficients or frequency subbands for natural images [129, 130, 131, 132] or videos [133], gradients for X-ray images [119], wavelet coefficients for natural [38, 39, 126], textured [122], or ultrasound images [134], tangential wavelet coefficients for three-dimensional mesh data [135], short time windows for speech signals [136] or frequency subbands for speech [127] or audio signals [128].

In this chapter, we go one step further and use multi-variate GGD with one scale and one shape parameter per dimension. The superior patch prior modeling capability of such a GGMM over a GMM is illustrated in Figure 7.1. The figure shows histograms of six orthogonal 1-D projections of subset of clean patches onto the eigenvectors of the covariance matrix of a single component of the GMM. To illustrate the difference in the shapes (ν) and scales (λ) of the distributions of each dimension, we have chosen a few projections corresponding to both the

most and the least significant eigenvalues. It can be seen that GGD is a better fit on the obtained histograms than a Gaussian model. Additionally, different dimensions of the patch follow a different GGD (*i.e.*, has a different shape and scale parameter). Hence, it does not suffice to model all the feature dimensions of a given cluster of patches as Laplacian or Gaussian. Therefore, we propose to model patch priors as GGMM distributed with a separate shape and scale parameters for each feature dimension of a GGD component. This differs from the recent related approach in [35] that considered GGMM where each component has a fixed shape parameter for all directions.

Contributions: The goal of this chapter is to measure the improvements obtained in image denoising tasks by incorporating a GGMM in EPLL algorithm. Unlike [35], that incorporates a GGMM prior in a posterior mean estimator based on importance sampling, we directly extend the maximum *a posteriori* formulation of Zoran and Weiss [15] for the case of GGMM priors. While such a GGMM prior has the ability to capture the underlying distribution of clean patches more closely, we will show that it introduces two major computational challenges in this case. The first one can be thought of as a classification task in which a noisy patch is assigned to one of the components of the mixture. The second one corresponds to an estimation task where a noisy patch is denoised given that it belongs to one of the components of the mixture. Due to the interaction of the noise distribution with the GGD prior, we first show that these two tasks lead to a group of one-dimensional integration and optimization problems, respectively. Specifically, for $x \in \mathbb{R}$, these problems are of the following forms

$$\int_{\mathbb{R}} \exp\left(-\frac{(t-x)^2}{2\sigma^2} - \frac{|t|^\nu}{\lambda_\nu^\nu}\right) \mathrm{d}t \qquad \text{and} \qquad \operatorname*{argmin}_{t\in\mathbb{R}} \frac{(t-x)^2}{2\sigma^2} + \frac{|t|^\nu}{\lambda_\nu^\nu}\,, \tag{7.1}$$

for some $\nu > 0$, $\sigma > 0$ and $\lambda_\nu > 0$. In general, they do not admit closed-form solutions but some particular solutions or approximations have been derived for the estimation/optimization problem [39, 137]. By contrast, up to our knowledge, little is known for approximating the classification/integration one (only crude approximations were proposed in [138]).

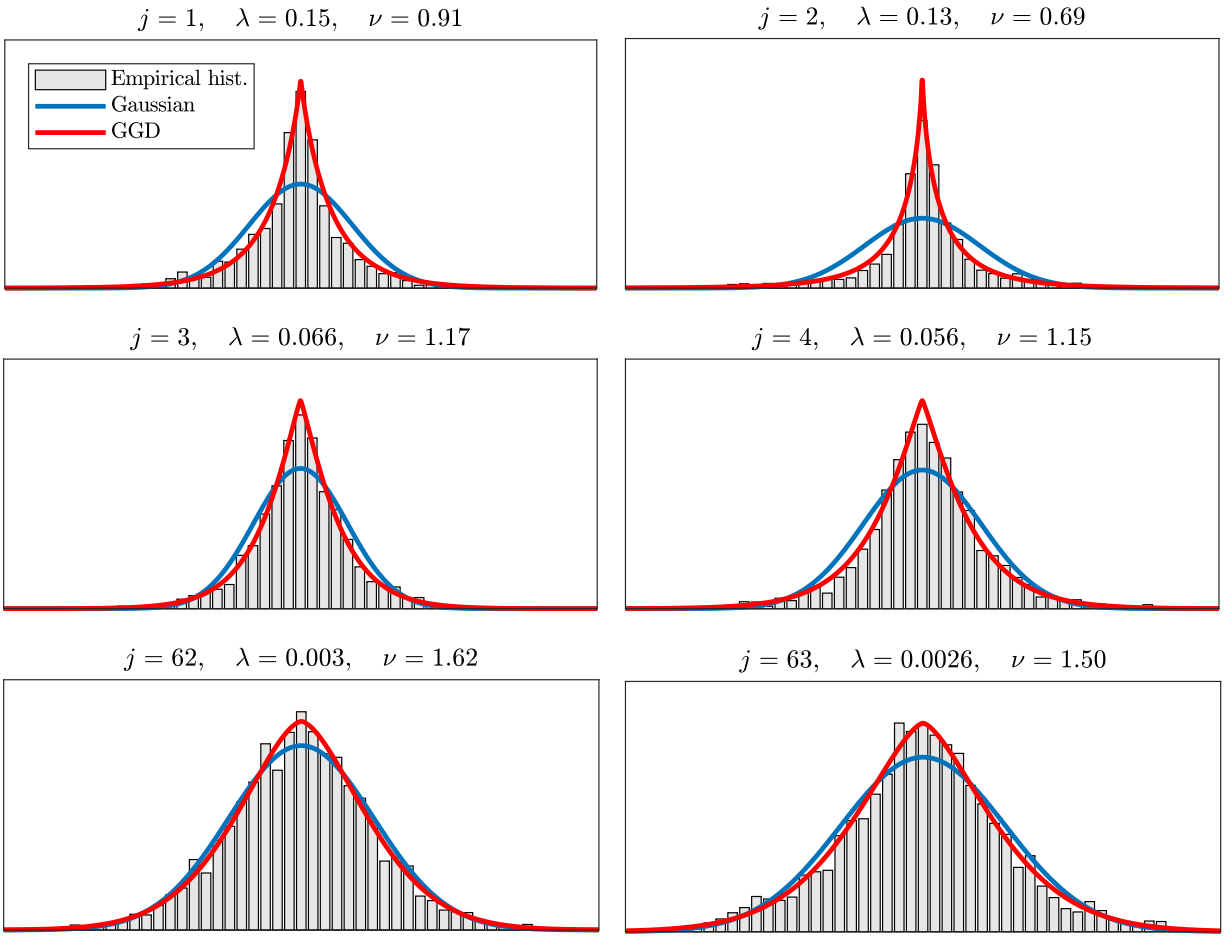


Figure 7.1: Histograms of the projection of $200,000$ clean patches on 6 eigenvectors $j = 1, 2, 3, 4, 62$ and 63 of the covariance matrix of one component k of the mixture (with weight $w_k = 1.3\%$). The contribution of each clean patch in the histograms is given by its membership values onto this component k (as obtained during the E-Step of EM). For each histogram, a generalized Gaussian distribution was adjusted by estimating the parameters λ and ν by moment estimation (as obtained during the M-Step of our modified EM exposed in Section 7.3.1). For comparisons, we have also provided illustrations of the best fit obtained with a Gaussian distribution.

Our contributions are both theoretical- and application-oriented. The major contribution of this paper, which is and theoretical in nature, is to develop an accurate approximation for the classification/integration problem. In particular, we show that our approximation error vanishes for $x \to 0$ and $x \to \pm\infty$ when $\nu = 1$, see Theorem 7.4.3 and Theorem 7.4.4. We next generalize this result for $\frac{2}{3} < \nu < 2$ in Theorem 7.4.5 and Theorem 7.4.6. In addition, we prove that the two problems enjoy some important desired properties in Proposition 7.4.2 and Proposition 7.5.1. These theoretical results allow the two quantities to be approximated by functions that can be quickly evaluated in order to be incorporated in fast algorithms. Our last contribution is experimental and concerns the performance evaluation of the proposed model in image denoising scenario. For reproducibility, we have released our implementation at https://bitbucket.org/cdeledalle/ggmm-epll.

Potential impacts beyond image denoising: It is important to note that the two main contributions presented in this work, namely approximations for classification and estimation problems, are general techniques that are relevant to a wider area of research problems than image restoration. In particular, our contributions apply to any problems where (i) the underlying clean data are modeled by a GGD or a GGMM, whereas (ii) the observed samples are corrupted by Gaussian noise. They are especially relevant in machine learning scenarios where a GGMM is trained on clean data but data provided during testing time is noisy. That is, our approximations can be used to extend the applicability of the aforementioned *clean* GGD/GGM based approaches to the less than ideal testing scenario where the data is corrupted by noise. For instance, using the techniques introduced in this paper, one could directly use the GGD based voice activity model of [127] into the likelihood ratio test based detector of [139] (which relies on solutions of the integration problem but was *limited* to Laplacian distributions). In fact, we suspect that many studies in signal processing may have limited themselves to Gaussian or Laplacian signal priors because of the complicated integration problem arising from the intricate interaction of GGDs with Gaussian noise. In this chapter, we demonstrate that this difficulty can be efficiently

overcome with our approximations. This leads us to believe that the impact of the approaches presented in this paper will not only be useful for image restoration but also aid a wider field of general signal processing applications.

Organization: After explaining the considered patch prior based restoration framework in Section 7.2, we derive our GGMM based restoration scheme in Section 7.3. The approximations of the classification and estimation problems are studied in Section 7.4 and Section 7.5, respectively. Finally, we present numerical experiments and results in Section 7.6.

7.2 Background

In this section we provide a detailed overview of the use of patch-based priors in Expected Patch Log-Likelihood (EPLL) framework and its usage under GMM priors.

7.2.1 Image restoration with patch based priors

Recall that image restoration is the problem of estimating an image $u \in \mathbb{R}^N$ (N is the number of pixels) from noisy linear observations $v = \mathcal{A}u + w$, where $\mathcal{A} : \mathbb{R}^N \to \mathbb{R}^M$ is a linear operator and $w \in \mathbb{R}^M$ is a noise component assumed to be white and Gaussian with variance σ^2. $\mathcal{A}$ can account for a variety degradation operators (e.g. blur) but for the purposes of this work which focuses on image denoising, $\mathcal{A}$ is the identity matrix.

To reduce noise and stabilize the inversion of $\mathcal{A}$, some *prior* information is used for the estimation of u. Recent techniques [45, 15, 46] include this *prior* information as a model for the distribution of patches found in natural clean images. We consider the EPLL framework [15] that restores an image by maximum *a posteriori* estimation over all patches, corresponding to the

following minimization problem:

$$\underset{u\in\mathbb{R}^n}{\operatorname{argmin}} \frac{P}{2\sigma^2}\|\mathcal{A}u - v\|^2 - \sum_{i=1}^{N} \log p\left(\mathcal{P}_i u\right) \tag{7.2}$$

where $\mathcal{P}_i : \mathbb{R}^N \to \mathbb{R}^P$ is the linear operator extracting a patch with P pixels centered at the pixel with location i (typically, $P = 8\times 8$), and $p(.)$ is the *a priori* probability density function (*i.e.*, the statistical model of noiseless patches in natural images). Since i scans all of the N pixels of the image, all patches contribute to the loss and many patches overlap. Allowing for overlapping is important because otherwise there would appear blocking artifacts. While the first term in eq. (7.2) ensures that $\mathcal{A}u$ is close to the observations v (this term is the negative log-likelihood under the white Gaussian noise assumption), the second term regularizes the solution u by favoring an image such that all of its patches fit the *prior* model of patches in natural images.

Optimization with half-quadratic splitting

Problem (7.2) is a large optimization problem where $\mathcal{A}$ couples all unknown pixel values of u and the patch prior is often chosen non-convex. Our method follow the choice made by EPLL of using a classical technique, known as half-quadratic splitting [110, 111], that introduces N auxiliary unknown vectors $z_i \in \mathbb{R}^P$, and alternatively consider the penalized optimization problem, for $\beta > 0$, as

$$\underset{\substack{u\in\mathbb{R}^n \\ z_1,\ldots,z_N\in\mathbb{R}^P}}{\operatorname{argmin}} \frac{P}{2\sigma^2}\|\mathcal{A}u - v\|^2 + \frac{\beta}{2}\sum_{i\in I}\|\mathcal{P}_i u - z_i\|^2 - \sum_{i\in I}\log p\left(z_i\right). \tag{7.3}$$

When $\beta \to \infty$, the problem (7.3) is equivalent to the original problem (7.2). In practice, an increasing sequence of β is considered, and the optimization is performed by alternating between the minimization for u and z_i. Though little is known about the convergence of this algorithm, few iterations produce remarkable results, in practice. We follow the EPLL settings prescribed in

[15] by performing 5 iterations of this algorithm with parameter β set to $\frac{1}{\sigma^2}\{1,4,8,16,32\}$ for each iteration, respectively. The algorithm is initialized using $\hat{u} = v$ for the first estimate.

Minimization with respect to u

Considering all z_i to be fixed, optimizing (7.3) for u corresponds to solving a linear inverse problem with a Tikhonov regularization. It has an explicit solution known as the linear minimum mean square estimator (or often referred to as Wiener filtering) which is obtained as:

$$\begin{aligned}\hat{u} &= \underset{u\in\mathbb{R}^n}{\operatorname{argmin}} \frac{P}{2\sigma^2}\|\mathcal{A}u - v\|^2 + \frac{\beta}{2}\sum_{i\in I}\|\mathcal{P}_i u - \hat{z}_i\|^2 \\ &= \left(\mathcal{A}^t\mathcal{A} + \frac{\beta\sigma^2}{P}\sum_{i\in I}\mathcal{P}_i^t\mathcal{P}_i\right)^{-1}\left(\mathcal{A}^t v + \frac{\beta\sigma^2}{P}\sum_{i\in I}\mathcal{P}_i^t\hat{z}_i\right), \end{aligned} \tag{7.4}$$

where $\mathcal{P}_i^t\mathcal{P}_i$ is a diagonal matrix whose i-th diagonal element corresponds to the number of patches overlapping the pixel of index i.

Minimization with respect to z_i Considering u to be fixed, optimizing (7.3) for z_i leads to:

$$\hat{z}_i \leftarrow \underset{z_i\in\mathbb{R}^P}{\operatorname{argmin}} \frac{\beta}{2}\|\tilde{z}_i - z_i\|^2 - \log p(z_i) \quad \text{where} \quad \tilde{z}_i = \mathcal{P}_i\hat{u}\,, \tag{7.5}$$

which corresponds to the maximum *a posterior* (MAP) denoising problem under the patch prior p of a patch $\tilde{z}_i$ contaminated by Gaussian noise with variance $1/\beta$. The solution of this optimization problem strongly depends on the properties of the chosen patch prior.

Our algorithm will follow the exact same procedure as EPLL by alternating between eq. (7.4)[1] and (7.5). In our proposed method, we will be using a generalized Gaussian mixture model (GGMM) to represent patch prior. The general scheme we will adopt to solve eq. (7.5) under GGMM prior is inspired from the one proposed by Zoran & Weiss [15] in the simpler case

[1] Since our study focuses only on denoising, we will consider $\mathcal{A} = \mathrm{Id}_N$.

of Gaussian mixture model (GMM) prior. For this reason, we will now introduce this simpler case of GMM prior before exposing the technical challenges arising from the use of GGMM prior in Section 7.3.

7.2.2 Patch denoising with GMM priors

The authors of [15, 17] suggested using a zero-mean Gaussian mixture model (GMM) prior[2], that, for any patch $z \in \mathbb{R}^P$, is given by

$$p(z) = \sum_{k=1}^{K} w_k \mathcal{N}_P(z; 0_P, \Sigma_k) \tag{7.6}$$

where K is the number of components, $w_k > 0$ are weights such that $\sum_k w_k = 1$, and $\mathcal{N}_P(0_P, \Sigma_k)$ denotes the multi-variate Gaussian distribution with zero-mean and covariance $\Sigma_k \in \mathbb{R}^{P \times P}$. A K-component GMM prior models image patches as being spread over K clusters that have ellipsoid shapes where each coefficient (of each component) follows a Gaussian distribution, *i.e.*, bell-shaped with small tails. In [15], the parameters w_k and Σ_k of the GMM are learned using the Expectation Maximization algorithm [48] on a dataset of 2 million clean patches of size 8×8 pixels that are randomly extracted from the training images of the Berkeley Segmentation Database (BSDS) [109]. The GMM learned in [15] has $K = 200$ zero-mean Gaussian mixture components.

Due to the multi-modality of the GMM prior, introducing this prior in eq. (7.5) makes the optimization problem highly non-convex:

$$\hat{z} \leftarrow \underset{z \in \mathbb{R}^P}{\operatorname{argmin}} \frac{\beta}{2} \|\tilde{z} - z\|^2 - \log \left[\sum_{k=1}^{K} w_k \mathcal{N}(z; 0_P, \Sigma_k) \right] . \tag{7.7}$$

To circumvent this issue, the EPLL framework introduced by Zoran et al. [15] uses an approxima-

[2]To enforce the zero-mean assumption, patches are first centered on zero, then denoised using eq. (7.5), and, finally, their initial means are added back. In fact, one can show that it corresponds to modeling $p(z - \bar{z})$ with a GMM where $\bar{z}_j = \frac{1}{P} \sum_i z_i$ for all $1 \leqslant j \leqslant P$.

tion[3]. In a nutshell, the approximated approach of EPLL provides a way to avoid the intractability of mixture models, thus making them available to model patch priors [15]. Section 7.2.2 provides an illustration of the EPLL framework and the steps involved in solving the optimization problem (7.7) namely:

(i) compute the posterior $p(k|\tilde{z})$[4] of each Gaussian component, $k = 1, 2 \ldots K$ for the given noisy patch $\tilde{z}$ with assumed noise variance of $1/\beta$,

(ii) select the component $k^\star$ that best explains the given patch $\tilde{z}$,

(iii) perform whitening by projecting $\tilde{z}$ over the main directions of that cluster (given by the eigenvectors of $\Sigma_{k^\star}$), and

(iv) apply a linear shrinkage on the coefficients with respect to the noise variance $1/\beta$ and the spread of the cluster (encoded by the eigenvalues).

The details of each of these steps will be discussed in Section 7.3.2 as part of the development of the proposed model which is more general.

In this chapter, we suggest using a mixture of generalized Gaussian distributions that will enable image patches to be spread over clusters that are bell shaped in some directions but can be peaky with large tails in others. While the use of GMM priors leads to piece-wise linear estimator (PLE) as a function of z (see [17]), our GGMM prior will lead to a piecewise non-linear shrinkage estimator.

[3] An alternative investigated in [32] is to replace the MAP denoising problem in (7.5) by the minimum mean square error (MMSE) estimator (*a.k.a.*, posterior mean). The MMSE estimator is defined as an integration problem and has a closed-form solution in case of GMM priors. In our experimental scenarios, this estimator did not lead to significant improvements compared to MAP and we thus did not pursue this idea.

[4] In [15], this is referred to as "conditional mixing weight".

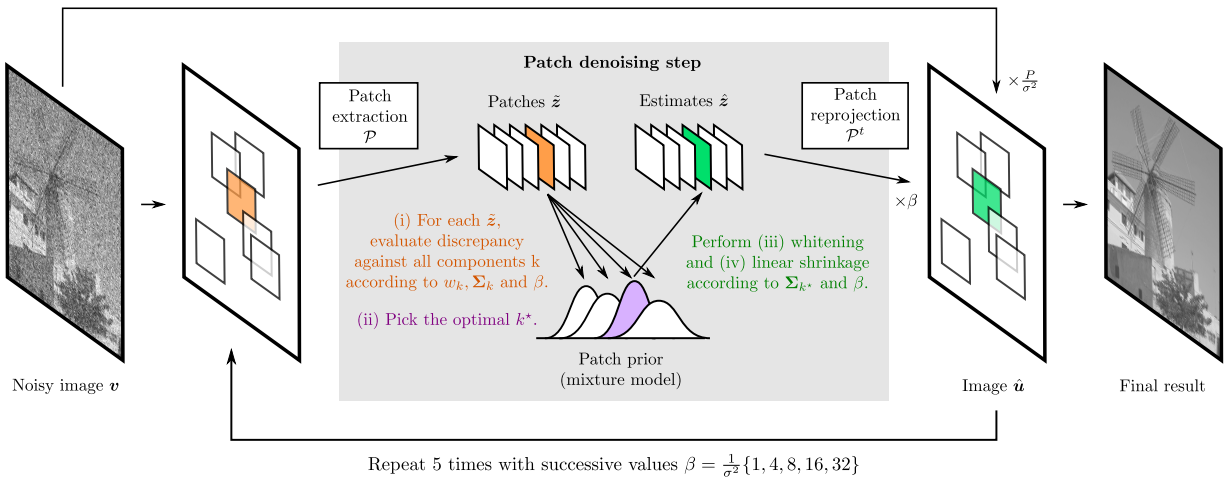


Figure 7.2: Illustration of EPLL framework for image denoising with a GMM prior. A large collection of (overlapping) patches are first extracted. For each patch, an optimal Gaussian component is picked based on a measure of discrepancy with the given patch. This Gaussian component is next used as a prior model to denoise the given patch by linear shrinkage in the corresponding eigenspace and depending on β. All estimated patches are finally aggregated together, weighted by β and combined with the original noisy image to produce a first estimate. The procedure is repeated 5 times with increasing values of β.

7.3 Generalized Gaussian Mixture Models

We aim to learn K orthogonal transforms such that each of them can map a subset (cluster) of clean patches into independent zero-mean coefficients. Instead of assuming the coefficient distributions to be bell shaped, we consider that both the scale and the shape of these distributions may vary from one coordinate to another (within the same transform). Our motivation to assume such a highly flexible model is based on the observation illustrated in Figure 7.1. Given one of such transform and its corresponding cluster of patches, we have displayed the histogram of the patch coefficients for six different coordinates. It can be clearly observed that the shape of the distribution varies depending on the coordinate. Some of them are peaky with heavy tails, and, therefore, would not be faithfully captured by a Gaussian distribution, as done in EPLL [15]. By contrast, some others have a bell shape, and so would not be captured properly by a peaky and heavy tailed distribution, as done for instance by sparse models [38, 39, 30, 45, 44, 47, 46]. This shows that one cannot simultaneously decorrelate and sparsify a cluster of clean patches for all

coordinates. Since some of the coordinates reveal sparsity while some others reveal Gaussianity, we propose to use a more flexible model that can capture such variations. We propose using a multi-variate zero-mean generalized Gaussian mixture model (GGMM)

$$p(z) = \sum_{k=1}^{K} w_k \mathcal{G}(z; 0_P, \Sigma_k, \nu_k) \tag{7.8}$$

where K is the number of components and $w_k > 0$ are weights such that $\sum_k w_k = 1$. The notation $\mathcal{G}(0_P, \Sigma, \nu)$ denotes the P-dimensional generalized Gaussian distribution (GGD) with zero-mean, covariance $\Sigma \in \mathbb{R}^{P \times P}$ (symmetric positive definite) and shape parameter $\nu \in \mathbb{R}^P$, whose expression is

$$\mathcal{G}(z; 0_P, \Sigma, \nu) = \frac{\mathcal{K}}{2|\Sigma_\nu|^{1/2}} \exp\left[-\|\Sigma_\nu^{-1/2} z\|_\nu^\nu\right] \quad \text{with} \quad \|x\|_\nu^\nu = \sum_{j=1}^{P} |x_j|^{\nu_j}, \tag{7.9}$$

$$\text{where} \quad \mathcal{K} = \prod_{j=1}^{P} \frac{\nu_j}{\Gamma(1/\nu_j)} \quad \text{and} \quad \Sigma_\nu^{1/2} = \Sigma^{1/2} \begin{pmatrix} \sqrt{\frac{\Gamma(1/\nu_1)}{\Gamma(3/\nu_1)}} & & \\ & \ddots & \\ & & \sqrt{\frac{\Gamma(1/\nu_P)}{\Gamma(3/\nu_P)}} \end{pmatrix}. \tag{7.10}$$

Denoting the eigen decomposition of matrix Σ by $\Sigma = U\Lambda U^t$ such that $U \in \mathbb{R}^{P \times P}$ is unitary and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_P)^2$ is diagonal with positive diagonal elements λ_j^2, $\Sigma^{1/2}$ in the above expression is defined as $\Sigma^{1/2} = U\Lambda^{1/2}$ and $\Sigma^{-1/2} = \Lambda^{-1/2}U^t$ is its inverse.

When ν is a constant vector with all entries equal to $\nu_j = 2$, $\mathcal{G}(0_P, \Sigma, \nu)$ is the multi-variate Gaussian distribution $\mathcal{N}(0_P, \Sigma)$ (as used in EPLL [15]). When all $\nu_j = 1$, it is the multi-variate Laplacian distribution and the subsequent GGMM is a Laplacian Mixture Model (LMM). When all $\nu_j < 1$, it is the multi-variate hyper-Laplacian distribution and the subsequent GGMM is a hyper-Laplacian Mixture Model (HLMM). Choosing $K = 1$ with a constant vector ν corresponds to ℓ_ν regularization [44, 47]. But as motivated earlier, unlike classical multivariate GGD models [123, 140, 35], we allow for the entries of ν to vary from one coordinate j to another. To the best

of our knowledge, the proposed work is the first one to consider this fully flexible model.

Proposition 7.3.1. *The multi-variate zero-mean GGD can be decomposed as*

$$\mathcal{G}(z;0_P,\Sigma,\nu)=\prod_{j=1}^{P}\mathcal{G}((U^t z)_j;0,\lambda_j,\nu_j) \tag{7.11}$$

where, for $x=U^t z$, the distribution of each of its components is given as

$$\mathcal{G}(x;0,\lambda,\nu)=\frac{\kappa}{2\lambda_\nu}\exp\left[-\left(\frac{|x|}{\lambda_\nu}\right)^\nu\right]$$
$$\textit{where}\quad \kappa=\frac{\nu}{\Gamma(1/\nu)}\quad \textit{and}\quad \lambda_\nu=\lambda\sqrt{\frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}},$$

where $x\mapsto\mathcal{G}(x;0,\lambda,\nu)$ is a real, even, unimodal, bounded and continuous probability density function. It is also differentiable everywhere except for $x=0$ when $\nu\leqslant 1$.

The proof follows directly by injecting the eigen decomposition of Σ in (7.9) and basic properties of $x\mapsto|x|^\nu$. Proposition 7.3.1 shows that, for each of the K clusters, eq. (7.9), indeed, models a prior that is separable in a coordinate system obtained by applying the whitening transform U^t. Not only is the prior separable for each coordinate j, but the shape (ν_j) and scale (λ_j) of the distribution may vary.

Before detailing the usage of GGMM priors in EPLL framework, we digress briefly to explain the procedure we used for training such a mixture of generalized Gaussian distributions where different scale and shape parameters are learned for each feature dimension.

7.3.1 Learning GGMMs

Parameter estimation is carried out using a modified version of the Expectation-Maximization (EM) algorithm [48]. EM is an iterative algorithm that performs at each iteration two steps, namely

Expectation step (E-Step) and Maximization stop (M-Step), and is known to monotonically increase the model likelihood and converge to a local optimum. For applying EM to learn a GGMM, we leverage standard strategies used for parameter estimation for GGD and/or GGMM that are reported in previous works [38, 141, 133, 132, 128, 136, 123, 142, 140, 143]. Our M-Step update for the shape parameter ν is inspired from Mallat's strategy [38] using statistics of the first absolute and second moments of GGDs. Since this strategy uses the method of moments for ν instead of maximum likelihood estimation, we refer to this algorithm as modified EM and the M-Step as Moment step. We also noticed that shape parameters $\nu < .3$ lead to numerical issues and $\nu > 2$ leads to local minima with several degenerate components. For this reason, at each step, we impose the constraint that the learned shape parameters satisfy $\nu \in [.3, 2]$. This observation is consistent with earlier works that have attempted to learn GGMM shape parameters from data [144]. Given n training clean patches of size P and an initialization for the K parameters $w_k > 0$, $\Sigma_k \in \mathbb{R}^{P\times P}$ and $\nu_k \in \mathbb{R}^P$, for $k = 1, \ldots, K$, our modified EM algorithm iteratively alternates between the following two steps:

- **Expectation step (E-Step)**
  - For all components $k = 1, \ldots, K$ and training samples $i = 1, \ldots, n$, compute:

$$\xi_{k,i} \leftarrow \frac{w_k \mathcal{G}(z_i; 0_P, \Sigma_k, \nu_k)}{\sum_{l=1}^{K} w_l \mathcal{G}(z_i; 0_P, \Sigma_l, \nu_l)} .$$

- **Moment step (M-Step)**
  - For all components $k = 1, \ldots, K$, update:

$$w_k \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i}}{\sum_{l=1}^{K} \sum_{i=1}^{n} \xi_{l,i}} \quad \text{and} \quad \Sigma_k \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i} z_i z_i^t}{\sum_{i=1}^{n} \xi_{k,i}} .$$

– Perform eigen decomposition of Σ_k:

$$\Sigma_k = U_k \Lambda_k U_k^t \quad \text{where} \quad \Lambda_k = \text{diag}(\lambda_{k,1}, \lambda_{k,2}, \ldots, \lambda_{k,P})^2 \ .$$

– For all components $k = 1, \ldots, K$ and dimensions $j = 1, \ldots, P$, compute:

$$\chi_{k,j} \leftarrow \frac{\sum_{i=1}^{n} \xi_{k,i} |(U_k^t z_i)_j|}{\sum_{i=1}^{n} \xi_{k,i}} \quad \text{and} \quad (\nu_k)_j \leftarrow \Pi_{[.3,2]} \left[F^{-1} \left(\frac{\chi_{k,j}^2}{\lambda_{k,j}^2} \right) \right] \ .$$

where $\Pi_{[a,b]}[x] = \min(\max(x,a),b)$ and $F(x) = \frac{\Gamma(2/x)^2}{\Gamma(3/x)\Gamma(1/x)}$ is a monotonic invertible function that was introduced in [38] (we used a lookup table to perform its inversion as done in [141, 133]). Note that $\chi_{k,j}^2$ and $\lambda_{k,j}^2$ corresponds to the first absolute and second moments for component k and dimension j, respectively.

For consistency purposes, we keep the training data and the number of mixture components in the models the same as that used in the original EPLL algorithm [15]. Specifically, we train our models on $n = 2$ million clean patches randomly extracted from Berkeley Segmentation Dataset (BSDS) [109]. We learn $K = 200$ zero-mean generalized Gaussian mixture components from patches of size $P = 8 \times 8$. We opted for a warm-start training by initializing our GGMM model with the GMM model from [15] and with initial values of shape parameters as 2. We run our modified EM algorithm for 100 iterations. As observed in Figure 7.1, the obtained GGMM models the underlying distributions of a cluster of clean patches much better than a GMM. In addition, we will see in Section 7.6 that our GGMM estimation did not lead to overfitting as it is also a better fit than a GMM for unseen clean patches.

7.3.2 Patch denoising with GGMM priors

We now explain why solving step (7.5) in EPLL is non-trivial when using a GGMM patch prior. In this case, for a noisy patch $\tilde{z}$ with variance σ^2, equation (7.5) becomes

$$\hat{z} \leftarrow \underset{z\in\mathbb{R}^P}{\operatorname{argmin}} \frac{1}{2\sigma^2}\|\tilde{z}-z\|^2 - \log\left[\sum_{k=1}^{K} w_k \mathcal{G}(z;0_P,\Sigma_k,\nu_k)\right] . \tag{7.12}$$

As for GMMs, due to the multi-modality of the GGMM prior, this optimization problem is highly non-convex. To circumvent this issue, we follow the strategy used by EPLL [15] in the specific case of Gaussian mixture model prior. The idea is to restrict the sum involved in the logarithm in eq. (7.12) to only one component $k^\star$.

If we consider the best $k^\star$ to be given (the strategy to select the best $k^\star$ will be discussed next), then eq. (7.12) is approximated by the following simpler problem

$$\hat{z} \quad \leftarrow \quad \underset{z\in\mathbb{R}^P}{\operatorname{argmin}} \quad \left\{ \frac{\|\tilde{z}-z\|^2}{2\sigma^2} - \log \mathcal{G}(z;0_P,\Sigma_{k^\star},\nu_{k^\star}) = \frac{\|\tilde{z}-z\|^2}{2\sigma^2} + \|\Sigma_{\nu_{k^\star}}^{-1/2} z\|_\nu^\nu \right\} \quad . \tag{7.13}$$

The main advantage of this simplified version is that, by virtue of Proposition 7.3.1, the underlying optimization becomes tractable and can be separated into P one-dimensional optimization problems, as:

$$\hat{z} = U_{k^\star}\hat{x} \quad \text{where} \quad \hat{x}_j = s(\tilde{x}_j;\sigma,\lambda_{k^\star,j},\nu_{k^\star,j}) \quad \text{with} \quad \tilde{x} = U_{k^\star}^t \tilde{z} \tag{7.14}$$

$$\text{and} \quad s(x;\sigma,\lambda,\nu) = s_{\sigma,\lambda}^\nu(x) \in \underset{t\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{2\sigma^2}(x-t)^2 + \frac{|t|^\nu}{\lambda_\nu^\nu} \quad \text{where} \quad \lambda_\nu = \lambda\sqrt{\frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}} , \tag{7.15}$$

where for all k, $\nu_{k,j} = (\nu_k)_j$ and $\lambda_{k,j} = (\lambda_k)_j$. While the problem is not necessarily convex, its solution $s_{\sigma,\lambda}^\nu$ is always uniquely defined almost everywhere (see, Section 7.5). We call this almost everywhere real function $s_{\sigma,\lambda}^\nu : \mathbb{R} \to \mathbb{R}$ shrinkage function. When $\nu = 2$, it is a linear function

that is often referred to as Wiener shrinkage. When $\nu \neq 2$, as we will discuss in Section 7.5, it is a non-linear shrinkage function that can be computed in closed form for some cases or with some approximations.

Now, we address the question of finding a strategy for choosing a relevant component $k^\star$ to replace the mixture distribution inside the logarithm. The optimal component $k^\star$ can be obtained by maximizing the posterior as

$$k^\star \in \underset{1\leqslant k\leqslant K}{\operatorname{argmax}}\ p(k \mid \tilde{z}) = \underset{1\leqslant k\leqslant K}{\operatorname{argmax}}\ w_k p(\tilde{z} \mid k) = \underset{1\leqslant k\leqslant K}{\operatorname{argmin}}\ -\log w_k - \log p(\tilde{z} \mid k) \tag{7.16}$$

where the weights of the GGMM corresponds to the prior probability $w_k = p(k)$. We next use the fact that the patch $\tilde{z}$ (conditioned on k) can be expressed as $\tilde{z} = z + n$ where z and n are two independent random variables from distributions $\mathcal{G}(0_P, \Sigma_k, \nu_k)$ and $\mathcal{N}(0_P, \sigma^2 \mathrm{Id}_P)$ respectively. It follows that the distribution of $\tilde{z}$ is the convolution of these latter two, and then

$$-\log p(\tilde{z} \mid k) = -\log \int_{\mathbb{R}^P} \mathcal{G}(\tilde{z}-z; 0_P, \Sigma_k, \nu_k) \cdot \mathcal{N}(z; 0_P, \sigma^2 \mathrm{Id}_P)\, \mathrm{d}z\ . \tag{7.17}$$

We next use Proposition 7.3.1 to separate this integration problem into P one-dimensional integration problems. We obtain

$$-\log p(\tilde{z} \mid k) = \sum_{j=1}^{P} f((U_k^t \tilde{z})_j; \sigma, \lambda_{k,j}, \nu_{k,j}) \tag{7.18}$$

where, for $\tilde{x} = U_k^t \tilde{z}$, the integration problem of each of its components reads as

$$f(x; \sigma, \lambda, \nu) = f^{\nu}_{\sigma,\lambda}(x) = -\log \int_{\mathbb{R}} \mathcal{G}(x-t; 0, \lambda, \nu) \cdot \mathcal{N}(t; 0, \sigma^2)\, \mathrm{d}t\ . \tag{7.19}$$

We call the real function $f^{\nu}_{\sigma,\lambda} : \mathbb{R} \to \mathbb{R}$ the *discrepancy function* which measures the goodness of fit of a GGD to the noisy value x. When $\nu = 2$, this function is quadratic with x. For $\nu \neq 2$, as we

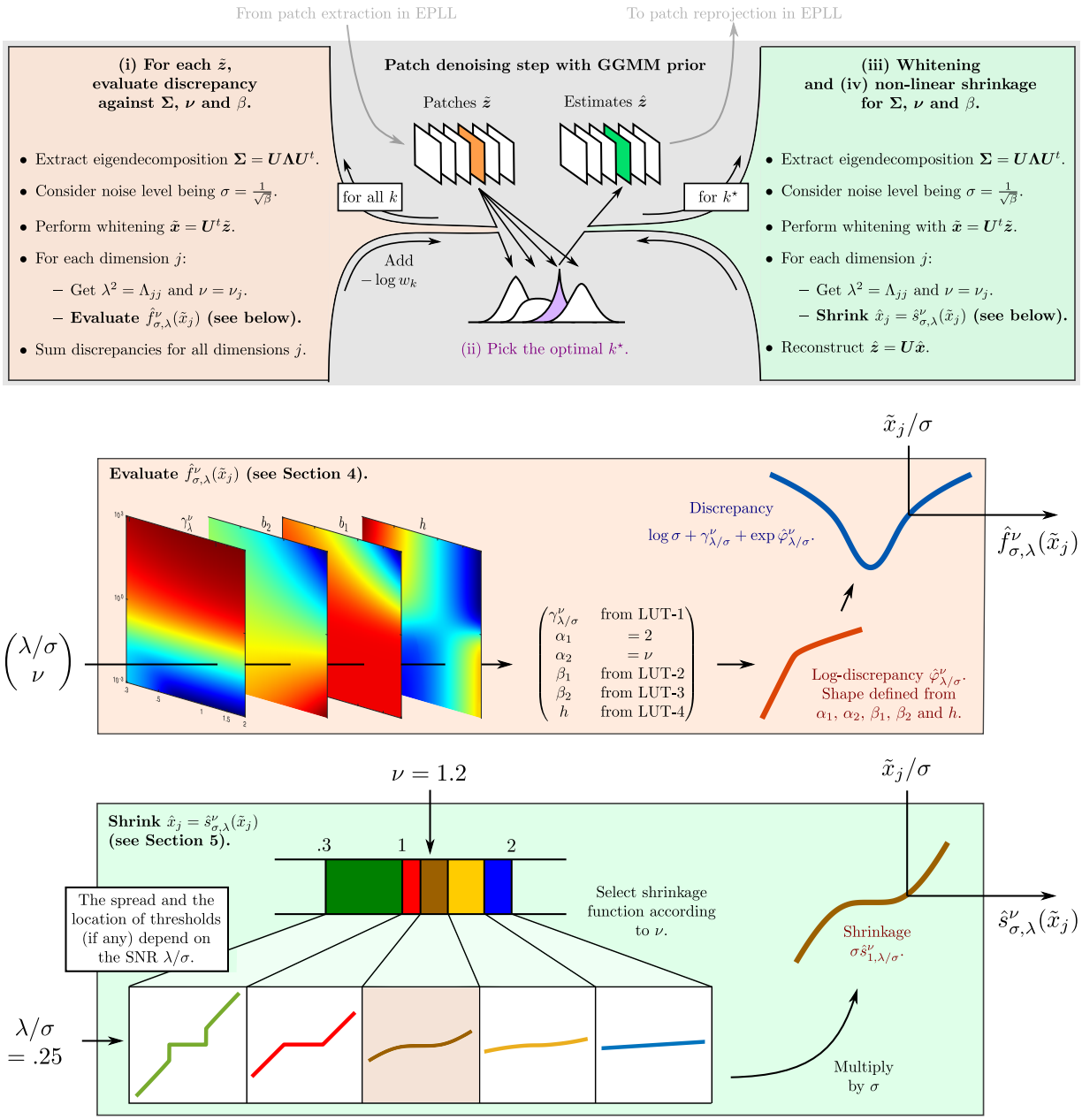


Figure 7.3: Illustration of our extension of EPLL to GGMM priors. The general procedure, illustrated in the top row, is similar to the original EPLL scheme described in Section 7.2.2 but relies on generalized Gaussian distributions instead of Gaussian distributions. The shape of the discrepancy function, illustrated in the second row, depends on the given scale and shape parameters (λ and ν) of the GGD components. In Section 7.4, we will see that it can be approximated based on six parameters, four of them retrieved from lookup tables (LUTs). Finally, the shrinkage function, illustrated in the bottom row, can be non-linear and depends on the selected GGD component. In Section 7.5, we will see that it can be approximated by one of five predefined parametric functions depending on the range in which the scale parameter ν lies. The values $\nu = 1.2$ and $\lambda/\sigma = .25$, shown in the bottom row, were chosen for the sake of illustration.

will discuss in Section 7.4, it is a non-quadratic function, that can be efficiently approximated based on an in-depth analysis of its asymptotic behavior.

Figure 7.3 illustrates the details of the patch denoising step under the GGMM-EPLL framework. It shows that the method relies on fast approximations $\hat{f}^{\nu}_{\sigma,\lambda}$ and $\hat{s}^{\nu}_{\sigma,\lambda}$ of the discrepancy and shrinkage functions, respectively.

The next two sections are dedicated to the analysis and approximations of the discrepancy function $f^{\nu}_{\sigma,\lambda}$ and the shrinkage function $s^{\nu}_{\sigma,\lambda}$, respectively.

7.4 Discrepancy function: analysis and approximations

From its definition given in eq. (7.19), the discrepancy function reads for $\nu > 0$, $\sigma > 0$ and $\lambda > 0$, as

$$f^{\nu}_{\sigma,\lambda}(x) = -\log \frac{1}{\sqrt{2\pi}\sigma} \frac{\nu}{2\lambda_{\nu}\Gamma(1/\nu)} - \log \int_{-\infty}^{\infty} \exp\left(-\frac{(x-t)^2}{2\sigma^2}\right) \exp\left[-\left(\frac{|t|}{\lambda_{\nu}}\right)^{\nu}\right] \mathrm{d}t \ . \quad (7.20)$$

It corresponds to the negative logarithm of the distribution of the sum of a zero-mean generalized Gaussian and a zero-mean Gaussian random variables. When $\nu = 2$, the generalized Gaussian random variable becomes Gaussian, and the resulting distribution is also Gaussian with zero-mean and variance $\sigma^2 + \lambda^2$, and then

$$f^{2}_{\sigma,\lambda}(x) = \frac{1}{2}\left[\log 2\pi + \log(\sigma^2+\lambda^2) + \frac{x^2}{\sigma^2+\lambda^2}\right] \ . \quad (7.21)$$

Remark 7.4.1. *For $\nu = 2$, a direct consequence of* (7.21) *is that $-\log p(\tilde{z} \mid k)$ is as an affine function of the Mahanalobis distance between $\tilde{z}$ and 0_P for the covariance matrix $\Sigma_k + \sigma^2 \mathrm{Id}_P$:*

$$-\log p(\tilde{z} \mid k) = \frac{1}{2}\left[P \log 2\pi + \log|\Sigma_k + \sigma^2 \mathrm{Id}_P| + \tilde{z}^t (\Sigma_k + \sigma^2 \mathrm{Id}_P)^{-1} \tilde{z}\right] \ . \quad (7.22)$$

When $\nu = 1$, the generalized Gaussian random variable becomes Laplacian, and the distribution resulting from the convolution also has a closed form which leads to the following discrepancy function

$$f^1_{\sigma,\lambda}(x) = \log(2\sqrt{2}\lambda) - \frac{\sigma^2}{\lambda^2} - \log\left[e^{\frac{\sqrt{2}x}{\lambda}}\,\mathrm{erfc}\left(\frac{x}{\sqrt{2}\sigma} + \frac{\sigma}{\lambda}\right) + e^{-\frac{\sqrt{2}x}{\lambda}}\,\mathrm{erfc}\left(-\frac{x}{\sqrt{2}\sigma} + \frac{\sigma}{\lambda}\right)\right], \tag{7.23}$$

refer to Appendix B.1 for derivation (note that this expression is given in [139]).

To the best of our knowledge, there are no simple expressions for other values of ν. One solution proposed by [138] is to express this in terms of the bi-variate Fox-H function [145]. This, rather cumbersome expression, is computationally demanding. In practice, this special function requires numerical integration techniques over complex lines [146], and is thus difficult to numerically evaluate it efficiently. Since, in our application, we need to evaluate this function a large number of times, we cannot utilize this solution.

In [138], the authors have also proposed to approximate this non-trivial distribution by another GGD. For fixed values of σ, λ and ν, they proposed three different numerical techniques to estimate its parameters λ' and ν' that best approximate either the kurtosis, the tail or the cumulative distribution function. Based on their approach, the discrepancy function $f^{\nu}_{\sigma,\lambda}(x)$ would thus be a power function of the form $|x|^{\nu'}$.

In this paper, we show that $f^{\nu}_{\sigma,\lambda}$ does, indeed, asymptotically behave as a power function for small and large values of x, but the exponent can be quite different for these two asymptotics. We believe that these different behaviors are important to be preserved in our application context. For this reason, $f^{\nu}_{\sigma,\lambda}$ cannot be modeled as a power function through a GGD distribution. Instead, we provide an alternative solution that is able to capture the correct behavior for both of these asymptotics, and that also permits fast computation.

7.4.1 Theoretical analysis

In this section, we perform a thorough theoretical analysis of the discrepancy function, in order to approximate it accurately. Let us first introduce some basic properties regarding the discrepancy function.

Proposition 7.4.2. *Let* $\nu > 0$, $\sigma > 0$, $\lambda > 0$ *and* $f^{\nu}_{\sigma,\lambda}$ *as defined in eq.* (7.19). *The following relations hold true*

$$f^{\nu}_{\sigma,\lambda}(x) = \log\sigma + f^{\nu}_{1,\lambda/\sigma}(x/\sigma)\ , \qquad \text{(reduction)}$$

$$f^{\nu}_{\sigma,\lambda}(x) = f^{\nu}_{\sigma,\lambda}(-x)\ , \qquad \text{(even)}$$

$$|x| \geqslant |y| \Leftrightarrow f^{\nu}_{\sigma,\lambda}(|x|) \geqslant f^{\nu}_{\sigma,\lambda}(|y|)\ , \qquad \text{(unimodality)}$$

$$\min_{x\in\mathbb{R}} f^{\nu}_{\sigma,\lambda}(x) = f^{\nu}_{\sigma,\lambda}(0) > -\infty\ . \qquad \text{(lower bound at 0)}$$

The proofs can be found in Appendix B.2. Based on Proposition 7.4.2, we can now express the discrepancy function $f^{\nu}_{\sigma,\lambda}(x) : \mathbb{R} \to \mathbb{R}$ in terms of a constant γ^{ν}_{λ} and another function $\varphi^{\nu}_{\lambda} : \mathbb{R}^{*}_{+} \to \mathbb{R}$, both of which can be parameterized by only two parameters $\lambda > 0$ and $\nu > 0$, as

$$f^{\nu}_{\sigma,\lambda}(x) = \log\sigma + \gamma^{\nu}_{\lambda/\sigma} + \begin{cases} e^{\varphi^{\nu}_{\lambda/\sigma}(|x/\sigma|)} & \text{if} \quad x \neq 0\ , \\ 0 & \text{otherwise}\ , \end{cases} \tag{7.24}$$

$$\text{where} \quad \varphi^{\nu}_{\lambda}(x) = \log\left[f^{\nu}_{1,\lambda}(x) - \gamma^{\nu}_{\lambda}\right] \quad \text{and} \quad \gamma^{\nu}_{\lambda} = f^{\nu}_{1,\lambda}(0)\ . \tag{7.25}$$

We call φ^{ν}_{λ} the log-discrepancy function.

At this point, let us consider an instructive toy example for the case when $\nu = 2$. In this case, from eq. (7.21), we can deduce that the log-discrepancy function is a log-linear function

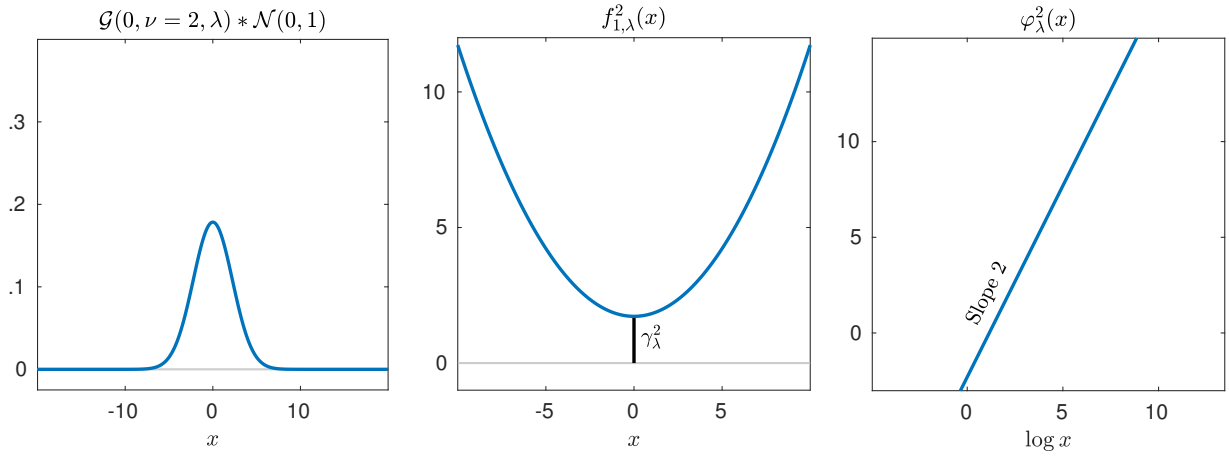


Figure 7.4: Convolution of two Gaussian distributions. From left to right: the convolution of a Gaussian distribution with standard deviation $\lambda = 2$ with a Gaussian distribution with standard deviation $\sigma = 1$, the corresponding discrepancy function and log-discrepancy function.

(*i.e.*, a linear function of $\log x$)

$$\varphi^2_\lambda(x) = \alpha \log x + \beta \,, \tag{7.26}$$

$$\text{and} \quad \gamma^2_\lambda = \tfrac{1}{2}\left[\log 2\pi + \log(1+\lambda^2)\right] \,, \tag{7.27}$$

$$\text{where} \quad \alpha = 2 \quad \text{and} \quad \beta = -\log 2 - \log(1+\lambda^2) \,. \tag{7.28}$$

Here, the slope $\alpha = 2$ reveals the quadratic behavior of the discrepancy function. Figure 7.4 gives an illustration of the resulting convolution (a Gaussian distribution), the discrepancy function (a quadratic function) and the log-discrepancy (a linear function with slope 2). Note that quadratic metrics are well-known to be non-robust to outliers, which is in complete agreement with the fact that Gaussian priors have thin tails.

Another example is the case of $\nu = 1$. From eq. (7.23), the log-discrepancy is given by

$$\varphi^1_\lambda(x) = \log\left[\log\left[2\,\mathrm{erfc}\left(\tfrac{1}{\lambda}\right)\right] - \log\left[e^{\frac{\sqrt{2}x}{\lambda}}\,\mathrm{erfc}\left(\tfrac{x}{\sqrt{2}}+\tfrac{1}{\lambda}\right) + e^{-\frac{\sqrt{2}x}{\lambda}}\,\mathrm{erfc}\left(-\tfrac{x}{\sqrt{2}}+\tfrac{1}{\lambda}\right)\right]\right] \,, \tag{7.29}$$

$$\text{and} \quad \gamma^1_\lambda = \frac{1}{2}\log 2 + \log\lambda - \frac{1}{\lambda^2} - \log\left[\mathrm{erfc}\left(\tfrac{1}{\lambda}\right)\right] \,. \tag{7.30}$$

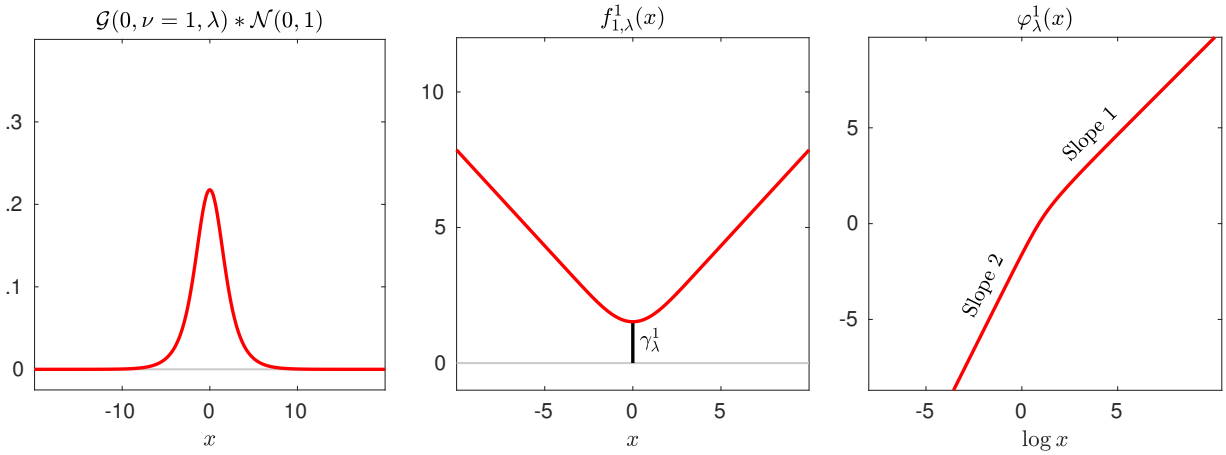


Figure 7.5: Convolution of Laplacian with Gaussian. From left to right: the convolution of a Laplacian distribution with standard deviation $\lambda = 2$ with a Gaussian distribution with standard deviation $\sigma = 1$, the corresponding discrepancy function and log-discrepancy function.

Unlike for $\nu = 2$, this function is not log-linear and thus $f^1_{\sigma,\lambda}$ is not a power function. Nevertheless, as shown by the next two theorems, it is also asymptotically log-linear for small and large values of x.

Theorem 7.4.3 *The function φ^1_λ is asymptotically log-linear in the vicinity of* 0

$$\varphi^1_\lambda(x) \underset{0}{\sim} \alpha_1 \log x + \beta_1 \,, \tag{7.31}$$

$$\textit{where} \quad \alpha_1 = 2 \quad \textit{and} \quad \beta_1 = -\log\lambda + \log\left[\frac{1}{\sqrt{\pi}}\frac{\exp\left(-\frac{1}{\lambda^2}\right)}{\operatorname{erfc}\left(\frac{1}{\lambda}\right)} - \frac{1}{\lambda}\right] . \tag{7.32}$$

The proof can be found in Appendix B.3.

Theorem 7.4.4 *The function φ^1_λ is asymptotically log-linear in the vicinity of* $+\infty$

$$\varphi^1_\lambda(x) \underset{\infty}{\sim} \alpha_2 \log x + \beta_2 \,, \tag{7.33}$$

$$\textit{where} \quad \alpha_2 = 1 \quad \textit{and} \quad \beta_2 = \frac{1}{2}\log 2 - \log\lambda \,. \tag{7.34}$$

The proof can be found in Appendix B.4.

Theorem 7.4.3 and Theorem 7.4.4 show that φ_λ^1 has two different asymptotics that can be approximated by a log-linear function. Interestingly, the exponent $\alpha_1 = 2$ in the vicinity of 0 shows that the Gaussian distribution involved in the convolution prevails over the Laplacian distribution and thus, the behavior of $f_{\sigma,\lambda}^1$ is quadratic. Similarly, the exponent $\alpha_2 = 1$ in the vicinity of $+\infty$ shows that the Laplacian distribution involved in the convolution prevails over the Gaussian distribution and the behavior of $f_{\sigma,\lambda}^1$ is then linear. These results are supported by Figure 7.5 which illustrates the resulting convolution, the discrepancy function (eq. (7.23)) and the log-discrepancy function (eq. (7.29)). Furthermore, the discrepancy function $f_{\sigma,\lambda}^1$ shares a similar behavior with the well-known Huber loss function [147] (also called smoothed ℓ_1), known to be more robust to outliers. This is again in complete agreement with the fact that Laplacian priors have heavier tails.

In the case $\frac{2}{3} < \nu < 2$, even though φ_λ^ν has no simple closed form expression, the similar conclusions can be made as a result of the next two theorems.

Theorem 7.4.5 *Let $\nu > 0$. The function φ_λ^ν is asymptotically log-linear in the vicinity of 0*

$$\varphi_\lambda^\nu(x) \underset{0}{\sim} \alpha_1 \log x + \beta_1 \ ,$$

$$\textit{where} \quad \alpha_1 = 2 \quad \textit{and} \quad \beta_1 = -\log 2 + \log\left(1 - \frac{\displaystyle\int_{-\infty}^{\infty} t^2 e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t}{\displaystyle\int_{-\infty}^{\infty} e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t}\right) \ .$$

The proof can be found in Appendix B.5.

Theorem 7.4.6 *Let $\frac{2}{3} < \nu < 2$, then φ_λ^ν is asymptotically log-linear in the vicinity of $+\infty$*

$$\varphi_\lambda^\nu(x) \underset{\infty}{\sim} \alpha_2 \log x + \beta_2 \ ,$$

$$\textit{where} \quad \alpha_2 = \nu \quad \textit{and} \quad \beta_2 = -\nu \log \lambda - \frac{\nu}{2} \log \frac{\Gamma(1/\nu)}{\Gamma(3/\nu)} \ .$$

The proof relies on a result of Berman (1992) [148] and is detailed in Appendix B.6.

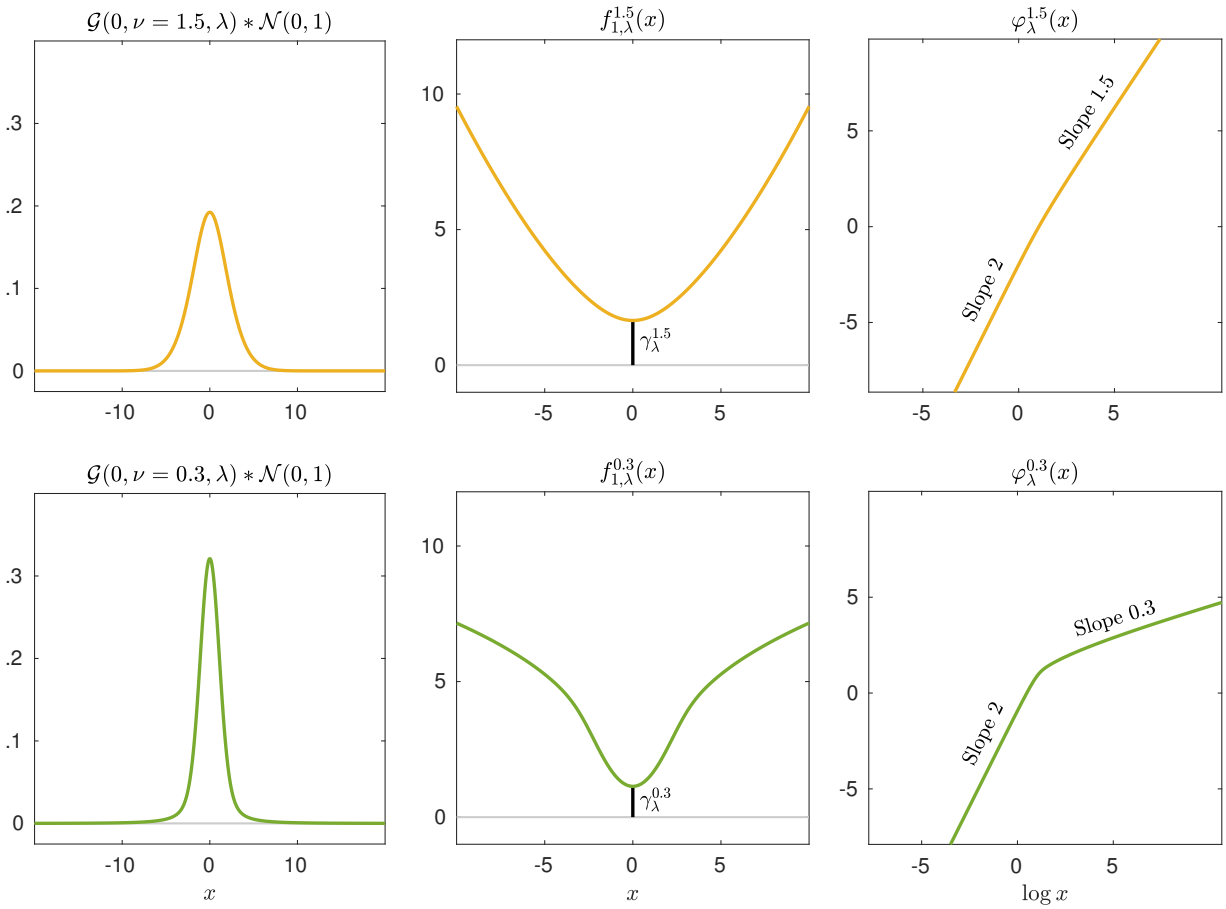


Figure 7.6: Convolution of generalized Gaussian with Gaussian. From left to right: the convolution of a generalized Gaussian distribution with standard deviation $\lambda = 2$ with a Gaussian distribution with standard deviation $\sigma = 1$, the corresponding discrepancy function and log-discrepancy function. From top to bottom: the GGD has a shape parameter $\nu = 1.5$ and .3, respectively.

Remark 7.4.7. *For $\nu > 2$, an asymptotic log-linear behavior with $\alpha_2 = 2$ and $\beta_2 = -\log 2$ can be obtained using exactly the same sketch of proof as the one of Theorem 7.4.6.*

Remark 7.4.8. *For $\nu = 2$, we have φ^2_λ is linear, $\beta_1 = -\log 2 - \log\left(1+\lambda^2\right)$ and $\beta_2 = -\log 2 - \log\lambda^2$, which shows that Theorem 7.4.6 cannot hold true for $\nu = 2$.*

Remark 7.4.9. *For $\nu = 1$, Theorem 7.4.3 and Theorem 7.4.4 coincide with Theorem 7.4.5 and Theorem 7.4.6.*

Remark 7.4.10. *For $0 < \nu \leqslant \frac{2}{3}$, though we did not succeed in proving it, our numerical simulations also revealed a log-linear asymptotic behavior for $x \to \infty$ in perfect agreement with the expression of α_2 and β_2 given in Theorem 7.4.6.*

Again, the exponent $\alpha_1 = 2$ in the vicinity of 0 shows that the Gaussian distribution involved in the convolution prevails over the generalized Gaussian distribution and the behavior of $f^{\nu}_{\sigma,\lambda}$ is then quadratic. Similarly, the exponent $\alpha_2 = \nu$ in the vicinity of $+\infty$ shows that the generalized Gaussian distribution involved in the convolution prevails over the Gaussian distribution and the behavior of $f^{\nu}_{\sigma,\lambda}$ is then a power function of the form x^{ν}. These results are supported by Figure 7.6 that illustrates the resulting convolution, the discrepancy function and the log-discrepancy function for $\nu = 1.5$ and $\nu = .3$. Moreover, the discrepancy function $f^{\nu}_{\sigma,\lambda}$ with $\nu \leqslant 1$ shares a similar behavior with well-known robust M-estimator loss functions [149]. In particular, the asymptotic case for $\nu \to 0$ resembles the Tukey's bisquare loss, known to be insensitive to outliers. This is again in complete agreement with GGD priors having larger tails as ν goes to 0.

Figure 7.7 shows the evolution of the log-discrepancy function for various values of ν in the context of three different signal-to-noise ratios λ/σ (SNR). One can observe that as the SNR decreases (resp., increases), the left (resp., right) asymptotic behavior starts dominating over the right (resp., left) asymptotes. In other words, for $\nu < 2$, the intersection of the two asymptotes

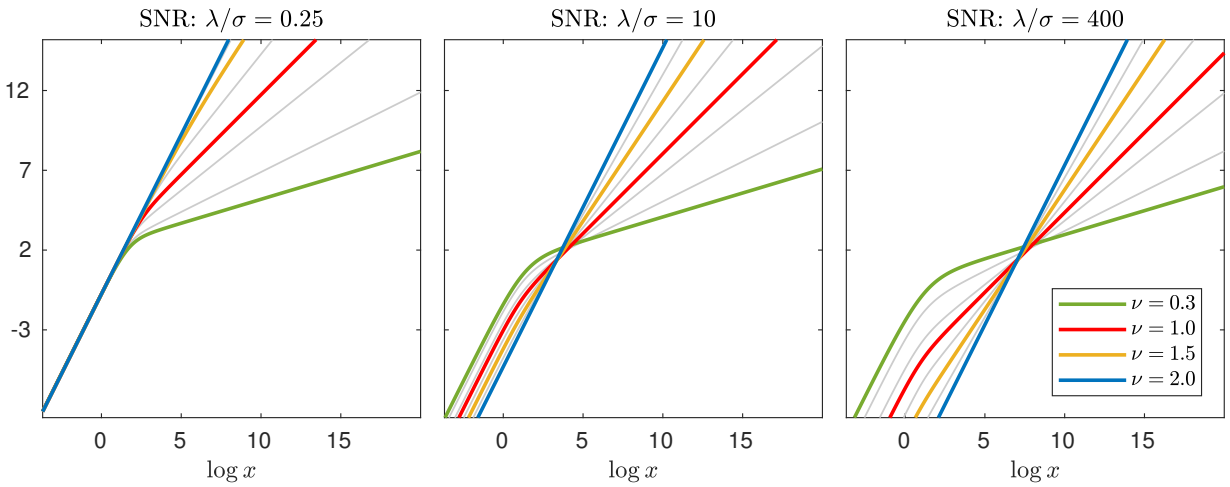


Figure 7.7: Illustrations of the log-discrepancy function for various $0.3 \leqslant \nu \leqslant 2$ and SNR λ/σ.

goes to $+\infty$ (resp., $-\infty$). Last but not least, for $0 < \nu \leqslant 2$, the log-discrepancy function φ_λ^ν is always concave and since $\alpha_2 \leqslant \alpha_1$ it is thus upper-bounded by its left and right asymptotes.

From Theorem 7.4.5, Theorem 7.4.6 and Remark 7.4.10, we can now build two asymptotic log-linear approximations for φ_λ^ν, with $0 < \nu \leqslant 2$, and subsequently an asymptotic power approximation for $f_{\sigma,\lambda}^\nu$ by using the relation (7.24). Next, we explain the approximation process of the in-between behavior, as well as its efficient evaluation.

7.4.2 Numerical approximation

We now describe the proposed approximation of the discrepancy function $f_{1,\lambda}^\nu$ through an approximation $\hat{\varphi}_\lambda^\nu$ of the log-discrepancy function as

$$\hat{f}_{1,\lambda}^\nu(x) = \gamma_\lambda^\nu + \exp \hat{\varphi}_\lambda^\nu(x) \quad \text{where} \quad \gamma_\lambda^\nu = f_{1,\lambda}^\nu(0)\ . \tag{7.35}$$

Based on our previous theoretical analysis, a solution preserving the asymptotic, increasing and concave behaviors of $\varphi^{\nu}_{1,\lambda}$ can be defined by making use of the following approximations

$$\hat{\varphi}^{\nu}_{\lambda}(x) = \alpha_1 \log|x| + \beta_1 - \texttt{rec}(\alpha_1 \log|x| + \beta_1 - \alpha_2 \log|x| - \beta_2) \ , \tag{7.36}$$

where `rec` is a so-called rectifier function that is positive, increasing, convex and satisfies

$$\lim_{x\to-\infty} \texttt{rec}(x) = 0 \quad \text{and} \quad \texttt{rec}(x) \underset{x\to\infty}{\sim} x \ . \tag{7.37}$$

In this paper, we consider the two following rectifying functions

$$\texttt{relu}(x) = \max(0,x) \quad \text{and} \quad \texttt{softplus}(x) = h\log\left[1+\exp\left(\frac{x}{h}\right)\right], \ h>0 \ , \tag{7.38}$$

as coined respectively in [150] and [151]. Using the function `relu` (Rectified linear unit) leads to an approximation $\hat{\varphi}^{\nu}_{\lambda}$ that is exactly equal to the asymptotes of φ^{ν}_{λ} with a singularity at their crossing point. In this paper, we will instead use the function `softplus` as it allows the approximation of φ^{ν}_{λ} to converge smoothly to the asymptotes without singularity. Its behavior is controlled by the parameter $h>0$. The smaller the value of h is, the faster the convergence speed to the asymptotes.

The parameter h should be chosen such that the approximation error between $\hat{\varphi}^{\nu}_{\lambda}(x)$ and $\varphi^{\nu}_{\lambda}(x)$ is as small as possible. This can be done numerically by first evaluating $\varphi^{\nu}_{\lambda}(x)$ with integration techniques for a large range of values x, and then selecting the parameter h by least square. Of course, the optimal value for h depends on the parameter λ and ν.

Figure 7.8 gives an illustration of our approximations of the log-discrepancy and the corresponding distribution obtained with `relu` and `softplus`. On this figure the underlying functions have been obtained by numerical integration for a large range of value of x. One can observe that using `softplus` provides a better approximation than `relu`.

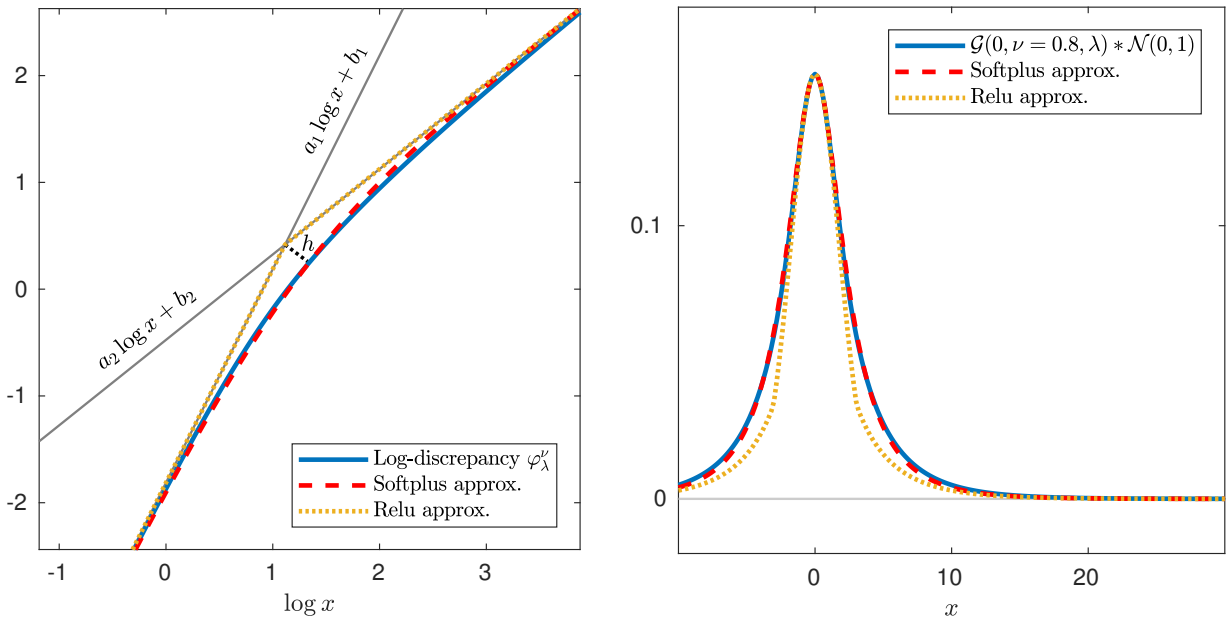


Figure 7.8: Illustrations of our approximations of φ_λ^ν and the corresponding underlying posterior distribution $\mathcal{N}(0, \nu, \lambda) * \mathcal{G}(0,1)$ (where $\nu = .8$ and $\lambda = 4$). The blue curves have been obtained by evaluating the convolution using numerical integration techniques for all x. The dashed curves are obtained using the proposed `relu`- and `softplus`-based approximations that have closed-form expressions.

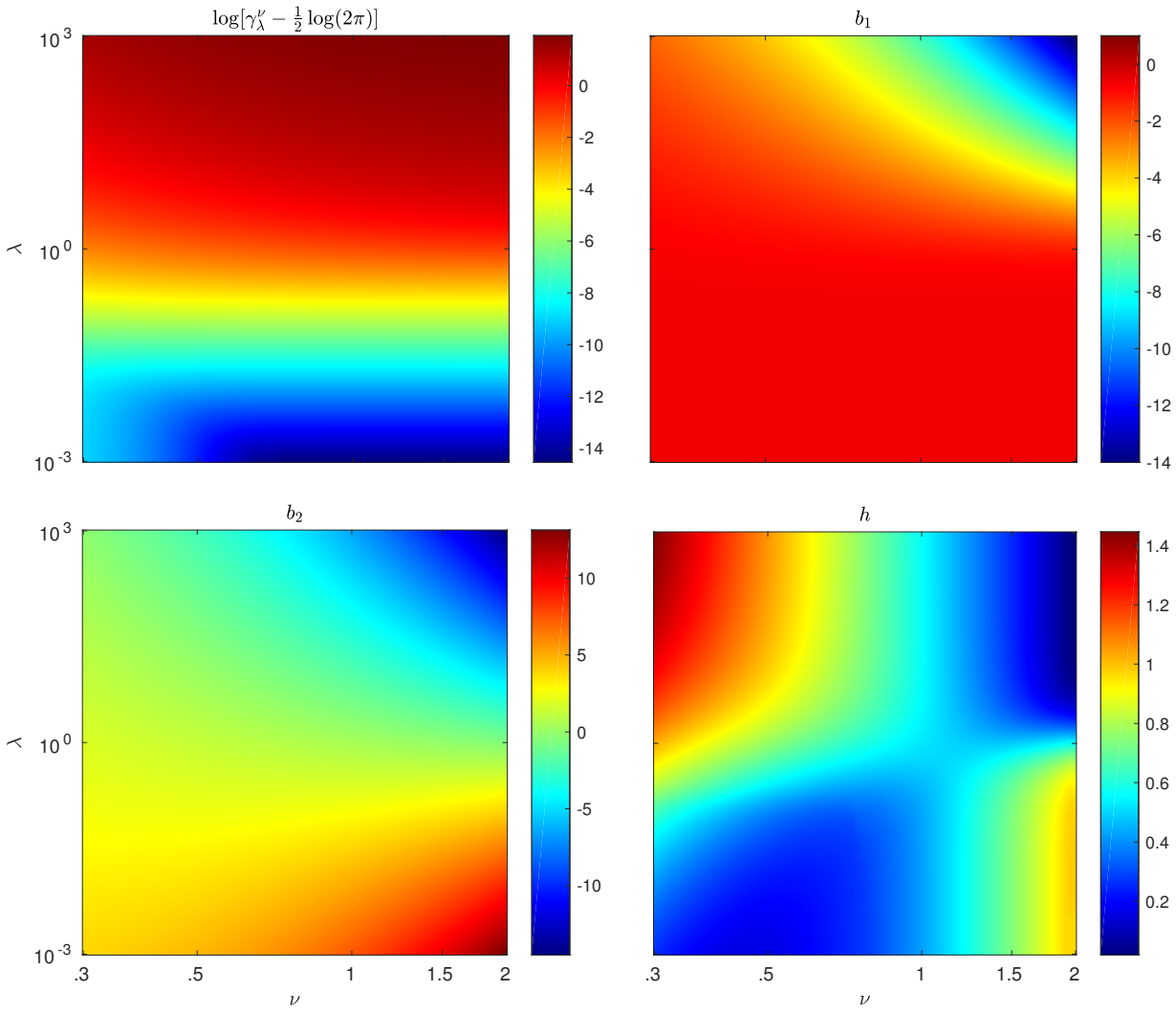


Figure 7.9: Lookup tables used to store the values of the parameters γ_λ^ν, β_1, β_2 and h for various $.3 \leqslant \nu \leqslant 2$ and $10^{-3} \leqslant \lambda \leqslant 10^3$. A regular grid of 100 values has been used for ν and a logarithmic grid of 100 values has been used for λ. This leads to a total of $10,000$ combinations for each of the four lookup tables.

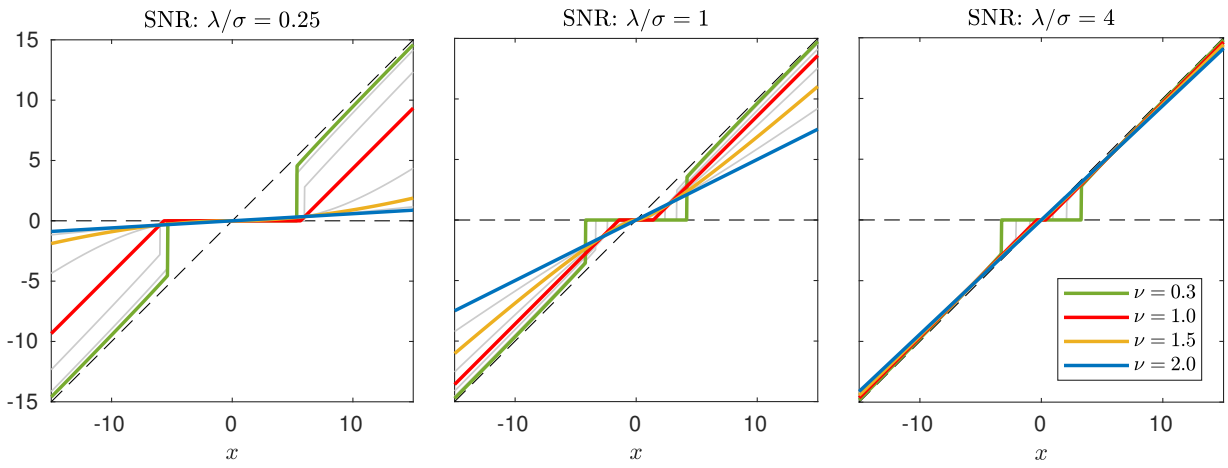


Figure 7.10: Illustrations of the shrinkage function for various $0.3 < \nu \leqslant 2$ and SNR λ/σ.

Our approximation for $\hat{f}^{\nu}_{1,\lambda}(x)$ is parameterized by six scalar values: γ^{λ}_{ν}, α_1, β_1, α_2, β_2 and h that depend only on the original parameters λ and ν. From our previous analysis, we have that $\alpha_1 = 2$ and $\alpha_2 = \nu$. The other parameters are non-linear functions of λ and ν. The parameters γ^{λ}_{ν}, β_1 and β_2 require either performing numerical integration or evaluating the special function Γ. As discussed, the parameter h requires numerical integration for various x and then optimization. For these reasons, these values cannot be computed during runtime. Instead, we pre-compute these four parameters offline for $10,000$ different combinations of λ and ν values in the intervals $[10^{-3}, 10^{3}]$ and $[0.3, 2]$, respectively (the choice for this range was motivated in Section 7.3.1). The resulting values are then stored in four corresponding lookup tables. During runtime, these parameters are retrieved online by bi-linear extrapolation and interpolation. The four lookup tables are given in Figure 7.9. We will see in Section 7.6 that using the approximation $\hat{f}^{\nu}_{1,\lambda}$ results in substantial acceleration without significant loss of performance as compared to computing $f^{\nu}_{1,\lambda}$ directly by numerical integration during runtime.

7.5 Shrinkage functions: analysis and approximations

Recall that from its definition given in eq. (7.15), the shrinkage function is defined for $\nu > 0$, $\sigma > 0$ and $\lambda > 0$, as

$$s^{\nu}_{\sigma,\lambda}(x) \in \operatorname*{argmin}_{t\in\mathbb{R}} \frac{(x-t)^2}{2\sigma^2} + \lambda^{-\nu}_{\nu}|t|^{\nu} \,. \tag{7.39}$$

7.5.1 Theoretical analysis

Except for some particular values of ν (see, Section 7.5.2), Problem (7.39) does not have explicit solutions. Nevertheless, as shown in [39], Problem (7.39) admits two (not necessarily distinct) solutions. One of them is implicitly characterized as

$$s^{\nu}_{\sigma,\lambda}(x) = \begin{cases} 0 & \text{if} \quad 0 < \nu \leqslant 1 \quad \text{and} \quad |x| \leqslant \tau^{\nu}_{\lambda} \,, \\ t^{\star} & \text{otherwise} \,, \end{cases} \tag{7.40}$$

$$\text{where} \quad t^{\star} = x - \operatorname{sign}(t^{\star})\nu\sigma^2\lambda^{-\nu}_{\nu}|t^{\star}|^{\nu-1} \,,$$

$$\text{and} \quad \tau^{\nu}_{\lambda} = \begin{cases} (2-\nu)(2-2\nu)^{-\frac{1-\nu}{2-\nu}}(\sigma^2\lambda^{-\nu}_{\nu})^{\frac{1}{2-\nu}} & \text{if} \quad \nu < 1 \,, \\ \sigma^2\lambda^{-1} & \text{otherwise } (\nu = 1) \,. \end{cases}$$

The other one is obtained by changing $|x| \leqslant \tau^{\nu}_{\lambda}$ to $|x| < \tau^{\nu}_{\lambda}$ in (7.40), and so they coincide for almost every $(x, \lambda, \sigma, \nu)$. As discussed in [39], for $\nu > 1$, $s^{\nu}_{\sigma,\lambda}(x)$ is differentiable, and for $\nu \leqslant 1$, the shrinkage exhibits a threshold τ^{ν}_{λ} that produces sparse solutions. Proposition 7.5.1 summarizes a few important properties.

Proposition 7.5.1. *Let $\nu > 0$, $\sigma > 0$, $\lambda > 0$ and $s^{\nu}_{\sigma,\lambda}$ as defined in eq.* (7.15). *The following*

relations hold true

$$s^{\nu}_{\sigma,\lambda}(x) = \sigma s^{\nu}_{1,\frac{\lambda}{\sigma}}\left(\frac{x}{\sigma}\right) , \qquad \text{(reduction)}$$

$$s^{\nu}_{\sigma,\lambda}(x) = -s^{\nu}_{\sigma,\lambda}(-x) , \qquad \text{(odd)}$$

$$s^{\nu}_{\sigma,\lambda}(x) \in \begin{cases} [0,x] & \textit{if} \quad x \geqslant 0 \\ [x,0] & \textit{otherwise} \end{cases} , \qquad \text{(shrinkage)}$$

$$x_1 \geqslant x_2 \Leftrightarrow s^{\nu}_{\sigma,\lambda}(x_1) \geqslant s^{\nu}_{\sigma,\lambda}(x_2) , \qquad \text{(increasing with } x\text{)}$$

$$\lambda_1 \geqslant \lambda_2 \Leftrightarrow s^{\nu}_{\sigma,\lambda_1}(x) \geqslant s^{\nu}_{\sigma,\lambda_2}(x) , \qquad \text{(increasing with } \lambda\text{)}$$

$$\lim_{\frac{\lambda}{\sigma}\to 0} s^{\nu}_{\sigma,\lambda}(x) = 0 , \qquad \text{(kill low SNR)}$$

$$\lim_{\frac{\lambda}{\sigma}\to +\infty} s^{\nu}_{\sigma,\lambda}(x) = x . \qquad \text{(keep high SNR)}$$

The proofs can be found in Appendix B.7. These properties show that $s^{\nu}_{\sigma,\lambda}$ is indeed a shrinkage function (non-expansive). It shrinks the input coefficient x according to the model ν and the modeled signal to noise ratio $\frac{\lambda}{\sigma}$ (SNR). When x is small in comparison to the SNR, it is likely that its noise component dominates the underlying signal, and is, therefore, shrunk towards 0. Similarly, when x is large, it will likely be preserved. This is even more likely when ν is small, since in this case large coefficients are favored by the *prior*. Illustrations of shrinkage functions for various SNR and ν are given in Figure 7.10.

7.5.2 Numerical approximations

The shrinkage function $s^{\nu}_{\sigma,\lambda}$, implicitly defined in (7.40) does not have a closed form expression in general. Nevertheless, for fixed values of x, σ, λ, ν, $s^{\nu}_{\sigma,\lambda}(x)$ can be estimated using iterative solvers such as Newton descent or Halley's root-finding method. These approaches

Table 7.1: Shrinkage function under generalized Gaussian priors

ν	Shrinkage $s^{\nu}_{\sigma,\lambda}(x)$	Remark
< 1	$\begin{cases} x - \gamma x^{\nu-1} + O(x^{2(\nu-1)}) & \text{if} \quad \lvert x\rvert \geqslant \tau^{\nu}_{\lambda} \\ 0 & \text{otherwise} \end{cases}$	$\approx$ Hard-thresholding [39]
1	$\text{sign}(x)\max\left(\lvert x\rvert - \dfrac{\sqrt{2}\sigma^2}{\lambda}, 0\right)$	Soft-thresholding [152]
$4/3$	$x + \gamma\left(\sqrt[3]{\dfrac{\zeta - x}{2}} - \sqrt[3]{\dfrac{\zeta + x}{2}}\right)$	[137]
$3/2$	$\text{sign}(x)\dfrac{\left(\sqrt{\gamma^2 + 4\lvert x\rvert} - \gamma\right)^2}{4}$	[137]
2	$\dfrac{\lambda^2}{\lambda^2 + \sigma^2} \cdot x$	Wiener (LMMSE)

$$\text{with} \quad \gamma = \nu\sigma^2\lambda_{\nu}^{-\nu} \quad \text{and} \quad \zeta = \sqrt{x^2 + 4\left(\frac{\gamma}{3}\right)^3}\,.$$

converge quite fast and, in practice, reach a satisfying solution within ten iterations. However, since in our application of interest we need to evaluate this function a large number times, we will follow a different path in order to reduce computation time (even though we have implemented this strategy).

As discussed earlier, $s^{\nu}_{\sigma,\lambda}$ is known in closed form for some values of ν, more precisely: $\nu = \{1, 4/3, 3/2, 2\}$ (as well as $\nu = 3$ but this is out of the scope of this study), see for instance [137]. When $\nu = 2$, we retrieve the linear minimum mean square estimator (known in signal processing as Wiener filtering) and related to Tikhonov regularization and ridge regression. This shrinkage is linear and the slope of the shrinkage goes from 0 to 1 as the SNR increases (see Figure 7.10). When $\nu = 1$, the shrinkage is the well-known soft-thresholding [152], corresponding to the maximum a posteriori estimator under a Laplacian prior. When $\nu < 1$, the authors of [39]

have shown that (i) the shrinkage admits a threshold with a closed-form expression (given in eq. (7.40)), and (ii) the shrinkage is approximately equal to hard-thresholding with an error term that vanishes when $|x| \to \infty$. All these expressions are summarized in Table 7.1.

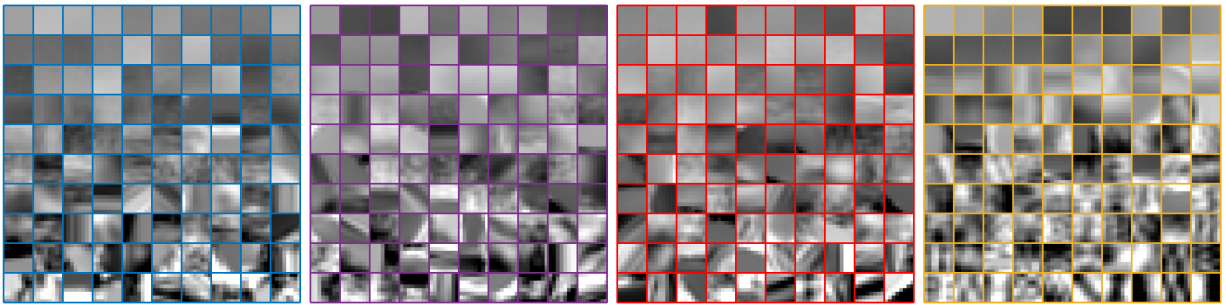
In order to keep our algorithm as fast as possible, we propose to use the approximation of the shrinkage given for $\nu < 1$ in [39]. Otherwise, we pick one of the four shrinkage functions corresponding to $\nu = \{1, 4/3, 3/2, 2\}$ by nearest neighbor on the actual value of $\nu \in [1, 2]$. Though this approximation may seem coarse compared to the one based on iterative solvers, we did not observe any significant loss of quality in our numerical experiments (see Section 7.6). Nonetheless, this alternative leads to 6 times speed-up while evaluating shrinkage.

7.6 Experimental evaluation

In this section we explain the methodology used to evaluate the GGMM model, and present numerical experiments to compare the performance of the proposed GGMM model over existing GMM-based image denoising algorithms. To demonstrate the advantage of allowing for a flexible GGMM model, we also present results using GGMM models with fixed shape parameters, $\nu = 1$ (Laplacian mixture model) and $\nu = 0.5$ (Hyper-Laplacian mixture model). For learning Laplacian mixture model (LMM) and hyper-Laplacian mixture model (HLMM), we use the same procedure as described in Section 7.3.1 but force all shape parameters to be equal to 1 or 0.5, respectively.

Model validation

As discussed in Section 7.3, Figure 7.1 illustrates the validity of our model choices with histograms of different dimensions of a single patch cluster. It clearly shows the importance of allowing the shape and scale parameter to vary across dimensions for capturing underlying patch distributions. Since GGMM (and obviously, GMM) falls into the class of *generative* models,



(a) GMM ($\nu = 2$) (b) GGMM ($.3 \leqslant \nu \leqslant 2$) (c) LMM ($\nu = 1$) (d) HLMM ($\nu = .5$)

Figure 7.11: Set of 100 patches, sorted by the norm of their gradient, and generated to be independently distributed according to (from left to right) a GMM, GGMM, LMM and HLMM. For ease of visualization, only the top eigendirections corresponding to 80% of the variance have been chosen. Near-constant patches with variance smaller than $\frac{2}{P}\|\Sigma_k\|_F^2$ have also been discarded.

one can also assess the *expressivity* of a model by analyzing the variability of generated patches and its ability to generate relevant image features (edges, texture elements etc.). This can be tested by selecting a component k of the GGMM (or GMM) with probability w_k and sampling patches from the GGD (or GD) as described in [153]. Figure 7.11 presents a collage of 100 patches independently generated by this procedure using GMM, GGMM, LMM and HLMM. As observed, patches generated from GGMM show greater balance between strong/faint edges, constant patches and subtle textures than the models that use constant shape parameters such as GGM, LMM and HLMM.

The superiority of our GGMM model over GMM, LMM or HLMM models can also be illustrated by comparing the log-likelihood (LL) achieved by these models over a set of clean patches from natural images. Note that, to maintain objectivity, the models have to be tested on data that is different than the dataset used during training. To this end, we compute the LL of the four above-mentioned models on all non-overlapping patches of 40 randomly selected images extracted from BSDS testing set [109], which is a different set than the training images used in the EM algorithm (parameter estimation/model learning). One can observe that not only GGMM is a better fit than GMM, LMM and HLMM on average for the 40 images, but it is also a better fit

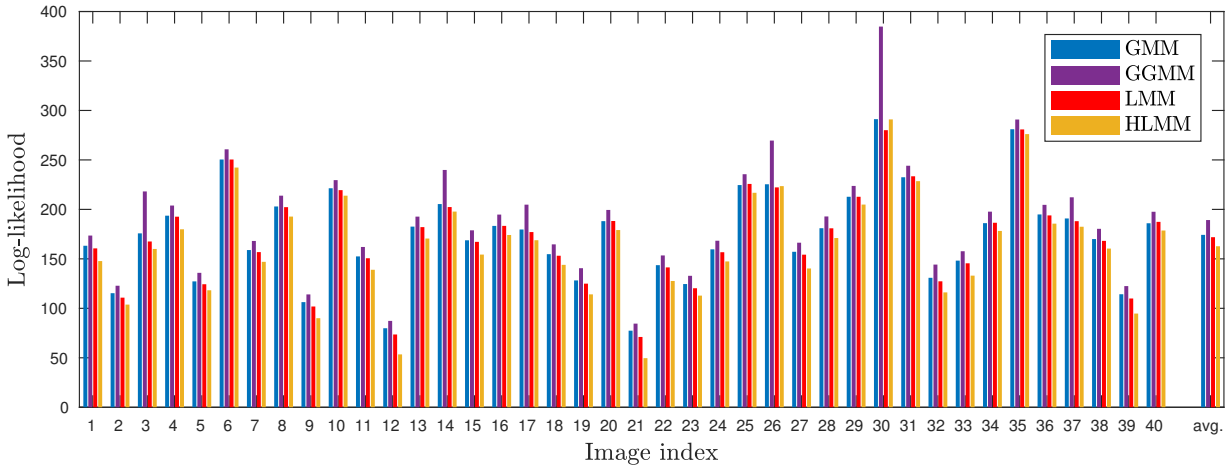


Figure 7.12: Average log-likelihood of all non-overlapping patches (with subtracted mean) of each of the 40 images of our validation subset of the testing BSDS dataset for the GMM, GGMM, LMM and HLMM. The total average over the 40 images is shown in the last column.

on each single image. Given that GGMM have a larger degree of freedom than GMM, this study proves that our learning procedure did not fall prey to over-fitting, and that the extra flexibility provided by GGMM was used to capture relevant and accurate image patterns.

Denoising evaluation

Following the verification of the model, we provide a thorough evaluation of our GGMM prior in denoising task by comparing its performance against EPLL that uses a GMM prior [15] and with our LMM and HLMM models explained above. For the ease of comparison, we utilize the pipeline and settings that was prescribed for the original EPLL [15] algorithm (see Section 7.2.1). To reduce the computation time of *all* EPLL-based algorithms, we utilize the random patch overlap procedure introduced by [5]. That is, instead of extracting all patches at each iteration, a randomly selected but different subset of overlapping patches consisting of only 3% of all possible patches is processed in each iteration. For the sake of reproducibility of our results, we have made our MATLAB/MEX-C implementation available online at \ourcodeurl.

The evaluation is carried out on classical images such as *Barbara*, *Cameraman*, *Hill*, *House*, *Lena* and *Mandrill*, and on 60 images taken from BSDS testing set [109] (the original

Table 7.2: Image denoising performance comparison of EPLL algorithm with GMM and GGMM priors. PSNR and SSIM values are obtained on the BSDS test set (average over 60 images), on six standard images corrupted with 5 different levels of noise (average over 10 noise realizations), and finally an average over these 66 images. BM3D algorithm results are also included for reference purposes.

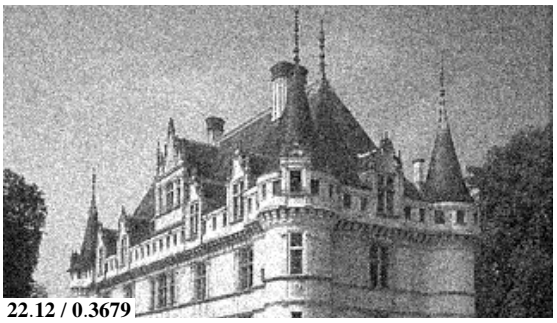
σ	Algo.	BSDS	barbara	camera man	hill	house	lena	mandrill	Avg.
					PSNR				
5	BM3D	37.33	38.30	38.28	36.04	39.82	38.70	35.26	37.36
	GMM	37.25	37.60	38.07	35.93	38.81	38.49	35.22	37.26
	GGMM	**37.33**	**37.73**	**38.12**	**35.95**	**38.94**	**38.52**	**35.23**	**37.33**
10	BM3D	33.06	34.95	34.10	31.88	36.69	35.90	30.58	33.15
	GMM	33.02	33.65	33.91	31.79	35.56	35.46	30.55	33.06
	GGMM	**33.10**	**33.87**	**34.01**	**31.81**	**35.72**	**35.59**	**30.58**	**33.15**
20	BM3D	29.38	31.73	30.42	28.56	33.81	33.02	26.60	29.50
	GMM	29.36	29.76	30.16	28.46	32.77	32.40	26.60	29.42
	GGMM	**29.43**	**30.02**	**30.24**	**28.48**	**33.03**	**32.59**	**26.64**	**29.50**
40	BM3D	26.28	27.97	27.16	25.89	30.69	29.81	23.07	26.38
	GMM	26.21	26.02	26.93	25.68	29.60	29.18	**23.25**	26.26
	GGMM	**26.26**	**26.17**	**27.03**	**25.70**	**29.89**	**29.42**	23.21	**26.32**
60	BM3D	24.81	26.31	25.24	24.52	28.74	28.19	21.71	24.90
	GMM	24.57	23.95	25.10	24.21	27.53	27.28	**21.57**	24.61
	GGMM	**24.64**	**24.03**	**25.17**	**24.25**	**27.80**	**27.52**	21.50	**24.67**
					SSIM				
5	BM3D	.9619	.9643	.9613	.9508	.9571	.9436	.9588	.9614
	GMM	.9626	.9616	**.9604**	**.9511**	**.9475**	**.9434**	**.9597**	.9618
	GGMM	**.9628**	**.9617**	.9602	.9507	.9469	.9425	.9593	**.9620**
10	BM3D	.9115	.9410	.9286	.8821	.9215	.9155	.8983	.9117
	GMM	**.9155**	.9298	.9307	**.8858**	**.8999**	.9107	**.9022**	**.9150**
	GGMM	.9154	**.9313**	**.9309**	.8839	.8992	**.9112**	.9007	.9149
20	BM3D	.8236	.9036	.8685	.7789	.8741	.8763	.7943	.8260
	GMM	**.8315**	.8687	**.8704**	**.7812**	.8596	.8639	**.8030**	**.8324**
	GGMM	.8297	**.8715**	.8699	.7766	**.8629**	**.8669**	.7991	.8308
40	BM3D	.7074	.8196	.7954	.6599	.8276	.8143	.6184	.7118
	GMM	**.7054**	.7509	.7780	**.6496**	.8025	.7918	**.6341**	**.7081**
	GGMM	.7018	**.7526**	**.7842**	.6430	**.8112**	**.7995**	.6192	.7048
60	BM3D	.6375	.7581	.7496	.5859	.7956	.7784	.4993	.6427
	GMM	**.6212**	.6534	.7174	**.5661**	.7507	.7350	**.5001**	**.6241**
	GGMM	.6174	**.6544**	**.7266**	.5592	**.7622**	**.7438**	.4782	.6207

Table 7.3: Image denoising performance comparison of EPLL algorithm with different priors. PSNR values are obtained on the BSDS test set (average over 60 images), on six standard images corrupted with 3 different levels of noise (average over 10 noise realizations), and finally an average over these 66 images.

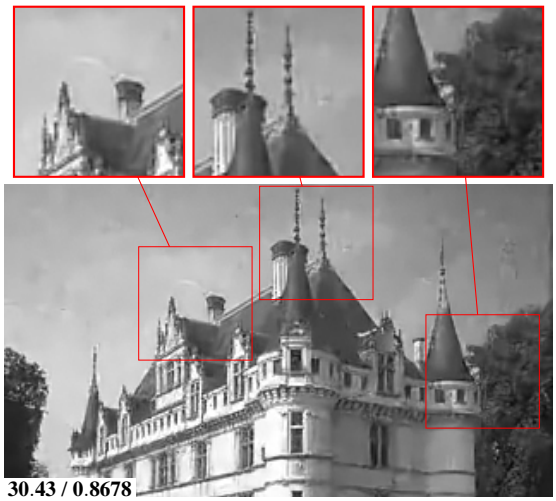
σ	Algo.	BSDS	barbara	camera man	hill	house	lena	mandrill	Avg.
		PSNR							
5	GMM	37.25	37.60	38.07	35.93	38.81	38.49	35.22	37.26
	LMM	37.31	**37.83**	38.11	35.89	38.93	38.49	35.18	37.32
	HLMM	36.85	37.42	37.66	35.39	38.37	38.08	34.77	36.86
	GGMM	**37.33**	37.73	**38.12**	**35.95**	**38.94**	**38.52**	**35.23**	**37.33**
20	GMM	29.36	29.76	30.16	28.46	32.77	32.40	26.60	29.42
	LMM	29.30	**30.18**	30.04	28.36	**33.22**	**32.72**	26.43	29.37
	HLMM	28.48	29.28	29.04	27.72	32.50	32.10	25.44	28.56
	GGMM	**29.43**	30.02	**30.24**	**28.48**	33.03	32.59	**26.64**	**29.50**
60	GMM	24.57	23.95	25.10	24.21	27.53	27.28	**21.57**	24.61
	LMM	24.55	23.94	24.96	24.23	**27.91**	**27.58**	21.35	24.59
	HLMM	23.95	23.16	23.72	23.84	27.10	26.94	20.67	23.97
	GGMM	**24.64**	**24.03**	**25.17**	**24.25**	27.80	27.52	21.50	**24.67**

BSDS test data contains 100 images, the other 40 were used for model validation experiments). All image have been corrupted independently by ten independent realizations of additive white Gaussian noise with standard deviation $\sigma = 5, 10, 20, 40$ and 60 (with pixel values between $[0, 255]$). The EPLL algorithm using mixture of Gaussian, generalized Gaussian priors are indicated as GMM and GGMM in Table 7.2. Results obtained with BM3D algorithm [28] are also included for reference purposes. To stay with the focus of this paper, *i.e.*, on the effect of image priors on EPLL-based algorithms, BM3D will be excluded from our performance comparison discussions. The denoising performance of the algorithms are measured in terms of Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM) [154]. As can be observed in Table 7.2, in general, GGMM prior provides better PSNR performance than the GMM prior. In terms of SSIM values (shown in the bottom part of Table 7.2), GGMM prior is comparable to GMM. In order to demonstrate the effect of fixed ν values compared to the more flexible GGMM prior, we compare the results of GGMM against GMM ($\nu = 2$), Laplacian Mixture Model (GGMM with

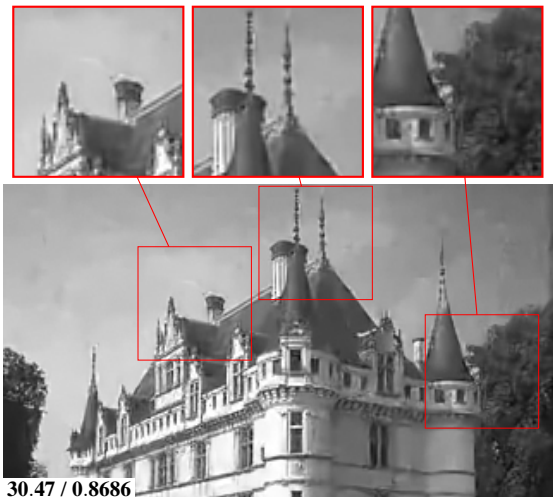
(a) Reference x



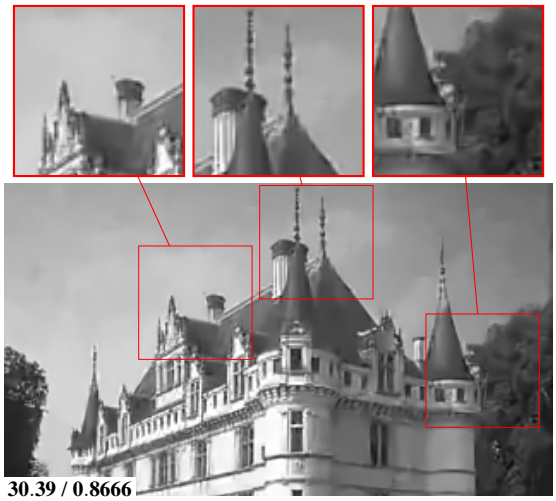
(b) Noisy y



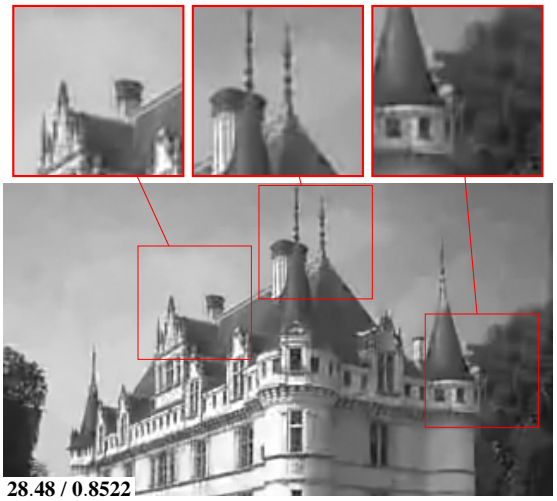
(c) GMM ($\nu = 2$)



(d) GGMM ($.3 \leqslant \nu \leqslant 2$)



(e) LMM ($\nu = 1$)



(f) HLMM ($\nu = 0.5$)

Figure 7.13: Qualitative comparison on *Castle*: (a) Close in on the image *Castle* from the BSDS testing dataset, (b) a noisy version degraded by additive white Gaussian noise with standard deviation $\sigma = 20$ and (c)-(f) results of EPLL under four patch priors: GMM, GGMM, LMM and HLMM, respectively. PSNR and SSIM are given in the bottom-left corner.

(a) Reference x

(b) Noisy y

(c) GMM ($\nu = 2$)

(d) GGMM ($.3 \leqslant \nu \leqslant 2$)

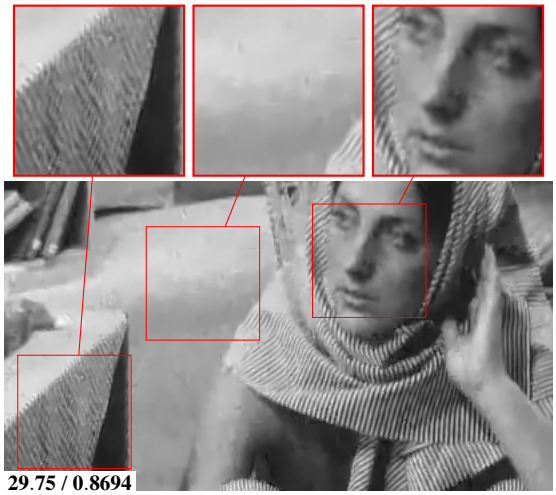
(e) LMM ($\nu = 1$)

(f) HLMM ($\nu = 0.5$)

Figure 7.14: Qualitative comparison on *Cameraman*: (a) Close in on the standard image *Cameraman*. (b) a noisy version degraded by additive white Gaussian noise with standard deviation $\sigma = 20$ and (c)-(f) results of EPLL under four patch priors: GMM, GGMM, LMM and HLMM, respectively. PSNR and SSIM are given in the bottom-left corner.

(a) Reference x

(b) Noisy y



(c) GMM ($\nu = 2$)

(d) GGMM ($.3 \leqslant \nu \leqslant 2$)

(e) LMM ($\nu = 1$)

(f) HLMM ($\nu = 0.5$)

Figure 7.15: Qualitative comparison on *Barbara*: (a) Close in on the standard image *Barbara*. (b) a noisy version degraded by additive white Gaussian noise with standard deviation $\sigma = 20$ and (c)-(f) results of EPLL under four patch priors: GMM, GGMM, LMM and HLMM, respectively. PSNR and SSIM are given in the bottom-left corner.

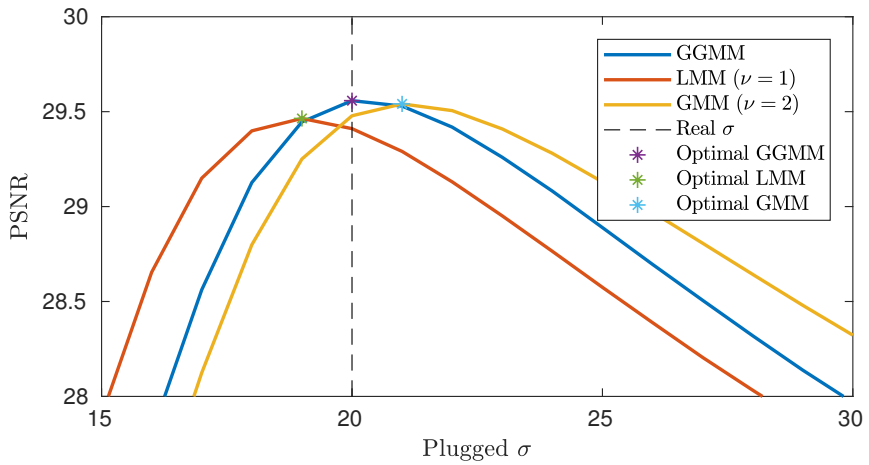


Figure 7.16: Evolution of performance of EPLL with a GGMM, LMM ($\nu = 1$) and a GMM ($\nu = 2$) under misspecification of the noise standard deviation σ. Performances are measured in terms of PSNR on the BSDS dataset corrupted by a Gaussian noise with standard deviation $\sigma = 20$. For each of the three priors, EPLL has been run assuming σ was ranging from 15 to 30.

$\nu = 1$) and hyper-Laplacian mixture model (GGMM with $\nu = 0.5$) priors in the same scenarios for $\sigma = 5, 20$ and 60. These results are shown in Table 7.3. GGMM prior provides better PSNR performance on average than the fixed-shape priors. The differences in denoising performance can also be verified visually in Figure 7.13, Figure 7.14 and Figure 7.15. The denoised images obtained using GGMM prior show much fewer artifacts as compared to GMM-EPLL results, in particular in homogeneous regions. On the other hand, GGMM prior is also able to better preserve textures than LMM and HLMM.

Prior fitness for image denoising

In this work, we have considered non-blind image denoising. That is, the noise standard deviation is assumed to be known. In this setting, if the restoration model is accurate, one should ideally achieve optimal restoration performance when using the true degradation. To verify this, we conducted a denoising task with image corrupted with noise with standard deviation $\sigma = 20$. We used GMM, LMM and GGMM priors in the restoration framework with assumed σ values ranging from 15 to 30. Figure 7.16 shows the evolution of average restoration performance over

40 images from BSDS testing set (kept aside for validation, as mentioned above) with varying noise variances. GGMM prior attains its best performance when the noise variance used in the restoration model matches with the ground truth $\sigma = 20$. In contrast to GGMM, GMM (resp., LMM) reaches its best performance at a larger (resp., lower) value of σ than the correct noise used during degradation. This is because GMM tends to under-smooth clean patches (resp., over-smooth) so that a larger (resp., lower) value of σ is required to compensate the mismatch between the assumed prior and the actual distribution in the restoration model. This indicates that GGMM is a better option to model distribution of image patches than GMM or LMM.

Influence of our approximations

All previous experiments using GGMM patch priors were conducted based on the two proposed approximations introduced in Section 7.4 and Section 7.5. In Figure 7.17 and Table 7.4, we provide a quantitative illustration of the speed-ups provided by these approximations and their effect on denoising performance. Timings were carried out with Matlab 2018a on an Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz (neither multi-core paralellization nor GPUs acceleration were used). Figure 7.17a shows the result obtained by calculating original discrepancy function via numerical integration and the shrinkage function via Halley's root-finding method. This makes the denoising process extremely slow and takes 10 hours and 29 minutes for denoising an image of size 128×128 pixels. The approximated discrepancy function provides 4 orders of magnitude speed-up with no perceivable drop in performance (Figure 7.17b). In addition, incorporating the shrinkage approximation provides further acceleration that allows the denoising to complete in less than 2 seconds with a very minor drop in PSNR/SSIM. As indicated in the detailed profiles on Table 7.4, the shrinkage approximation provides an acceleration of six-fold to the shrinkage calculation step itself and leads to an overall speed-up of 1.5 due to the larger bottleneck caused by discrepancy function calculation. The approximately 23,000$\times$ speed-up realized without any perceivable drop in denoising performance underscores the efficacy of our

(a) No approx. (10h 29m) (b) Approx. discrepancy (2s54) (c) Approx. disc. & shrink. (1s63)

Figure 7.17: Influence of our approximations on denoising performance. Results obtained by GGMM-EPLL on a 128×128 cropped image of the BSDS testing dataset damaged by additive white Gaussian noise with $\sigma = 20$. These are obtained respectively by (a) evaluating the classification and shrinkage problem with numerical solvers (numerical integration and Halley's root-finding method), (b) approximating the classification problem only, and (c) approximating both problems. PSNR and SSIM are given in the bottom-left corner. Running time (averaged on ten runs) of the overall GGMM-EPLL are indicated on the captions: our accelerations lead to a speed-up of $\times 15,000$ and $\times 1.5$ respectively.

proposed approximations.

7.7 Conclusions and Discussion

In this work, we suggest using a mixture of generalized Gaussians for modeling the patch distribution of clean images. We provide a detailed study of the challenges that one encounters when using a highly flexible GGMM prior for image restoration in place of a more common GMM prior. We identify the two main bottlenecks in the restoration procedure when using EPLL and GGMM – namely, the patch classification step and the shrinkage step. One of the main contributions of this paper, is the thorough theoretical analysis of the classification problem allowing us to introduce an asymptotically accurate approximation that is computationally efficient. In order to tackle the shrinkage step, we collate and extend the existing solutions under GGMM prior.

Table 7.4: Runtime profiles (averaged over ten runs) of GGMM-EPLL corresponding to the denoising experiment shown in Figure 7.17. These profiles are obtained respectively by evaluating the classification and shrinkage problem with numerical solvers (numerical integration and Halley's root-finding method), approximating the classification problem only, and approximating both problems. Profiles are split into discrepancy, shrinkage and patch extration/reprojection. Speed-up with respect to the no-approximation (first column) are indicated in parenthesis and major accelerations in green. Percentage of time taken for each step with respect to the overall execution time (first row), are indicated below each time reading and bottlenecks are indicated in red.

	No approximations	Approx. discrepancy		Approx. disc. & shrink.
Total	10h 29m 15s 100%	2.54s (×15,000) 100%	$\xrightarrow{\times 1.5}$	1.63s (×23,000) 100%
Discrepancy	10h 29m 14s >99.99%	1.44s (×26,000) 56.69%		1.44s (×26,000) 88.34%
Shrinkage	1.08s <0.001%	1.08s (×1) 42.52%		0.17s (×6.3) 10.43%
Patch extraction and reprojection	0.02s <0.001%	0.02s (×1) 0.79%		0.02s (×1) 1.23%

Our numerical experiments indicate that our flexible GGMM patch prior is a better fit for modeling natural images than GMM and other mixture distributions with constant shape parameters such as LMM or HLMM. In image denoising tasks, we have shown that using GGMM priors, often, outperforms GMM when used in the EPLL framework.

Acknowledgments

Parts of this chapter is a reprint of materials in the journal paper "Image restoration with generalized Gaussian mixture model patch priors", C-A. Deledalle, S. Parameswaran, and T. Q. Nguyen [6]. The dissertation author is the secondary investigator and author of this paper.

The authors would like to thank Charles Dossal and Nicolas Papadakis for fruitful discussions.

Part of the experiments presented in this paper were carried out using the PlaFRIM

experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr/).

Chapter 8

Conclusion

8.1 Summary of contributions

Due to ubiquitousness cameras and imagery data, there is a constant demand for *efficient* restoration algorithms that can post-process and correct for degradations in images and videos captured in unideal environmental conditions and/or by unideal sensors. Although there are many solutions proposed for image restoration, the lack of care taken to ensure algorithm scalability have led to a deluge of powerful algorithms that are computationally non-viable options that will take tens-hundreds of minutes on standard size images.

In this dissertation, we proposed algorithms that provide competent performance while being practically feasible for large scale applications. First, we demonstrated that, if runtime constraints are ignored, state-of-the-art patch-wise denoising filter design strategies can be improved by incorporating a robust patch matching metric using recommender system principles and, in case of videos, using temporal correspondences among patches. Following this, for the sake of algorithm scalability and flexibility, we replace patch-wise formulation with whole image denoising frameworks and investigate the usage of learned patch priors with progressively better expressivity and genericity.

We introduced the FED algorithm which can be thought of as using multiple independent isotropic Gaussian distributions to represent the patch prior distribution. Due to its simplicity it has limited expressivity and therefore, the prior has to be learned using a category-specific (targeted) database. To relax the category specific constraint, next we used the well-known GMM prior to capture the rich variations that one encounters in a generic patch database. The FEPLL algorithm utilizes the GMM prior along with novel approximations achieving more than two orders of magnitude runtime gains compared to the state-of-the-art GMM-based EPLL framework [15]. We continued this work by using a fully flexible GGMM to model natural image patch prior with a separate shape and scale parameter for each dimension. We analyzed the challenges presented by a non-Gaussian prior, and proposed approximate solutions that are asymptotically accurate and computationally efficient. The successful application of GGMM prior in image restoration has shed light into many future directions that my dissertation research can be extended which are briefly discussed below.

8.2 Future work

Given that GGMM is persistently a better prior than GMM (in terms of log-likelihood), one would expect the GGMM-EPLL to outperform GMM-EPLL consistently, but it falls short of its expected potential. We postulate that this under-performance is caused by the EPLL strategy that we use for optimization. That is, even though the GGMM prior may be improving the quality of the global solution, the half quadratic splitting strategy used in EPLL is not guaranteed to return a better solution due to the non-convexity of the underlying problem. For this reason, further research is needed in designing specific optimization strategies for GGMM-EPLL leveraging the better expressivity of the proposed prior model for denoising and other general restoration applications.

Another direction of future research can focus on extending this work to employ GGMMs

or GGDs in other model-based signal processing tasks. Of these tasks, estimating the parameters of GGMM directly on noisy observations is a problem of particular interest, that could benefit from our approximations. Learning GMM priors on noisy patches has been shown to be useful in patch-based image restoration when clean patches are not available *a priori*, or to further adapt the model to the specificities of a given noisy image [17, 32, 33]. Another open problem is to analyze the asymptotic behavior of the minimum mean square estimator (MMSE) shrinkage with GGD prior, as an alternative to MAP shrinkage. This could be useful to design accurate approximations for other general inference frameworks. Last but not least, characterizing the exact asymptotic behaviors of the convolution of two arbitrary GGDs, as investigated in [138], is still an open question. To the best of our knowledge, our study is the first attempt towards this goal but in ours one of the GGD is always Gaussian (noise). Extending our study to the general GGD case (or even specific cases such as Laplacian) is a challenging problem that is of major interest in signal processing tasks where noise is not Gaussian but instead follows another GGD.

Appendix A

Additional analysis and results of FEPLL algorithm

A.1 Flat tail approximation based acceleration

First, we demonstrate the superiority of the proposed flat tail approximation over the commonly used approach of ignoring the least significant eigendirections. In Supplemental Figure A.1, we show that the naïve approach of zeroing out (or ignoring) coefficients of the least significant eigendirections is inferior to the proposed approach of replacing these coefficients by the mean. Our approach can provide reasonably good performance even for very low values of ρ. For example, when $\rho = 0.20$ our approach leads to a PSNR of 27.5 compared to 25.1 obtained by the zeroing out strategy.

In a second experiment, we analyze the influence of the parameter $\rho \in (0, 1]$ used in the flat tail approximation. Supplemental Figure A.2 shows the curve of PSNR as a function of speed-up for varying values of ρ. This experiment is repeated twice either with or without enabling the other two accelerations. Visual results highlight that as ρ decreases to zero, residual noise starts appearing around salient structures. From these experiments, the choice of $\rho = 0.95$

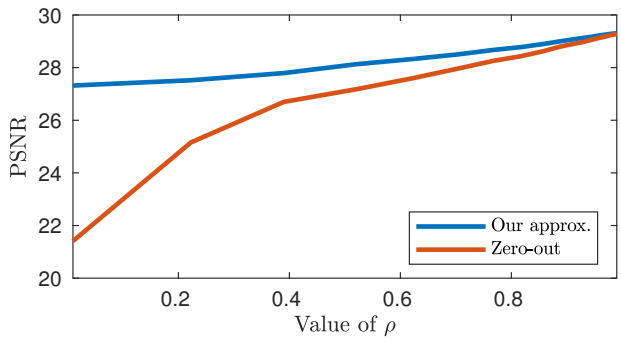
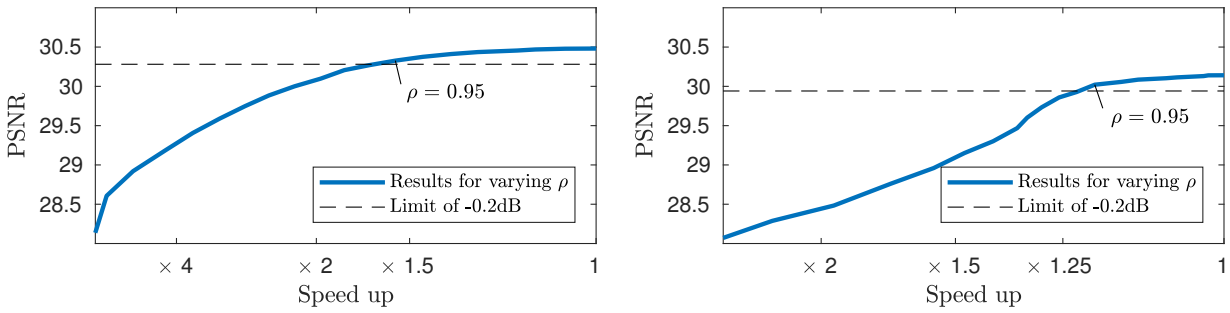


Figure A.1: Naïve approach *vs.* the proposed flat-tail approximation: Comparison between the naïve approach of zeroing out (ignoring) coefficients of the least significant eigendirections versus the proposed flat-tail approximation approach of replacing the coefficients with mean. Performance is compared in terms of PSNR obtained while denoising with different values of ρ (percent energy captured) used for thresholding singular values.

leads to a good trade-off in terms of speed and visual quality for a drop of PSNR lower than 0.2dB.

A.2 Search tree acceleration

In the second supplementary experiment, we analyze the influence of the clustering method used to successively collapse the GMM model at a given level into a smaller model at a lower level. We compare the proposed approach based on Multiple Traveling Salesmen Problem [112] (MTSP) using the symmetric KL divergence as a semi-metric to a hierarchical GMM clustering algorithm similar to K-means based on KL divergence as proposed in [113] and the standard hierarchical agglomerated clustering (HAC) using symmetric KL divergence. Supplemental Figure A.3 shows that MTSP leads to a well-balanced tree, with a height of 7 (almost a binary tree except for the last level due to 200 mixture components). In comparison, using [113]'s K-Means inspired algorithm provides a tree of height 7 but is not balanced and HAC leads to a tree of height 59 with comb structured branches. Please note that a shorter tree is preferred for a faster Gaussian selection step. The tree built using the proposed MTSP strategy not only leads to better PSNR/SSIM, but also provides a more stable computation time for all



(a) (b)

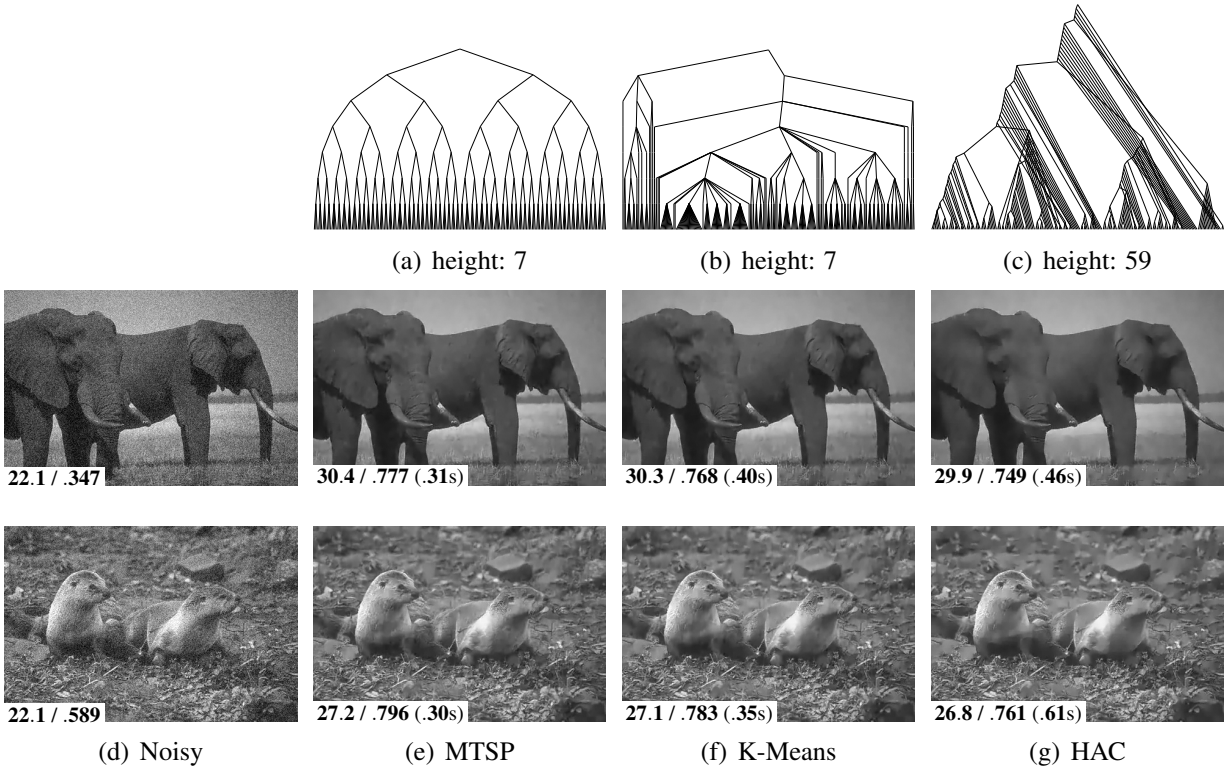
(c) Noisy (d) $\rho = 0.5$ (e) $\rho = 0.8$ (f) $\rho = 0.95$ (g) $\rho = 1$

Figure A.2: Influence of the parameter ρ on denoising performance. Top: PSNR as a function of speed-up for varying values of the proportion ρ used in the flat tail approximation when (a) no other accelerations are used and (b) the other two accelerations are also used. The parameter offering the best speed-up for a drop of at most -0.2dB is indicated. Bottom: (c) a noisy image with $\sigma = 20$ and (d-g) FEPLL results obtained for increasing value of ρ (includes the other two accelerations as well).

images of a fixed size (irrespective of content) due to its balanced structure. This is confirmed in Supplemental Figure A.4 that displays box-plots obtained from computation time statistics based on five independent runs of the algorithm on 40 different images of the BSDS dataset (all images have the same size). In contrast to our MTSP based strategy, the trees built using [113]'s method and HAC strategy lead to computation times that vary drastically depending on the image content.

A.3 Stochastic patch sub-sampling

The last analysis focuses on the influence of the period $1 \leqslant s \leqslant \sqrt{P}$ used in the stochastic patch sub-sampling. In addition, we also compare against regular sub-sampling. Supplemental



(a) height: 7 (b) height: 7 (c) height: 59

(d) Noisy (e) MTSP (f) K-Means (g) HAC

Figure A.3: Influence of clustering techniques on the search tree and the results. (left) Multiple Traveling Salesmen Problem [112] (MTSP), (center) Hierarchical K-means like clustering proposed in [113] and (right) Hierarchical agglomerated clustering (HAC). (a-c) The corresponding trees. (d-i) The corresponding results on two denoising problems with $\sigma = 20$ involving two different images of size 481×321. PSNR/SSIM and time are indicated for each result.

Figure A.5 shows the curve of PSNR as a function of speed-up for varying values of s where $P = 8$. The experiment has been performed in both cases where either the other two accelerations are disabled or enabled. Visual results highlight that as s increases to $\sqrt{P}$, blocky artifacts start appearing especially when using a regular patch sub-sampling. In comparison, the stochastic sub-sampling leads to competing results even for $s = \sqrt{P}$, which corresponds to a regular grid that is globally shifted by a random shift at each iteration of FEPLL. These experiments show that the choice of $s = 6$ leads to good trade-off in terms of speed and visual quality for a PSNR drop of less than 0.2dB.

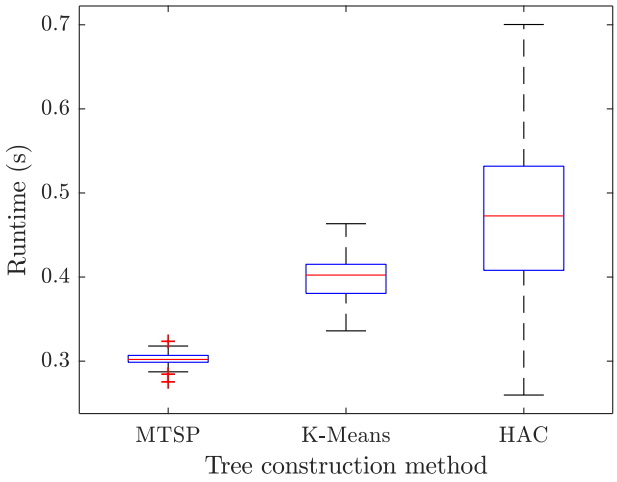


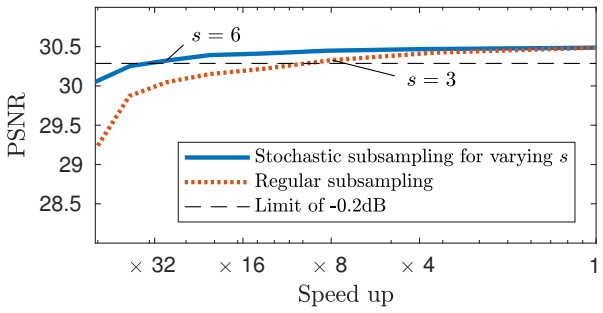
Figure A.4: Timing scatter obtained over 5 runs for 40 images using Gaussian trees built using different clustering methods we tested. "MTSP" refers to the Multiple Traveling Salesman (proposed), "K-Means" is the method introduced by [113] and "HAC" is a simple Hierarchical agglomerative clustering approach.

A.4 Additional results

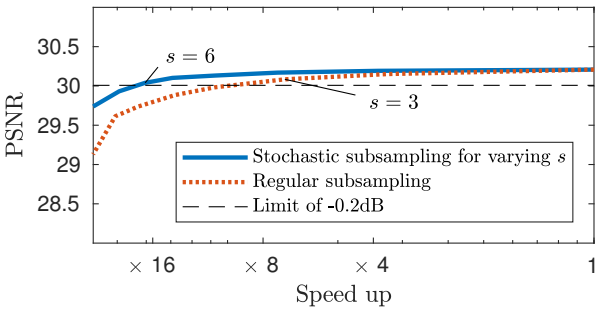
Denoising results As the first set of additional results, we provide the visual results obtained on five of the standard images: Barbara, Boat, Couple, Fingerprint, Lena and Mandrill, in the setting $\sigma = 20$. Results are displayed in Supplemental Figure A.6.

Deconvolution results As the second set additional results, we show deconvolution results for five of testing images from the BSDS dataset [109] with a Gaussian blur setting of 3 pixels (standard deviation 3) and noise level $\sigma = 2$. Results are displayed in Supplemental Figure A.7.

Super-resolution results As the last set of additional results, super-resolution results are given for five of the testing images from the BSDS dataset [109] for super-resolution by a factor of 3 with a noise level of $\sigma = 2$. Note that the sub-sampling operator also includes a small Gaussian blur of 0.5 pixels as well as a Kaiser window apodization for anti-aliasing. See Supplemental Figure A.8.



(a) without the other accelerations



(b) with the other accelerations

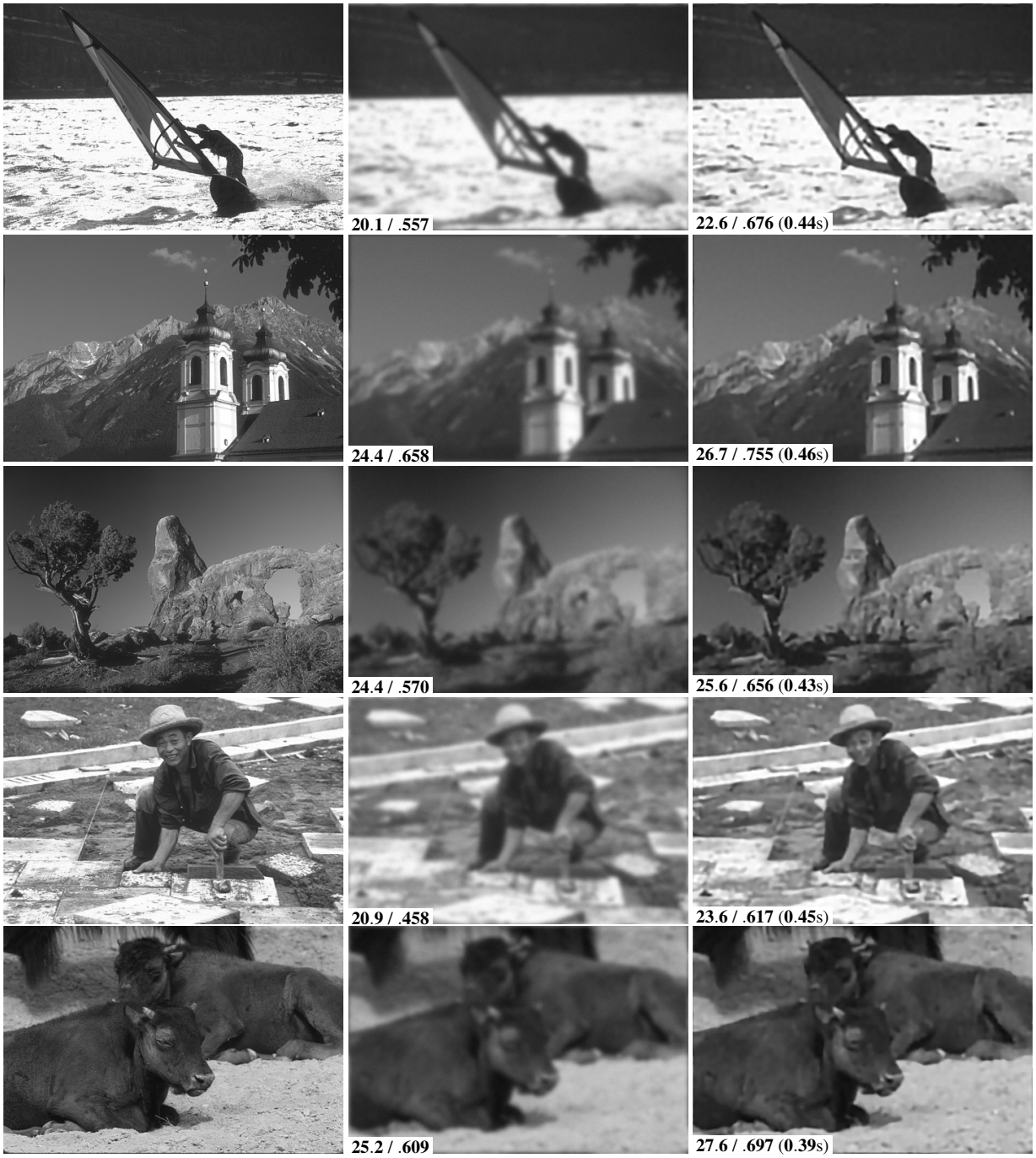
(c) Reference (d) Stochastic $s = 2$ (e) Stochastic $s = 4$ (f) Stochastic $s = 6$ (g) Stochastic $s = 8$

(h) Noisy (i) Regular $s = 2$ (j) Regular $s = 4$ (k) Regular $s = 6$ (l) Regular $s = 8$

Figure A.5: Influence of random patch subsampling parameter s on performance. PSNR as a function of speed-up for varying values of (top) the patch extraction expected period s when (a) no other accelerations are used and (b) the other two accelerations are also used. Stochastic and regular sub-sampling are compared. The parameter offering the best speed-up within a drop of at most -0.2dB is indicated. (c) A reference image, (h) its noisy version with $\sigma = 20$ and results obtained for increasing value of s (including the other two accelerations as well) for (d-g) the stochastic version and (i-l) the regular version.

(a) Reference (b) Noisy (c) FEPLL (d) EPLLc (e) BM3D

Figure A.6: Denoising results where $\sigma = 20$ obtained on five standard images, from top to bottom: Barbara, Boat, Couple, Fingerprint, Lena and Mandrill.



(a) Reference image x (b) Blurry image y (c) Deblurred result $\hat{x}$

Figure A.7: Illustration of a deblurring problem of a Gaussian convolution of width 3 pixels with a noise of standard deviation $\sigma = 2$. (a) The original high-resolution (HR) image x. (b) The low-resolution (LR) image $y = \mathcal{A}x + w$ and its bicubic interpolation. (c) The super-resolution result $\hat{x}$ obtained by our Fast EPLL.

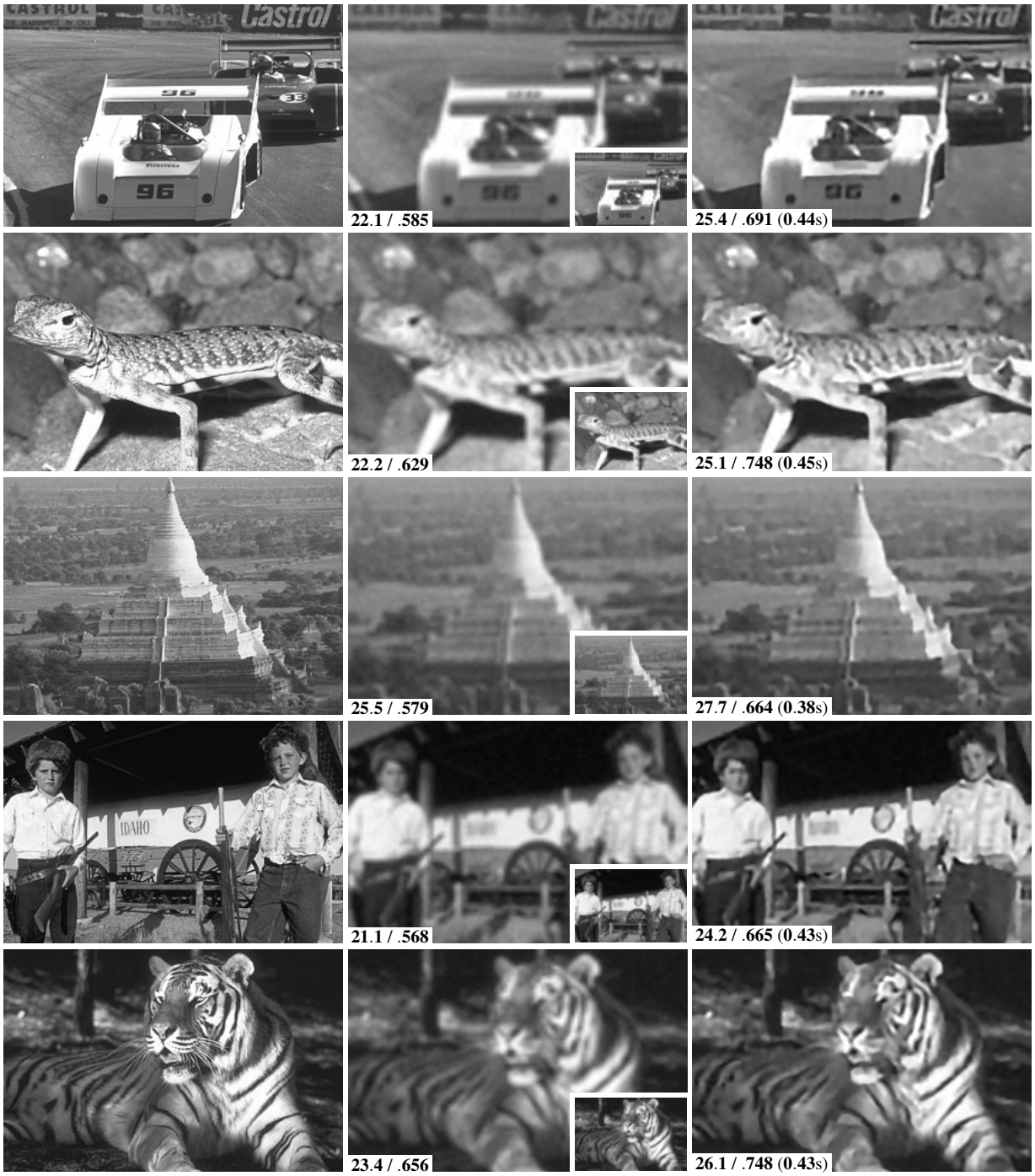


Figure A.8: Illustration of a super-resolution by a factor $\times 3$ with a noise of standard deviation $\sigma = 2$. (a) The original high-resolution (HR) image x. (b) The low-resolution (LR) image $y = \mathcal{A}x + w$ and its bicubic interpolation. (c) The super-resolution result $\hat{x}$ obtained by our Fast EPLL.

Appendix B

Proofs for Theorems in Chapter 7

B.1 Proof of equation (7.23)

Proof 1 *For* $\nu = 1$*, using* $\Gamma(1) = 1$ *and* $\Gamma(3) = 2$*, we obtain*

$$f^1_{\sigma,\lambda}(x) = \log(2\sqrt{\pi}\sigma\lambda) - \log \int_{-\infty}^{\infty} e^{-\frac{t^2}{2\sigma^2} - \frac{\sqrt{2}|x-t|}{\lambda}} \, \mathrm{d}t \ , \tag{B.1}$$

$$= \log(2\sqrt{\pi}\sigma\lambda) - \log \left[e^{-\frac{\sqrt{2}x}{\lambda}} \int_{-\infty}^{x} e^{-\frac{t^2}{2\sigma^2} + \frac{\sqrt{2}t}{\lambda}} \, \mathrm{d}t + e^{\frac{\sqrt{2}x}{\lambda}} \int_{x}^{\infty} e^{-\frac{t^2}{2\sigma^2} - \frac{\sqrt{2}t}{\lambda}} \, \mathrm{d}t \right] . \tag{B.2}$$

Since $\mathrm{erf}'(t) = \frac{2e^{-t^2}}{\sqrt{\pi}}$*, it follows that for* $a > 0$ *and* $b > 0$

$$\frac{\partial}{\partial t} \left[-\frac{\sqrt{\pi a}}{2} e^{\frac{a}{4b^2}} \mathrm{erf}\left(-\frac{t}{\sqrt{a}} - \frac{\sqrt{a}}{2b} \right) \right] = e^{-\frac{t^2}{a} - \frac{t}{b}} \ . \tag{B.3}$$

Therefore we have with $a = 2\sigma^2$ *and* $b = \lambda/\sqrt{2}$

$$\int_{-\infty}^{x} e^{-\frac{t^2}{2\sigma^2} + \frac{\sqrt{2}t}{\lambda}} \, \mathrm{d}t = \frac{-\sqrt{\pi}\sigma e^{\frac{\sigma^2}{\lambda^2}}}{\sqrt{2}} \left[\mathrm{erf}\left(-\frac{t}{\sqrt{2}\sigma} + \frac{\sigma}{\lambda} \right) \right]_{-\infty}^{x} = \frac{\sqrt{\pi}\sigma e^{\frac{\sigma^2}{\lambda^2}}}{\sqrt{2}} \mathrm{erfc}\left(-\frac{x}{\sqrt{2}\sigma} + \frac{\sigma}{\lambda} \right) \ , \tag{B.4}$$

since $\lim_{t\to\infty} \operatorname{erf}(t) = 1$ *and* $\operatorname{erfc}(t) = 1 - \operatorname{erf}(t)$. *Similarly, we get*

$$\int_x^\infty e^{-\frac{t^2}{2\sigma^2} - \frac{\sqrt{2}t}{\lambda}} \, \mathrm{d}t = \frac{-\sqrt{\pi}\sigma e^{\frac{\sigma^2}{\lambda^2}}}{\sqrt{2}} \operatorname{erfc}\left(\frac{x}{\sqrt{2}\sigma} + \frac{\sigma}{\lambda}\right) . \tag{B.5}$$

Plugging these two last expressions in (B.2) *and rearranging the terms conclude the proof.*

B.2 Proof of Proposition 7.4.2

Proof 2 *Starting from the definition of* $f^{\nu}_{\sigma,\lambda}$ *and using the change of variable* $t \to \sigma t$, *eq.* (reduction) *follows as*

$$\begin{aligned} f^{\nu}_{\sigma,\lambda}(x) &= -\log \int_{\mathbb{R}} \sigma \frac{\kappa}{2\lambda_\nu} \exp\left[-\left(\frac{\sigma|t|}{\lambda_\nu}\right)^\nu\right] \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\sigma t)^2}{2\sigma^2}\right] \mathrm{d}t \ , \\ &= \log\sigma - \log \int_{\mathbb{R}} \frac{\kappa}{2\lambda_\nu/\sigma} \exp\left[-\left(\frac{|t|}{\lambda_\nu/\sigma}\right)^\nu\right] \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{(x/\sigma - t)^2}{2}\right] \mathrm{d}t = \log\sigma + f^{\nu}_{1,\lambda/\sigma}\left(\frac{x}{\sigma}\right) . \end{aligned} \tag{B.6}$$

Properties (even) *and* (unimodality) *hold since the convolution of two real even unimodal distributions is even unimodal [155, 156]. Property* (lower bound at 0) *follows from* (even), (unimodality) *and the fact that the convolution of continuous and bounded real functions are continuous and bounded.*

B.3 Proof of Theorem 7.4.3

Lemma B.3.1. *Let* $a > 0$ *and* $b > 0$. *For* x *in the vicinity of* 0, *we have*

$$\frac{1}{2abx} \log\left[\frac{\operatorname{erfc}(ax+b) + e^{-4abx}\operatorname{erfc}(-ax+b)}{2\operatorname{erfc}(b)}\right] = -1 + \left(ab - \frac{ae^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}\right)x + o(x) . \tag{B.7}$$

Proof 3 *Since* $\operatorname{erfc}'(x) = -\frac{2e^{-x^2}}{\sqrt{\pi}}$ *and* $\operatorname{erfc}''(x) = \frac{2xe^{-x^2}}{\sqrt{\pi}}$*, using second order Taylor's expansion for* x *in the vicinity of* 0*, it follows that*

$$\operatorname{erfc}(ax+b) \underset{0}{\sim} \operatorname{erfc}(b) - \frac{2ae^{-b^2}}{\sqrt{\pi}}x + \frac{2a^2be^{-b^2}}{\sqrt{\pi}}x^2 \,, \tag{B.8}$$

$$\textit{and} \quad \operatorname{erfc}(-ax+b) \underset{0}{\sim} \operatorname{erfc}(b) + \frac{2ae^{-b^2}}{\sqrt{\pi}}x + \frac{2a^2be^{-b^2}}{\sqrt{\pi}}x^2 \,. \tag{B.9}$$

We next make the following deductions

$$e^{-4abx}\operatorname{erfc}(-ax+b)) \underset{0}{\sim} e^{-4abx}\operatorname{erfc}(b) + e^{-4abx}\frac{2ae^{-b^2}}{\sqrt{\pi}}x + e^{-4abx}\frac{2a^2be^{-b^2}}{\sqrt{\pi}}x^2 \,, \tag{B.10}$$

$$\begin{aligned}\operatorname{erfc}(ax+b)+e^{-4abx}\operatorname{erfc}(-ax+b)) \underset{0}{\sim} (1+e^{-4abx})\left(\operatorname{erfc}(b)+\frac{2a^2be^{-b^2}}{\sqrt{\pi}}x^2\right)\\ -(1-e^{-4abx})\frac{2ae^{-b^2}}{\sqrt{\pi}}x \,,\end{aligned}$$

$$\begin{aligned}\underbrace{\frac{\operatorname{erfc}(ax+b)+e^{-4abx}\operatorname{erfc}(-ax+b))}{2\operatorname{erfc} b}}_{A(x)} \underset{0}{\sim} \frac{1+e^{-4abx}}{2}\left(1+\frac{2a^2be^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}x^2\right)\\ -(1-e^{-4abx})\frac{ae^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}x \,.\end{aligned}$$

The left-hand side $A(x)$ *of this last equation is then, in the vicinity of* $x = 0$*, equals to*

$$A(x) = \frac{1+e^{-4abx}}{2}\left(1+\frac{2a^2be^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}x^2\right) - (1-e^{-4abx})\frac{ae^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}x + o(x^2) \,. \tag{B.11}$$

We next use second-order Taylor's expansion for e^{-4abx}, leading to

$$A(x) = (1-2abx+4a^2b^2x^2+o(x^2))\left(1+\frac{2a^2be^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}x^2\right) \tag{B.12}$$

$$-(4abx-8a^2b^2x^2+o(x^2))\frac{ae^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}x+o(x^2)\ ,$$

$$= 1-2abx+\left(4a^2b^2-\frac{2a^2be^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}\right)x^2+o(x^2)\ . \tag{B.13}$$

By using the second-order Taylor's expansion of $\log(1+x)$*, it follows that*

$$\log[A(x)] = -2abx+\left(4a^2b^2-\frac{2a^2be^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}\right)x^2-2a^2b^2x^2+o(x^2)\ . \tag{B.14}$$

Dividing both sides by $2abx$ *then concludes the proof,*

$$\frac{1}{2abx}\log\left[\frac{\operatorname{erfc}(ax+b)+e^{-4abx}\operatorname{erfc}(-ax+b))}{2\operatorname{erfc}b}\right] = -1+\left(ab-\frac{ae^{-b^2}}{\operatorname{erfc}(b)\sqrt{\pi}}\right)x+o(x)\ . \tag{B.15}$$

Proof 4 (Proof of Theorem 7.4.3) *We first rewrite* $\varphi_\lambda^1(x)$ *as*

$$\varphi_\lambda^1(x) = \log\left[\log\left[2\operatorname{erfc}\left(\frac{1}{\lambda}\right)\right]-\log\left[e^{\frac{\sqrt{2}x}{\lambda}}\operatorname{erfc}\left(\frac{x}{\sqrt{2}}+\frac{1}{\lambda}\right)+e^{-\frac{\sqrt{2}x}{\lambda}}\operatorname{erfc}\left(-\frac{x}{\sqrt{2}}+\frac{1}{\lambda}\right)\right]\right]\ ,$$

$$= \log\left[\frac{\sqrt{2}x}{\lambda}\right]+\log\left[-1-\frac{\lambda}{\sqrt{2}x}\log\left[\frac{\operatorname{erfc}\left(\frac{x}{\sqrt{2}}+\frac{1}{\lambda}\right)+e^{-\frac{2\sqrt{2}x}{\lambda}}\operatorname{erfc}\left(-\frac{x}{\sqrt{2}}+\frac{1}{\lambda}\right)}{2\operatorname{erfc}\left(\frac{1}{\lambda}\right)}\right]\right]\ . \tag{B.16}$$

Next, using Lemma B.3.1 with $a = 1/\sqrt{2}$ and $b = 1/\lambda$, it follows that

$$\varphi_\lambda^1(x) = \log\left[\frac{\sqrt{2}x}{\lambda}\right] + \log\left[-\left(\frac{1}{\sqrt{2}\lambda} - \frac{e^{-\frac{1}{\lambda^2}}}{\sqrt{2}\operatorname{erfc}(\frac{1}{\lambda})\sqrt{\pi}}\right)x + o(x)\right] , \tag{B.17}$$

$$= \log\left[\frac{x^2}{\lambda}\right] + \log\left[\frac{e^{-\frac{1}{\lambda^2}}}{\operatorname{erfc}(\frac{1}{\lambda})\sqrt{\pi}} - \frac{1}{\lambda} + o(1)\right] , \tag{B.18}$$

$$= \log\left[\frac{x^2}{\lambda}\right] + \log\left[\frac{1}{\sqrt{\pi}}\frac{e^{-\frac{1}{\lambda^2}}}{\operatorname{erfc}(\frac{1}{\lambda})} - \frac{1}{\lambda}\right] + o(1) , \tag{B.19}$$

where the last equation follows from the first-order Taylor expansion of $\log(a+x)$.

B.4 Proof of Theorem 7.4.4

Lemma B.4.1. *Let $a > 0$ and $b > 0$. For x in the vicinity of $+\infty$, we have*

$$\frac{1}{2abx}\log\left[\frac{\operatorname{erfc}(ax+b) + e^{-4abx}\operatorname{erfc}(-ax+b)}{2}\right] = -2 + o(1) . \tag{B.20}$$

Proof 5 *We have* $\lim_{x\to+\infty} \operatorname{erfc}(x) = 0$ *and* $\lim_{x\to+\infty} \operatorname{erfc}(-x) = 2$, *it follows that*

$$\operatorname{erfc}(-ax+b) \underset{+\infty}{\sim} 2 , \tag{B.21}$$

$$e^{-4abx}\operatorname{erfc}(-ax+b) \underset{+\infty}{\sim} 2e^{-4abx} , \tag{B.22}$$

$$\frac{\operatorname{erfc}(ax+b) + e^{-4abx}\operatorname{erfc}(-ax+b)}{2} \underset{+\infty}{\sim} e^{-4abx} , \tag{B.23}$$

$$\log\left[\frac{\operatorname{erfc}(ax+b) + e^{-4abx}\operatorname{erfc}(-ax+b)}{2}\right] \underset{+\infty}{\sim} -4abx , \tag{B.24}$$

$$\frac{1}{2abx}\log\left[\operatorname{erfc}(ax+b) + e^{-4abx}\operatorname{erfc}(-ax+b)\right] \underset{+\infty}{\sim} -2 , \tag{B.25}$$

where we have used the knowledge that $f \sim g$ implies that $\log f \sim \log g$.

Proof 6 (Proof of Theorem 7.4.4) *By writing φ_λ^1 as in eq.* (B.16) *and using Lemma B.4.1 with*

$a = 1/\sqrt{2}$ and $b = 1/\lambda$, it follows that

$$\varphi_\lambda^1(x) = \log\left[\frac{\sqrt{2}x}{\lambda}\right] + \log\left[1 + \frac{\lambda}{\sqrt{2}x}\log\operatorname{erfc}\left(\frac{1}{\lambda}\right) + o(1)\right] = \log\left[\frac{\sqrt{2}x}{\lambda}\right] + o(1)\ .$$

B.5 Proof of Theorem 7.4.5

Proof 7 *We first decompose the term $\exp\left(-\frac{(x-t)^2}{2}\right)$ involved in the definition of $f_{1,\lambda}^\nu$ and rewrite e^{xt} using its power series*

$$f_{1,\lambda}^\nu(x) = -\log\frac{1}{\sqrt{2\pi}}\frac{\nu}{2\lambda_\nu\Gamma(1/\nu)} - \log\int_{-\infty}^{\infty}\exp\left(-\frac{(x-t)^2}{2}\right)\exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right]\mathrm{d}t\ , \tag{B.26}$$

$$= -\log\frac{1}{\sqrt{2\pi}}\frac{\nu}{2\lambda_\nu\Gamma(1/\nu)} - \log\int_{-\infty}^{\infty}e^{-\frac{x^2}{2}}e^{xt}e^{-\frac{t^2}{2}}\exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right]\mathrm{d}t\ , \tag{B.27}$$

$$= -\log\frac{1}{\sqrt{2\pi}}\frac{\nu}{2\lambda_\nu\Gamma(1/\nu)} + \frac{x^2}{2} - \log\int_{-\infty}^{\infty}\sum_{k=0}^{\infty}\frac{(xt)^k}{k!}e^{-\frac{t^2}{2}}\exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right]\mathrm{d}t\ . \tag{B.28}$$

For x in the vicinity of 0, we can consider $|x| \leqslant 1$, and then

$$\int_{-\infty}^{\infty}\sum_{k=0}^{\infty}\left|\frac{(xt)^k}{k!}e^{-\frac{t^2}{2}}\exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right]\right|\mathrm{d}t \leqslant \int_{-\infty}^{\infty}\sum_{k=0}^{\infty}\frac{|t|^k}{k!}e^{-\frac{t^2}{2}}\exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right]\mathrm{d}t\ , \tag{B.29}$$

$$\leqslant \int_{-\infty}^{\infty}e^{-\frac{t^2}{2}+|t|-\left(\frac{|t|}{\lambda_\nu}\right)^\nu}\mathrm{d}t\ < \infty\ . \tag{B.30}$$

Then, Fubini's theorem applies and we get

$$f_{1,\lambda}^\nu(x) = -\log\frac{1}{\sqrt{2\pi}}\frac{\nu}{2\lambda_\nu\Gamma(1/\nu)} + \frac{x^2}{2} - \log\sum_{k=0}^{\infty}\int_{-\infty}^{\infty}\frac{(xt)^k}{k!}e^{-\frac{t^2}{2}}\exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right]\mathrm{d}t\ , \tag{B.31}$$

$$= -\log\frac{1}{\sqrt{2\pi}}\frac{\nu}{2\lambda_\nu\Gamma(1/\nu)} + \frac{x^2}{2} - \log\sum_{k=0}^{\infty}\frac{x^k}{k!}\int_{-\infty}^{\infty}t^k e^{-\frac{t^2}{2}}\exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right]\mathrm{d}t\ . \tag{B.32}$$

By definition, we have

$$\gamma_\lambda^\nu \triangleq f_{1,\lambda}^\nu(0) = -\log \frac{1}{\sqrt{2\pi}} \frac{\nu}{2\lambda_\nu \Gamma(1/\nu)} - \log \int_{-\infty}^{\infty} \exp\left(-\frac{t^2}{2}\right) \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t \ . \tag{B.33}$$

Moreover, when k is odd, we have

$$\int_{-\infty}^{\infty} t^k e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t = 0 \ . \tag{B.34}$$

Using third-order Taylor expansion of $\log(1+x)$ *for x in the vicinity of* 0*, it follows that*

$$f_{1,\lambda}^\nu(x) = \gamma_\lambda^\nu + \frac{x^2}{2} - \log\left(1 + \frac{x^2}{2} \frac{\int_{-\infty}^{\infty} t^2 e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t}{\int_{-\infty}^{\infty} e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t} + o(x^3)\right) \ , \tag{B.35}$$

$$= \gamma_\lambda^\nu + \frac{x^2}{2}\left(1 - \frac{\int_{-\infty}^{\infty} t^2 e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t}{\int_{-\infty}^{\infty} e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t}\right) + o(x^3) \ . \tag{B.36}$$

Finally, using first-order Taylor's expansion for $\log(1+x)$*, we conclude the proof as*

$$\varphi_\lambda^\nu(x) = \log\left[\frac{x^2}{2}\left(1 - \frac{\int_{-\infty}^{\infty} t^2 e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t}{\int_{-\infty}^{\infty} e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t} + o(x)\right)\right] \ , \tag{B.37}$$

$$= 2\log x - \log 2 + \log\left(1 - \frac{\int_{-\infty}^{\infty} t^2 e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t}{\int_{-\infty}^{\infty} e^{-\frac{t^2}{2}} \exp\left[-\left(\frac{|t|}{\lambda_\nu}\right)^\nu\right] \mathrm{d}t}\right) + o(x) \ . \tag{B.38}$$

B.6 Proof of Theorem 7.4.6

We first recall in a lemma, a result extracted from Corollary 3.3 in [148].

Lemma B.6.1 (Berman). *Let p and q be differentiable real probability density functions. Define*

for x large enough $u(x) = p^{-1}(q(x))$ and define v and w as

$$v(x) = -\frac{\partial}{\partial x}\log p(x) \quad \textit{and} \quad w(x) = -\frac{\partial}{\partial x}\log q(x)\ . \tag{B.39}$$

Assume v and w are positive continuous function and regularly oscillating, i.e.:

$$\lim_{\substack{x,x'\to\infty \\ x/x'\to 1}} \frac{v(x)}{v(x')} = 1 \quad \textit{and} \quad \lim_{\substack{x,x'\to\infty \\ x/x'\to 1}} \frac{w(x)}{w(x')} = 1\ . \tag{B.40}$$

Suppose that we have

$$\lim_{x\to\infty} \frac{w(x)}{v(x)} = 0 \quad \textit{and} \quad \lim_{x\to\infty} u(x)w(x) = +\infty\ , \tag{B.41}$$

then, for $x \to \infty$, we have

$$\log \int_{-\infty}^{+\infty} p(x-t)q(t)\ \mathrm{d}t \sim \log q(x)\ . \tag{B.42}$$

Proof 8 (Proof of Theorem 7.4.6) *Using the definition of the discrepancy function,*

$$f^{\nu}_{1,\lambda}(x) = -\log \int_{\mathbb{R}} \mathcal{G}(t;0,\lambda,\nu)\cdot\mathcal{N}(x-t;0,1)\ \mathrm{d}t = -\log \int_{-\infty}^{+\infty} p(x-t)q(t)\ \mathrm{d}t\ . \tag{B.43}$$

with $q(x) = \mathcal{G}(x;0,\lambda,\nu)$ and $p(x) = \mathcal{N}(x;0,1)$. We have

$$v(x) = -\frac{\partial}{\partial x}\log p(x) = 2x \quad \textit{and} \quad w(x) = -\frac{\partial}{\partial x}\log q(x) = \frac{\nu x^{\nu-1}}{\lambda_\nu^\nu}\ . \tag{B.44}$$

Remark that v and w are positive continuous, and

$$\lim_{\substack{x,x'\to\infty \\ x/x'\to 1}} \frac{v(x)}{v(x')} = \lim_{\substack{x,x'\to\infty \\ x/x'\to 1}} \frac{x}{x'} = 1 \quad \textit{and} \quad \lim_{\substack{x,x'\to\infty \\ x/x'\to 1}} \frac{w(x)}{w(x')} = \lim_{\substack{x,x'\to\infty \\ x/x'\to 1}} \left(\frac{x}{x'}\right)^{\nu-1} = 1\ . \tag{B.45}$$

For $x > 0$ large enough and $y > 0$ small enough

$$y = p(x) \Leftrightarrow y = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \Leftrightarrow x = \sqrt{-\log(2\pi) - 2\log y}\,. \tag{B.46}$$

Then, from Proposition 7.3.1, we have

$$u(x)w(x) = \frac{\nu x^{\nu-1}}{\lambda_\nu^\nu} \sqrt{-\log(2\pi) - 2\log q(x)} \tag{B.47}$$

$$= \frac{\nu x^{\nu-1}}{\lambda_\nu^\nu} \sqrt{-\log(2\pi) - 2\log\left[\frac{\kappa}{2\lambda_\nu}\right] + 2\left(\frac{x}{\lambda_\nu}\right)^\nu} \sim \frac{\nu\sqrt{2}x^{\frac{3}{2}\nu-1}}{\lambda_\nu^{\frac{3}{2}\nu}}\,, \tag{B.48}$$

and thus, as $\nu > \frac{2}{3}$, $\lim_{x\to\infty} u(x)w(x) = \infty$. Moreover, for $\nu < 2$, we have

$$\lim_{x\to\infty} \frac{w(x)}{v(x)} = \lim_{x\to\infty} \frac{\nu}{2\lambda_\nu} x^{\nu-2} = 0\,. \tag{B.49}$$

It follows that Lemma B.6.1 applies, and then for large x

$$f_{1,\lambda}^\nu(x) \sim -\log q(x) \sim \left(\frac{x}{\lambda_\nu}\right)^\nu\,. \tag{B.50}$$

Using that $\lambda_\nu = \lambda\sqrt{\frac{\Gamma(1/\nu)}{\Gamma(3/\nu)}}$, we conclude the proof since

$$\varphi_\lambda^\nu(x) = \log\left[f_{1,\lambda}^\nu(x) - \gamma_\lambda^\nu\right] \sim \nu\log x - \nu\log\lambda_\nu\,. \tag{B.51}$$

B.7 Proof of Proposition 7.5.1

Proof 9 *Starting from the definition of* $s^{\nu}_{\sigma,\lambda}$ *and using the change of variable* $t \to \sigma t$*, eq.* (reduction) *follows as*

$$s^{\nu}_{\sigma,\lambda}(x) = \operatorname*{argmin}_{t\in\mathbb{R}} \frac{(x-t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|t|^\nu = \sigma \operatorname*{argmin}_{t\in\mathbb{R}} \frac{(x-\sigma t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|\sigma t|^\nu \,, \tag{B.52}$$

$$= \sigma \operatorname*{argmin}_{t\in\mathbb{R}} \frac{(x/\sigma-t)^2}{2} + (\lambda_\nu/\sigma)^{-\nu}|t|^\nu = \sigma s^{\nu}_{1,\lambda/\sigma}(x/\sigma) \,. \tag{B.53}$$

For eq. (odd)*, we use the change of variable* $t \to -t$

$$s^{\nu}_{\sigma,\lambda}(-x) = \operatorname*{argmin}_{t\in\mathbb{R}} \frac{(-x-t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|t|^\nu = -\operatorname*{argmin}_{t\in\mathbb{R}} \frac{(-x+t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|t|^\nu \,, \tag{B.54}$$

$$= -\operatorname*{argmin}_{t\in\mathbb{R}} \frac{(x-t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|t|^\nu = -s^{\nu}_{\sigma,\lambda}(x) \,. \tag{B.55}$$

We now prove eq. (shrinkage)*. Let* $t = s^{\nu}_{\sigma,\lambda}(x)$*, and since* t *minimizes the objective, then*

$$\lambda_\nu^{-\nu}|t|^\nu \leqslant \frac{(x-t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|t|^\nu \leqslant \frac{(x-x)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|x|^\nu = \lambda_\nu^{-\nu}|x|^\nu \,, \tag{B.56}$$

which implies that $|t| \leqslant |x|$*. Let* $x > 0$ *and assume* $t = s^{\nu}_{\sigma,\lambda}(x) < 0$*. Since* t *minimizes the objective, then*

$$\frac{(x-t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|t|^\nu \leqslant \frac{(x+t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|t|^\nu \tag{B.57}$$

which implies that $-xt \leqslant xt$ *and leads to a contradiction. Then for* $x > 0$*,* $s^{\nu}_{\sigma,\lambda}(x) \in [0, x]$*, which concludes the proof since* $s^{\nu}_{\sigma,\lambda}$ *is odd.*

We now prove (increasing with x)*. Let* $x_1 > x_2$ *and define* $t_1 = s^{\nu}_{\sigma,\lambda}(x_1)$ *and* $t_2 = s^{\nu}_{\sigma,\lambda}(x_2)$*.*

Since t_1 and t_2 minimize their respective objectives, the following two statements hold

$$\frac{(x_1-t_1)^2}{2\sigma^2}+\lambda_\nu^{-\nu}|t_1|^\nu \leqslant \frac{(x_1-t_2)^2}{2\sigma^2}+\lambda_\nu^{-\nu}|t_2|^\nu \,, \tag{B.58}$$

$$\textit{and} \quad \frac{(x_2-t_2)^2}{2\sigma^2}+\lambda_\nu^{-\nu}|t_2|^\nu \leqslant \frac{(x_2-t_1)^2}{2\sigma^2}+\lambda_\nu^{-\nu}|t_1|^\nu \,. \tag{B.59}$$

Summing both inequalities lead to

$$(x_1-t_1)^2+(x_2-t_2)^2 \leqslant (x_1-t_2)^2+(x_2-t_1)^2 \,, \tag{B.60}$$

$$\Rightarrow \quad -2x_1t_1-2x_2t_2 \leqslant -2x_1t_2-2x_2t_1 \,, \tag{B.61}$$

$$\Rightarrow \quad t_1(x_1-x_2) \geqslant t_2(x_1-x_2) \quad \Rightarrow \quad t_1 \geqslant t_2 \quad \textit{(since } x_1 > x_2\textit{)} \,. \tag{B.62}$$

We now prove (increasing with λ). *Let $\lambda_1 > \lambda_2$ and define $t_1 = s^\nu_{\sigma,\lambda_1}(x)$ and $t_2 = s^\nu_{\sigma,\lambda_2}(x)$. Since t_1 and t_2 minimize their respective objectives, the following expressions hold*

$$\frac{(x-t_1)^2}{2\sigma^2}+\lambda_{\nu,1}^{-\nu}|t_1|^\nu \leqslant \frac{(x-t_2)^2}{2\sigma^2}+\lambda_{\nu,1}^{-\nu}|t_2|^\nu \,, \tag{B.63}$$

$$\textit{and} \quad \frac{(x-t_2)^2}{2\sigma^2}+\lambda_{\nu,2}^{-\nu}|t_2|^\nu \leqslant \frac{(x-t_1)^2}{2\sigma^2}+\lambda_{\nu,2}^{-\nu}|t_1|^\nu \,. \tag{B.64}$$

Again, summing both inequalities lead to

$$\lambda_{\nu,1}^{-\nu}|t_1|^\nu+\lambda_{\nu,2}^{-\nu}|t_2|^\nu \leqslant \lambda_{\nu,1}^{-\nu}|t_2|^\nu+\lambda_{\nu,2}^{-\nu}|t_1|^\nu \,, \tag{B.65}$$

$$\Rightarrow \quad (\lambda_{\nu,1}^{-\nu}-\lambda_{\nu,2}^{-\nu})|t_1|^\nu \leqslant (\lambda_{\nu,1}^{-\nu}-\lambda_{\nu,2}^{-\nu})|t_2|^\nu \,, \tag{B.66}$$

$$\Rightarrow \quad |t_1|^\nu \geqslant |t_2|^\nu \quad \textit{(since } \lambda_1 > \lambda_2 \textit{ and } \nu > 0\textit{)} \,. \tag{B.67}$$

We now prove (keep high SNR). *Consider $x > 0$. Since $\lambda \mapsto s^\nu_{\sigma,\lambda}(x)$ is a monotonic function and $s^\nu_{\sigma,\lambda}(x) \in [0,x]$ for all λ, it converges for $\lambda \to \infty$ to a value $\omega \in [0,x]$. Assume*

$0 < \omega < x$ *and let* $0 < \varepsilon < \max(\omega, x-\omega)$. *By definition of the limit, for* λ *big enough*

$$0 < \omega - \varepsilon < t \triangleq s^{\nu}_{\sigma,\lambda}(x) < \omega + \varepsilon \; . \tag{B.68}$$

It follows that $x - t > x - (w + \varepsilon) > 0$, *and then*

$$\frac{(x-(\omega+\varepsilon))^2}{2\sigma^2} + \lambda_{\nu}^{-\nu}|\omega-\varepsilon|^{\nu} < \frac{(x-t)^2}{2\sigma^2} + \lambda_{\nu}^{-\nu}|t|^{\nu} \; . \tag{B.69}$$

Moreover, since $\omega + \varepsilon \neq x$, *we have for* λ *big enough*

$$\lambda_{\nu}^{-\nu}|x|^{\nu} < \frac{(x-(\omega+\varepsilon))^2}{2\sigma^2} + \lambda_{\nu}^{-\nu}|\omega-\varepsilon|^{\nu} \; . \tag{B.70}$$

Combining the two last inequalities shows that

$$\frac{(x-x)^2}{2\sigma^2} + \lambda_{\nu}^{-\nu}|x|^{\nu} < \frac{(x-t)^2}{2\sigma^2} + \lambda_{\nu}^{-\nu}|t|^{\nu} \; , \tag{B.71}$$

which is in contradiction with the fact that t *minimizes the objective. As a consequence,* $\omega = x$, *which concludes the proof since* $s^{\nu}_{\sigma,\lambda}(x)$ *is odd and satisfies* (reduction).

We now prove (kill low SNR). *Consider* $x > 0$. *Since* $\lambda \mapsto s^{\nu}_{\sigma,\lambda}(x)$ *is a monotonic function and* $s^{\nu}_{\sigma,\lambda}(x) \in [0,x]$ *for all* λ, *it converges for* $\lambda \to 0^+$ *to a value* $\omega \in [0,x]$. *Assume* $0 < \omega < x$ *and let* $0 < \varepsilon < \max(\omega, x-\omega)$. *Again, we have for* λ *small enough*

$$\frac{(x-(\omega+\varepsilon))^2}{2\sigma^2} + \lambda_{\nu}^{-\nu}|\omega-\varepsilon|^{\nu} < \frac{(x-t)^2}{2\sigma^2} + \lambda_{\nu}^{-\nu}|t|^{\nu} \; . \tag{B.72}$$

Moreover, since $\omega \neq \varepsilon$, *we have for* λ *small enough*

$$\frac{x^2}{2\sigma^2} < \frac{(x-(\omega+\varepsilon))^2}{2\sigma^2} + \lambda_{\nu}^{-\nu}|\omega-\varepsilon|^{\nu} \; . \tag{B.73}$$

Combining the two last inequalities shows that

$$\frac{(x-0)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|0|^\nu < \frac{(x-t)^2}{2\sigma^2} + \lambda_\nu^{-\nu}|t|^\nu \ , \tag{B.74}$$

which is in contradiction with the fact that t minimizes the objective. As a consequence, $\omega = 0$*, which concludes the proof since* $s^\nu_{\sigma,\lambda}(x)$ *is odd and satisfies* (reduction)*.*

Bibliography

[1] G. D. Gopen and J. A. Swan, "The science of scientific writing," 1990.

[2] S. Parameswaran, E. Luo, and T. Nguyen, "Patch matching for image denoising using neighborhood-based collaborative filtering," *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[3] S. Parameswaran, E. Luo, and T. Q. Nguyen, "Targeted video denoising for decompressed videos," in *Image Processing (ICIP), 2017 IEEE International Conference on*, pp. 2981–2985, IEEE, 2017.

[4] S. Parameswaran, E. Luo, C.-A. Deledalle, and T. Q. Nguyen, "Fast external denoising using pre-learned transformations," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pp. 1025–1033, IEEE, 2017.

[5] S. Parameswaran, C.-A. Deledalle, L. Denis, and T. Q. Nguyen, "Accelerating gmm-based patch priors for image restoration: Three ingredients for a 100× speed-up," *IEEE Transactions on Image Processing*, 2018 (In press).

[6] C.-A. Deledalle, S. Parameswaran, and T. Q. Nguyen, "Image denoising with generalized gaussian mixture model patch priors," *SIAM Journal on Imaging Sciences*, 2018 (In press).

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

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, pp. 2080–2095, August 2007.

[9] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, pp. 3736–3745, December 2006.

[10] J. Boulanger, J.-B. Sibarita, C. Kervrann, and P. Bouthemy, "Non-parametric regression for patch-based fluorescence microscopy image sequence denoising," in *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pp. 748–751, IEEE, 2008.

[11] M. Lebrun, A. Buades, and J.-M. Morel, "A nonlocal bayesian image denoising algorithm," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1665–1688, 2013.

[12] C. Knaus and M. Zwicker, "Dual-domain image denoising," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pp. 440–444, IEEE, 2013.

[13] L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern Recognition*, vol. 43, pp. 1531–1549, April 2010.

[14] C.-A. Deledalle, V. Duval, and J. Salmon, "Non-local methods with shape-adaptive patches (nlm-sap)," *Journal of Mathematical Imaging and Vision*, vol. 43, no. 2, pp. 103–120, 2012.

[15] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *International Conference on Computer Vision*, pp. 479–486, IEEE, November 2011.

[16] H. Burger, C. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[17] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2481–2499, 2012.

[18] Y.-Q. Wang and J.-M. Morel, "Sure guided gaussian mixture image denoising," *SIAM Journal on Imaging Sciences*, vol. 6, no. 2, pp. 999–1034, 2013.

[19] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 860–867, IEEE, 2005.

[20] Y. Chen, R. Ranftl, and T. Pock, "Insights into analysis operator learning: From patch-based sparse models to higher order mrfs," *IEEE Transactions on Image Processing*, vol. 23, pp. 1060–1072, March 2014.

[21] P. Ochs, Y. Chen, T. Brox, and T. Pock, "ipiano: Inertial proximal algorithm for nonconvex optimization," *SIAM Journal on Imaging Sciences*, vol. 7, no. 2, pp. 1388–1419, 2014.

[22] Y. Zhou, "Explore the power of external data in denoising task,"

[23] E. Luo, S. H. Chan, and T. Q. Nguyen, "Adaptive image denoising by targeted databases," *IEEE Transactions on Image Processing*, vol. 24, pp. 2167–2181, July 2015.

[24] E. Luo, S. H. Chan, and T. Q. Nguyen, "Image denoising by targeted external databases," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2450–2454, IEEE, May 2014.

[25] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

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

[27] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 60–65, IEEE, 2005.

[28] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[29] L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern Recognition*, vol. 43, pp. 1531–1549, Apr. 2010.

[30] M. Aharon, M. Elad, and A. Bruckstein, "*k*-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[31] A. Van Den Oord and B. Schrauwen, "The student-t mixture as a natural image patch prior with application to image compression.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2061–2086, 2014.

[32] A. M. Teodoro, M. S. Almeida, and M. A. Figueiredo, "Single-frame image denoising and inpainting using gaussian mixtures.," in *ICPRAM (2)*, pp. 283–288, 2015.

[33] A. Houdard, C. Bouveyron, and J. Delon, "High-Dimensional Mixture Models For Unsupervised Image Denoising (HDMI)." Preprint hal-01544249, Aug. 2017.

[34] G. Singh and J. Singh, "Adapted non-gaussian mixture model for image denoising," *Journal of Basic and Applied Engineering Research*, vol. 4, no. 6, 2017.

[35] M. Niknejad, J. M. Bioucas-Dias, and M. A. Figueiredo, "Class-specific image denoising using importance sampling," in *IEEE International Conference on Image Processing*, IEEE, 2017.

[36] F. J. Anscombe, "The transformation of poisson, binomial and negative-binomial data," *Biometrika*, vol. 35, no. 3/4, pp. 246–254, 1948.

[37] C.-A. Deledalle, J. Salmon, and A. S. Dalalyan, "Image denoising with patch based pca: local versus global," in *BMVC 2011-22nd British Machine Vision Conference*, pp. 25–1, BMVA Press, 2011.

[38] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[39] P. Moulin and J. Liu, "Analysis of multiresolution image denoising schemes using generalized gaussian and complexity priors," *IEEE Transactions on Information Theory*, vol. 45, no. 3, pp. 909–919, 1999.

[40] M. J. Shensa, "The discrete wavelet transform: wedding the a trous and mallat algorithms," *IEEE Transactions on signal processing*, vol. 40, no. 10, pp. 2464–2482, 1992.

[41] R. R. Coifman and D. L. Donoho, "Translation-invariant de-noising," *Wavelets and statistics*, vol. 103, pp. 125–150, 1995.

[42] R. J. Duffin and A. C. Schaeffer, "A class of nonharmonic fourier series," *Transactions of the American Mathematical Society*, vol. 72, no. 2, pp. 341–366, 1952.

[43] S. Mallat, *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[44] M. Elad, P. Milanfar, and R. Rubinstein, "Analysis versus synthesis in signal priors," *Inverse problems*, vol. 23, no. 3, p. 947, 2007.

[45] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[46] J. Sulam and M. Elad, "Expected patch log likelihood with a sparse prior," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 99–111, Springer, 2015.

[47] R. Rubinstein, T. Peleg, and M. Elad, "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model," *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 661–677, 2013.

[48] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[49] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Transactions on Image processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[50] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 492–526, 2010.

[51] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '99*, (New York, New York, USA), pp. 230–237, ACM Press, August 1999.

[52] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, June 2005.

[53] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1–19, January 2009.

[54] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, pp. 133–151, July 2001.

[55] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the Tenth International Conference on World Wide Web - WWW '01*, (New York, New York, USA), pp. 285–295, ACM Press, April 2001.

[56] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 43–52, IEEE, October 2007.

[57] C. E. Thomaz and G. A. Giraldi, "A new ranking method for principal components analysis and its application to face image analysis," *Image and Vision Computing*, vol. 28, pp. 902–913, June 2010.

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

[59] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, pp. 2080–2095, August 2007.

[60] A. Buades, B. Coll, and J.-M. Morel, "Denoising image sequences does not require motion estimation," in *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*, pp. 70–74, Sept 2005.

[61] C. Liu and W. T. Freeman, "A high-quality video denoising algorithm based on reliable motion estimation," in *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ECCV'10, (Berlin, Heidelberg), pp. 706–719, Springer-Verlag, 2010.

[62] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3d transform-domain collaborative filtering," in *Signal Processing Conference, 2007 15th European*, pp. 145–149, Sept 2007.

[63] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Processing Letters*, vol. 12, pp. 839–842, 2005.

[64] P. Chatterjee and P. Milanfar, "Is Denoising Dead?," *IEEE Transactions on Image Processing*, vol. 19, pp. 895–911, April 2010.

[65] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'11)*, pp. 2833–2840, IEEE, June 2011.

[66] H. C. Burger, C. Schuler, and S. Harmeling, "Learning how to combine internal and external denoising methods," in *Pattern Recognition* (J. Weickert, M. Hein, and B. Schiele, eds.), vol. 8142 of *Lecture Notes in Computer Science*, pp. 121–130, Springer Berlin Heidelberg, 2013.

[67] S. H. Chan, T. Zickler, and Y. M. Lu, "Monte Carlo non-local means: random sampling for large-scale image filtering.," *IEEE transactions on image processing*, vol. 23, pp. 3711–25, August 2014.

[68] I. Mosseri, M. Zontak, and M. Irani, "Combining the Power of Internal and External Denoising," in *IEEE International Conference on Computational Photography (ICCP)*, pp. 1–9, 2013.

[69] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *International Conference on Computer Vision*, pp. 479–486, IEEE, Nov. 2011.

[70] E. Luo, S. H. Chan, and T. Q. Nguyen, "Image denoising by targeted external databases," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2450–2454, IEEE, May 2014.

[71] E. Luo, S. H. Chan, and T. Q. Nguyen, "Adaptive image denoising by targeted databases.," *IEEE transactions on image processing*, vol. 24, pp. 2167–81, July 2015.

[72] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE Transactions on Image Processing*, vol. 22, pp. 119–133, Jan 2013.

[73] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian, "Video denoising, deblocking, and enhancement through separable 4-d nonlocal spatiotemporal transforms," *IEEE Transactions on Image Processing*, vol. 21, pp. 3952–3966, Sept 2012.

[74] A. Buades, J.-L. Lisani, and M. Miladinovc, "Patch-based video denoising with optical flow estimation," *Transactions on Image Processing*, vol. 25, pp. 2573–2586, June 2016.

[75] M. D. Rodriguez, J. Ahmed, and M. Shah, "Action mach: a spatio-temporal maximum average correlation height filter for action recognition," in *In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[76] K. Soomro and A. R. Zamir, *Computer Vision in Sports*, ch. Action Recognition in Realistic Sports Videos, pp. 181–208. Cham: Springer International Publishing, 2014.

[77] N. Ben-Zrihem and L. Zelnik-Manor, "Riann: Approximate nearest neighbor fields in video," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'15)*, 2015.

[78] C. Barnes, F.-L. Zhang, L. Lou, X. Wu, and S.-M. Hu, "Patchtable: Efficient patch queries for large datasets and applications," in *ACM Transactions on Graphics (Proc. SIGGRAPH)*, Aug. 2015.

[79] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings 8th International Conference on Computer Vision*, vol. 2, pp. 416–423, July 2001.

[80] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *European Conference on Computer Vision*, pp. 241–257, Springer, 2016.

[81] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. Bourdev, and R. Fergus, "Web scale photo hash clustering on a single machine," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[82] M. Aharon, M. Elad, and A. Bruckstein, "k -svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, pp. 4311–4322, Nov 2006.

[83] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th annual international conference on machine learning*, pp. 689–696, ACM, 2009.

[84] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[85] M. Weber, M. Welling, and P. Perona, "Towards automatic discovery of object categories," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 101–108 vol.2, 2000.

[86] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, April 2004.

[87] A. Danielyan, A. Foi, V. Katkovnik, and K. Egiazarian, "Image upsampling via spatially adaptive block-matching filtering," in *Signal Processing Conference, 2008 16th European*, pp. 1–5, IEEE, 2008.

[88] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *IEEE 12th International Conference on Computer Vision*, 2009.

[89] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the nonlocal-means to super-resolution reconstruction," *IEEE Transactions on image processing*, vol. 18, no. 1, pp. 36–51, 2009.

[90] A. Singh, F. Porikli, and N. Ahuja, "Super-resolving noisy images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2846–2853, 2014.

[91] K. Egiazarian and V. Katkovnik, "Single image super-resolution via bm3d sparse coding," in *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pp. 2849–2853, IEEE, 2015.

[92] E. López-Rubio, "Superresolution from a single noisy image by the median filter transform," *SIAM Journal on Imaging Sciences*, vol. 9, no. 1, pp. 82–115, 2016.

[93] T. Guillemot, A. Almansa, and T. Boubekeur, "Covariance Trees for 2D and 3D Processing," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2014.

[94] E. Luo, S. H. Chan, and T. Q. Nguyen, "Adaptive image denoising by mixture adaptation," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4489–4503, 2016.

[95] N. Cai, Y. Zhou, S. Wang, B. W.-K. Ling, and S. Weng, "Image denoising via patch-based adaptive gaussian mixture prior method," *Signal, Image and Video Processing*, vol. 10, no. 6, pp. 993–999, 2016.

[96] V. Papyan and M. Elad, "Multi-scale patch-based image restoration," *IEEE Transactions on image processing*, vol. 25, no. 1, pp. 249–261, 2016.

[97] Y. Ren, Y. Romano, and M. Elad, "Example-based image synthesis via randomized patch-matching," *arXiv preprint arXiv:1609.07370*, 2016.

[98] A. Danielyan, A. Foi, V. Katkovnik, and K. Egiazarian, "Spatially adaptive filtering as regularization in inverse imaging: Compressive sensing super-resolution and upsampling," *Super-Resolution Imaging*, pp. 123–154, 2010.

[99] V. Katkovnik and K. Egiazarian, "Nonlocal image deblurring: Variational formulation with nonlocal collaborative L0-norm prior," in *Local and Non-Local Approximation in Image Processing, 2009. LNLA 2009. International Workshop on*, pp. 46–53, IEEE, 2009.

[100] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2774–2781, 2014.

[101] J. Jancsary, S. Nowozin, and C. Rother, "Regression tree fields – an efficient, non-parametric approach to image labeling problems," in *25th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[102] Y. Chen, W. Yu, and T. Pock, "On learning optimized reaction diffusion processes for effective image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5261–5269, 2015.

[103] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[104] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1646–1654, 2016.

[105] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[106] J. Rick Chang, C.-L. Li, B. Poczos, B. Vijaya Kumar, and A. C. Sankaranarayanan, "One network to solve them all–solving linear inverse problems using deep projection models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5888–5897, 2017.

[107] Y. Wang, S. Cho, J. Wang, and S.-F. Chang, "Discriminative indexing for probabilistic image patch priors," in *European Conference on Computer Vision*, pp. 200–214, Springer, 2014.

[108] D. Rosenbaum and Y. Weiss, "The return of the gating network: combining generative models and discriminative training in natural image priors," in *Advances in Neural Information Processing Systems*, pp. 2683–2691, 2015.

[109] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, pp. 416–423, IEEE, 2001.

[110] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp. 932–946, 1995.

[111] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Advances in Neural Information Processing Systems*, pp. 1033–1041, 2009.

[112] J. Kirk, "Source code for "multiple traveling salesmen problem - genetic algorithm", https://fr.mathworks.com/matlabcentral/fileexchange/

19049-multiple-traveling-salesmen-problem-genetic-algorithm." MATLAB Central File Exchange. Retrieved May 6, 2014, 2008.

[113] J. Goldberger and S. T. Roweis, "Hierarchical clustering of a mixture model.," in *NIPS*, vol. 17, pp. 505–512, 2004.

[114] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Transactions on Graphics (TOG)*, vol. 5, no. 1, pp. 51–72, 1986.

[115] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *arXiv preprint arXiv:1608.03981*, 2016.

[116] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2862–2869, 2014.

[117] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

[118] J. Yang, Y. Zhang, and W. Yin, "An efficient tvl1 algorithm for deblurring multichannel images corrupted by impulsive noise," *SIAM Journal on Scientific Computing*, vol. 31, no. 4, pp. 2842–2865, 2009.

[119] C. Bouman and K. Sauer, "A generalized gaussian image model for edge-preserving map estimation," *IEEE Transactions on image processing*, vol. 2, no. 3, pp. 296–310, 1993.

[120] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," in *ACM transactions on graphics (TOG)*, vol. 25, pp. 787–794, ACM, 2006.

[121] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE transactions on image processing*, vol. 9, no. 9, pp. 1532–1546, 2000.

[122] M. N. Do and M. Vetterli, "Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance," *IEEE transactions on image processing*, vol. 11, no. 2, pp. 146–158, 2002.

[123] L. Boubchir and J. Fadili, "Multivariate statistical modeling of images with the curvelet transform," in *ISSPA*, pp. 747–750, 2005.

[124] H. Ji, S. Huang, Z. Shen, and Y. Xu, "Robust video restoration by joint sparse and low rank matrix approximation," *SIAM Journal on Imaging Sciences*, vol. 4, no. 4, pp. 1122–1142, 2011.

[125] T. Michaeli and M. Irani, "Blind deblurring using internal patch recurrence," in *European Conference on Computer Vision*, pp. 783–798, Springer, 2014.

[126] Q. Cheng and T. S. Huang, "Robust optimum detection of transform domain multiplicative watermarks," *IEEE Transactions on Signal Processing*, vol. 51, no. 4, pp. 906–924, 2003.

[127] S. Gazor and W. Zhang, "Speech probability distribution," *IEEE Signal Processing Letters*, vol. 10, no. 7, pp. 204–207, 2003.

[128] J. A. Dominguez-Molina, G. González-Farías, R. M. Rodríguez-Dagnino, and I. C. Monterrey, "A practical procedure to estimate the shape parameter in the generalized gaussian distribution," *technique report I-01-18_eng. pdf, available through http://www. cimat. mx/reportes/enlinea/I-01-18_eng. pdf*, vol. 1, 2003.

[129] P. H. Westerink, J. Biemond, and D. E. Boekee, "Subband coding of color images," in *Subband Image Coding*, pp. 193–227, Springer, 1991.

[130] N. Tanabe and N. Farvardin, "Subband image coding using entropy-coded quantization over noisy channels," *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 5, pp. 926–943, 1992.

[131] F. Müller, "Distribution shape of two-dimensional dct coefficients of natural images," *Electronics Letters*, vol. 29, no. 22, pp. 1935–1936, 1993.

[132] B. Aiazzi, L. Alparone, and S. Baronti, "Estimation based on entropy matching for generalized gaussian pdf modeling," *IEEE Signal Processing Letters*, vol. 6, no. 6, pp. 138–140, 1999.

[133] K. Sharifi and A. Leon-Garcia, "Estimation of shape parameter for generalized gaussian distributions in subband decompositions of video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 1, pp. 52–56, 1995.

[134] A. Achim, A. Bezerianos, and P. Tsakalides, "Novel bayesian multiscale method for speckle removal in medical ultrasound images," *IEEE transactions on medical imaging*, vol. 20, no. 8, pp. 772–783, 2001.

[135] S. Lavu, H. Choi, and R. Baraniuk, "Estimation-quantization geometry coding using normal meshes," in *Data Compression Conference, 2003. Proceedings. DCC 2003*, pp. 362–371, IEEE, 2003.

[136] K. Kokkinakis and A. K. Nandi, "Exponent parameter estimation for generalized gaussian probability density functions with application to speech modeling," *Signal Processing*, vol. 85, no. 9, pp. 1852–1858, 2005.

[137] C. Chaux, P. L. Combettes, J.-C. Pesquet, and V. R. Wajs, "A variational formulation for frame-based inverse problems," *Inverse Problems*, vol. 23, no. 4, p. 1495, 2007.

[138] H. Soury and M.-S. Alouini, "New results on the sum of two generalized gaussian random variables," in *Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on*, pp. 1017–1021, IEEE, 2015.

[139] S. Gazor and W. Zhang, "A soft voice activity detector based on a laplacian-gaussian model," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 5, pp. 498–505, 2003.

[140] F. Pascal, L. Bombrun, J.-Y. Tourneret, and Y. Berthoumieu, "Parameter estimation for multivariate generalized gaussian distributions," *IEEE Transactions on Signal Processing*, vol. 61, no. 23, pp. 5960–5971, 2013.

[141] K. A. Birney and T. R. Fischer, "On the modeling of dct and subband image data for compression," *IEEE transactions on Image Processing*, vol. 4, no. 2, pp. 186–193, 1995.

[142] O. M. M. Mohamed and M. Jaïdane-Saïdane, "Generalized gaussian mixture model," in *Signal Processing Conference, 2009 17th European*, pp. 2273–2277, IEEE, 2009.

[143] R. Krupiński, "Approximated fast estimator for the shape parameter of generalized gaussian distribution for a small sample size," *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 63, no. 2, pp. 405–411, 2015.

[144] A. A. Roenko, V. V. Lukin, I. Djurovic, and M. Simeunovic, "Estimation of parameters for generalized gaussian distribution," in *Communications, Control and Signal Processing (ISCCSP), 2014 6th International Symposium on*, pp. 376–379, IEEE, 2014.

[145] P. Mittal and K. Gupta, "An integral involving generalized function of two variables," in *Proceedings of the Indian academy of sciences-section A*, vol. 75, pp. 117–123, Springer, 1972.

[146] K. P. Peppas, "A new formula for the average bit error probability of dual-hop amplify-and-forward relaying systems over generalized shadowed fading channels," *IEEE Wireless Communications Letters*, vol. 1, no. 2, pp. 85–88, 2012.

[147] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[148] S. M. Berman, "The tail of the convolution of densities and its application to a model of HIV-latency time," *The Annals of Applied Probability*, vol. 2, no. 2, pp. 481–502, 1992.

[149] P. J. Huber, "Robust statistics," in *International Encyclopedia of Statistical Science*, pp. 1248–1251, Springer, 2011.

[150] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[151] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," in *Advances in neural information processing systems*, pp. 472–478, 2001.

[152] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[153] M. Nardon and P. Pianca, "Simulation techniques for generalized gaussian densities," *Journal of Statistical Computation and Simulation*, vol. 79, no. 11, pp. 1317–1329, 2009.

[154] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[155] A. Wintner, "Asymptotic distributions and infinite convolutions. edwards brothers," *Ann Arbor, MI. In Bickel, PJ*, 1938.

[156] S. Purkayastha, "Simple proofs of two results on convolutions of unimodal distributions," *Statistics & probability letters*, vol. 39, no. 2, pp. 97–100, 1998.