Low-level Image Priors and Laplacian Preconditioners for Applications in Computer Graphics and Computational Photography

by

Dilip Krishnan

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May, 2013

Rob Fergus

To Meghana

Acknowledgements

I thank my advisor, Rob Fergus, for teaching me computer vision and machine learning. From him, I have learnt the process of conducting and writing about research. I shall always remember his admonishments of "don't waffle" when reviewing my writing or presentations. His enthusiastic endorsements of my work and many personal kindnesses have made my PhD studies a most enjoyable experience. Thanks, Rob!

I am grateful to Rick Szeliski for mentoring me during a 2010 summer internship at Microsoft Research, which started us on a fruitful collaboration on a very interesting topic. I also thank him for being on my thesis committee and supporting my career. It has been wonderful to collaborate with Raanan Fattal. I have learnt much from him on multigrid, wavelets and numerical linear algebra. Raanan's humor and tolerance of my mistakes makes working with him even more enjoyable.

Thanks to Michael Overton for teaching me much about optimization and agreeing to write me reference letters even though I did not get good results from our work together! I thank those who have offered me technical advice and guidance for which I am grateful: Bill Freeman, Anat Levin, Yair Weiss, Denis Zorin, Yann LeCun, Chris Bregler and Fredo Durand.

Many PhD students and postdocs made the 12th floor of 715 Broadway a fun place to be. I have also learnt much from discussions with them. Thanks to: Koray Kavukcuoglu, Karol Gregor, David Eigen, Joan Bruna, Matt Zeiler, Nathan Silberman, Arthur Szlam, Clement Farabet and Shravan Veerapaneni. I thank my parents and my wife's parents for their support and encouragement, and Eka for bringing me immense joy in the last two years.

I dedicate this thesis to my dear wife, Meghana, for her love, understanding and many sacrifices in support of my studies.

Abstract

In the first part of this thesis, we develop novel image priors and efficient algorithms for image denoising and deconvolution applications. Our priors and algorithms enable fast, high-quality restoration of images corrupted by noise or blur. In the second part, we develop effective preconditioners for Laplacian matrices. Such matrices arise in a number of computer graphics and computational photography problems such as image colorization, tone mapping and geodesic distance computation on 3D meshes.

The first prior we develop is a spectral prior that models correlations between different spectral bands. We introduce a prototype camera and flash system, used in conjunction with the spectral prior, to enable taking photographs at very low light levels. Our second prior is a sparsity-based measure for blind image deconvolution. This prior gives lower costs to sharp images than blurred ones, enabling the use simple and efficient Maximum a-Posteriori algorithms.

We develop a new algorithm for the non-blind deconvolution problem. This enables extremely fast deconvolution of images blurred by a known blur kernel. Our algorithm uses Fast Fourier Transforms and Lookup Tables to achieve real-time deconvolution performance with non convex gradient-based priors. Finally, for certain image restoration problems with no clear formation model, we demonstrate how learning a direct mapping between original/corrupted patch pairs enables effective restoration.

We develop multi-level preconditioners to solve discrete Poisson equations. Existing multilevel preconditioners have two major drawbacks: excessive bandwidth growth at coarse levels; and the inability to adapt to problems with highly varying coefficients. Our approach tackles both these problems by introducing sparsification and compensation steps at each level. We interleave the selection of fine and coarse-level variables with the removal of weak connections between potential fine-level variables (sparsification) and compensate for these changes by strengthening nearby connections. By applying these operations before each elimination step and repeating the procedure recursively on the resulting smaller systems, we obtain highly efficient schemes. The construction is linear in time and memory. Numerical experiments demonstrate that our new schemes outperform state of the art methods, both in terms of operation count and wall-clock time, over a range of 2D and 3D problems.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Summary of contributions 5

2 Literature Survey **7**

2.1 Inverse problems 7

2.2 Image priors 9

2.3 Image denoising 12

2.4 Multiple images for denoising and other applications 14

2.5 Non-blind deconvolution 15

2.6 Blind deconvolution 19

2.7 Localized corruption removal 22

2.8 Preconditioners and solvers for Laplacian matrices 24

3 Dark Flash Photography **30**

3.1 Introduction 30

3.2 Related work 31

3.3 Dark flash hardware 33

3.4 Dark flash processing 37

3.4.1 Spectral constraints 37

3.4.2 Spatial-spectral cost function 38

3.4.3 Pre & post-processing 39

3.5 Results 40

3.5.1 Comparison experiments 40

3.5.2 Fluorescence 41

3.5.3 Photometric flash measurements 42

3.6 Other applications 42

3.6.1 Estimation of spectral reflectance 42

3.6.2 Color-band denoising 43

3.7 Discussion 43

4 Fast Image Deconvolution Using Hyper-Laplacian Priors **56**

4.1 Introduction 56

4.2 Algorithm 58

4.2.1 **x** sub-problem . . . 59
4.2.2 **w** sub-problem . . . 60
4.2.3 Summary of algorithm . . . 62
4.3 Results . . . 63
4.4 Discussion . . . 64

5 Blind Deconvolution Using a Normalized Sparsity Measure 68
5.1 Introduction . . . 68
5.2 Motivation . . . 69
5.3 Approach . . . 71
5.3.1 Blind Kernel Estimation . . . 72
5.3.2 Image recovery . . . 74
5.3.3 Speed and robustness . . . 75
5.3.4 Extension to in-plane rotation . . . 75
5.3.5 Extension to 3-D rotations . . . 76
5.4 Experiments . . . 76
5.4.1 Spatially invariant kernel . . . 76
5.4.2 In-Plane rotation . . . 78
5.4.3 3-D Rotation . . . 79
5.5 Connections with the blind equalization literature . . . 79
5.6 Discussion . . . 85

6 Removing Localized Corruption from Natural Images 87
6.1 Introduction . . . 87
6.2 Approach . . . 88
6.2.1 Gaussian mixture model . . . 89
6.2.2 Joint sparse coding . . . 89
6.2.3 Neural network . . . 90
6.2.4 Mean-Covariance RBM . . . 90
6.3 Datasets . . . 92
6.3.1 Synthetic corruption . . . 92
6.3.2 Water droplets dataset . . . 93
6.4 Results . . . 93
6.4.1 Water droplet removal . . . 94
6.5 Discussion . . . 95

7 Efficient Preconditioning of Laplacian Matrices for Computer Graphics 98
7.1 Introduction . . . 98
7.2 Mathematical background . . . 101
7.2.1 Laplacian matrices . . . 101
7.2.2 Hierarchical preconditioning . . . 103
7.3 Sparsification and coloring . . . 104
7.3.1 Matrix sparsification for the HSC preconditioner . . . 106

7.3.2 Compensation for ABF and HSC . . . 107
7.3.3 Coloring algorithm . . . 112
7.3.4 Updating the HSC preconditioner for diagonal shifts . . . 116
7.3.5 Efficient multilevel eigensolver . . . 116
7.4 Results . . . 118
7.4.1 2D Problems . . . 120
7.4.2 3D Meshes . . . 121
7.5 Discussion . . . 122

8 Conclusions **126**

Appendices **128**
A Coarse Level matrices are Laplacian . . . 128
B Bounds on Energy Deviation after Sparisification . . . 129
C Characterization of Sparsified Spaces and Compensation . . . 130

List of Figures

1.1 Noise at Low Light Levels . . . 2
1.2 Many solutions to the blind deconvolution problem . . . 3
1.3 Raindrop image . . . 4
1.4 Problem which gives rise to a Laplacians . . . 4

2.1 Spectral Magnitude of derivative filters . . . 10
2.2 2D FFT of Derivative Filter and Morlet Wavelet . . . 11
2.3 Fields of Experts . . . 11
2.4 KSVD Dictionary Elements . . . 12
2.5 BM3D Result . . . 13
2.6 Flash/no-flash processing . . . 15
2.7 Handling of Saturated Areas in Non-blind Deconvolution . . . 18
2.8 Spatially invariant blind deconvolution . . . 19
2.9 Inverted costs under standard priors . . . 20
2.10 Cho-Lee deblurring workflow . . . 21
2.11 Power spectrum of natural images . . . 22
2.12 Snow removal . . . 23
2.13 Thin occluder removal . . . 24
2.14 Graph Laplacian Correspondence . . . 25
2.15 Half-Octave Coarsening . . . 27

3.1 Illustration of dark flash system . . . 32
3.2 Camera/Flash spectrum . . . 35
3.3 Flash brightness . . . 36
3.4 $I^f(\lambda)$ and $H(\lambda)$, see text for details. . . . 36
3.5 Motivation of spectral prior . . . 38
3.6 Denoising of faces - 1 of 2 . . . 45
3.7 Denoising of faces - 2 of 2 . . . 46
3.8 Denoising of scenes . . . 47
3.9 Denoising of scenes . . . 48
3.10 Denoising of person . . . 49
3.11 Comparison with Sony NightShot . . . 50
3.12 Comparison with Sony NightShot . . . 51
3.13 Need for UV and IR illumination . . . 52
3.14 Comparison with other methods . . . 53

3.15 Varying α . . . 54
3.16 Estimating spectral reflectance . . . 54
3.17 Comparison with other techniques . . . 55
3.18 Candelight scene denoising . . . 55

4.1 Heavy-tailed Distributions . . . 58
4.2 Crops from Two Images . . . 65

5.1 Motivation for our new prior . . . 70
5.2 A visualization of ℓ_1 , ℓ_1/ℓ_2 and ℓ_0 functions . . . 71
5.3 Cumulative histograms . . . 77
5.4 Recovery of 27×27 kernel . . . 78
5.5 . . . 79
5.6 Recovery of a real-world kernel . . . 80
5.7 Recovery of a real-world kernel . . . 80
5.8 In-plane rotational deblurring . . . 81
5.9 3D rotational deblurring . . . 81
5.10 Blind Equalization Model . . . 82
5.11 Gaussian, Sub-Gaussian and Super-Gaussian Signals . . . 82
5.12 Blurring increases kurtosis of a sub-Gaussian signal . . . 83
5.13 Blurring decreases kurtosis of a super-Gaussian signal . . . 84
5.14 Illustration of Shalvi-Weinstein algorithm . . . 84
5.15 Proof of optimality . . . 84
5.16 Sensitivity of Kurtosis measure . . . 86

6.1 Examples of synthetic corruptions . . . 93
6.2 Comparison of approaches on an image corrupted with synthetic "snow". . . . 95
6.3 Two test images corrupted by real rain drops . . . 97

7.1 Two-dimensional discrete Poisson problems. . . . 99
7.2 Geodesic Distance Computation . . . 99
7.3 Mesh Segmentation . . . 100
7.4 Effect of Sparsification . . . 105
7.5 ABF Sparsification . . . 106
7.6 Sparsification and Compensation in a Triangle . . . 109
7.7 Modeling Fourier Modes . . . 110
7.8 Scheme Progression . . . 113
7.9 Mesh Segmentation using Spectral Embedding . . . 117
7.10 Independence to System Size . . . 119
7.11 Performance comparison for 2D problems with varying degrees of homogenity . . 124
7.12 Example of mesh smoothing . . . 125

List of Tables

4.1 SNR and Running Time Comparisons 64
4.2 Run-time Comparisons 64
4.3 SNR and Runtime Comparisons 66

6.1 Mean PSNR over 10 test images 94
6.2 Log progabilities 96

7.1 Effect of compensation on condition numbers 112
7.2 Total wall-clock time taken for 2D grids 120
7.3 Condition numbers achieved by the solvers 120
7.4 Wall clock time to compute the lowest 3 eigenvectors 122
7.5 Wall clock time to compute geodesic distances 122
7.6 Wall clock time to smooth noisy meshes 123

Chapter 1

Introduction

1.1 Motivation

The first part of this thesis considers a range of inverse problems. Inverse problems in low-level computer vision are almost always ill-posed. Examples of such problems are image denoising, blind deconvolution, and superresolution. Non-parametric methods tackle the ill-posed nature of inverse problems by the use of more data, but a non-parametric approach is often not possible. In a parametric model, the inverse problem can be made well-posed by the use of *image priors*. These priors quantify important properties of the latent uncorrupted image. From a practical perspective, priors must be efficient to use on megapixel images.

When images are captured using a camera, short exposure times are needed to avoid motion blur. This leads to high noise at low light levels; an example is given in Figure 1.1. The problem of image denoising has probably received the most attention of any low-level vision problem. This is for good reason: noise is visually annoying, and causes problems for other algorithms such as depth estimation, image segmentation or object recognition. Single-image non-parametric denoising methods have reached close to optimal performance, as shown recently in [98]. These performance levels are, however, still not satisfactory for extremely noisy images resulting from low-light photography. The authors of [98] suggest that further improvements require the use of parametric methods.

Image blur results from the opposite situation to image noise: exposure times are long. Image blur has a variety of causes. Camera shake during the exposure, or moving objects in the scene, all result in a blurred image. In astronomical imaging from ground-based telescopes, blur occurs due to atmospheric turbulence. The atmosphere acts as a low-pass filter, thereby causing point light sources to appear as blobs. In fluorescence microscopy, imaged objects may be out of focus due to tiny movements, and often appear blurred. Image deblurring is therefore an important and difficult problem.

At each pixel in the blurred image, the blur is a result of convolution of the image with a *blur*

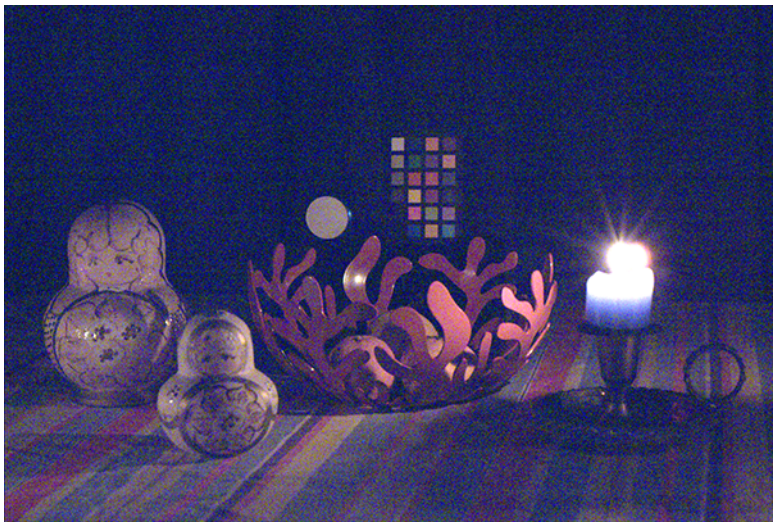


Figure 1.1: An image taken with illumination from a a candlelight. High noise levels are clearly visible, especially in the blue channel. Even state-of-the-art denoising methods do not perform well at such low signal-to-noise ratios.

kernel. This kernel may be spatially varying or spatially invariant. Spatially invariant blur is due to camera translation. Spatially varying blur has many causes, including camera rotation, and object motion. The blurred image is expressed as $y = K(x)$, where K is a spatially variant operator. In the spatially invariant case, K is a matrix (linear operator) with shifted versions of the same blur kernel in each row. In this case, the blur formation model is written as: $y = x \oplus k$, where x is the sharp image, k is the blur kernel and y is the observed blurred image. When both x and k are unknown, the problem is called *blind* deconvolution. If k is known and x is unknown, it is *non-blind* deconvolution.

For the non-blind deconvolution problem, successful low-level priors have been developed. By "low-level", we mean that the priors are simple functions of the image pixel values. These priors usually take the form of heavy-tailed distributions on the marginal statistics of the uncorrupted images. These statistics are derived by convolving the images with a set of fixed or learned filters and give good quality results. However, the priors are often non-convex and numerically expensive to use in a minimization framework. The question then becomes one of speed: how can we deconvolve megapixel images fast, but without losing output quality?

The blind deconvolution problem is considerably harder, because the number of unknowns exceeds the number of known variables (when only a single image is provided). Figure 1.2 shows that the same blurred image can be explained as the convolution of many different image and kernel combinations. High-quality results therefore require priors that are well-tuned to the specific problem of deblurring. It is now well known [101] that using a simple Maximum a-Posteriori (MAP) framework with the same priors effective for non-blind deconvolution lead to trivial solutions in blind deconvolution. In these trivial solutions, the output image is the corrupted input and the blur kernel is the "delta" kernel (the first row in Figure 1.2).

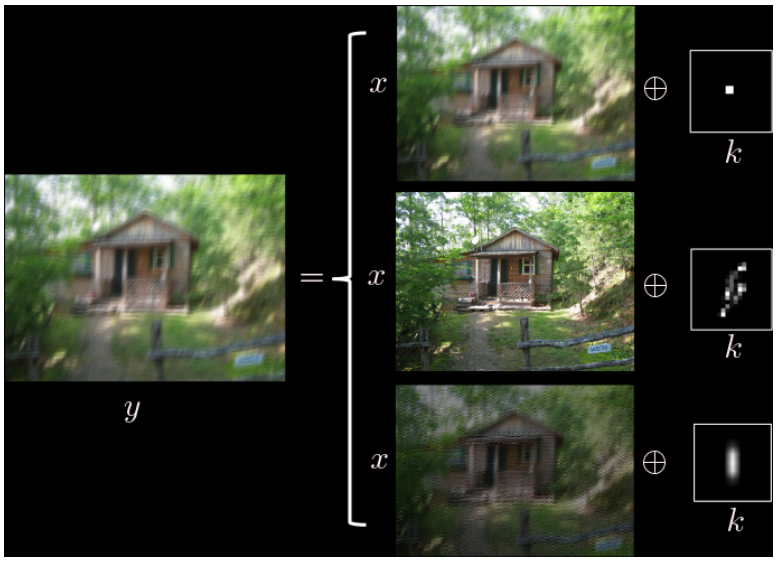


Figure 1.2: The blind deconvolution problem is ill-posed. The observed blurred image y can be explained as the convolution of many different sharp images x with kernels k. To get good deconvolution results, priors are required which prefer the true sharp solution (in this case, the middle row).

For denoising and deconvolution problems, the formation model is well understood: additive (or multiplicative) noise in the former case, and convolution with a blur kernel in the latter. For many other types of corruption, it is very difficult to develop a corruption model. Such corruptions often arise due to natural phenomena such as rain, dust/dirt or fog. For example, imagine taking a photograph through a window with rain drops or dust on it ; an example is given in Figure 1.3. Every pixel in the image has a certain probability of being corrupted. The formation model needs to account for both the probability and the intensity of corruption at every pixel. It is unclear how one may formulate a tractable model of the corruption process. Such restoration problems are of significant practical interest and not much progress has been made in solving them.

In the second part of this thesis, we develop preconditioners for the efficient solution of Laplacian linear systems. Discrete elliptic partial differential equations (PDEs) arise in many computer graphics and vision problems. An example is shown in Figure 1.4. The solutions to these problems usually involve a large linear system of equations. The common factor in these linear systems is the presence of *Laplacian* matrices. Laplacians have a one-to-one association with weighted graphs defined on a domain [146]. In the case of 2D images, the vertices of the underlying graph are the image pixels and the graph edge weights are a function of image intensities. In the case of 3D surfaces, the graph structure is associated with the finite element discretization of the domain.

Laplacians measure the smoothness of a vector x over the 2D or 3D surface. The large size of 2D images and 3D meshes (usually in the millions of vertices) give rise to large Laplacian matrices. Efficient solvers are required to solve these linear systems in reasonable time and

Figure 1.3: An image taken through a window with raindrops. The raindrops are random in size, color and position; it is hard to specify a formation model.

Figure 1.4: Image Colorization (figure taken from [97]): the user scribbles some patterns on a gray-scale image (left). These scribbles are propagated through the entire image while respecting the gradients (right). Finding the colorized image requires solving a Laplacian linear system.

memory requirements. Direct solvers are robust and give machine-precision accurate solutions. However, they tend to be memory intensive and are complex to port to different environments, e.g. Graphics Processing Units (GPU). Furthermore, the machine-precision accuracy of direct solvers is not necessary for most problems in graphics. This opens the door to significant efficiency gains using iterative solvers such as Preconditioned Conjugate Gradient (PCG). Our focus is on developing efficient and practical preconditioners for Laplacians arising from a number of 2D and 3D problems.

1.2 Summary of contributions

We summarize the main contributions of this thesis, addressing the problems outlined in the previous section. Chapter 2 presents a literature survey and Chapter 8 presents conclusions and directions for future work.

1. **We present a novel mechanism for low-light photography (Chapter 3).**
   A new approach to the specific problem of taking photographs in low light conditions is presented. Long exposures lead to image blur, but short exposures lead to noise. The use of a flash allows short exposures without noise, but it dazzles people who are in front of the camera and changes the illumination. We develop a prototype camera and flash that uses near infra-red and near ultra-violet light to provide illumination. The resulting flash is 200 times dimmer than a conventional camera flash. A pair of images is captured, one with and one without the flash. The non-flash image is then denoised taking into account the spectral correlations between the pair of images. For this purpose, we develop a novel spectral image prior. The resulting algorithm is able to effectively denoise images captured at light levels close to 1 Lux.

2. **We develop an algorithm for non-blind deconvolution, which is significantly faster than state of the art methods without loss of quality (Chapter 4).**
   It is well known that the distribution of image gradients follow heavy-tailed distributions. These distributions are very effective priors for non-blind deconvolution. However, due to their non-convexity, the resulting optimization problems are computationally expensive to solve. We present a simple and effective operator splitting scheme which enables the use of Fast Fourier Transforms (FFT) and Look Up Tables (LUT). Our resulting algorithm is orders of magnitude faster than existing methods, without any loss of quality.

3. **We propose a new prior for blind deconvolution which overcomes a key problem with existing priors (Chapter 5).**
   The heavy-tailed priors used in non-blind deconvolution are commonly used for blind deconvolution as well. It is observed that using these priors leads to a trivial solution. Instead, we develop a novel prior based on the ratio of the l_1 and l_2 norms of image gradients. We show that this prior overcomes the problem of trivial solutions, and enables the use of much more efficient algorithms without loss of quality. Our prior has interesting connections with the blind equalization literature in digital communications. We examine similarities and differences between our approach and a classical blind equalization method.

4. **We study methods to remove localized corruptions from images (Chapter 6).**
   We present a learning-based approach to the restoration of images corrupted by raindrops or dust. Developing formation models for such problems seems difficult. Instead, we use a training set consisting of pairs of clean and corrupted patches to learn mappings from the corrupted patches to the clean patches. We experiment with two techniques to learn the mappings: a conditional Gaussian Mixture Model (GMM); and a joint GMM. The joint GMM gives promising reconstruction results.

5. **We develop efficient preconditioners for Laplacian matrices arising from a range of computer graphics problems (Chapter 7).**
   We present two new multi-level preconditioning schemes which draw on ideas from the multigrid and combinatorial preconditioning literature. Our preconditioners can be constructed in linear time and memory. They achieve excellent condition numbers and wall-clock time performance on a wide range of Laplacians. We also develop an efficient multi-scale eigensolver based on one of these preconditioners. We compare our resulting linear system solvers and eigensolver to a number of state of the art iterative and direct solvers. In all cases considered, we equal or outperform other solvers. Our solvers are simple to implement on parallel platforms such as GPUs.

Chapter 2

Literature Survey

Extensive research into low-level image priors for inverse problems has been conducted in both the image processing and computer vision literature. In this chapter, we review some of the most relevant work. We also review the literature on image denoising and deconvolution (both blind and non-blind).

Preconditioners for matrices arising in elliptic PDE problems have also been studied extensively. Most of these preconditioners are developed under the umbrella of multigrid or hierarchical preconditioners. Recently, the study of linear solvers for Laplacian systems has seen fundamental theoretical breakthroughs in the linear-time construction of preconditioners. A different approach to solving PDE's is by transforming them into integral equations, and solving the integral equation using Fast Multipole Methods. These methods are outside the scope of this thesis and will not be considered.

2.1 Inverse problems

Most parametric formulations of inverse problems explain the corrupted image y as the output of a corruption process A on the latent image x. With the addition of white Gaussian noise, this process can be written as:

$$y = Ax + n \tag{2.1}$$

The operator A can explain image blurring [55], noise [129], image downsampling [182] and other processes. A may be a nonlinear function instead of a matrix [8]. Given the observation y, the inverse problem is the recovery of a solution $\hat{x}$ which is close to x, usually in terms of mean square error. When A is known, the problem of recovering x is called non-blind; when A is unknown and must also be estimated, the problem is called blind.

A probabilistic model of the inverse recovery process may be formulated using Bayes' rule. In

the blind setting, using Bayes' theorem [175] gives us:

$$p(x, A|y) \propto p(y|x, A)p(x)p(A) \tag{2.2}$$

where we assume that x and A are independent of each other; and for a variable x, $p(x)$ refers to the probability of x. Then $p(x)$ and $p(A)$ are the *prior distributions* on the image and the corruption process, respectively. $p(y|x, A)$ is called the *likelihood*. In the non-blind setting, since y and A are given, we are interested in estimating x, using the following relationship:

$$p(x|y, A) \propto p(y|x, A)p(x) \tag{2.3}$$

Hence, finding a good approximation to $p(x)$ is of great importance in both blind and non-blind settings. $p(A)$ is of importance in some problems such as spatially varying blind deconvolution with object motion where the dimensionality of A may be of the same order as x. In this thesis, we do not consider such problems and assume very simple parametric forms for $p(A)$ (such as uniform distributions). The distributions $p(x, A|y)$ and $p(x|y, A)$ are called *posterior* distributions.

Once the posterior distribution is specified, the solution to the inverse problem is commonly determined in one of two ways: Maximum a-Posteriori (MAP) or Variational approximation. In the MAP framework, we look for the values of the unknowns that correspond to modes of the distribution (i.e. the most probable values of the unknowns). Therefore, the corresponding optimization problems for the blind and non-blind problems are, respectively:

$$(\hat{x}, \hat{A}) = \arg\max_{x,A} p(x, A|y) \quad \text{or} \tag{2.4}$$

$$\hat{x} = \arg\max_{x} p(x|y, A) \tag{2.5}$$

One may directly attempt to find the MAP solutions using the probabilistic form. An equivalent approach is to transform the probability maximization problem into an energy minimization problem, usually because the distributions are written in the form of products of exponentials. Then, taking a logarithm of the distribution gives the minimization problems:

$$(\hat{x}, \hat{A}) = \arg\min_{x,A} -\log p(x, A|y) = \arg\min_{x,A} -\log p(y|x, A) - \log p(x) - \log p(A) \quad \text{or} \tag{2.6}$$

$$\hat{x} = \arg\min_{x} -\log p(x|y, A) = \arg\min_{x} -\log p(y|x, A) - \log p(x) \tag{2.7}$$

In the variational approach to solving the inverse problem, the solutions are found as an approximation to the expected value of the unknowns under the given distribution. The expected solution to Eq. 2.3 is given by:

$$\hat{x} = \int x \; p(x|y, A)dx \tag{2.8}$$

In practice, $p(x|y, A)$ is usually intractable and so the variational approach approximates $p(x|y, A)$ with a tractable distribution $q(x|y, A)$. This approximation is usually determined by minimizing the KL divergence between $p(x|y, A)$ and $q(x|y, A)$ [113]. The variational approach has the

advantage of taking into account uncertainties associated with different solutions. Thus, a mode of the distribution with relatively low density does not necessarily have preference over a solution in a region with high probability density.

Stochastic approximation methods such as Markov Chain Monte Carlo [121] attempt to evaluate the integrals by drawing samples from the true posterior distribution. On the other hand, parametric approximation methods [113] approximate the true posterior distribution with other, more tractable distributions. The computational effort is spent in computing the parameters of the approximation.

The MAP and variational methods are the most common techniques to solving inverse problems in vision. But there are other techniques for solving parametric inverse problems which are problem-specific. An example are the spectral methods in blind deconvolution Section 2.6.

2.2 Image priors

Image priors quantify certain properties of natural images. *Low-level* priors are functions of the pixels in an image. Usually they are non-linear and non-convex.

Derivative-based priors. Whether the MAP or variational methods are used, image priors $p(x)$ are necessary to break the ill-posed nature of the inverse problem. Most priors have been constructed based on trying to fit the distributions of certain statistics of natural images. One of the earliest examples of such a statistic was that the amplitude of the power spectrum of an image x followed a power law distribution [57]: $|X(\omega)|^2 \propto 1/\omega^2$, where ω is the frequency and $X(\omega)$ is the Fourier transform of x. This is a phenomenon that is generally observed in many natural images. In the spatial domain, this can be approximated by using a Gaussian prior on derivative filters [13, 185, 116]: $-\log p(x) \propto \|x \oplus f\|^2$ where f is a derivative filter. Unfortunately, these Gaussian priors are insensitive to phase shifts in x (due to the use of the modulus sign). In practice, this means simple Gaussian priors are not very robust in inverse problems.

A natural extension of Gaussian priors is to use a non-Gaussian function. One of the first non-Gaussian, and probably the most famous prior, is the TV norm of Rudin et al. [137]. Here the exponent of 2 is replaced with 1, to give $-\log p(x) \propto \|x \oplus f\|_1$. The motivation of Rudin et al. was the minimization of the total variation (TV) in an image, which means that piecewise constant, rather than oscillatory, signals are preferred in x. This is a reasonable description of many natural images, especially urban and indoor scenes. This prior indeed performs much better than the Gaussian prior, especially for image denoising and non-blind deconvolution problems. It is also convex, which can be exploited to develop efficient algorithms. Some recent efficient algorithms based on the Bregman distance have been proposed [183, 64], that allow fast solution to inverse problems using TV priors.

Using an exponent even lower than 1 leads to hyper-Laplacian priors. Unlike the TV norm, these priors were motivated by the distribution of the gradients of many natural images [56]. An example is shown in Figure 4.1. The use of an exponent less than 1 leads to non convex

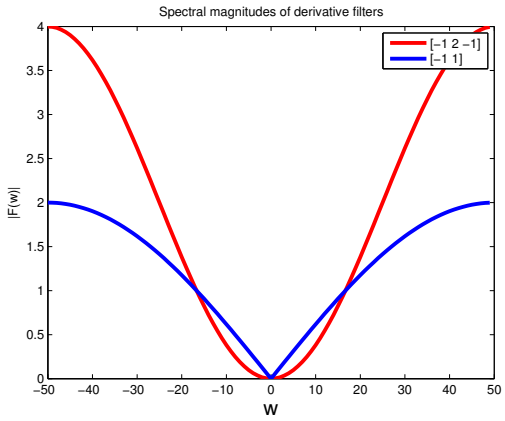


Figure 2.1: Spectral magnitude of derivative filters for image of size 512×512: derivative filters have very broad spectral signatures. Hence their use as image priors contains statistics from all scales and orientations of the image.

pseudo-norm priors of the form $-\log p(x) = \|x \oplus f\|^{\alpha}$ which are significantly harder to optimize than the convex Gaussian or TV norm priors. Hyper-Laplacian image priors have been used in a range of settings: super-resolution [155], transparency separation [100] and motion deblurring [95]. The exponent used is usually in the range of 0.5 to 0.8.

Wavelet-based priors. The frequency spectrum of a derivative filter is broad, as seen in Figure 2.1. This means that statistics at all scales and orientations are combined in derivative-based priors. Wavelets and associated constructions such as curvelets have, by design, much better frequency and spatial localization properties. In Figure 2.2, we show the Fourier spectrum of a derivative filter compared to that of a Morlet wavelet for an image of size 512×512. These priors have been popular with the applied mathematics and signal processing communities [149, 24]. These transforms can provide a sparser representations of images than derivative filters, especially for highly textured images. Sparsity here refers to the number of coefficients of the transformed image which are close to zero. The resulting priors are of the form $-\log p(x) = \|\Psi x\|_1$, where Ψx is the wavelet/curvelet transform of x. Curvelets have also been used as priors for motion blur kernels [22], owing to the assumption that many motion blur kernels are directional and spatially sparse i.e. $-\log p(k) = \|\Psi k\|$, where k is a blur kernel and Ψ is a curvelet transform. A recent interesting paper adapts the non-parametric image denoising algorithm BM3D [39] into a parametric prior that is wavelet-like in it's construction [41].

Learnt filters. The priors discussed so far are fixed filters based on certain statistical or mathematical properties of images. Learning-based approaches try to learn filters from large databases of natural image patches. Zhu and Mumford [189] learn arbitrary energy functions for a set of oriented derivative filters via Gibbs sampling in a maximum likelihood approach. Roth and Black [136] introduce the Fields of Experts model that employs student-T potential functions and learn

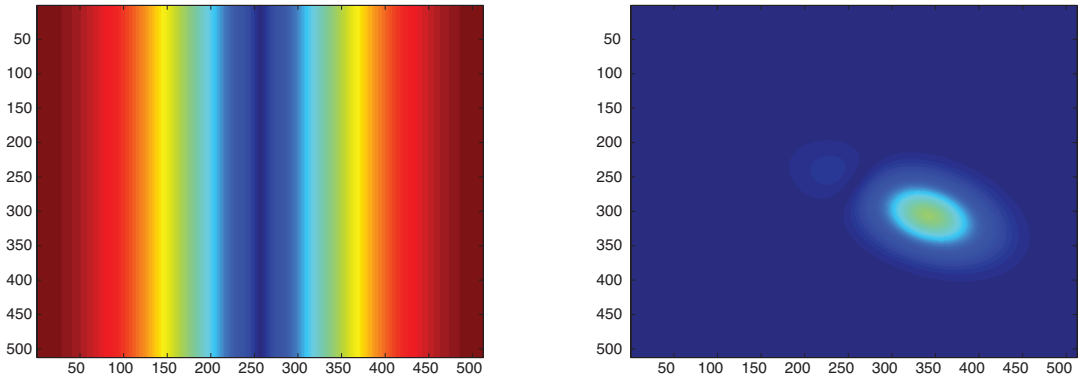


Figure 2.2: The frequency magnitude spectrum of a horizontal derivative filter (left) and a wavelet (right). Wavelets have much better localization in frequency. This means that they can discriminate better between different types of signals such as textures and piecewise smooth signals.

Figure 2.3: From Roth and Black [136]: Examples of filters learnt using the Fields of Experts model.

the filters using contrastive divergence (Figure 2.3). In this case the prior is given by

$$-\log p(x) = \sum_{i \in patches} \sum_{j \in filters} |x \oplus f_j|_i \tag{2.9}$$

Weiss and Freeman [169] propose a simpler learning scheme for the Fields of Experts model that allows the efficient training of large filters. Raj and Zabih [131] propose a discrete Markov random field based smoothness prior that can efficiently be minimized using graph cuts. Zoran and Weiss [191] have recently introduced a model that uses Mixtures of Gaussians over image patches:

$$-\log p(x) = \sum_{i \in patches} \log \sum_{k \in comp.} \pi_k \mathcal{N}(P_i \mathbf{x} | \mu_\mathbf{k}, \mathbf{\Sigma_k}) \tag{2.10}$$

The parameters of the model are learnt by training over a large set of 2 million image patches. Since $-\log p(x)$ does not have a simple form, they introduce approximations to enable inference of the MAP solution in applications.

Dictionaries. In the Fields of Experts and other filter-based models, sparsity is enforced over the response of each filter learnt by the model. These filters do not resemble image patches. Another related approach to learning priors over image patches has been the use of image dictionaries. The idea here is to represent an image patch directly as a sparse linear combination of canonical patches. The canonical patches are overcomplete representations known as dictionar-

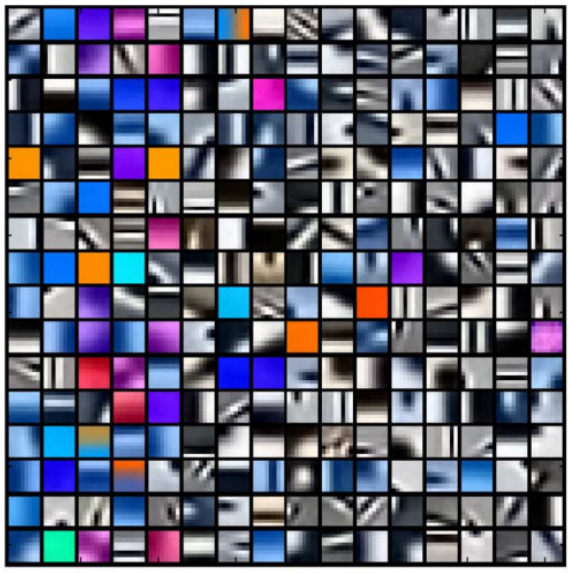


Figure 2.4: From Mairal et al. [110]: Image patches learnt from the color KSVD model. The dictionary elements are RGB patches of size 8×8.

ies. An example is the KSVD algorithm of [47, 110]. Examples of dictionary elements learnt by KSVD are shown in Figure 2.4. Dictionary-based methods do not lend themselves to a probabilistic interpretation since the representation of a patch depends on solving a sparsity inducing minimization problem.

Modern images are almost always multi-spectral (usually RGB). Exploiting inter-spectral correlations can provide strong cues to develop priors. Recently Chakrabarti and Zickler [26] have studied the statistical properties of hyper spectral images for natural world scenes. They also propose a basis in which hyperspectral image patches may be efficiently represented. It may be possible to extend such representations into useful multi-spectral image priors.

An implicit assumption in constructing priors been that the distributions of corrupted images are significantly different from that of uncorrupted images. In other words, the probability distribution generated by $p(x)$ gives a higher probability to clean images than to corrupted images. In the case of additive Gaussian noise, this seems to hold empirically, although the priors were not designed with this explicit goal in mind. But as we shall see in Chapter 5, this is not so for image blur and new priors are required.

2.3 Image denoising

Image denoising has been intensively investigated for many decades. Significant advances have been made in the last 10 years. We present a brief literature survey of some of the state-of-the-art methods. Image denoising corresponds to a corruption process $A = I$ in Eq. 2.1. Broadly speaking, denoising algorithms can be divided into parametric and non-parametric methods.

Figure 2.5: Image denoising example achieved using BM3D [39]. The original image on the left has noise of standard deviation 30 added to give the middle image (PSNR 18.6). The denoised image is on the right (PSNR 31.3).

Parametric methods work by introducing image prior and likelihood terms, and maximizing the posterior distribution. Non-parametric methods rely on a training set of patches which give an empirical estimate of the posterior. The most influential and effective non-parametric methods are the non-local means [21] and BM3D [39]. In the non-local means method, a patch is denoised by using weighted averaging of a set of patches that are similar to the source patch in Euclidean distance and intensity. The BM3D denoising also works by grouping together similar patches. However, instead of averaging in the spatial domain, a group of patches is transformed using a specially developed shape-adaptive Discrete Cosine Transform (DCT). Then thresholding is applied to the resulting coefficients. Inverting the thresholded coefficients gives a clean sample for a group of pixels. Multiple samples are averaged over multiple neighborhoods to give the final denoised result. Figure 2.5 gives an example of a noisy image and denoising achieved with BM3D.

In the computer graphics community, bilateral filtering [159, 125] has proved quite popular. This is a non-parametric non-local method which is based on very similar principles as non-local means. It is used in both 2D image denoising and 3D mesh denoising [58]. The idea of the bilateral filter is to replace a noisy pixel with a weighted average of a number of other pixels. These other pixels are weighted based on their geometric distance to the noisy pixel, and their gray-level distance. The weights are based on two Gaussians with variances that are user-defined. The non-local means method can be considered a generalization of the bilateral filter where neighborhoods around pixels are used to compute distances (instead of distances over single pixels). This makes the distance computation more robust (at the expense of greater computation). The cross-bilateral filter is a generalization of the bilateral filter where two images are used, and the weights corresponding to gray-level distances are computed on the second image (where they are more reliable due to lack of noise).

Parametric methods rely on image prior models to achieve effective denoising. The KSVD algorithm [110], the GMM model of Zoran and Weiss [191] and the Gaussian Scale Mixtures (GSM) Model of [129] are state-of-the-art parametric denoising methods. As explained in Section 2.2,

the KSVD method uses a dictionary-based image prior. The Gaussian Scale Mixture (GSM) model of Portilla et al. [129] uses a prior based on a wavelet decomposition of the image along with the statistical dependence of nearby coefficients in the decomposition. The denoising process uses this prior in a Bayesian framework to estimate the most likely denoised coefficient from the set of noisy wavelet coefficients (taking into account statistical dependence between coefficients). The GMM model of Zoran and Weiss [191] uses a Mixture of Gaussians, giving results as good as BM3D. This model (called EPLL) has a further advantage that it may be used for problems other than denoising, such as non-blind deconvolution.

In two recent papers, Levin et al. [98, 99] have shown that single-image denoising results obtained by current state-of-the-art algorithms such as BM3D and EPLL are close to optimal for most patch sizes and noise levels. To overcome this limit therefore requires the use of more than one image with statistically different properties. A number of recent papers have taken this approach for image denoising and other applications, and we review this literature in the next section.

2.4 Multiple images for denoising and other applications

When multiple images are captured at different spectral bands or under varying illumination, the statistics images vary between bands. However, there are strong correlations and dependencies, such as the presence of edges at the same spatial location. This allows denoising to potentially be more effective than using one image. In the flash/no-flash technique ([1, 128, 46]), two images are captured: one without flash and the other with a standard camera flash. Agrawal et al. [1] focused on the removal of artifacts such as flash hotspots or self-reflections. An example is shown in Figure 2.6. Petschnigg et al. [128] use the flash/no-flash pair to denoise the ambient image using the edges in the noise-free flash image as a guide for denoising the noisy ambient. Eisemann et al. [46] combine the flash and no-flash images to to give a new image with the ambience of the no-flash and the color information from the flash image. Both these approaches are similar in that they use a cross-bilateral filter [125] and detail transfer.

Bennett et al. [11], show how video captured in low-light conditions can be denoised using continuous IR illumination. They make use of temporal smoothing to achieve high quality results. Wang et al. [166] show how IR illumination can be used to relight faces in well-lit scenes. Both these works rely on cross-bilateral filtering to combine the IR and visible signals.

Yuan [187] use a pair of images, one noisy and one blurred, to provide better deblurring performance. Their idea is to use the (noisy) edges from the noisy image to stabilize the blind deconvolution process. However, they assume perfect registration between the pair of images, so any object motion would cause problems for their setup. [156] go even further, capturing a stack of noisy images. However, clearly the image registration problem is even more pronounced.

Tai et al. [154] capture two streams of video and use a high-frame rate, low-resolution video to deblur a low-frame rate, high-resolution video. They use optical flow between the high-frame rate video frames to determine motion blur. Their setup requires the use of an optical bench to

Figure 2.6: Self-reflection removal from am ambient image using a flash/no-flash pair (image taken from[1]).

ensure the images to the two cameras are aligned.

2.5 Non-blind deconvolution

In this section, we review existing literature on non-blind deconvolution in the spatially invariant and spatially varying settings. The classical non-blind deconvolution algorithms are Richardson-Lucy [109, 134] and Wiener deconvolution [174]. Richardson-Lucy is a iterative non-blind algorithm that finds a maximum likelihood solution under the assumption of multiplicative (Poisson) noise. There is no image prior assumed in the original version of this algorithm. Due to this, it is well-known that the classical Richardson-Lucy can lead to artifacts due to noise amplification if iterated for too long. Iterations are terminated early to avoid this behavior. Wiener deconvolution finds a maximum a-posteriori solution under the assumption of Gaussian priors and additive white Gaussian noise. Due to the use of quadratic likelihood term and Gaussian priors, Wiener deconvolution can be efficiently implemented in the Fourier domain. This is simple to see: the posterior term in Eq. 2.3 is written as a product of the likelihood and prior terms. Under the assumption of additive Gaussian noise with variance η^2, the likelihood term can be written as:

$$p(y|x,k) \propto e^{-\frac{1}{2\eta^2}\|x \oplus k - y\|^2}$$

and the Gaussian prior is written as: $p(x) \propto e^{-\|x\|^2}$, where we have replaced the general operator A in Eq. 2.3 with the specific blur kernel k so that $Ax \equiv x \oplus k$.

Taking the negative log of the product of $p(y|x,k)$ and $p(x)$, gives a cost function of the form: $\min_x \lambda\|x \oplus k - y\|^2 + \|x\|^2$, where λ is a constant that weights the importance of the two terms; $\lambda =$

$1/\sigma^2$, where *sigma* is the noise level. By Parseval's theorem: $\|x\|^2 = \|X(\omega)\|^2$. Furthermore, the Fourier transform makes a convolution in the spatial domain into a point wise multiplication in the frequency domain. This gives us an equivalent frequency domain cost function: $\min_X \lambda\|X(\omega) \circ K(\omega) - Y(\omega)\|^2 + \|X(\omega)\|^2$, where $\circ$ refers to point-wise multiplication. Taking the gradient of this with respect to $X(\omega)$ gives us a closed form solution for $X(\omega)$, which results in Weiner deconvolution:

$$X(\omega) = \frac{K(\omega) \circ Y(\omega)}{1 + \lambda|K(\omega)|^2} \tag{2.11}$$

However, the assumption of Gaussian priors on images is not very accurate (see Figure 4.1) and leads to mediocre performance of Weiner deconvolution. More sophisticated image priors such as the heavy-tailed priors and GMM model were described in Section 2.2. The use of these priors in the model Eq. 2.3 leads to a non-quadratic and almost always a non-convex problem. As a result, the minimization of the resulting cost function is not possible with simple frequency domain transformations.

For priors such as the heavy-tailed derivative priors and Fields of Experts, the prior is of the form $-\log p(x) = \sum_i |(x \oplus f)_i|^\alpha$ where i corresponds to entries of the vector x i.e. the priors are *point-wise*. For point-wise priors, the Iterative Least Squares (IRLS) algorithm [42] is an effective numerical scheme for minimization of the posterior $p(x|y,k)$. As the name suggests, the idea is to solve a series of weighted least squares problems, where the weights are modified at every iteration. Suppose the likelihood term is of the form $-\log p(y|x,A) = \|Ax - y\|^2$, then the minimization problem is given by:

$$x^* = \arg\min_x \lambda\|Ax - y\|^2 + \sum_i |(x \oplus f)_i|^\alpha \tag{2.12}$$

Eq. 2.12 is not a quadratic problem when $\alpha < 2$. The idea behind IRLS is to solve a series of weighted least squares problems of the form $x^{k+1} = \arg\min_x \|W^k(Ax-y)\|^2$, where the diagonal entries of the matrix W^k are determined by the current iterate x^k. Solving the weighted least squares gives us a new iterate x^{k+1}. Let $\rho_i(x) = |(x \oplus f)|_i$. W^k is a diagonal matrix with (diagonal) entries $W^k_{i,i} = \frac{\partial\rho(x)}{\partial x_i}/\rho_i(x)$, x^k being the iterate at the kth iteration. Given W^k,a least squares problem is solved to give the $x^{k+1} = \arg\min_x \|W^k(Ax - y\|^2$. This inner problem can be solved using preconditioned conjugate gradients (PCG) [138] to give x^{k+1} and then W^{k+1} is computed. IRLS is a very effective algorithm which enjoys convergence guarantees under certain assumptions on A [42].

IRLS can be quite slow for large problems, as we will see in Chapter 4. For the convex TV regularizer [137], faster algorithms have been proposed, based on *half-quadratic splitting* , originally proposed by Geman and colleagues [61, 62]. Wang et al. [167] showed how how splitting could be used with a total-variation (TV) norm for non-blind deconvolution. Instead of creating a quadratic problem using re-weighting, they introduce auxiliary variables to decouple the

likelihood and prior terms. With a TV norm prior, the original problem Eq. 2.12 is given by:

$$x^* = \arg\min_x \lambda \|Ax - y\|^2 + \sum_i \|(\nabla x)_i\| \tag{2.13}$$

where ∇ is the discrete differential operator and gives us a vector output at each pixel (consisting of the gradient values). Auxiliary variables w_i are introduced to modify the problem to:

$$\min_{x,w} \lambda \|Ax - y\|^2 + \frac{\beta}{2} \sum_i \|w_i - (\nabla x)_i\|^2 + \sum_i \|w_i\| \tag{2.14}$$

This problem is still convex in x and w_i and may be solved by alternating minimization. First fix the current value of x and update w_i (for all i); then fix w_i and update x. The sub-problem in x is now quadratic and the sub-probem in the w_i's is an ℓ_1 minimization problem. Each of these sub-problems may be solved very efficiently when A is a convolution. The x sub-problem can be solved fast using FFT's; and the w sub-problem can be solved using a point-wise shrinkage algorithm [167]. For convex problems, this leads to algorithms with provable convergence rates. We adopt these techniques in our work on non convex non-blind deconvolution Chapter 4. IRLS cannot be used for the EPLL prior of Zoran et al. [191]. They also use a numerical scheme based on half-quadratic splitting,

Introducing auxiliary variables to simplify the solution of optimization problems has been studied in other contexts. In the numerical solution of partial differential equations, these techniques are called *operator splitting* [130]. In the optimization community, a similar family of methods is termed *Alternating Direction Method of Multipliers* [17] and proximal method algorithms [37]. While these methods differ in their assumptions and details, the overall idea is to introduce new variables into a problem, and then solve a sequence of subproblems which are tied together through the newly introduced variables.

Schmidt et al. [139] have recently introduced an algorithm for non-blind deconvolution which directly tries to find a minimum mean square estimate from the posterior distribution, using a sampling-based approach. This approach is computationally demanding, but has the advantage of not requiring hyperparameter tuning. [28] also present a Bayesian approach to non-blind deconvolution.

Cho et al. [34] try to improve reconstructed image quality by develop a global prior based on matching gradient distributions of the unknown image to a reference gradient distribution. Their algorithm uses a two-step process to first update a penalty term, which is then used to reconstruct the latent image. Their method leads to greater perceptual improvement, especially of textured regions in the image.

There are a number of practical issues that arise in image deconvolution. The most serious is the presence of noise, which is usually a problem in low-light situations. In MAP approaches, this is handled by a step of noise estimation followed by setting the λ hyperparameter (e.g. in Eq. 2.13) in a heuristic manner. Setting this parameter too low leads to an over regularized (usually blurry) result. Setting it too high leads to ringing artifacts which manifest as oscillations around sharp

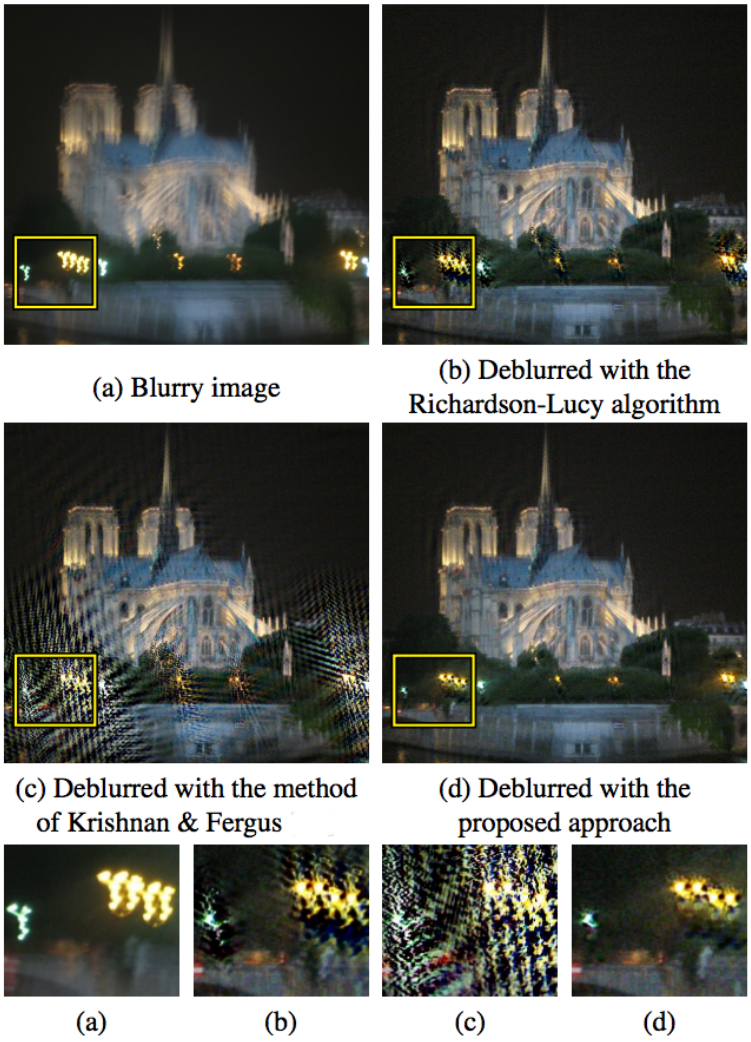


Figure 2.7: A figure taken from [172] which shows artifacts that are caused if saturated pixels are not handled. The smaller figures at the bottom show zooms taken from the areas marked by yellow boxes. "Krishnan and Fergus" refers to our algorithm from Chapter 4 which does not handle saturated pixels.

edges. Bayesian methods such as [139] handle this in a more principled manner by integrating out the noise level. This however comes at a significant computational cost.

Another practical issue is the presence of clipped (saturated) regions in the blurry image. A simple non-blind deconvolution will lead to significant artifacts around clipped pixels. Whyte et al. [172] introduce a non-blind deconvolution scheme to handle artifacts arising due to saturated pixels (see Figure 2.7). Cho et al. [33] use an EM-like approach to alternately estimate the latent image and the saturated pixels. At each step, the estimated saturated pixel map is used to formulate a constrained deblurring problem. Harmeling et al. [68] use thresholding to ignore saturated pixels during deconvolution in a multi-frame deconvolution framework.

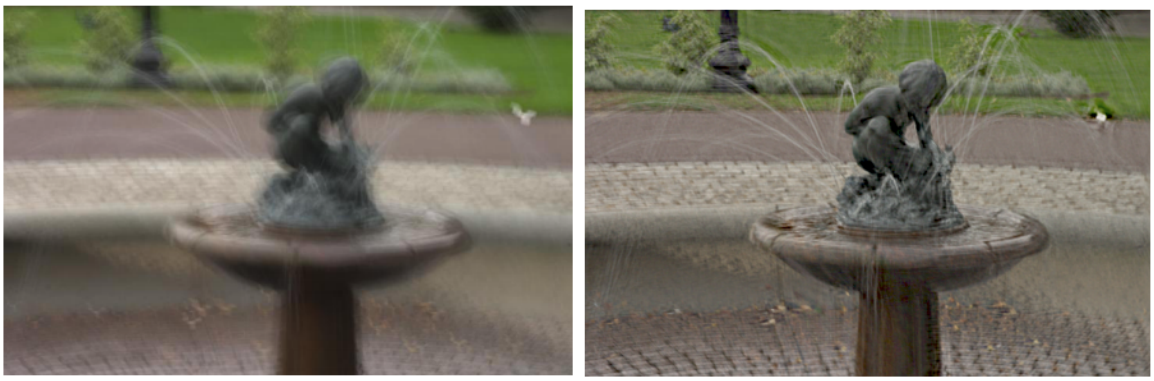


Figure 2.8: An example from [55] showing spatially invariant blind deconvolution of an image. On the left is the blurred image and the deblurrred image is shown on the right.

2.6 Blind deconvolution

Blind deconvolution has been studied for a long time in both the image processing and data communications communities, being known as *blind equalization* in the latter. An early survey is given in [91]. The simplest form of the blind deconvolution problem is when the blur is assumed to be due to translational motion of the camera, so that the model for the blurred image is $y = x \oplus k + n$ i.e. the blurring operator A is given by a filter k, and n is additive Gaussian noise. When only a single observation y is given, the number of unknowns is greater than the number of observed variables. Multiple solutions are therefore possible (see Figure 1.2). This is also easy to see from the form of the equation $y = x \oplus k$ - we can perturb k and find an x to satisfy the formation model approximately (within noise level tolerance). An example of blind deconvolution from [55] is shown in Figure 2.8.

A more straightforward way to overcome the ill-posedness is to increase the number of observations. This is the approach taken in [187] where a pair of images is captured, one being blurred with a long exposure and the other being noisy due to a short exposure. Now there are more observed variables than unknowns. The noisy image has sharp edges that provide a strong constraint to aid the blind deconvolution problem. However, this method requires perfect registration of the blurred/noisy pair. Another method to estimate the blur kernel is to use additional hardware. This is the approach taken in [76, 154]. In the former, gyroscopes and accelerometers attached to the camera allow measurement of camera motion. In the latter, 2 cameras are used - this approach is explained in Section 2.4.

Single-image methods have been studied for many years. When a single image is used, the image and kernel priors take on great importance. Single-image methods may be into three main classes: Variational, Maximum A-Posteriori (MAP) and and spectral methods. The variational and MAP methods were introduced in Section 2.2. Spectral methods use frequency-domain information and do not have a probabilistic interpretation.

In two recent papers [101, 103], Levin et al. have explained the reasons behind the failure of naive

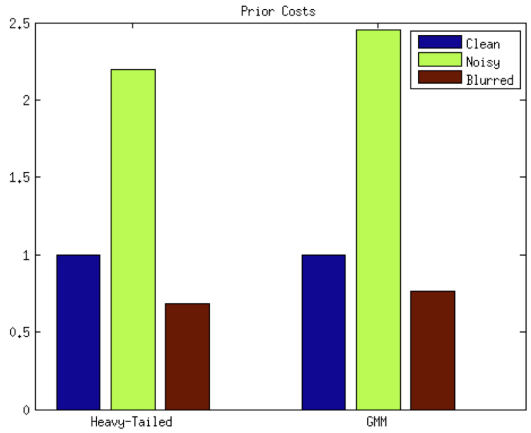


Figure 2.9: Inverted costs under standard priors. The heavy-tailed prior [96] and the EPLL prior [191] give a higher cost (lower probability) to noisy images than clean ones. But, the situation is inverted for image blur: the blurred image is given a lower cost (higher probability) than the clean image.

MAP methods and the success behind variational algorithms such as that of [55]. The problem lies with the use of image priors that are unsuitable for blind deconvolution. Existing image priors such as EPLL [191], heavy-tailed priors [96] and Fields of Experts [136] give a higher probability to blurred images than to sharp ones. Hence this biases simple MAP estimators towards the trivial solution, where the output image x is the same as the input y and the kernel k is the δ kernel (identity). An example of this situation is shown in Figure 2.9. Here we see that the EPLL prior and heavy-tailed gradent priors give a lower cost to the blurred image than the sharp image.

Most MAP methods overcome this fundamental problem by introducing various heuristics to prevent the collapse into a trivial solution. In [32], shock filtering is used to enhance the edges of the image iterates. Shock filtering [123] is a non-linear process which smooths out small edges, while enhancing (increasing the gradient of) larger edges. This edge map with some edges smoothed and others enhanced is used to determine a blur kernel, following which a new estimate of the latent image is computed. The shock filtering is shown to stabilize the MAP estimation process and gives good blind deconvolution performance. The workflow of Cho and Lee is shown in Figure 2.10. The idea of shock filtering is also used in other papers [68, 69, 181]. Unfortunately there is no clear understanding of why non-linear filtering works well to stabilize the MAP processing. A plausible, although incomplete, explanation is that keeping only strong edges while suppressing others prevents instability due to noise.

A much more principled way of handling the blind deconvolution problem is the variational approach, used in [55, 103, 104]. As explained in Section 2.1, the variational approach finds an (approximate) expected value of k by marginalizing out over x in the posterior distribution $p(x, k|y)$. Since $p(x, k|y)$ is usually intractable, this marginalization is carried out by using an approximation: $p(x, k|y) \approx q_1(x)q_2(k)$, where $q_1(x)$ and $q_2(k)$ are more tractable distributions (usually Gaussians or mixture of Gaussians). The parameters of q_1 and q_2 are chosen to approx-

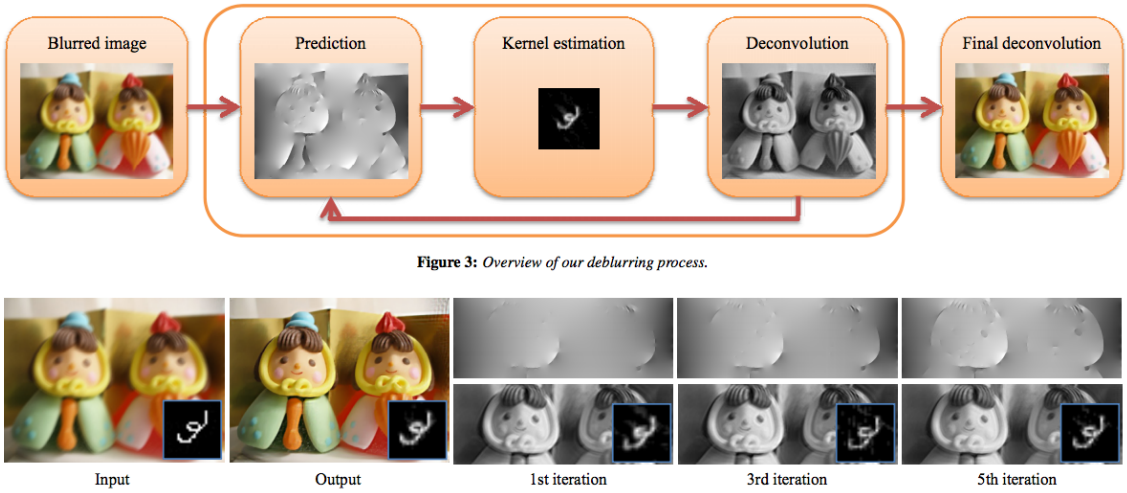


Figure 3: *Overview of our deblurring process.*

Figure 2.10: The deblurring workflow from Cho et al. [32].

imate closely the posterior $p(x, k|y)$. Levin et al. [103] provide an EM-based method to perform this marginalization more efficiently and robustly than in [55].

Due to the commutativity of the convolution operator, the formation model $y = x \oplus k + n$ can be reasonably approximated as $\nabla y = \nabla x \oplus k + n$ where ∇x is the gradient of x. When heavy-tailed gradient priors are used, they are of the form $-\log p(x) = \|\nabla x\|^{\alpha}$, in which case the posterior can be expressed as $p(\nabla x, k|\nabla y)$. It has been empirically observed in [103] that the gradient-space approach provides much better solutions than the image-space approach. However, the gradient-space approach ignores the dependence between gradients in different directions and integrability constraints. It is possible that higher-quality results may be achieved by taking this dependence into account. A disadvantage of the gradient-space approach is that the output of the blind deconvolution only results in a useful kernel k. The original blurred image must then be deconvolved with a non-blind technique using the kernel k from the first step. This adds additional computational expense.

A third class of blind deconvolution methods are spectral methods which cannot be interpreted in probabilistic terms. These methods rely on the properties of the Fourier spectrum of natural images. A recent paper in this class is [63]. It is well-known that natural images exhibit a power-law decay in their spectrum; given an image x with Fourier transform $X(\omega)$, $|X(\omega)|^2 \propto \|\omega\|^{-2}$. However, for many images, it is shown in [63] that the constant in the decay is actually dependent on the frequency ω (see Figure 2.11). This is due to the presence of long edges. That is $\|X(\omega)\|^2 \approx c_{\theta(\omega)}\|\omega\|^2$, where $\theta(\omega)$ is the angle of the vector ω. The authors present a sophisticated algorithm to determine the orientation specific constants $c_{\theta(\omega)}$, and thereby the kernel k. The key advantage of this technique is that only the statistics of the blurred image y are required. The sharp image x is never repeatedly estimated as happens in the MAP and variational methods. This makes their technique very fast and robust to scene content. However, it is unclear how to extend this method to spatially varying blur, since the Fourier techniques

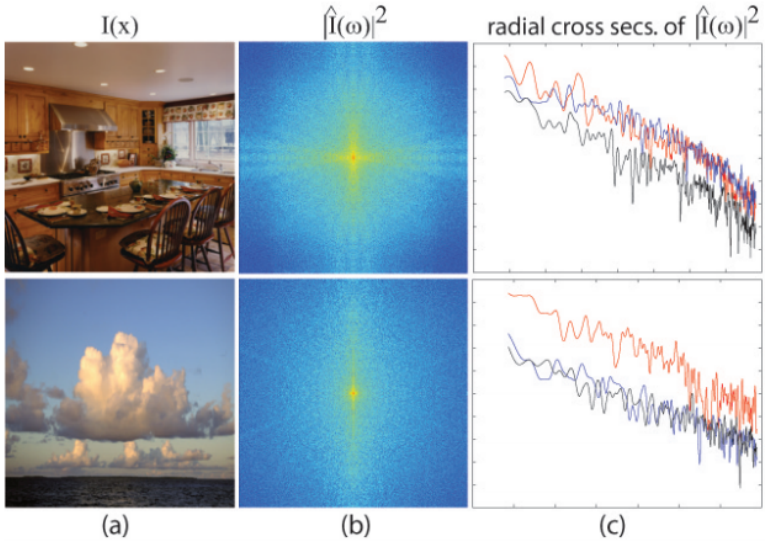


Figure 2.11: A figure from [63] showing: (a) two natural images; (b) their power spectra; (c) log-log plots of the cross-sections of the power spectra. According to the classical theory, the spectrum in column (b) should be close to a constant and the curves with each plot in (c) should overlap. This is clearly not the case and there is a multiplicative offset.

are global in nature.

Until now, we have considered spatially invariant blur. However, in practice, camera rotations and object motion in the scene lead to spatially varying blur. A number of recent papers have addressed the spatially varying blur problem. Whyte et al. [173] modified the variational approach of [55] to address the case of camera shake that is not purely translational. They achieved this by parametrizing the camera motion in terms of 3D rotations of the camera. Hirsch et al. [69] address general spatially varying blur by dividing the image into overlapping patches and computing a blur kernel for each patch. Their method assumes that the blur kernel varies smoothly across the image, and is therefore also restricted to camera shake. In contrast, Levin [95] provides a simple algorithm to handle motion blur that is unidirectional and constant. Her algorithm is based on the empirical observation that the horizontal and vertical gradients of an image exhibit similar statistical properties (for most images which have significant gradients in all directions). Furthermore, unidirectional blur (for example, horizontal blur) changes the statistics of one of these two derivatives. The amount of horizontal blur may then be determined by blurring the image in the vertical direction until the statistics of the blurred gradients match. Deconvolving images blurred by general object motions remains an open research problem.

2.7 Localized corruption removal

Images can be corrupted in many ways. Additive white Gaussian noise and motion blur are two such corruption types. Localized corruptions are caused due to dirt on the camera lens, or taking

Figure 2.12: A figure from [8] showing: (Left) A frame from a video sequence shot during snow; and (Right) the same frame with snow removed.

pictures through a transparent surface with dirt or dust. Another example is taking images or video during rain or snow. The restoration of such images is of importance for vision-based driver assistance, robotics, and aesthetics. The main stumbling block in these problems is the lack of a formation model for the corruption process: the corruption may occur randomly at different pixels and at different locations.

To overcome this, most existing approaches use a combination of physics-based models and multiple video frames for restoration. The removal of rain from videos has been investigated by Garg and Nayar [59], and Barnum et al. [8]. These methods are based on spatio-temporal filtering of a video sequence to detect and remove rain. The approach of Barnum et al. [8] can also be extended to snow removal, and snow and rain enhancement. Figure 6.2 shows an example of snow removal from their paper.

Garg and Nayar [60] also consider how the optics of the camera (exposure and depth of field) may be adjusted to reduce or remove the appearance of rain. This approach works well for rain drops and is real-time. However, it requires control over the camera aperture and depth of field settings which is often not possible. Furthermore, it is not clear whether this approach may be extended to snow removal.

Gu et al. [66] have considered the problem of removing image artifacts caused by "thin occluders", usually due to the presence of dust or dirt on camera lenses or other objects such as fences. They rely on the defocus property of lens and make the assumption that the artifact is predominantly a low frequency aberration. They use multiple video frames (and assuming perfect registration) and simple point wise operations to detect and reconstruct the artifacts. An example from their paper in shown in Figure 2.13.

Jancsary et al. [72] have proposed a non-parametric model based on Regression Tree Fields where the parameters of the model are learnt from a training set. The loss function that guides the training process may be a mean square error between the predicted output and the true (uncorrupted) output images, as a function of the input image or other kinds of loss functions.

Figure 2.13: A figure from [66] showing: (Left) A frame from a video sequence with thin lens occluders causing low frequency artifacts; and (Right) the same frame with corruptions removed.

They show results on removing synthetically generated localized corruption.

Removing localized corruption can be considered a form of blind inpainting, where the position of the corrupted regions is not given, unlike traditional inpainting [2]. Dong et al. [45] show how salt-and-pepper noise can be removed, but the approach does not extend to multi-pixel corruption.

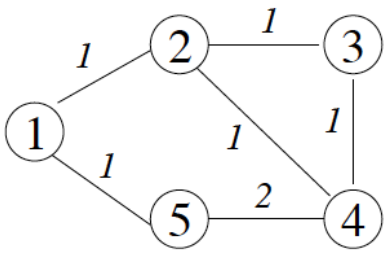
2.8 Preconditioners and solvers for Laplacian matrices

The solution of linear systems involving Laplacian matrices has attracted significant research interest in recent years. This is due to the close connection between Laplacians and graphs, and the fact that Laplacians arise in the solution of discrete partial differential equations (PDEs). Consider an undirected graph G given by a triplet (V, E, w) where V is the set of vertices, E is the set of edges, and w is a weight function that assigns a real value to every edge. For a vector $x \in \mathcal{R}^{|V|}$, the Laplacian measures of the smoothness of x over the graph, and the quadratic form is given by:

$$E_L(x) \equiv x^T L x = \sum_{(u,v)\in E} w_{u,v}(x(u) - x(v))^2 \tag{2.15}$$

The Laplacian L associated with the graph G is defined as follows. The off-diagonal entries of L are given by $L_{uv} = -w_{u,v}, u \neq v$ and the diagonal entries of each row are given by $L(uu) = \sum_{u\neq v} w_{u,v}$. This definition arises naturally from Eq. 2.15. Graphs and Laplacians therefore have a one-to-one correspondence. Figure 2.14 demonstrates this correspondence.

Due to the above definition, Laplacians are symmetric and positive definite when the edge weights w are non-negative. Laplacians corresponding to graphs with non-negative edges are also symmetric diagonally dominant (SDD), and are called M-matrices. This means that the diagonal



$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 4 & -2 \\ -1 & 0 & 0 & -2 & 3 \end{pmatrix}$$

Figure 2.14: A graph with 5 vertices and the corresponding Laplacian matrix, which is a 5×5 matrix. The weights on the edges are indicated by the numbers on each edge. The vertices are numbered within the circles and form the rows of the Laplacian matrix, in order. For example, node 1 corresponds to the first row of the matrix and therefore it has non-zero entries in columns 2 and 5, since it is connected to nodes 2 and 5 in the graph.

entry in each row of L is greater than or equal to the absolute sum of the off-diagonals:

$$L_{ii} \geq \sum_{j \neq i} |L_{ij}| \tag{2.16}$$

SDD matrices are positive definite. The theoretical research community has focused on the solution of SDD linear systems [161, 148, 147, 82, 81] . Systems involving SDD matrices can be shown to be equivalent to systems involving larger M-matrices. However, not all Laplacians are SDD. Particularly, many Laplacians arising in computer graphics and computational photography are not SDD.

Laplacians arise in a number of applications. Wardetzy et al. [168], provide a comprehensive taxonomy of the most common Laplacian matrices arising in 2D and 3D computer graphics. The most important of these are Laplacians arising from discrete partial differential equations. Examples of these applications are given in Chapter 7. Laplacians also arise in other domains such as image segmentation and graph clustering. Due to their wide applicability, the efficient solution of linear systems involving Laplacians is of importance.

State of the art sparse direct solvers [43] are based on the nested dissection method [105]. From a theoretical perspective, nested dissection has strong guarantees on fill-in for systems arising from planar graphs. The best known algorithms for planar graphs run in time $O(n^{1.5})$ and are incorporated as direct solvers in software packages such as MATLAB. General dense direct solvers such as Gaussian elimination and LU factorization run in $O(n^3)$ time [79]. The latter however, is often used for matrix preconditioning by dropping terms from the complete LU factorization [138].

While direct solvers provide machine precision accuracy, the modern versions are complex to code and are expensive in computation and memory requirement. The machine precision accuracy is not necessary in many applications. Therefore, one may consider the use of iterative solvers. Iterative linear solvers such as Jacobi, Gauss-Seidel and Conjugate Gradients (CG) are applicable for diagonally dominant and positive semi-definite matrices [79] including the Laplacian matrices we are considering. Since these methods consist of matrix-vector multiplications, each iteration

runs in $O(n)$ time when solving sparse systems. However, the number of iterations taken by these methods depends on the condition number of the Laplacian matrix.

The condition number of a symmetric positive definite matrix is defined as

$$\kappa(L) = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{E_L(\mathbf{x}_{\max})}{E_L(\mathbf{x}_{\min})} = \frac{\max_{\mathbf{x}} E_L(\mathbf{x})}{\min_{\mathbf{x}} E_L(\mathbf{x})}, \tag{2.17}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximal and minimal non-zero eigenvalues of L and $\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$ are their corresponding eigenvectors. The last equality results from the fact that eigenvalues of L are the extremal values of E_L.

The number of iterations it takes iterative solvers to achieve a certain accuracy depends on $\kappa(L)$: $O(\kappa)$ iterations for Jacobi and Gauss-Seidel and $O(\sqrt{\kappa})$ for conjugate gradients [138]. In the case of homogeneous matrices, $\kappa(L) = O(l^2)$, where l is the domain's length, $l \propto \sqrt[d]{n}$, and d is the spatial dimension ($d=2$ for images). Inhomogenous Laplacian (defined below) often contain approximate zero modes [160], which, according to (2.17), lead to very high values of $\kappa(L)$.

The condition number of a matrix can be reduced by converting the linear system, $L\mathbf{x} = \mathbf{b}$, into a related problem by multiplying it with a *preconditioning matrix* Q^{-1} such that the condition number of $Q^{-1}L$ is significantly lower than that of L. In order to achieve effective preconditioning, the matrix Q^{-1} must meet several additional requirements. Iterative linear solvers such as Jacobi, Gauss-Seidel and CG consist of repeated matrix-vector multiplications with $Q^{-1}L$, which is typically computed in succession, with L and then with Q^{-1}. Therefore, multiplying a vector with Q^{-1} must not be significantly more expensive than multiplication with L. For example, in the case of sparse Laplacian matrices, these operations must cost no more than $O(n)$. Note that this does not require Q^{-1} to be sparse; there just needs to be an efficient procedure for multiplying it with vectors, which is the case with various hierarchical schemes (Section 7.2.2). Another important aspect is that $Q^{-1}L$ must meet solver-specific requirements, for example maintaining the positive definiteness of L in case of the Preconditioned Conjugate Gradient (PCG) method.

Since iterative solvers are often the method of choice, the search for efficient linear system solvers boils down the construction of efficient preconditioners for Laplacian matrices. The classical preconditioners are based on multigrid and multilevel ideas. The fundamental idea here is to recursively construct smaller (coarser) versions of the original problem in such a manner that the coarser versions enable a good approximate solution to the fine problem. This is visually depicted in Figure 2.15.

The fundamental reason that multigrid methods are useful is as follows. When a linear system $Lx = b$ needs to be solved, the solution x can be expressed as a linear combination of the eigenvectors of L: $x = \sum_i \alpha_i \mathbf{v}_i$, where $\mathbf{v}_i$ are the eigenvectors of L. The components α_i corresponding the largest eigenvectors are quickly resolved by iterative methods. However, it is well known that the components corresponding to the smallest eigenvectors are extremely slow to be resolved. This is because these components correspond to *long range* interactions between variables. To overcome this slow convergence, it is useful to create a coarser version of the problem $Lx = b$

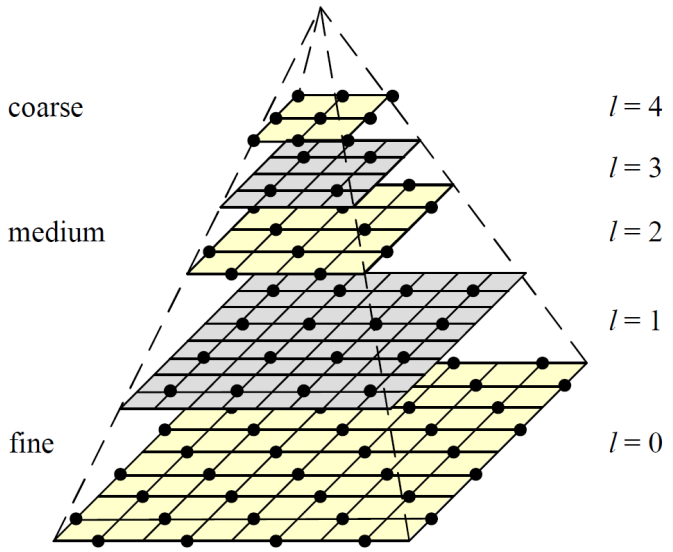


Figure 2.15: A multilevel pyramid with regular half-octave coarsening [153].

using a subset of variables of x. If this coarser problem can be solved, then by interpolation we can find an approximate solution to the fine level problem, and thereby accelerate the solution process. Repeating this logic recursively gives rise to a multigrid method.

There are three key components to a multigrid method: coarsening, smoothing and interpolation. The coarsening step decides which variables form the coarse-level problem, and which stay at the fine level. The smoothing step corresponds to an iterative solver such as Jacobi or Gauss-Seidel. This is applied at every level of the hierarchy to improve the iterative solution. The interpolation provides coarse-to-fine transfer of the solution. Various choices for these components give rise to different multigrid methods. The multigrid hierarchy may be used as a standalone solver or as a preconditioner. In most cases, the hierarchy works best as a preconditioner for CG.

The earliest versions of multigrid preconditioners were developed for *homogenous* Laplacian matrices. Homogenous Laplacians correspond to graphs with all edges having the same weight. The resulting Laplacians have all (non-zero) off-diagonals with the same value. The optimal multigrid scheme for homogenous problems is the geometric multigrid method (GMG) [18]. It is optimal in the sense that the matrix preconditioned by a GMG preconditioner achieves a constant condition number regardless of the matrix size. This is theoretically proven [160] using Fourier analysis techniques. As the name suggests, GMG takes advantage of the geometry of a problem. It is thus only useful for problems where an underlying grid structure is available, such as in the case of 2D images. In the case of a 2D grid, a coarse level created by selecting every fourth grid point, to give *full octave* coarsening.

The hierarchical basis methods [186, 152] use a similar approach to GMG, where a multi-level basis is used to precondition the matrix when solved by an iterative solver. Linear running time is achieved by applying iterative solvers in combination with multigrid or multi-level methods.

Inhomogenous Laplacians arise from graphs with different weights on the edges. The multigrid theory for such problems is much less well-developed. However, these methods are empirically shown to work well. The condition number of inhomogenous Laplacian matrices is often considerably higher than their homogeneous counterparts; clusters of strongly-connected variables that are weakly connected to the rest of the system introduce very weak modes, known as approximate zero modes, which increase the condition number [160]. Here "strongly-connected" refers to variables with large edge weights and "weakly-connected" refers to variables with small edge weights. Spatially homogeneous solvers, such as GMG, fail to capture these spatially irregular modes and do not perform effectively on such problems.

The algebraic multigrid method (AMG) [19] generalizes its geometric counterpart and has a better ability to isolate and rescale the weak modes in spatially varying matrices. Adaptive coarse-grid selection plays a key role in AMG's success in capturing these modes, but at the same time, it leads to a growth in the bandwidth of non-zero matrix elements at coarse levels. This is an important practical problem because successive levels are not significantly lower in cost if the smaller size of the problem is offset by significantly higher bandwidths. The reason for the increase in bandwidth is easy to see: whenever a variable is eliminated, the neighbors of that variables become connected to each other.

The aggregation-based AMG [160] and its smoothed version [162] limit the number of non-zero elements in the prolongation matrices that relate successive levels. This is equivalent to limiting how many coarse neighbors of a fine level variable are used to interpolate the fine level solution. This avoids the growth in the matrix bandwidth but also lowers its preconditioning abilities and increases the number of required cycles. The recently developed lean AMG (LAMG) solver [108] falls into this category but offers a more sophisticated agglomeration rule as well as a correction step that improves the representation of weak modes at coarser levels. The use of fixed coarsening is another way to avoid growth in the matrix bandwidth [188, 120]. A hierarchical basis analogue of the AMG that also employs regular grid selection is described [153]. A careful comparison, however, shows that methods that use adaptive grid selection offer better overall performance on highly irregular problems [89, 84].

Several multigrid solvers have been adapted for specific computer graphics purposes. A streaming multigrid solver capable of solving very large problems, arising from processing gigapixel images, is described in [77]. Farbman et al. [50] describe a highly optimized pyramid-based solver for tone mapping and interpolation. A multigrid framework for the simulation of high-resolution elastic deformable models, supporting linear and co-rotational linear elasticity, is described in [190].

The multigrid based iterative solvers are very amenable to parallel processing and so can be significantly optimized, such as with GPU processing, for example. [143] develop a multigrid solver to handle mesh deformation problems. They show significant speedup over direct solvers for meshes of upto 3 million vertices. [14] present a GPU-based multigrid solver. However, they restrict their numerical experiments to small grids with less than 200K vertices.

Recent work in theoretical computer science has led to the development of preconditioners for

graph-Laplacian matrices. A general umbrella term for these methods is *combinatorial preconditioning* [146]. The key is to construct a sparser approximation of the original matrix that is easy to invert and to then use it as a preconditioning matrix. For example, Vaidya [161] suggested preconditioning with the Laplacian of a maximum spanning tree derived from the graph of the original Laplacian. The justification for this is that inverting a tree is computationally very efficient. This construction, however, does not offer an attractive bound on the resulting condition number. Boman and Hendrickson [15] showed that better bounds are attained using low-stretch spanning trees. A low-stretch spanning tree is one where the edges are "evenly" distributed in their weight. This intuitively makes sense as it avoids extremely large or extremely small edges in the tree. Extremely small edges would effectively decouple variables. However, decoupled variables are not a problem for iterative solvers since they can be simultaneously solved. On the other extreme, including extremely large edges would lead to the exclusion of small edges. However, constructions of low-stretch spanning trees require $O(n \log n)$ operations and still do not guarantee that the resulting condition number is independent of n.

Vaidya [161] also suggests improving the preconditioning by adding $O(n)$ edges to the spanning tree. This construction, which is known as an ultra-sparsifier, solves sparse Laplacian matrices in $O(n \log^{15} n)$ time and is constructed in nearly linear time in a seminal work [148]. The fastest known ultra-sparsifier based solver is described in [82] and runs in $O(n \log^2 n (\log \log n)^2)$ time. A recent paper [78] uses a non hierarchical approach for solving linear systems involving SDD matrices.

A fundamental theorem in combinatorial matrix preconditioning shows that the effectiveness of Q^{-1} as a preconditioning matrix depends on how well E_Q approximates E_L. More specifically, given $a, b > 0$ such that

$$\forall \mathbf{x}, \;\; a E_Q(\mathbf{x}) \leq E_L(\mathbf{x}) \leq b E_Q(\mathbf{x}), \tag{2.18}$$

$\kappa(Q^{-1}L) \leq b/a$ [16, Prop. 2.4].

The theoretical works based on combinatorial preconditioning provide a solid foundation for nearly-linear time solvers. Unfortunately, to date no practical solver has been developed based on these ideas. A number of practical solvers, however, have been developed in the multigrid community. Some examples are PyAMG [10] and HyPre [49]. Recently, Koutis et al. [83] used the notion of conductance from the support theory of graphs to derive an aggregation-based AMG method. They call the algorithm Combinatorial Multigrid (CMG).

There are also specialized methods that work well in some applications normally solved using Laplacian matrices. The edge-avoiding wavelets in [52] offer fast running times for edge-preserving interpolation and tone mapping. However, they consist of a regular sampling strategy and produce results of limited accuracy. Edit propagation using KD-trees in [180] is another example.

Chapter 3

Dark Flash Photography

3.1 Introduction

The work described in this chapter is joint work with Rob Fergus. It was published in SIGGRAPH 2009 [85], and resulted in a patent filing [54].

The introduction of digital camera sensors has transformed photography, permitting new levels of control and flexibility over the imaging process. Coupled with cheap computation, this has precipitated a wide range of novel photographic techniques, collectively known as Computational Photography. Modern camera sensors, be they in a cellphone or a high-end DSLR, use either a CCD or CMOS sensor based on silicon. The raw sensor material responds to light over a wide range of wavelengths, typically 350–1200nm. Colored dyes are deposited onto the sensor pixels in a Bayer pattern, resulting in 3 groups of pixels (red, green and blue). Each responds to a limited range of wavelengths, approximating the sensitivities of the three types of cone cell in our retina. However, silicon is highly sensitive to infra-red (IR) wavelengths and it is difficult to manufacture dyes that have sufficient attenuation in this region, thus an extra filter is placed on top of most sensors to block IR light. This gives a sensor that records only over the range 400-700nm, matching our own color perception, but a considerable restriction of the intrinsic range of the device.

One solution to capturing photographs in low light conditions is to use a flash unit to add light to the scene. Although it provides the light to capture otherwise unrecordable scenes, the flash makes the photographic process intrusive. The sudden burst of light not only alters the illumination but disturbs any people present, making them aware that a photo has just been taken and possibly dazzling them if they happen to be looking toward the camera. For example, a group photo in a dark restaurant or bar using a bright camera flash leaves the subjects unable to see clearly for some moments afterward. From an aesthetic perspective, the illumination of the flash often alters the look of the photograph, rendering it "flat" and unappealing.

In this paper we introduce a camera/flash system that is based around off-the-shelf consumer

equipment, with a number of minor modifications. First, the camera is a standard DSLR with the IR-block filter removed, thus restoring much of the original spectral range of the sensor. Second, we use a modified flash that emits light over a wider spectral range than normal, which we filter to remove visible wavelengths. This *dark flash* allows us to add light to the scene in such a way that it can be recorded by the camera, but not by our own visual system. Using the dark flash we can illuminate a dimly lit scene without dazzling people present, or significantly disturbing those around. Furthermore, it allows a fast shutter speed to be used, thus avoiding camera shake. However, the difficulty is that people want images with colors that match their visual experience and this will not be the case for images captured using the dark flash.

To overcome this, we acquire a pair of images in the manner of flash/no-flash photography [46, 128], one using the dark flash and the second using ambient illumination alone. For the latter to be blur-free a fast shutter speed must be used, resulting in high noise levels in dim light. A key observation is that if the non-visible and visible channels are close in wavelength, strong correlations will exist between them. We introduce a novel *spectral prior* that exploits correlations between spectral bands. Using this constraint, the edge structure of the dark flash image can be used to remove the noise from the ambient image, yielding a high quality result that lacks the shadow and specularity artifacts present in the flash image. Figure 3.1 illustrates our overall scheme.

We also show how our camera/flash hardware and spectral constraints can be used in a range of additional applications, including: inferring spectral reflectance functions of materials in the scene and denoising individual color channels of images captured with standard cameras.

3.2 Related work

Our approach can be regarded as a multi-spectral version of the flash/no-flash technique introduced by [1], [128] and [46]. These papers were reviewed in Chapter 2. However, [1] did not use their technique for denoising, but for flash artifact removal. [128] use the cross-bilateral filteral for denoising the ambient, but as we show, the cross-bilateral filter works poorly when the flash has non-overlapping spectral channels.

The closest work to ours is that of [11], who show how video captured in low-light conditions can be denoised using continuous IR illumination. However, they make use of temporal smoothing to achieve high quality results, something not possible in our photography setting. [166] show how IR illumination can be used to relight faces in well-lit scenes. Both these works differ from ours in a number of ways: (i) they use complex optical bench based setups with twin cameras and beam-splitters – we use a single portable DSLR camera and temporally multiplex instead; (ii) both use IR alone rather than the near-UV and IR that we use (both being necessary for high quality reconstructions); (iii) both rely on cross-bilateral filtering to combine the IR and visible signals, an approach which we demonstrate to have serious short-comings. In contrast, we propose a principled mechanism for propagating information between spectral bands. We integrate this into a unified cost function that combines the denoising and detail transfer mechanisms, treated

Figure 3.1: Our camera and flash system offers dazzle-free photography by hiding the flash in the non-visible spectrum. A pair of images are captured at a blur-free shutter speed, one using a multi-spectral flash (F), the other using ambient illumination (A) which in this case is 1/100th of that required for a correct exposure. The pair are combined to give an output image (R) which is of comparable quality to a reference long exposure shot (L). The figures in this paper are best viewed on screen, rather than in print.

separately in cross-bilateral filtering and related methods, such as [51]. This allows us to operate in dimmer conditions than those considered by [46] or [128].

The methods cited above use a (bright) visible flash, whereas ours is designed to be almost invisible by using illumination outside the visible range. Although [128] pondered the possibility of using an infra-red flash, they did not go on to explore it. Extra challenges are posed by our flash using different wavelengths to the ambient illumination. As [46] transfer the colors from the flash image, their approach will not work in our scenario. The cross-bilateral filter, unlike our proposed approach, does not explicitly model the spectral correlations between the flash and ambient images.

Infra-red imaging has a long history in areas such as astronomy and night-vision. In consumer photography the most prominent use has been the Sony Nightshot where the IR-block filter can be switched out to use the near-IR part of the spectrum. The images are monochrome (with a greenish tint) and no attempt is made to restore natural colors to them. Other imaging approaches use Far-IR wavelengths to record the thermal signature of people or vehicles. However, this requires specialized optics and sensors and thus has limited relevance to consumer photography. Ultra-violet (UV) photography has received little attention, other than from flower photography enthusiasts [135]. Many flowers that look plain to humans have vibrant patterns under UV light to attract insects sensitive to these wavelengths.

Multi-spectral recording using visible wavelengths has been explored by several authors. [126]

used multiplexed illumination via arrays of colored LEDs to recover spectral reflectance functions of the scene at video frame rates. Our system can be used in a similar manner for still scenes, being able to estimate the reflectance functions beyond the visible range. [115] use a diffraction grating in conjunction with an LCD mask to give control over the color spectrum for applications including metamer detection and adaptive color primaries.

Our processing of the flash/no-flash pair exploits the correlations *between* nearby spectral bands. Most work on image priors has focused on capturing spatial correlations *within* a band. For example, priors based on the heavy tailed distributions of image gradients have proven highly effective in a wide range of problems such as denoising [129], deblurring [55] and separating reflections [100]. However, models that exploit dependencies between color channels are less common. The K-SVD denoising approach of [2] does so implicitly by vector quantizing color patches. The fields-of-experts approach of [136] has also been extended to model color images [112] and uses color marginal filters. However, neither of these approaches explicitly model the inter-channel correlations, unlike our method. Explicit spectral models are used in color constancy problems and joint spatial-spectral models have been proposed [145, 25] for this task, but these assume a noise-free image. [118] measured the spatial gradients of far IR images gathered with a specialized camera, demonstrating their similarity to those of visible light images. Spectral priors have also been used with near-IR in a tomography application [20].

Flash-based methods are not the only solution to taking pictures in low-light levels. Wide aperture lenses gather more light but are heavy and expensive, making them impractical for most photographers. Limited physical footprints on mobile phones prevent the possibility of large apertures in these environments. Anti-shake hardware can be used to capture blur-free images at slow shutter speeds. Recently developed "inverted CMOS" sensors have much greater light-gathering capabilities. These techniques can be combined with our approach to extend performance to even lower light levels. Software-based deblurring techniques [55, 75] can only cope with modest levels of blur and typically have artifacts in their output. Denoising techniques [159, 129] have similar performance issues, and cannot cope with the noise levels we address in this paper. Joint denoising/deblurring techniques, such as that of Yuan [187], provide better performance but still require a problematic deconvolution operation, which can introduce artifacts. Methods that register and combine a stack of noisy images, such as [156], have the inconvenience of needing to capture far more than two images. Finally, a visible flash can be made non-dazzling by using a diffuser and aiming at the ceiling. This methods works well but is limited to indoors settings with a fairly low ceiling of neutral color.

3.3 Dark flash hardware

In our approach we capture a pair of images, one with the dark flash (F) and another using ambient lighting alone (A). The pixel value p in channel j of image F depends on three terms: the spectral response of each camera channel $C_j(\lambda)$ at wavelength λ; the illumination spectrum of the dark flash $I^f(\lambda)$; and the surface reflectance function $S(p, \lambda)$ at the point in the scene.

These combine in a linear fashion:

$$F_j(p) = \int C_j(\lambda) I^f(\lambda) S(p, \lambda) \; d\lambda \tag{3.1}$$

with $j = \{1, 2, 3\}$ being the index of the camera channel. Note we assume even illumination (i.e. $I^f(\lambda)$ does not depend on p). The ambient image A is formed in a similar fashion, using illumination $I^a(\lambda)$ which scales with the exposure interval. A_1, A_2 and A_3 record red, green and blue wavelengths respectively under typical illumination. Through the choice of flash and camera, we can control $I^f(\lambda)$ and the channel sensitivities $C_j(\lambda)$.

A primary design constraint is that off-the-shelf consumer hardware should be used where possible, making the system cheap and easily reproducible. Our camera is a Fuji IS Pro, which is marketed for applications involving UV and IR work since it lacks an IR sensor filter. The flash is a Nikon SB-14UV. We use a standard Nikon 50mm f/1.8 lens, which transmits light down to 350nm, hence is not the limiting factor in the camera's UV response. A MaxMax CC3 filter was attached to the lens at all times. The purpose of this filter is to block IR light above 850nm, which would otherwise distort the colors of the ambient image (as the naked sensor's response extends out to 1100nm). This filter does not block either visible light or the dark flash. The flash is a clone of the Nikon SB-14UV, adapted from a standard SB-14 by removing the UV absorbent coating on the Xenon flash tube. A Hoya U360 filter was attached to the flash at all times to filter out visible light. The standard visible flash used in comparisons was equipped with a MaxMax CC1 filter to block its significant IR output.

The response functions $C_j(\lambda)$ in Figure 3.2(a) include the filter and lens. These filters remain in place for both shots, thus the pair of images can be taken in quick succession, limited only by the 3 frames/sec rate of the camera. In Section 3.7, we show how the camera hardware can be modified to allow capturing only one image. The flash is used at full power for all shots, the cycle time being sufficiently long that it does not fire for the second shot, giving an image with ambient illumination alone. The system is no more complex to operate than a standard DSLR (see Figure 3.3(top left) for a picture of the system).

We now describe the form of $I^f(\lambda)$ and how it can be recorded by the camera while remaining largely invisible to humans. The spectral response of each camera channel $C_j(\lambda)$ is shown in Figure 3.2(a). Note that with no IR sensor filter, the responses extend considerably beyond the visible range (400-700nm). The spectrum of the dark flash $I^f(\lambda)$ is shown in Figure 3.2(b). It has two distinct emission lobes, both just outside the visible range. The first, consisting of UV light, couples with the small part of channel $j = 3$'s response extending below 400nm. The second lobe in the IR region between 700 and 800nm is picked up by channel $j = 1$ which responds strongly. Thus, the dark flash allows the recording of two independent measurements at each location in a scene within a single image: one in UV recorded in F_3, the other in IR recorded in F_1.

The flash/no-flash image pair captures the scene at 5 different spectral bands, assuming the ambient illumination is dim compared to the output of the flash: 1. UV (370–400nm) in F_3; 2. Blue ($\sim$ 400–500nm) in A_3; 3. Green ($\sim$ 500–600nm) in A_2; 4. Red ($\sim$ 600-700nm) in A_1 and

5. IR (700nm–800nm), recorded in F_1. In Figure 3.3, we show a Macbeth color chart in each of these five bands.

For comparison purposes, we also use a standard visible flash whose power is adjusted to give comparable camera exposure to the dark flash. In Figure 3.3(top) we attempt to show the relative perceived brightness of the dark and visible flashes by capturing them using a standard DSLR whose spectral response is close to that of our eyes (thus the brightness in the image should correspond to our perception). See Section 3.5.3 for a quantitative analysis of their relative brightness.

Safety issues. As shown in Figure 3.2(b), our dark flash emits energy just outside visible wavelengths, centered around 380nm with negligible energy below 360nm or above 400nm (until the IR lobe at 700nm). The health hazard posed by UV light depends strongly on the wavelength, those close to visible (400nm) being orders of magnitude safer than the shorter wavelength components of sunlight. Our flash is very close to visible, even closer than black-lights found in bars and nightclubs, which have a broader spectral width centered at 360nm. In the USA, the acknowledged regulations regarding the safe daily exposure to UV light are given in the Threshold Limit Values (TLV) booklet, published by the government body ACGIH [157]. We carefully measured the absolute spectral irradiance of our flash using a spectrometer.

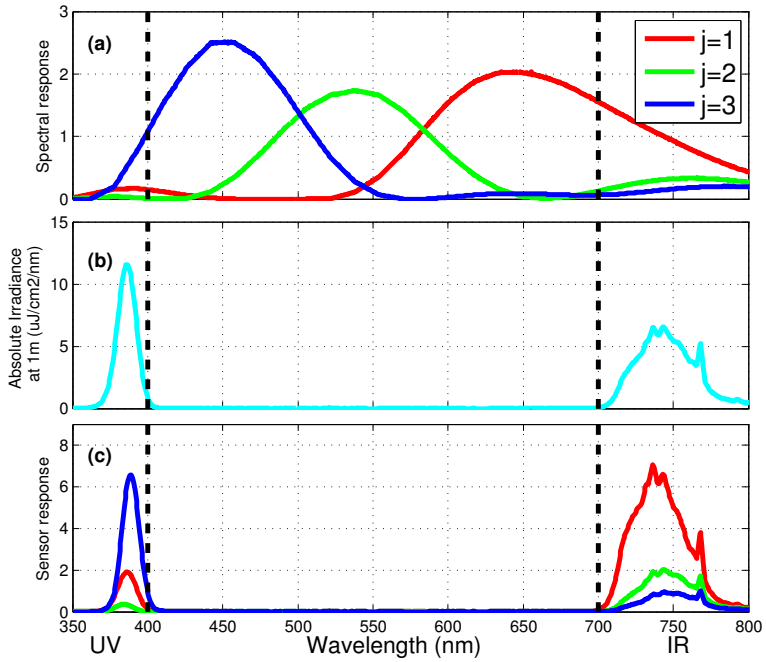


Figure 3.2: (a) Spectral response curves $C_j(\lambda)$, $j = \{1, 2, 3\}$ for each of the camera's three color channels. (b) Absolute irradiance 1m from the dark flash $I^f(\lambda)$. (c) Spectrum received by the camera sensor when imaging a perfect white surface ($S(p, \lambda)$=1) illuminated by the dark flash. The curves are the product of those shown in (a) and (b). The recorded pixel values for the three channels are the integrals of these curves (see Eq. 3.1). Note under the dark flash: no channel records in the visible range (black dashed lines); channel j=3 measures in the UV and channel j=1 responds to IR.

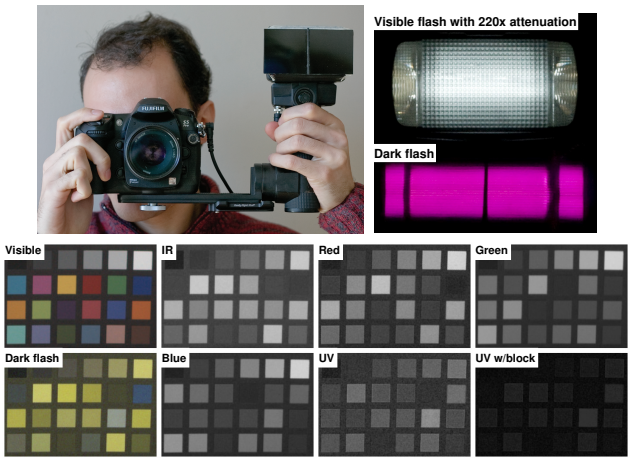
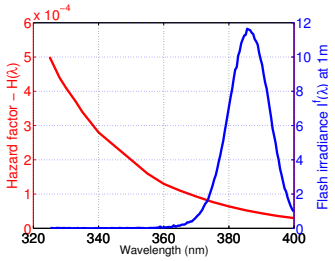


Figure 3.3: Top left: Our camera and dark flash system. Top right: The perceived brightness of the dark flash and a visible flash that gives a comparable camera exposure. To capture them in a single image, it was necessary to attenuate the visible flash by a factor of 220 using neutral density filters. Without these, the dark flash would not be visible in a non-saturated 8-bit image. Bottom: A color chart captured with a pair of flash images (visible and dark), separated out into five spectral bands. The bottom right subplot shows the UV band with a UV-block filter attached to the camera that has a sharp cut-off at 400nm. The low intensities in this band show that our camera is genuinely recording UV light, not blue light from fluorescence caused by the UV part of the flash. See Section 3.5.2 for further discussion.

The threshold limit values (TLVs) for UV radiation 180–400nm incident on the eye (the most sensitive part of the body) over any 8 hour period are given by the formula on p.155 of [157], reproduced in Eqn. 3 below. It relates the maximum number of flashes to the effective irradiance E_{Eff}, relative to a monochromatic source at 270nm. E_{Eff} is computed, using Eqn. 4 below, from the spectral irradiance of the flash $I^f(\lambda)$ (units: $\mu\text{J/cm}^2\text{/nm/flash}$) and a hazard weighting function $H(\lambda)$ (which is 1 at 270nm), given on p.157 of [157]. In Figure 3.4, we show $I^f(\lambda)$ and $H(\lambda)$. Integrating over the product of the two and inserting E_{Eff} into Eqn. 3, we arrive at the value of 130,000 flashes. Note that this number scales with the inverse square of distance, so at 2m the max safe limit would be 520,000 flashes.



$$\text{Max flashes} = \frac{3000}{E_{\text{Eff}}}$$

$$E_{\text{Eff}} = \int H(\lambda) I^f(\lambda)\, d\lambda$$

Figure 3.4: $I^f(\lambda)$ and $H(\lambda)$, see text for details.

Putting the above numbers in another way, if we assume that 30 minutes outside in the sun results in the maximum permissible UV dose on a bright summer day, then each flash is equivalent to being outside for 1/100th second. Hence our dark flash poses no significant safety hazard.

3.4 Dark flash processing

The pair of images, F and A are captured using a shutter speed sufficient to avoid camera shake. We assume that the ambient illumination is weak, thus A will typically be very noisy and the illumination in F will be dominated by the dark flash $I^f(\lambda)$. We seek an image R whose edges are close to those in F and whose intensities are close to a denoised version of A, hopefully being similar to a long-exposure shot of the scene L.

Standard approaches to denoising use spatial priors that enforce sparsity on image gradients [129]. In the flash/no-flash scenario, F contains high-frequency details that can assist the denoising process. But unlike conventional flash/no-flash photography, our flash and ambient illuminations $I^f(\lambda)$ and $I^a(\lambda)$ are by design almost non-overlapping, thus the colors in F will be quite different to those in the ambient image A or the long-exposure L. We propose a solution that uses the strong correlations between color channels as a constraint in an optimization scheme which computes R from A and F.

3.4.1 Spectral constraints

Consider the 1-D example in Figure 3.5 which shows a scanline across 3 squares in the color chart from Figure 3.3. Figure 3.5(a) shows the intensities from the red channel of a long exposure shot (L_1, in magenta) and IR from the dark flash (F_1, in black). Although the intensities are quite different, the edges are aligned, since the spectral reflectance at red and IR wavelengths are correlated with one another. The alignment of the edges is apparent in Figure 3.5(b) where the gradients along the scanline ∇F_1 and ∇L_1 are shown ($\nabla F_1(p) = F_1(p) - F_1(p-1)$, the difference between adjacent pixels p). As is widely known, this gradient signal is sparse, being close to zero everywhere but a few locations. Now, if we consider the *difference* between the two gradient signals $\nabla F_1 - \nabla L_1$ (Figure 3.5(c)) then this too will be sparse, as shown by shape of the histogram in Figure 3.5(d). Now consider a dark flash and noisy ambient image pair, shown in Figure 3.5(e)–(h). The difference between gradients $\nabla F_1 - \nabla A_1$ (in Figure 3.5(g)) is now no longer sparse, as shown by it's Gaussian-shaped histogram in Figure 3.5(h).

Reflecting the sparse distribution of $\nabla F_1 - \nabla L_1$ in Figure 3.5(d), our spectral constraints take the form of a sparse norm on the gradient difference between channels in the reconstructed image R and the flash image F_1, i.e. $|\nabla R_j - \nabla F_1|^\alpha$ where $\alpha \leq 1$. This encourages the edge structures in R_j to align spatially with those in F_1 while allowing their magnitudes to differ. Thus, when transitioning between two materials, it does not matter if the spectral reflectances are different in visible and IR/UV bands, provided that there is an significant edge in IR/UV. If an ℓ_2 norm were used, this would not be the case, and ∇R_j and ∇F_1 would have to closely match, even at material transitions, so causing artifacts in R_j (see Figure 3.15). While a conventional spatial prior, such as $|\nabla R_j|^\alpha, \alpha < 1$, would also reduce noise, it would not encourage the edges to align with those of F which are close to those of the desired solution L.

We also impose a similar constraint to the UV channel: $|\nabla R_j - \nabla F_3|^\alpha$, recalling that F_3 records

UV and F_1 records IR. For R_3 (the blue channel), this will be a strong constraint since, in terms of wavelength, blue is much closer to UV than to IR. In this example, we have only considered 1-D gradients but in the real problem we use both x and y gradients, with separate terms for each. For brevity, we use ∇ to refer to both ∇_x and ∇_y.

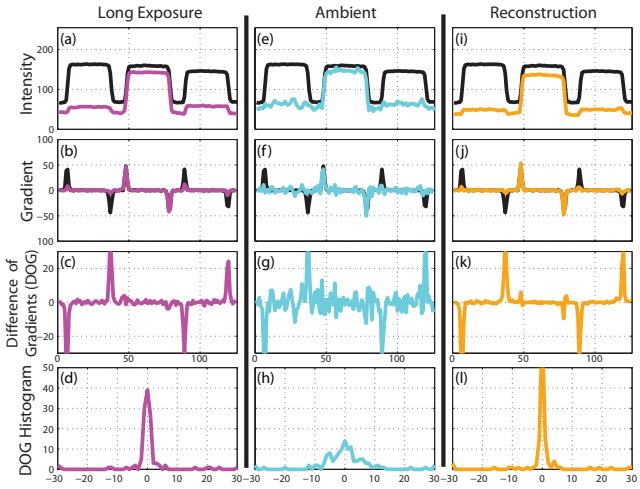


Figure 3.5: 1-D example of the spectral constraints in our model, using a a scan line across 3 squares in the color chart of Figure 3.3. See text for explanation.

3.4.2 Spatial-spectral cost function

Our cost function consists of three main terms: (i) **Likelihood**: the intensities of the reconstruction R_j should be close to those of the noisy ambient image A under an ℓ_2 norm, assuming a Gaussian noise model. (ii) **Spatial prior**: ∇R_j should be small under a sparse norm, reflecting the heavy-tailed nature of image gradients. The spatial prior term helps to give a further boost to image quality. (iii) **Spectral constraint**: ∇R_j should be close to both ∇F_1 (IR) and ∇F_3 (UV) under a sparse norm, as explained above.

As with existing flash/no-flash techniques, we use a shadow and specularity mask $m(p)$ which removes artifacts from the flash image. Details of the mask construction are given in Section 3.4.3 below. The overall cost function for each channel j is:

$$\underset{R_j}{\operatorname{argmin}} \sum_p \Big[\underbrace{\mu_j\, m(p)(R_j(p) - A_j(p))^2}_{\text{Likelihood}} + \underbrace{\kappa\, m(p)|\nabla R_j(p)|^\alpha}_{\text{Spatial}} + \underbrace{|\nabla R_j(p) - \nabla F_1(p)|^\alpha}_{\text{IR Spectral}} + \underbrace{|\nabla R_j(p) - \nabla F_3(p)|^\alpha}_{\text{UV Spectral}} \Big] \tag{3.2}$$

In our experiments, unless otherwise stated, we use $\kappa = 1$, $\alpha = 0.7$. We solve for each each channel j separately. $m(p)$ has the effect of increasing the weight on the likelihood and spatial terms in regions of shadows or specularities. We also assumed the UV and IR spectral terms to have equal weight for all channels j. Hence the weighting on the reconstruction term for each

channel μ_j is the only important parameter in the model and strongly depends on the noise level of the ambient image A. Since the blue channel is often significantly noisier than the others, we use a different value for μ_3 than for μ_1 and μ_2 (which are set to be the same). Intuitively, if μ_j is set to a large value then the colors of R will be close to those of A at the expense of increased noise. Conversely, if μ_j is small then the noise in R is reduced, but the colors will deviate from those in A. Choosing the value of μ_j can be done semi-automatically from the level of under-exposure of A (given by the camera's exposure meter) and the camera's ISO setting. If needed, the value may be fine-tuned on a small image patch, before processing the entire image. Typical values range from $\mu_j = 5$ (high noise) to $\mu_j = 40$ (low noise).

Returning to our 1-D example in Figure 3.5, we show the scanline across the color chart for our reconstructed image R in Figure 3.5(i)–(l). Despite the spectral reflectances of the squares being quite different, the intensities of R_1 shown in orange in Figure 3.5(i) closely match those of the desired solution L_1 in Figure 3.5(a). Note that R_1 is kept close to A_1 (shown in Figure 3.5(e)) by the likelihood term, while the sparse norm on the spectral terms removes the noise.

Eq. 3.2 may be optimized for any $\alpha \leq 1$ with the fast numerical algorithm of Chapter 4, which takes less than a minute for a megapixel image to process all 3 color channels. This is faster than the cross-bilateral filter while giving superior quality results.

3.4.3 Pre & post-processing

Pre-processing. All images were captured in RAW mode. They were then demosaiced and manually white-balanced using some neutral-colored object (e.g. a wall or calibration target) in the scene. The mask $m(p)$ was built using the same methods used in [128], namely the shadows were detected by finding areas where $|F - A|$ is very small. Specularities were found by looking for pixels saturated in F_1 (IR channel). In areas of shadow/specularity $m(p) = 5$ and $m(p) = 1$ in all other areas, smoothly varying between the two at the boundaries. In high noise conditions, we apply a small Gaussian smoothing to A_j to break up any spurious image structure formed by the noise. The optimization is then performed on the linear tonescale images (i.e. without gamma correction).

Post-processing. If the ambient light levels are very low, the colors in the ambient image can become imbalanced, particularly with a blue tint due to excessive noise levels in the blue channel. Hence the output of the optimization will also have a similar color cast and will not look similar to a long-exposure shot L. To compensate for this, we use an additional color correction operation that applies a global color mapping to R. To generate this mapping function, we determined the tone response curve of our camera for each color channel using a stack of images taken over a wide range of exposures [44]. Particular care was taken when fitting the parametric model to the low intensity part of the curve. In this regime, the sensor noise causes the curve to be non-linear, in turn giving rise to the color casts observed in very noisy images (e.g. Figure 3.6). By passing each R_j through its appropriate mapping function, we can infer the true value of each pixel, yielding colors close to those in a long-exposure shot L. Finally, we gamma-correct the images

for display, using $\gamma = 1.8$.

3.5 Results

3.5.1 Comparison experiments

We compare our method to a range of different hardware and software approaches. In Figure 3.13 we explore in turn the importance of having UV and IR in our dark flash by removing the corresponding spectral term in the cost function of Eq. 3.2. The figure shows the need for both the UV and IR components, since if either is removed, the adjacent spectral bands (blue and red, respectively) in R become degraded.

For the dark flash system to be practical it must achieve high quality reconstructions in low levels of ambient illumination. In Figure 3.6, Figure 3.7, Figure 3.8 and Figure 3.9, we show 4 test examples: two portrait shots and two still scenes. The test images were captured using two different types of ambient illumination (tungsten and compact fluorescent) and contain a wide range of materials and colors. The images in Figure 3.6, Figure 3.7, Figure 3.8 and Figure 3.9 are high resolution so are best viewed under magnification, in order that fine details and noise may be seen. To show how the noise levels vary across color channel we show a small region in two of the images, separated out into its constituent color planes. This typically reveals the blue channel to be far noisier than the others.

In Figure 3.10, Figure 3.12 and Figure 3.11 we show an example of denoising with different skin tones.The scene illumination was below 1 Lux in all cases. This is dimmer than the illumination form a candle. In Figure 3.10, we compare our result to that of a state-of-the-art denoising method, BM3D [40]. Our method is able to suppress the noise and recover edges better than BM3D, althoughthere are some color artifacts on the coat. In Figure 3.12 and Figure 3.11, we compare our result with the output of a Sony camera, a state-of-the-art camera enabling low-light photography. The Sony images are taken without flash. Our output has less noise and better image recovery.

To make comparisons straightforward, the shutter speed used to capture the flash/no-flash pair is varied, thus simulating different levels of ambient illumination. In practice however, the shutter speed would be set to the slowest level that avoids camera shake, irrespective of the level of ambient light. As the light levels drop, the ambient image becomes noisier (the dark flash image F stays constant, however) thus making the reconstruction harder. Three different noise scenarios are explored: (i) Low, where it is possible to achieve reconstructions close to a long exposure reference shot; (ii) Medium, where the reconstruction is acceptable in terms of quality and (iii) High, where a significant degradation in quality is visible and the failure modes of the algorithm are evident. At each noise level, the degree of under-exposure of the ambient image A, relative to the long exposure reference L, is quoted. These range from 1/32nd of ambient illumination (Figure 3.8(top)), down to 1/256th for the portrait shots. Assuming 1/30th of a second is required to avoid camera shake, the results are equivalent to taking pictures in conditions where

exposures ranging from 1 second to 8 seconds would otherwise be required. Techniques that permit blur-free photography at slow shutter speeds, such as image stabilizers, would extend the range of operation of the dark flash system to even longer equivalent exposures.

Ensuring accurate alignment between F and A is an important practical issue since the spectral constraints require this. While a range of software approaches for image registration exist (e.g. [6]), any commercial implementation of the system would use a hardware approach based on sensors that can capture pairs of images with virtually no delay between them (e.g. Fuji Finepix Z10fd), guaranteeing good alignment. With our prototype, we sidestep this issue and capture the shots using a tripod. It is difficult to draw comparisons with Petschnigg et al. [128] since they do not specify the exposures used to capture their images, but qualitatively the majority of their examples correspond to our low noise case, with a single case being equivalent to our medium noise level.

At high noise levels, some color deviations and loss of detail can be observed. This is a consequence of low μ_j values which give the likelihood term little weight in the optimization. At all noise levels, our reconstructions contain some artifacts that result from the dark flash illumination. If a material absorbs both UV and IR strongly, then F will contain no gradients to guide the reconstruction. Examples of this include: the freckles on the man in Figure 3.1 & Figure 3.7 and the red lips of the doll in Figure 3.8. Fortunately, this is relatively uncommon, as demonstrated by the range of colors and materials in our shots, the vast majority of which are accurately recovered. In particular, human skin and hair, two materials relevant to the dark flash application, are plausibly reproduced.

In Figure 3.14 we compare our algorithm to alternate methods, using the mid-noise case. First, we use the processing pipeline based on the cross-bilateral filter and detail enhancement, as described in [128]. Using the dark flash/ambient image pair with their system, the results obtained are inferior to our approach. The range term in the cross-bilateral filter causes the edge strength in the flash image F to directly influence the smoothing of the ambient image A. Thus it will only operate correctly if the edges in F and A are closely matched in magnitude, an unrealistic assumption since spectral reflectances typically differ between bands. In contrast, our model permits the edge magnitudes to differ when $\alpha \leq 1$ in Eq. 3.2, giving a reconstruction of superior quality. Second, we tried two approaches that attempt to directly denoise the ambient image: (i) bilateral filtering [159] and (ii) a commercial denoising tool, Noise Ninja [35]. Both methods perform poorly compared to the flash/no-flash approaches.

In Figure 3.15 we explore how the value of α in Eq. 3.2 effects the reconstruction. When a non-sparse norm is used ($\alpha = 2$), the ambient colors bleed. This can be prevented by using $\alpha \leq 1$, with some improvement in quality for $\alpha = 0.7$.

3.5.2 Fluorescence

Certain materials fluoresce when illuminated by the UV component of our flash, the most common instances being white items of clothing such as the stripes in Figure 3.6. Fluorescence manifests

itself as visible blue light that gives an unnaturally bright intensity in F_3 in that part of the scene. Experimentally, we find the phenomenon to be relatively rare: our test scenes contain a wide range of materials, natural and man-made, yet it only occurs in a few locations. It is certainly not the dominant source of signal in F_3, as demonstrated by Figure 3.3(bottom). Where it does occur, it can produce some minor purple artifacts. Another drawback is that other people observing the subjects during the photograph may see a glow from the clothing, thus making the flash not so invisible to them, although the subjects themselves, if looking at the camera, will not notice this.

3.5.3 Photometric flash measurements

One of the main objectives of our dark flash is that it should be as unnoticeable as possible to human subjects. We measured the dark flash output with a spectrometer to determine the spectral irradiance (shown in Figure 3.2(b)) 1m from the flash. This was then converted to photometric units, using the photopic luminosity function of Vos [163]. The luminous exposure for the dark flash was 1.6 lux seconds. A visible flash set to produce an image V of similar intensity to a dark flash image F had luminous exposure of 362 lux seconds, a factor of 226 times brighter. This ratio agrees closely with the experiment of Figure 3.3(top right) where an attenuation of 220 times was required to make the visible flash of comparable brightness to the dark flash. In Figure 3.6 and Figure 3.7, we show images D captured with a visible flash attenuated by this factor. The resulting images are unacceptably noisy.

Subjectively, people report that when looking directly at the flash they see a weak purple light that does not dazzle, or leave an after-image. They also report that if not looking directly at the dark flash, the burst of light is very easy to miss. By contrast, when using a visible flash that gives a comparable scene exposure, the burst of light is highly dazzling and leaves a strong after-image.

3.6 Other applications

Although our main focus has been the dark flash application, both the hardware and software elements of our system can be used in a variety of other ways.

3.6.1 Estimation of spectral reflectance

By taking two images, one with the dark flash, the other with a visible flash, we can obtain 5 different spectral measurements at each point in the scene: UV,B,G,R,IR as opposed to 3 obtained with a conventional camera. The spectral reflectances of real world materials can be accurately modeled in a low-dimensional subspace using PCA with relatively few components [164]. Using a spectrometer and reflectance probe, we measured 255 different materials in the real world and computed a set of 5 PCA basis functions for the range 360–800nm. We then used the constrained

least squares formulation introduced in [126] to solve for the spectral reflectance functions for all points in the scene ($S(p, \lambda)$ in Eq. 3.1). In Figure 3.16(left), we show the estimated spectral reflectance for four squares from the color chart in Figure 3.8, along with ground truth. Note that we are able to accurately infer the spectrum beyond the visible range. In Figure 3.16(right) we compare the RMS error between our spectra and the ground truth over the visible range. We achieve very similar total error to the approach of Park et al. [126]: 0.82 and 0.79 respectively, compared to 1.19 when using R,G,B channels alone.

3.6.2 Color-band denoising

The spectral constraints used in our dark flash approach can be applied to images captured by standard cameras. One example, as shown in Figure 3.17, is for conventional flash/no-flash processing, using a visible flash/ambient pair. When using our algorithm in this configuration, the spectral constraint reduces to a single term linking each channel in the flash image to its corresponding channel in the ambient, hence the term no longer links between different spectral bands. Our algorithm yields better results than the cross-bilateral based method.

Another application is where one color channel is much noisier than the others. For example, candle-light is very weak in the blue part of the spectrum, compared to red and green. Hence when trying to white balance a candle-lit image, the blue channel must be multiplied by a large factor, increasing the noise levels. Using spectral constraints, the blue channel can be denoised using the red and green channels (in place of F_1 and F_3 in Eq. 3.2). This gives a superior result to denoising the blue channel using spatial priors and likelihood alone. See Figure 3.18 for this technique applied to a candle-lit image captured with an unmodified Canon 40D.

3.7 Discussion

We have demonstrated a camera and flash system that can take pictures in low light conditions using a flash that is far less noticeable and disruptive than a conventional one. The system uses standard hardware for the most part, combined with novel image processing techniques. The spectral constraints are a powerful way of combining the images, yielding good quality results in low light conditions. In addition, we have shown that the hardware and software techniques introduced in this paper can be used in a number of other applications.

Our hardware is a prototype and can be improved in a number of ways. An obvious limitation is the need to take two images of the scene. This precludes the capture of fast moving scenes and adds to the overall complexity of the system. However, by modifying the Bayer pattern on the sensor to include UV-only and IR-only pixels (for a total of 5 channels), we would be able to implement the dark flash concept using a single image. Additionally, our large flash unit could be replaced with compact UV and IR LEDs giving a more controllable pulse duration and a more precise spectral emission, perhaps further reducing the visibility of the flash. This would also permit the dark flash concept to be implemented in small platforms such as cell-phones,

where a flash is often needed due to poor low-light performance on account of the small sensor size. Recently, new cameras with "inverted CMOS" sensors have been introduced which improve the low-light performance of standard cameras. Our algorithm may be used in conjunction with these new sensors to enable photographs at even lower light levels than conventional sensors.

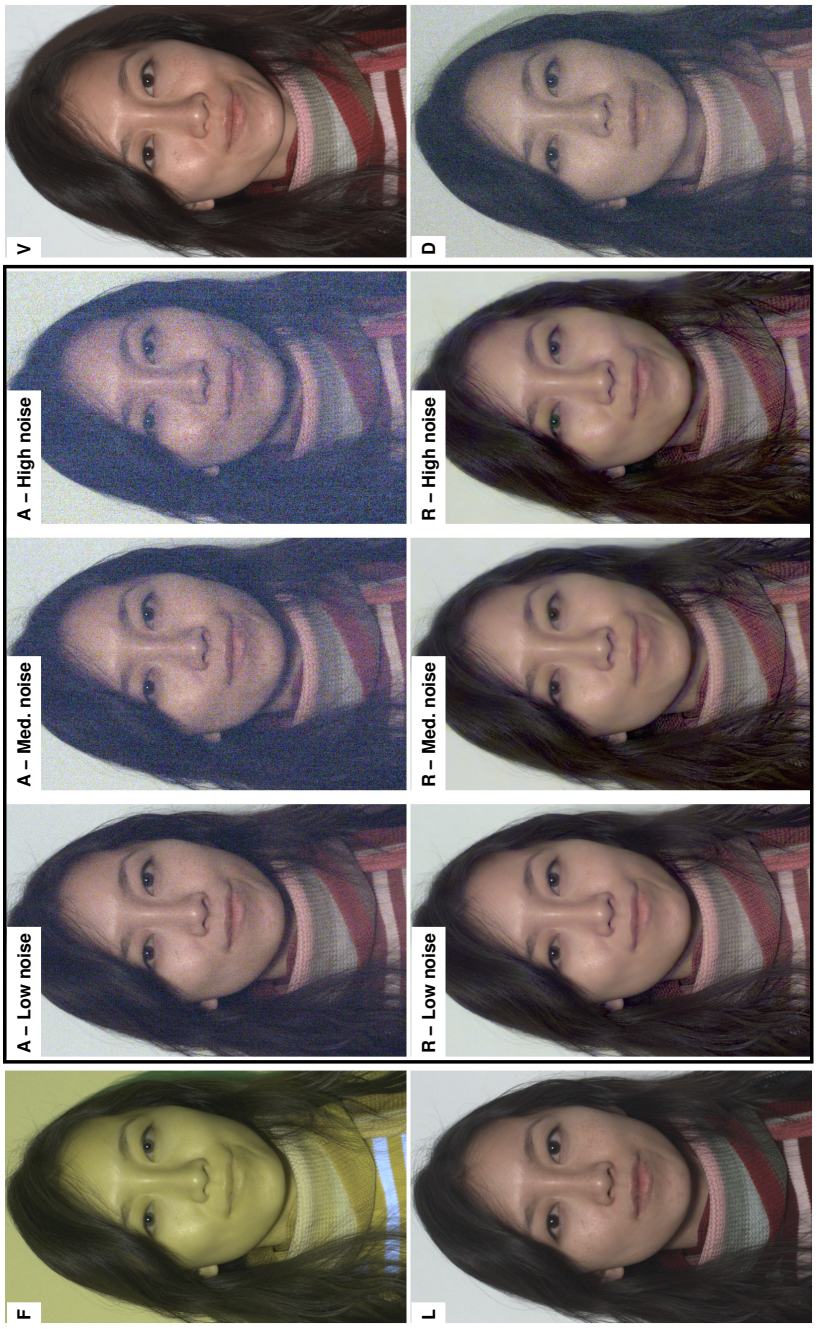


Figure 3.6: A portrait shot captured with our camera/flash under tungsten illumination. Column 1 shows the dark flash shot (F) and long exposure reference (L). Our results are shown in Columns 2,3 & 4. For each ambient image (A) of decreasing exposure (yielding increased noise), we show the reconstructed output (R). Column 5 shows a visible flash image (V), along with a visible flash shot (D) attenuated with neutral density filters so that it is comparably dazzling to F. The Low, Medium and High noise levels correspond to 6, 7 and 8 stops of underexposure respectively (corresponding to 1/64th, 1/128th and 1/256th of ambient long exposure).

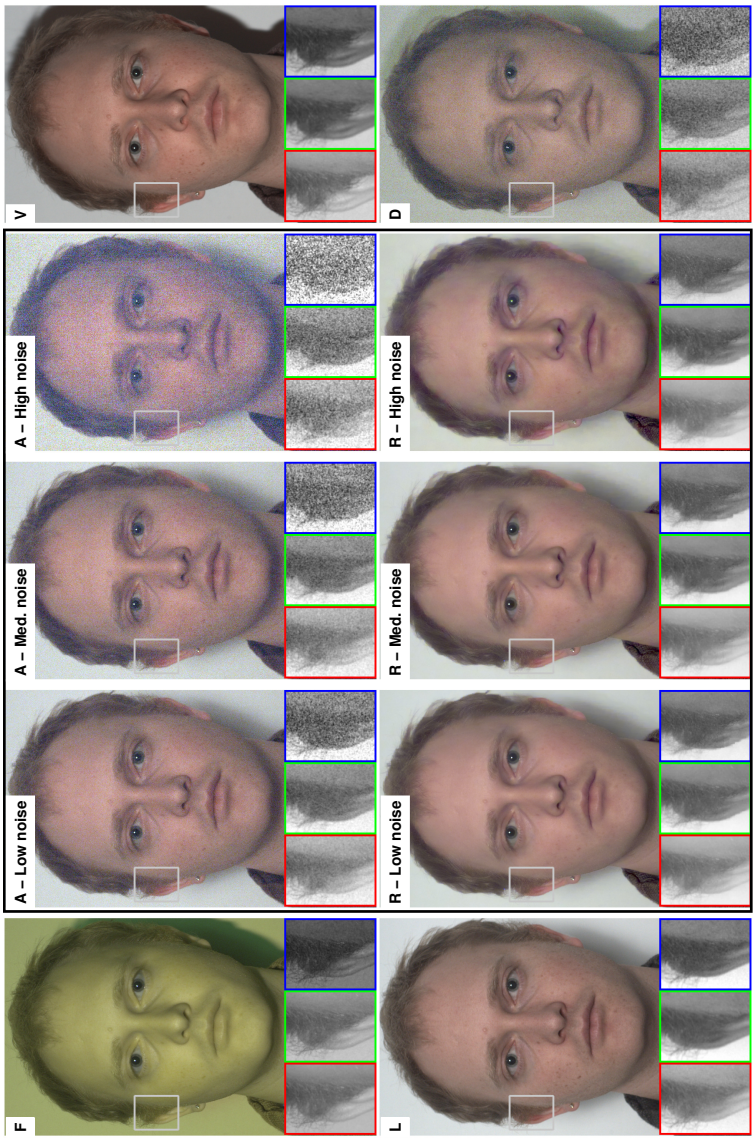


Figure 3.7: A portrait shot captured with our camera/flash under tungsten illumination. Column 1 shows the dark flash shot (F) and long exposure reference (L). Our results are shown in Columns 2,3 & 4. For each ambient image (A) of decreasing exposure (yielding increased noise), we show the reconstructed output (R). Column 5 shows a visible flash image (V), along with a visible flash shot (D) attenuated with neutral density filters so that it is comparably dazzling to F. The Low, Medium and High noise levels correspond to 6, 7 and 8 stops of underexposure respectively (corresponding to 1/64th, 1/128th and 1/256th of ambient long exposure). We also show a zoomed-in section, separated into red, green, blue color channels.

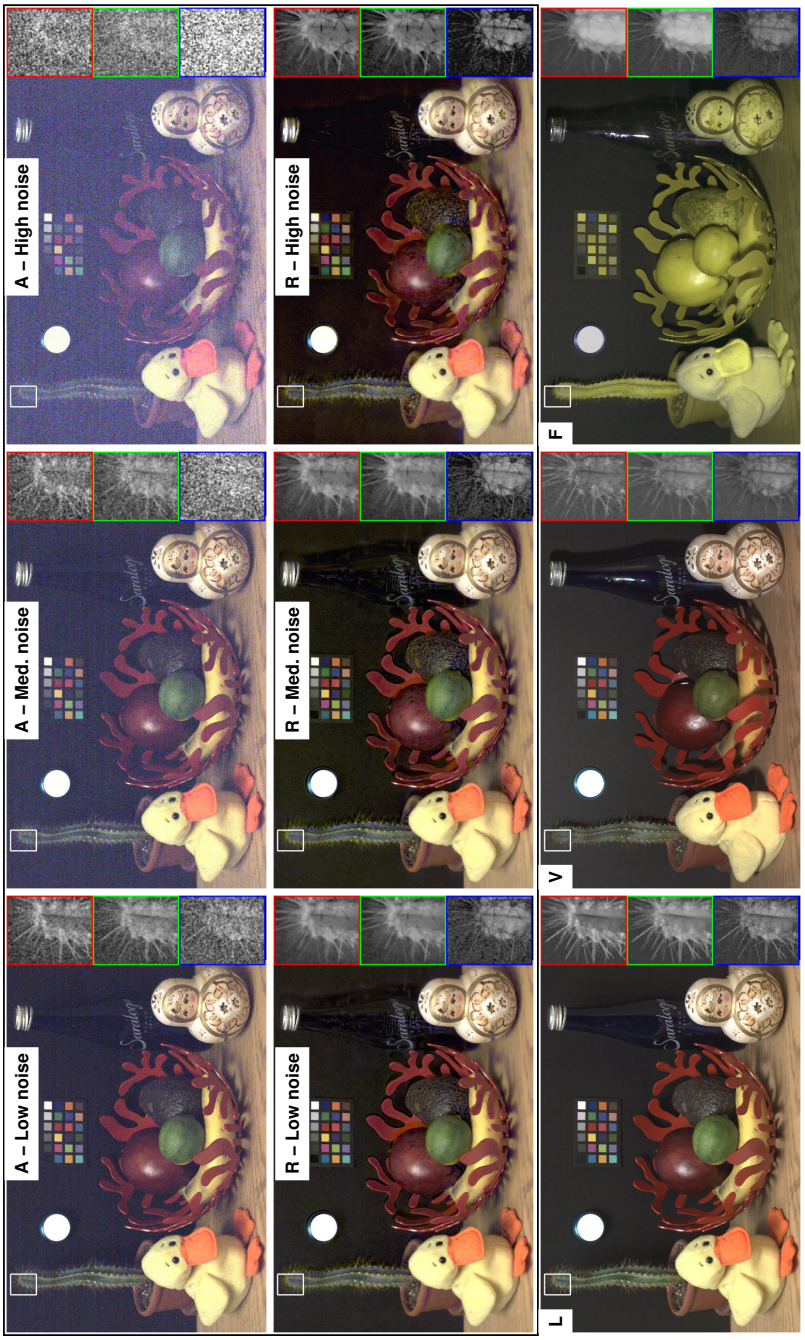


Figure 3.8: A scene captured with our camera/flash under fluorescent illumination. Rows 1 & 2 show shots under ambient illumination (A) of decreasing exposure (yielding increased noise) and our reconstructed output (R). Row 3 shows, from left to right: Long exposure reference (L), Visible flash shot (V) and dark flash shot (F). Low, Medium and High noise levels correspond to 5, 6 and 7 stops of underexposure respectively (equating to 1/32nd, 1/64th and 1/128th of ambient long exposure).

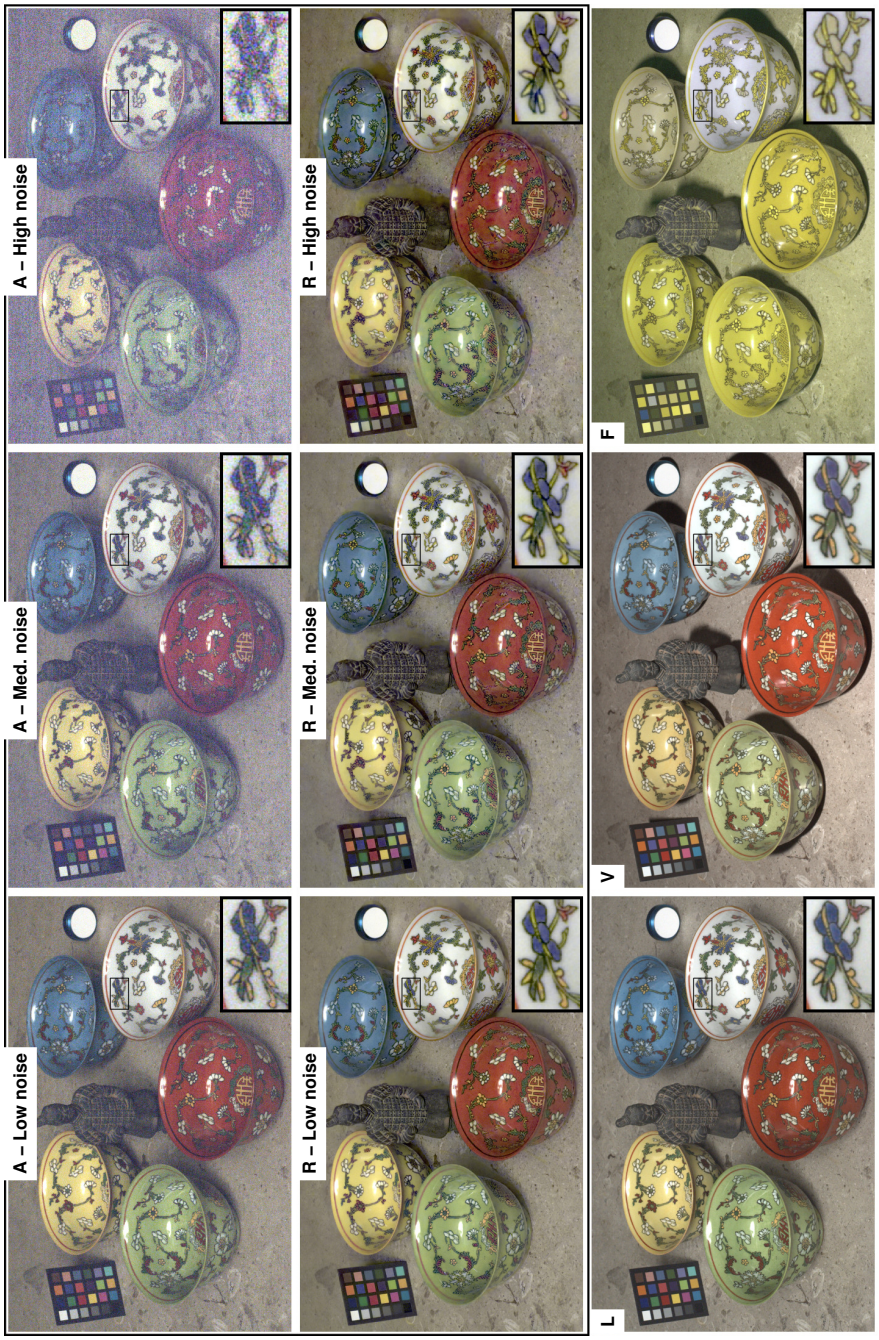


Figure 3.9: A scene captured with our camera/flash under fluorescent illumination. Rows 1 & 2 show shots under ambient illumination (A) of decreasing exposure (yielding increased noise) and our reconstructed output (R). Row 3 shows, from left to right: Long exposure reference (L), Visible flash shot (V) and dark flash shot (F). Low, Medium and High noise level correspond to 5.5, 6.5 and 7.5 stops underexposed (corresponding to 1/45th, 1/90th and 1/180th of ambient long exposure).

Figure 3.10: Denoising at extremely low light levels (0.9 Lux). Top-left: Flash shot; Top-right: Ambient; Bottom-left: Our reconstruction; Bottom-right: Best reconstruction of BM3D [40].

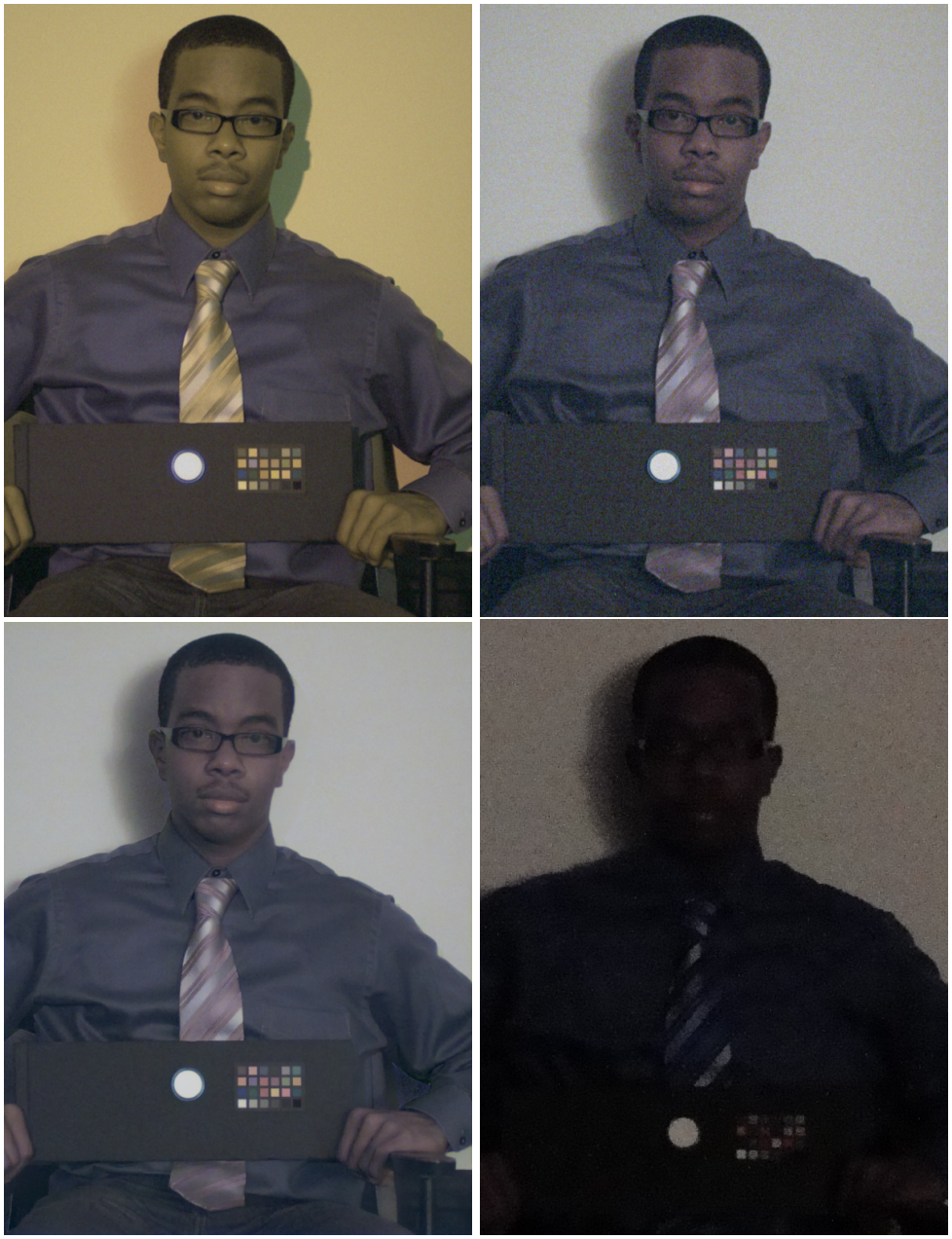


Figure 3.11: Denoising at extremely low light levels (0.7 Lux). Top-left: UV/IR Flash shot; Top-right: Noisy Ambient; Bottom-left: Our reconstruction; Bottom-right: Image taken from Sony NightShot camera.

Figure 3.12: Denoising at extremely low light levels (0.7 Lux). Top-left: UV/IR Flash shot; Top-right: Noisy Ambient; Bottom-left: Our reconstruction; Bottom-right: Image taken from Sony NightShot camera.

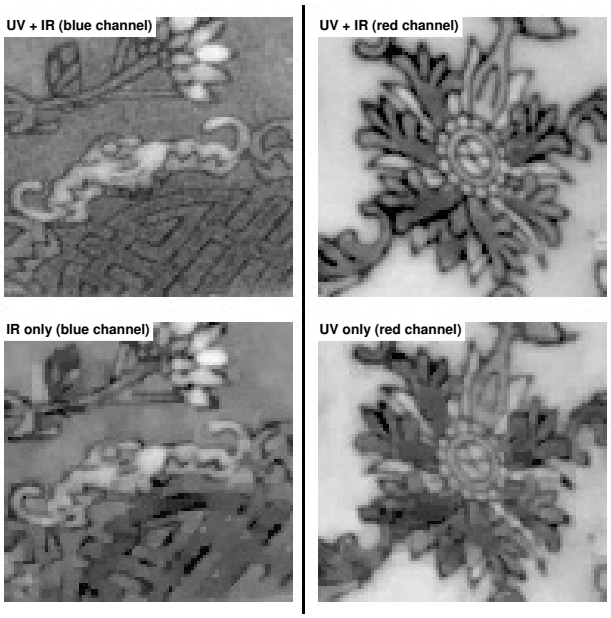


Figure 3.13: Closeup of Figure 3.8 , showing the need for both spectral terms in Eq. 3.2. Top left: Blue channel of reconstructed image R using both UV and IR spectral terms. Bottom left: Blue channel using only IR spectral term. Top right: Red channel of reconstructed image R using both UV and IR spectral terms. Bottom right: Red channel using only UV spectral term. Note that the removal of the flash in the adjacent band causes a degraded result.

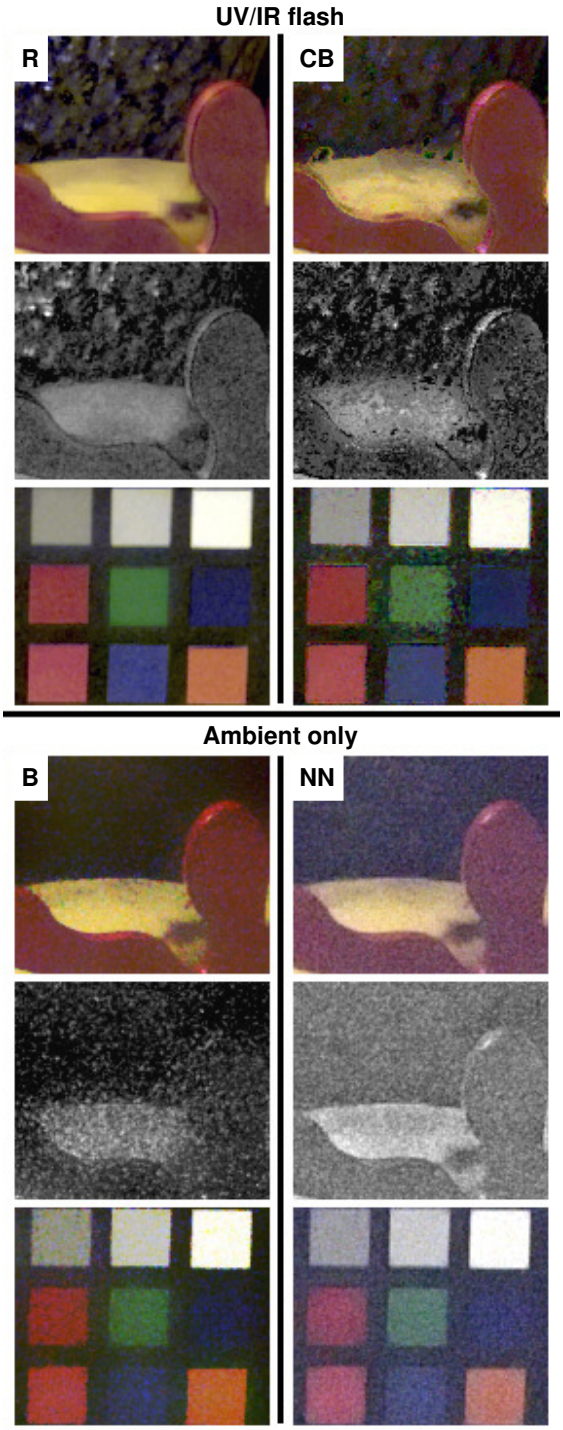


Figure 3.14: Comparison of our approach to different processing methods, showing two crops from Figure 3.8, along with the blue channel of the first crop. The top set uses a dark flash / ambient image pair, while the bottom uses the ambient image only. Key. R: Our reconstruction using spectral constraints. CB: Pipeline from [128] based on cross-bilateral filter and detail enhancement. B: Bilateral filter of ambient image [159]. NN: Noise Ninja commercial denoising plugin for Photoshop [35]. Our reconstruction approach produces superior results to the cross-bilateral approach and the standard denoising methods.

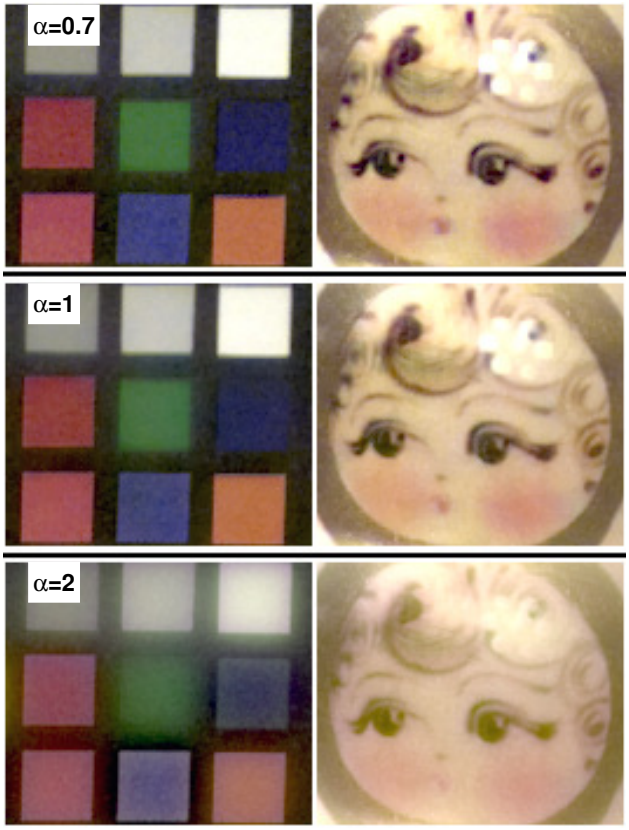


Figure 3.15: Effect of varying α in Eq. 3.2. For values ≤ 1, R contains crisp edges, even if the spectral reflectances of the materials in visible and non-visible wavelengths differ somewhat, as is typically the case. Setting $\alpha = 2$ has the undesirable effect of causing the colors to bleed between regions. When $\alpha = 2$ the spectral constraints force the edges in the UV/IR flash and ambient to be the same, an unrealistic assumption given that they are captured at different wavelengths.

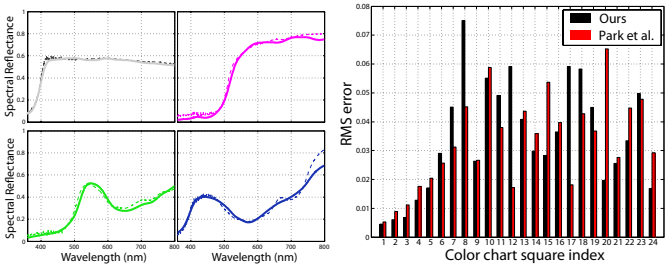


Figure 3.16: Using a dark/visible flash pair we are able to accurately infer the spectral reflectance of objects. Left: Spectra of four different squares from the color chart in Figure 3.8. Solid line is inferred spectrum, dashed line is ground truth. Line colors correspond to square color. Right: RMS estimation errors for all 24 squares in color chart over 400-700nm range, compared to results of multi-spectral illumination approach of Park et al. [126].

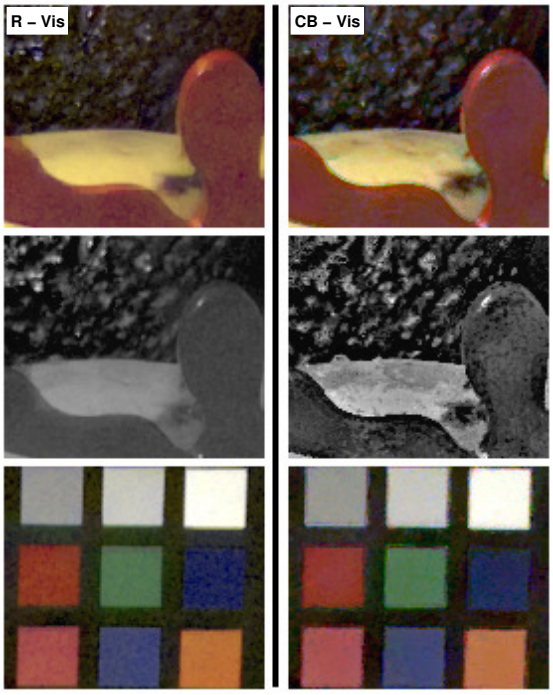


Figure 3.17: The model in Eq. 3.2 being used in a visible flash/no-flash setting. The two crops are taken from Figure 3.8, with the center row showing the the blue channel of the first row. R - Vis: reconstruction with our model using spectral constraints. CB - Vis: Pipeline from [128] based on cross-bilateral filter and detail enhancement.

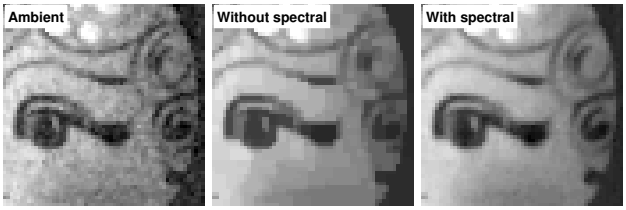


Figure 3.18: Close up of scene in Figure 3.8 illuminated by candlelight. Left: Blue channel of white-balanced ambient shot, showing high noise due to lack of blue wavelengths in candlelight. Middle: Denoising of ambient using likelihood and spatial priors only. Right: Denoising of ambient using spectral constraints from the red and green channels, in addition to the likelihood and spatial priors. The spectral constraints significantly improve performance.

Chapter 4

Fast Image Deconvolution Using Hyper-Laplacian Priors

4.1 Introduction

In this chapter, we consider the development of a novel and efficient non-blind deconvolution algorithm. Non-blind deconvolution seeks to deblur an image when the blur kernel is *known*. There are many approaches, both in hardware and software, to estimate the blur kernel; these will be considered in (Chapter 5). This is joint work with Rob Fergus and was published in NIPS 2009 [87].

Natural image statistics have been used very effectively as priors for problems in image processing, computer vision and computational photography. Some examples are denoising [129], deblurring [55], transparency separation [100] and super-resolution [155]. Further examples are provided in Chapter 2. Priors based on natural image statistics can regularize ill-posed problems to yield high-quality results. However, digital cameras now have sensors that record images with tens of megapixels (MP), e.g. the latest Canon DSLRs have over 20MP. Solving the above tasks for such images in a reasonable time frame (i.e. a few minutes or less), poses a severe challenge to existing algorithms. This is usually (but not always) because the regularizers are non-convex. In this paper we focus on one particular problem: non-blind deconvolution, and propose an algorithm that is practical for very large images while still yielding high quality results.

Numerous non-blind deconvolution approaches exist, varying greatly in their speed and sophistication. Simple filtering operations such as classical Wiener deconvolution are very fast but typically yield poor results. The reasons for this are twofold: blurring attenuates high frequencies and a naive boosting of these frequencies leads to ringing artifacts; real-world blurred images have noise in them and Wiener deconvolution is highly sensitive to the noise parameter setting. Most of the best-performing approaches [96, 139] solve globally for the corrected image, encouraging the marginal statistics of a set of filter outputs to match those of uncorrupted images.

These statistics act as a prior to regularize the problem. For these methods, a trade-off exists between accurately modeling the image statistics and being able to solve the ensuing optimization problem efficiently. If the marginal distributions are assumed to be Gaussian, a closed-form solution exists in the frequency domain and FFTs can be used to recover the image very quickly - this is essentially Wiener deconvolution.

However, real-world images typically have marginals that are non-Gaussian, as shown in Figure 4.1, and thus the output is often of mediocre quality. A common approach is to assume the marginals have a Laplacian distribution. Since the likelihood (image formation) term is quadratic and the Laplacian form is convex, the overall resulting problem is convex. This allows a number of fast ℓ_1 and related TV-norm methods [137, 167] to be deployed, which give good results in a reasonable time. However, studies of real-world images have shown the marginal distributions have significantly heavier tails than a Laplacian, being well modeled by a hyper-Laplacian [56, 96, 144]. Although such priors give the best quality results, they are typically far slower than methods that use either Gaussian or Laplacian priors. This is a direct consequence of the problem becoming non-convex for hyper-Laplacians with $\alpha < 1$, meaning that many of the fast ℓ_1 or ℓ_2 tricks are no longer applicable. Instead, standard optimization methods such as conjugate gradient (CG) must be used. One variant that works well in practice is iteratively reweighted least squares (IRLS) [150] that solves a series of weighted least-squares problems with CG, each one an ℓ_2 approximation to the non-convex problem at the current point. In both cases, typically hundreds of CG iterations are needed, each involving an expensive convolution of the blur kernel with the current image estimate.

Recent papers have highlighted the benefits of using non-convex regularizers in the context of compressive sensing [30, 29, 31, 42]. The results from these papers can be summarized as: it is beneficial to use non-convex regularizers of the form $\|x\|^p, p < 1$, to promote greater sparsity in the solution x; (ii) using $p < 1$ can give sparser solutions than $p = 1$; and (iii) IRLS can give very good performance even in the non convex problems. In the scenario of non-blind deconvolution, we are forced into the regime of non-convex regularizers owing to the nature of the problem. An interesting direction for future work would be to see if results from the compressive sensing literature can be used to further improve non-blind deconvolution.

We introduce an efficient scheme for non-blind deconvolution of images using a hyper-Laplacian image prior for $0 < \alpha \leq 1$. Our algorithm uses an alternating minimization scheme where the non-convex part of the problem is solved in one phase, followed by a quadratic phase which can be efficiently solved in the frequency domain using FFTs. We focus on the first phase where at each pixel we are required to solve a non-convex separable minimization. We present two approaches to solving this sub-problem. The first uses a lookup table (LUT); the second is an analytic approach specific to two values of α. For $\alpha = 1/2$ the global minima can be determined by finding the roots of a cubic polynomial analytically. In the $\alpha = 2/3$ case, the polynomial is a quartic whose roots can also be found efficiently in closed-form. Both IRLS and our approach solve a series of approximations to the original problem. However, in our method each approximation is solved by alternating between the two phases above a few times, thus avoiding the expensive CG descent used by IRLS. This allows our scheme to operate several orders of magnitude faster. Although

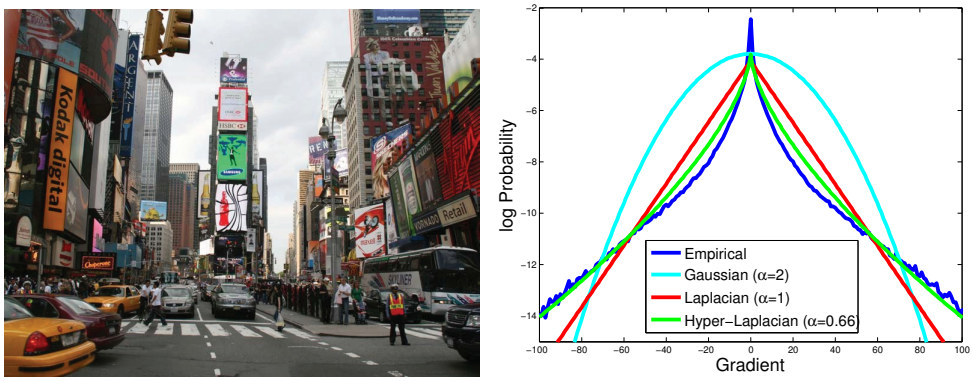


Figure 4.1: A hyper-Laplacian with exponent $\alpha = 2/3$ is a better model of image gradients than a Laplacian or a Gaussian. **Left:** A typical real-world scene. **Right:** The empirical distribution of gradients in the scene (blue), along with a Gaussian fit (cyan), a Laplacian fit (red) and a hyper-Laplacian with $\alpha = 2/3$ (green). Note that the hyper-Laplacian fits the empirical distribution closely, particularly in the tails.

we focus on the problem of non-blind deconvolution, it would be straightforward to adapt our algorithm to other related problems, such as denoising or super-resolution.

The alternating minimization that we adopt is closely related to half-quadratic splitting, [61, 62, 167]. We also use a half-quadratic minimization, but the per-pixel sub-problem is quite different. With the TV norm it can be solved with a straightforward shrinkage operation. In our work, as a consequence of using a sparse prior, the problem is non-convex and solving it efficiently is one of the main contributions of this paper.

Chartrand [29, 30] has introduced non-convex compressive sensing, where the usual ℓ_1 norm on the signal to be recovered is replaced with a ℓ_p quasi-norm, where $p < 1$, resulting in a non-convex per-pixel sub-problem. To solve this, a Huber approximation (see [29]) to the quasi-norm is used. This allows the derivation of a generalized shrinkage operator to solve the non-convex sub-problem efficiently. However, this approximates the original sub-problem.

4.2 Algorithm

We now introduce the non-blind deconvolution problem. $\mathbf{x}$ is the original uncorrupted linear grayscale image of N pixels; $\mathbf{y}$ is an image degraded by blur and/or noise, which we assume to be produced by convolving $\mathbf{x}$ with a blur kernel $\mathbf{k}$ and adding zero mean Gaussian noise. We assume that $\mathbf{y}$ and $\mathbf{k}$ are given and seek to reconstruct $\mathbf{x}$. Given the ill-posed nature of the task, we regularize using a penalty function $|.|^\alpha$ that acts on the output of a set of filters $f_1, \ldots, f_j$ applied to $\mathbf{x}$. A weighting term λ controls the strength of the regularization. From a probabilistic perspective, we seek the MAP estimate of $\mathbf{x}$: $p(\mathbf{x}|\mathbf{y}, \mathbf{k}) \propto p(\mathbf{y}|\mathbf{x}, \mathbf{k})p(\mathbf{x})$, the first term being a Gaussian likelihood and second being the hyper-Laplacian image prior. Maximizing $p(\mathbf{x}|\mathbf{y}, \mathbf{k})$ is equivalent to minimizing the cost $-\log p(\mathbf{x}|\mathbf{y}, \mathbf{k})$:

$$\min_{\mathbf{x}} \sum_{i=1}^{N} \left(\frac{\lambda}{2} (\mathbf{x} \oplus \mathbf{k} - \mathbf{y})_i^2 + \sum_{j=1}^{J} |(\mathbf{x} \oplus f_j)_i|^\alpha \right) \tag{4.1}$$

where i is the pixel index, and $\oplus$ is the 2-dimensional convolution operator. For simplicity, we use two first-order derivative filters $f_1 = [1\ \text{-}1]$ and $f_2 = [1\ \text{-}1]^T$, although additional ones can easily be added (e.g. learned filters [124, 136], or higher order derivatives). For brevity, we denote $F_i^j \mathbf{x} \equiv (\mathbf{x} \oplus f_j)_i$ for $j = 1, .., J$.

Using the half-quadratic penalty method [61, 62, 167], we now introduce auxiliary variables w_i^1 and w_i^2 (together denoted as $\mathbf{w}$) at each pixel that allow us to move the $F_i^j \mathbf{x}$ terms outside the $|.|^\alpha$ expression, giving a new cost function:

$$\min_{\mathbf{x},\mathbf{w}} \sum_{i} \left(\frac{\lambda}{2} (\mathbf{x} \oplus \mathbf{k} - \mathbf{y})_i^2 + \frac{\beta}{2} \left(\|F_i^1 \mathbf{x} - w_i^1\|_2^2 + \|F_i^2 \mathbf{x} - w_i^2\|_2^2 \right) + |w_i^1|^\alpha + |w_i^2|^\alpha \right) \tag{4.2}$$

where β is a weight that we will vary during the optimization, as described in Section 4.2.3. As $\beta \to \infty$, the solution of Eq. 4.2 converges to that of Eq. 4.1. Minimizing Eq. 4.2 for a fixed β can be performed by alternating between two steps, one where we solve for $\mathbf{x}$, given values of $\mathbf{w}$ and vice-versa. The novel part of our algorithm lies in the $\mathbf{w}$ sub-problem, but first we briefly describe the $\mathbf{x}$ sub-problem and its straightforward solution.

4.2.1 x sub-problem

Given a fixed value of $\mathbf{w}$ from the previous iteration, Eq. 4.2 is quadratic in $\mathbf{x}$. The optimal $\mathbf{x}$ is thus:

$$\left({F^1}^T F^1 + {F^2}^T F^2 + \frac{\lambda}{\beta} K^T K \right) \mathbf{x} = {F^1}^T \mathbf{w}^1 + {F^2}^T \mathbf{w}^2 + \frac{\lambda}{\beta} K^T \mathbf{y} \tag{4.3}$$

where $K\mathbf{x} \equiv \mathbf{x} \oplus \mathbf{k}$. Assuming circular boundary conditions, we can apply 2D Fast Fourier Transforms (FFT) which diagonalize the convolution matrices F^1, F^2, K, enabling us to find the optimal $\mathbf{x}$ directly:

$$\mathbf{x} = \mathcal{F}^{-1} \left(\frac{\mathcal{F}(F^1)^* \circ \mathcal{F}(\mathbf{w}^1) + \mathcal{F}(F^2)^* \circ \mathcal{F}(\mathbf{w}^2) + (\lambda/\beta) \mathcal{F}(K)^* \circ \mathcal{F}(\mathbf{y})}{\mathcal{F}(F^1)^* \circ \mathcal{F}(F^1) + \mathcal{F}(F^2)^* \circ \mathcal{F}(F^2) + (\lambda/\beta) \mathcal{F}(K)^* \circ \mathcal{F}(K)} \right) \tag{4.4}$$

where $*$ is the complex conjugate; $\circ$ denotes component-wise multiplication; $\mathcal{F}$ is the forward FFT; and $\mathcal{F}^{-1}$ is the inverse FFT. The division is also performed component-wise. Solving Eq. 4.4 requires only 3 FFT's at each iteration since many of the terms can be precomputed. The form of this sub-problem is identical to that of [167].

4.2.2 w sub-problem

Given a fixed $\mathbf{x}$, finding the optimal $\mathbf{w}$ consists of solving $2N$ independent 1D problems of the form:

$$w^* = \arg\min_w \ |w|^\alpha + \frac{\beta}{2}(w - v)^2 \tag{4.5}$$

where $v \equiv F_i^j \mathbf{x}$. We now describe two approaches to finding w^*.

Lookup table

For a fixed value of α, w^* in Eq. 4.5 only depends on two variables, β and v, hence can easily be tabulated off-line to form a lookup table. The cost function in Eq. 4.5 is smooth away from 0, and so interpolation using a LUT is accurate for the purposes of image deconvolution. We numerically solve Eq. 4.5 for $10,000$ different values of v over the range encountered in our problem ($-0.6 \leq v \leq 0.6$). This is repeated for different β values, namely integer powers of $\sqrt{2}$ between 1 and 256. Although the LUT gives an approximate solution, it allows the $\mathbf{w}$ sub-problem to be (approximately) solved very quickly for any $\alpha > 0$.

Analytic solution

For some specific values of α, it is possible to derive exact analytical solutions to the $\mathbf{w}$ sub-problem. For $\alpha = 2$, the sub-problem is quadratic and thus easily solved. If $\alpha = 1$, Eq. 4.5 reduces to a 1-D shrinkage operation [167]. For some special cases of $1 < \alpha < 2$, there exist analytic solutions [179]. Here, we address the more challenging case of $\alpha < 1$ and we now describe a way to solve Eq. 4.5 for two special cases of $\alpha = 1/2$ and $\alpha = 2/3$. For non-zero w, setting the derivative of Eq. 4.5 w.r.t w to zero gives:

$$\alpha|w|^{\alpha-1}\text{sign}(w) + \beta(w - v) = 0 \tag{4.6}$$

For $\alpha = 1/2$, this becomes, with successive simplification:

$$|w|^{-1/2}\text{sign}(w) + 2\beta(w - v) = 0 \tag{4.7}$$

$$|w|^{-1} = 4\beta^2(v - w)^2 \tag{4.8}$$

$$w^3 - 2vw^2 + v^2 w - \text{sign}(w)/4\beta^2 = 0 \tag{4.9}$$

At first sight Eq. 4.9 appears to be two different cubic equations with the $\pm 1/4\beta^2$ term, however we need only consider one of these as v is fixed and w^* must lie between 0 and v. Hence we can replace sign(w) with sign(v) in Eq. 4.9:

$$w^3 - 2vw^2 + v^2 w - \text{sign}(v)/4\beta^2 = 0 \tag{4.10}$$

For the case $\alpha = 2/3$, using a similar derivation, we arrive at:

$$w^4 - 3vw^3 + 3v^2w^2 - v^3w + \frac{8}{27\beta^3} = 0 \tag{4.11}$$

there being no sign(w) term as it conveniently cancels in this case. Hence w^*, the solution of Eq. 4.5, is either 0 or a root of the cubic polynomial in Eq. 4.10 for $\alpha = 1/2$, or equivalently a root of the quartic polynomial in Eq. 4.10 for $\alpha = 2/3$. Although it is tempting to try the same manipulation for $\alpha = 3/4$, this results in a 5^{th} order polynomial, which can only be solved numerically (since no analytic solution exists). Other larger values of α result in polynomials of order above 4, and therefore our analytic approach is limited to these two special cases above.

Finding the roots of the cubic and quartic polynomials: Analytic formulae exist for the roots of cubic and quartic polynomials [170, 171] and they form the basis of our approach, as detailed in Algorithm 4.3 and Algorithm 4.3. In both the cubic and quartic cases, the computational bottleneck is the cube root operation. Analternative way of find the roots of the polynomials Eq. 4.10 and Eq. 4.11 is to use a numerical root-finder such as Newton-Raphson. In our experiments, we found Newton-Raphson to be slower and less accurate than either the analytic method or the LUT approach (see [88] for futher details).

Selecting the correct roots: Given the roots of the polynomial, we need to determine which one corresponds to the global minima of Eq. 4.5. When $\alpha = 1/2$, the resulting cubic equation can have: (a) 3 imaginary roots; (b) 2 imaginary roots and 1 real root, or (c) 3 real roots. In the case of (a), the $|w|^\alpha$ term means Eq. 4.5 has positive derivatives around 0 and the lack of real roots implies the derivative never becomes negative, thus $w^* = 0$. For (b), we need to compare the costs of the single real root and $w = 0$, an operation that can be efficiently performed using Eq. 4.13 below. In (c) we have 3 real roots. Examining Eq. 4.7 and Eq. 4.8, we see that the squaring operation introduces a spurious root above v when $v > 0$, and below v when $v < 0$. This root can be ignored, since w^* must lie between 0 and v. The cost function in Eq. 4.5 has a local maximum near 0 and a local minimum between this local maximum and v. Hence of the 2 remaining roots, the one further from 0 will have a lower cost. Finally, we need to compare the cost of this root with that of $w = 0$ using Eq. 4.13.

We can use similar arguments for the $\alpha = 2/3$ case. Here we can potentially have: (a) 4 imaginary roots, (b) 2 imaginary and 2 real roots, or (c) 4 real roots. In (a), $w^* = 0$ is the only solution. For (b), we pick the larger of the 2 real roots and compare the costs with $w = 0$ using Eq. 4.13, similar to the case of 3 real roots for the cubic. Case (c) never occurs: the final quartic polynomial Eq. 4.11 was derived with a cubing operation from the analytic derivative. This introduces 2 spurious roots into the final solution, both of which are imaginary, thus only cases (a) and (b) are possible.

In both the cubic and quartic cases, we need an efficient way to pick between $w = 0$ and a real root that is between 0 and v. We now describe a direct mechanism for doing this which does not involve the expensive computation of the cost function in Eq. 4.5[1].

[1]This requires the calculation of a fractional power, which is slow, particularly if $\alpha = 2/3$.

Let r be the non-zero real root. 0 must be chosen if it has lower cost in Eq. 4.5. This implies:

$$\begin{aligned} |r|^{\alpha} + \frac{\beta}{2}(r-v)^2 &> \frac{\beta v^2}{2} \\ \text{sign}(r)|r|^{\alpha-1} + \frac{\beta}{2}(r-2v) &\lessgtr 0 \quad , \quad r \lessgtr 0 \end{aligned} \tag{4.12}$$

Since we are only considering roots of the polynomial, we can use Eq. 4.6 to eliminate $\text{sign}(r)|r|^{\alpha-1}$ from Eq. 4.6 and Eq. 4.12, yielding the condition:

$$r \lessgtr 2v\frac{(\alpha-1)}{(\alpha-2)} \quad , \quad v \gtrless 0 \tag{4.13}$$

since $\text{sign}(r) = \text{sign}(v)$. So $w^* = r$ if r is between $2v/3$ and v in the $\alpha = 1/2$ case or between $v/2$ and v in the $\alpha = 2/3$ case. Otherwise $w^* = 0$. Using this result, picking w^* can be efficiently coded, e.g. lines 12–16 of Algorithm 4.3. Overall, the analytic approach is slower than the LUT, but it gives an exact solution to the $\mathbf{w}$ sub-problem.

4.2.3 Summary of algorithm

We now give the overall algorithm using a LUT for the $\mathbf{w}$ sub-problem. As outlined in Algorithm 1 below, we minimize Eq. 4.2 by alternating the $\mathbf{x}$ and $\mathbf{w}$ sub-problems T times, before increasing the value of β and repeating. Starting with some small value β_0 we scale it by a factor β_{Inc} until it exceeds some fixed value β_{Max}. In practice, we find that a single inner iteration suffices ($T = 1$), although more can sometimes be needed when β is small.

Algorithm 1 Fast image deconvolution using hyper-Laplacian priors

Require: Blurred image $\mathbf{y}$, kernel $\mathbf{k}$, regularization weight λ, exponent α (¿0)
Require: β regime parameters: $\beta_0, \beta_{\text{Inc}}, \beta_{\text{Max}}$
Require: Number of inner iterations T.
1: $\beta = \beta_0$, $\mathbf{x} = \mathbf{y}$
2: Precompute constant terms in Eq. 4.4.
3: **while** $\beta < \beta_{\text{Max}}$ **do**
4: $iter = 0$
5: **for** $i = 1$ to T **do**
6: Given $\mathbf{x}$, solve Eq. 4.5 for all pixels using a LUT to give $\mathbf{w}$
7: Given $\mathbf{w}$, solve Eq. 4.4 to give $\mathbf{x}$
8: **end for**
9: $\beta = \beta_{\text{Inc}} \cdot \beta$
10: **end while**
11: **return** Deconvolved image $\mathbf{x}$

As with any non-convex optimization problem, it is difficult to derive any guarantees regarding the convergence of Algorithm 1. However, we can be sure that the global optimum of each sub-problem will be found, given the fixed $\mathbf{x}$ and $\mathbf{w}$ from the previous iteration. Like other methods that use this form of alternating minimization[61, 62, 167], there is little theoretical guidance for setting the β schedule. We find that the simple scheme shown in Algorithm 1 works well to minimize Eq. 4.2 and its proxy Eq. 4.1. We also note that empirically, the results are quite

robust to the choice of β schedule. The experiments in Section 4.3 show our scheme achieves very similar SNR levels to IRLS, but at a greatly lower computational cost.

4.3 Results

We evaluate the deconvolution performance of our algorithm on images, comparing them to numerous other methods: (i) ℓ_2 (Gaussian) prior on image gradients; (ii) Lucy-Richardson [133]; (iii) the algorithm of Wang et al. [167] using a total variation (TV) norm prior and (iv) a variant of [167] using an ℓ_1 (Laplacian) prior; (v) the IRLS approach of Levin et al. [96] using a hyper-Laplacian prior with $\alpha = 1/2, 2/3, 4/5$. Note that only IRLS and our method use a prior with $\alpha < 1$. We note that the sampling-based deconvolution approach in [139] outperforms both our method and IRLS, although running times are much slower. However, the unavailability of code prevents us from comparing our method to theirs, and so we refer the readers to their paper for experiments.

For the IRLS scheme, we used the implementation of [96] with default parameters, the only change being the removal of higher order derivative filters to enable a direct comparison with other approaches. Note that IRLS and ℓ_2 directly minimize Eq. 4.1, while our method, and the TV and ℓ_1 approaches of [167] minimize the cost in Eq. 4.2, using $T = 1, \beta_0 = 1, \beta_{\text{Inc}} = 2\sqrt{2}, \beta_{\text{Max}} = 256$. In our approach, we use $\alpha = 1/2$ and $\alpha = 2/3$, and compare the performance of the LUT and analytic methods as well. All runs were performed with multithreading enabled (over 4 CPU cores).

We evaluate the algorithms using a set of blurry images, created in the following way. 7 in-focus grayscale real-world images were downloaded from the web. They were then blurred by real-world camera shake kernels from [101]. 1% Gaussian noise was added, followed by quantization to 255 discrete values. In any practical deconvolution setting the blur kernel is never perfectly known. Therefore, the kernel passed to the algorithms was a minor perturbation of the true kernel, to mimic kernel estimation errors. In experiments with non-perturbed kernels (not shown), the results are similar to those in Tables 4.3 and 4.1 but with slightly higher SNR levels. See Figure 4.2 for an example of a kernel from [101] and its perturbed version. Our evaluation metric was the SNR between the original image $\hat{\mathbf{x}}$ and the deconvolved output $\mathbf{x}$, defined as $10 \log_{10} \frac{\|\hat{\mathbf{x}} - \mu(\hat{\mathbf{x}})\|^2}{\|\hat{\mathbf{x}} - \mathbf{x}\|^2}$, $\mu(\hat{\mathbf{x}})$ being the mean of $\hat{\mathbf{x}}$.

In Table 4.1 we compare the algorithms on 7 different images, all blurred with the same 19×19 kernel. For each algorithm we exhaustively searched over different regularization weights λ to find the value that gave the best SNR performance, as reported in the table. In Table 4.3 we evaluate the algorithms with the same 512×512 image blurred by 8 different kernels (from [101]) of varying size. Again, the optimal value of λ for each kernel/algorithm combination was chosen from a range of values based on SNR performance. Table 4.2 shows the running time of several algorithms on images up to 3072×3072 pixels. Figure 4.2 shows a larger 27×27 blur being deconvolved from two example images, comparing the output of different methods.

The tables and figures show our method with $\alpha = 2/3$ and IRLS with $\alpha = 4/5$ yielding higher quality results than other methods. However, our algorithm is around 300 to 1000 times faster than IRLS depending on whether the analytic or LUT method is used. This speedup factor is independent of image size, as shown by Table 4.2. The ℓ_1 method of [167] is the best of the other methods, being of comparable speed to ours but achieving lower SNR scores. The SNR results for our method are almost the same whether we use LUTs or analytic approach. Hence, in practice, the LUT method is preferred, since it is approximately 5 times faster than the analytic method and can be used for any value of α.

Image #	Blurry	ℓ_2	Lucy	TV	ℓ_1	IRLS α=1/2	IRLS α=2/3	IRLS α=4/5	Ours α=1/2	Ours α=2/3
1	6.42	14.13	12.54	15.87	16.18	14.61	15.45	16.04	16.05	**16.44**
2	10.73	17.56	15.15	19.37	19.86	18.43	19.37	20.00	19.78	**20.26**
3	12.45	19.30	16.68	21.83	22.77	21.53	22.62	22.95	23.26	**23.27**
4	8.51	16.02	14.27	17.66	18.02	16.34	17.31	17.98	17.70	**18.17**
5	12.74	16.59	13.28	19.34	20.25	19.12	19.99	20.20	**21.28**	21.00
6	10.85	15.46	12.00	17.13	17.59	15.59	16.58	17.04	17.79	**17.89**
7	11.76	17.40	15.22	18.58	18.85	17.08	17.99	18.61	18.58	**18.96**
Av. SNR gain		6.14	3.67	8.05	8.58	7.03	7.98	8.48	8.71	**8.93**
Av. Time (secs)		79.85	1.55	0.66	0.75	354	354	354	L:1.01 A:5.27	L:1.00 A:4.08

Table 4.1: Comparison of SNRs and running time of 9 different methods for the deconvolution of 7 576×864 images, blurred with the same 19×19 kernel. L=Lookup table, A=Analytic. The best performing algorithm for each kernel is shown in bold. Our algorithm with $\alpha = 2/3$ beats IRLS with $\alpha = 4/5$, as well as being much faster. On average, both these methods outperform ℓ_1, demonstrating the benefits of a sparse prior.

Image size	ℓ_1	IRLS α=4/5	Ours (LUT) α=2/3	Ours (Analytic) α=2/3
256×256	0.24	78.14	0.42	0.7
512×512	0.47	256.87	0.55	2.28
1024×1024	2.34	1281.3	2.78	10.87
2048×2048	9.34	4935	10.72	44.64
3072×3072	22.40	-	24.07	100.42

Table 4.2: Run-times of different methods for a range of image sizes, using a 27×27 kernel. Our algorithm is more than 300 times faster than the IRLS method of [96].

4.4 Discussion

We have described an image deconvolution scheme that is fast, conceptually simple and yields high quality results. Our algorithm takes a novel approach to the non-convex optimization problem arising from the use of a hyper-Laplacian prior, by using a splittting approach that allows the non-convexity to become separable over pixels. Using a LUT to solve this sub-problem allows for orders of magnitude speedup in the solution over existing methods. Our Matlab implementation is available online at `http://cs.nyu.edu/~dilip/research/fast-deconvolution/`.

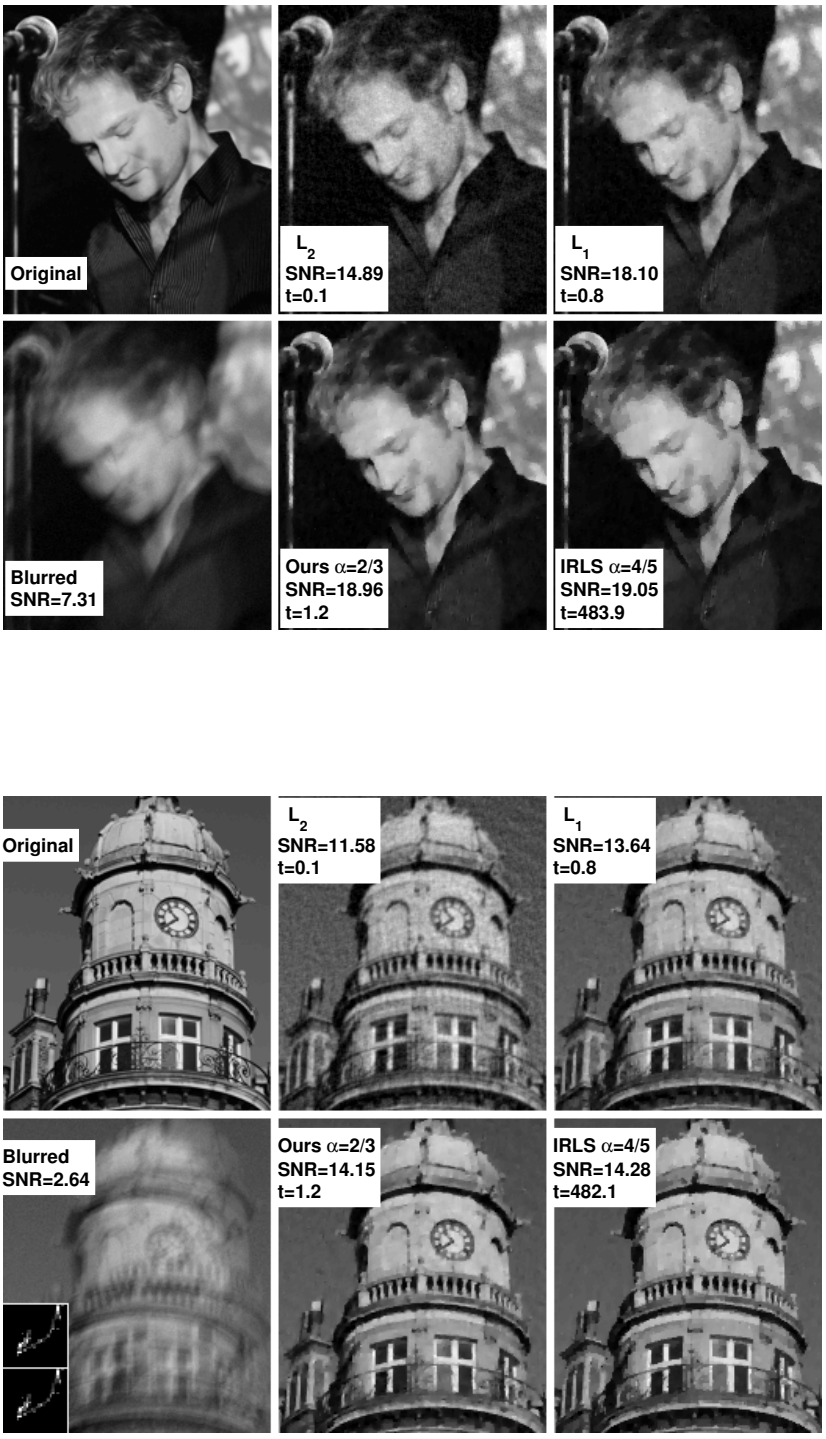


Figure 4.2: Crops from two images (#1 & #5) being deconvolved by 4 different algorithms, including ours using a 27×27 kernel (#7). In the bottom left inset, we show the original kernel from [101] (lower) and the perturbed version provided to the algorithms (upper), to make the problem more realistic. This figure is best viewed on screen, rather than in print.

Kernel # / size	Blurry	ℓ_2	Lucy	TV	ℓ_1	IRLS α=1/2	IRLS α=2/3	IRLS α=4/5	Ours α=1/2	Ours α=2/3
#1: 13×13	10.69	17.22	14.49	19.21	19.41	17.20	18.22	18.87	19.36	**19.66**
#2: 15×15	11.28	16.14	13.81	17.94	18.29	16.17	17.26	18.02	18.14	**18.64**
#3: 17×17	8.93	14.94	12.16	16.50	16.86	15.34	16.36	16.99	16.73	**17.25**
#4: 19×19	10.13	15.27	12.38	16.83	17.25	15.97	16.98	17.57	17.29	**17.67**
#5: 21×21	9.26	16.55	13.60	18.72	18.83	17.23	18.36	18.88	19.11	**19.34**
#6: 23×23	7.87	15.40	13.32	17.01	17.42	15.66	16.73	17.40	17.26	**17.77**
#7: 27×27	6.76	13.81	11.55	15.42	15.69	14.59	15.68	**16.38**	15.92	16.29
#8: 41×41	6.00	12.80	11.19	13.53	13.62	12.68	13.60	**14.25**	13.73	13.68
Av. SNR gain		6.40	3.95	8.03	8.31	6.74	7.78	8.43	8.33	**8.67**
Av. Time (sec)		57.44	1.22	0.50	0.55	271	271	271	L:0.81 A:2.15	L:0.78 A:2.23

Table 4.3: Comparison of SNRs and running time of 9 different methods for the deconvolution of a 512×512 image blurred by 7 different kernels. L=Lookup table, A=Analytic. Our algorithm beats all other methods in terms of quality, with the exception of IRLS on the largest kernel size. However, our algorithm is far faster than IRLS, being comparable in speed to the ℓ_1 approach.

Algorithm 2: Solve Eq. 4.5 for $\alpha = 1/2$

Require: Target value v, Weight β

$\epsilon = 10^{-6}$

{Compute intermediary terms m, t_1, t_2, t_3}

$m = -\text{sign}(v)/4\beta^2$

$t_1 = 2v/3$

$t_2 = \sqrt[3]{-27m - 2v^3 + 3\sqrt{3}\sqrt{27m^2 + 4mv^3}}$

$t_3 = v^2/t_2$

{Compute 3 roots, r_1, r_2, r_3:}

$r_1 = t_1 + 1/(3 \cdot 2^{1/3}) \cdot t_2 + 2^{1/3}/3 \cdot t_3$

$r_2 = t_1 - (1 - \sqrt{3}i)/(6 \cdot 2^{1/3}) \cdot t_2 - (1 + \sqrt{3}i)/(3 \cdot 2^{2/3}) \cdot t_3$

$r_3 = t_1 - (1 + \sqrt{3}i)/(6 \cdot 2^{1/3}) \cdot t_2 - (1 - \sqrt{3}i)/(3 \cdot 2^{2/3}) \cdot t_3$

{Pick global minimum from $(0, r_1, r_2, r_3)$}

$r = [r_1, r_2, r_3]$

$c_1 = (\texttt{abs}(\texttt{imag}(r)) < \epsilon)$ {Root must be real}

$c_2 = \texttt{real}(r)\texttt{sign}(v) > (2/3 \cdot \texttt{abs}(v))$ {Root must obey bound of Eq. 4.13}

$c_3 = \texttt{real}(r)\texttt{sign}(v) < \texttt{abs}(v)$ {Root $< v$}

$w^* = \texttt{max}((c_1 \,\&\, c_2 \,\&\, c_3)\texttt{real}(r)\texttt{sign}(v))\texttt{sign}(v)$

return w^*

Algorithm 3: Solve Eq. 4.5 for $\alpha = 2/3$

Require: Target value v, Weight β

$\epsilon = 10^{-6}$

{Compute intermediary terms $m, t_1, \ldots, t_7$:}

$m = 8/(27\beta^3)$

$t_1 = -9/8 \cdot v^2$

$t_2 = v^3/4$

$t_3 = -1/8 \cdot mv^2$

$t_4 = -t_3/2 + \sqrt{-m^3/27 + m^2v^4/256}$

$t_5 = \sqrt[3]{t_4}$

$t_6 = 2(-5/18 \cdot t_1 + t_5 + m/(3 \cdot t_5))$

$t_7 = \sqrt{t_1/3 + t_6}$

{Compute 4 roots, r_1, r_2, r_3, r_4:}

$r_1 = 3v/4 + (t_7 + \sqrt{-(t_1 + t_6 + t_2/t_7)})/2$

$r_2 = 3v/4 + (t_7 - \sqrt{-(t_1 + t_6 + t_2/t_7)})/2$

$r_3 = 3v/4 + (-t_7 + \sqrt{-(t_1 + t_6 - t_2/t_7)})/2$

$r_4 = 3v/4 + (-t_7 - \sqrt{-(t_1 + t_6 - t_2/t_7)})/2$

{Pick global minimum from $(0, r_1, r_2, r_3, r_4)$}

$r = [r_1, r_2, r_3, r_4]$

$c_1 = (\texttt{abs}(\texttt{imag}(r)) < \epsilon)$ {Root must be real}

$c_2 = \texttt{real}(r)\texttt{sign}(v) > (1/2 \cdot \texttt{abs}(v))$ {Root must obey bound in Eq. 4.13}

$c_3 = \texttt{real}(r)\texttt{sign}(v) < \texttt{abs}(v)$ {Root $< v$}

$w^* = \max((c_1 \& c_2 \& c_3)\texttt{real}(r)\texttt{sign}(v))\texttt{sign}(v)$

return w^*

A potential drawback to our method, common to the TV and ℓ_1 approaches of [167], is its use of frequency domain operations which assume circular boundary conditions, something not present in real images. These give rise to boundary artifacts which can be overcome to some extent with the edge tapering operations that we use. However, our algorithm is suitable for very large

images where the boundaries are a small fraction of the overall image. Furthermore, the use of FFT and LUT operations makes this algorithm suitable for deployment in mobile platforms.

Although we focus on deconvolution, our scheme can be adapted to a range of other problems which rely on natural image statistics. For example, by setting $\mathbf{k} = 1$ the algorithm can be used to denoise, or if $\mathbf{k}$ is a defocus kernel it can be used for super-resolution. The speed offered by our algorithm makes it practical to perform these operations on the multi-megapixel images from modern cameras. Our algorithm has been used in a number of other works on denoising and non-blind image deconvolution [191, 68, 27]. A GPU implementation has been developed [80] which achieves real-time performance for images of size 710×470 on an NVIDIA GeForce GTX 260 GPU: nearly 2 orders of magnitude speedup over our CPU implementation.

Schmidt et al. [139] have recently developed a Bayesian approach to non-blind deconvolution which outperforms our method in terms of PSNR. Their sampling-based technique is, however, significantly slower than our algorithm. There are a number of ways in which the quality of our method may be improved: the use of higher order filters such as the Fields of Experts [136]; improved handling of the boundary to minimize artifacts due to FFT; and a more finely tuned β schedule. A practical deblurring algorithm must be robust to outliers such as saturated pixels in the blurred image. Recent papers such [172] addressed this issue although with different optimization schemes. By modifying the likelihood term appropriately, our algorithm has been used for spatially varying deconvolution [68, 69].

Chapter 5

Blind Deconvolution Using a Normalized Sparsity Measure

5.1 Introduction

In this chapter, we develop a novel image prior for the blind deconvolution problem. This is joint work with Rob Fergus and Terence Tay and was published in CVPR 2011 [90].

As described in Chapter 2, a wide range of parametric image priors have been proposed for inverse problems such as denoising, non-blind deconvolution and super resolution [100, 136, 191]. These priors are developed by fitting parametric distributions or learning schemes to some statistics of *sharp* uncorrupted images. However, when used in an optimization setting for blind deconvolution, they prove to be ineffective. As noted by Levin et al. [101] and Fergus et al. [55], somewhat counter-intuitively, the above image models all favor blurry images to sharp images. In other words, blurry images have lower cost (are more probable) than sharp images. This is a direct result of the learned/chosen potential functions decreasing toward zero: since blur attenuates high frequencies, the response of any derivative-type filter will also be reduced and consequently will have a lower cost under the model. This phenomenon is illustrated in Figure 5.1, where we show the cost (negative log probability) of a sharp image blurred by different amounts. As a consequence, for blind deconvolution, which is highly ill-posed and hence reliant on the image prior standard maximum a-posterori (MAP) based approaches do not work. Correspondingly, a number of more complex methods have been proposed. These include: marginalization over all possible images [55, 101, 114]; dynamic adaptation of the cost function [142]; alpha-matt extraction [74]; re-weighting of the image edges [32]; determination of the edge locations using shock filtering [117].

In this paper we introduce a novel type of image prior that favors sharp images over blurry and show how this prior can be used in a framework for blind deconvolution. Compared to other methods, our approach is very simple – it requires none of the complexities needed by other

methods to overcome shortcomings of existing priors in an MAP setting. The resulting scheme is also quick since it can take advantage of existing fast ℓ_1 methods to estimate the kernel and sharp image.

5.2 Motivation

The regularization function we propose is the ratio of the ℓ_1 norm to the ℓ_2 norm on the high frequencies of an image. ℓ_1/ℓ_2 is an unusual function and its relevance to blind image deconvolution is not immediately clear. We first motivate its use in this setting, before explaining our method.

First consider the ℓ_1 norm. The ℓ_1 norm is widely used to impose signal sparsity, but it is scale variant so the norm can be minimized by simply reducing the signal. In an image setting, the ℓ_1 norm is typically used to penalize the high frequency bands. As image noise presents itself in these bands, boosting their ℓ_1 norm, minimizing the norm is a way of denoising the image. However, in the case of image blur, the opposite situation holds since blur attenuates the high frequency bands so reducing their ℓ_1 norm. Consequently, in a blind deconvolution setting where the kernel is only loosely constrained, minimizing the ℓ_1 norm on the high frequencies of the image will result in a blurry image (and a delta function kernel). This behavior, studied in [101], is illustrated in Figure 5.1.

The simplest interpretation of the ℓ_1/ℓ_2 function is that it is a normalized version of ℓ_1 , making it scale invariant. If applied to the high frequency bands of an image, it is equivalent to the ℓ_1 norm of the edges rescaled by their total energy. Although blur decreases both the ℓ_1 and ℓ_2 norms, crucially the latter is reduced more, thus the *ratio* of the two will be increased by blur (see the magenta curve in Figure 5.1).

To understand why this is so, consider the visualization of the ℓ_1/ℓ_2 function for a two dimensional signal in Figure 5.2. The minima lie along the axes with the cost increasing smoothly in between. The high frequency bands of natural scenes are typically sparse in that the magnitudes are mostly either zero or very small, but occasionally large. If these bands are represented as a single high dimensional vector, it would be close to the axes in many dimensions and have a low ℓ_1/ℓ_2 value. Blur smears out the large magnitude elements and reduces the number of zero elements in the vector, so rotating it diagonally away from the axes, increasing the ℓ_1/ℓ_2 value.

Given that the ℓ_1/ℓ_2 function behaves correctly for blur, it is natural to wonder if added noise or sharpening operations might result in a lower cost than the true image. Noise added to the signal increases the ℓ_1/ℓ_2 value, as do sharpening operations (see left side of Figure 5.1).

Most of the energy in images is contained in the low and mid frequency bands, which are barely affected by blur. As a consequence, the ℓ_1/ℓ_2 function will not be changed significantly if measuring the entire image (i.e. all frequency bands). Instead, the ℓ_1/ℓ_2 function must be applied to just the high frequency part of the image if it is to discriminate between sharp and blurry images.

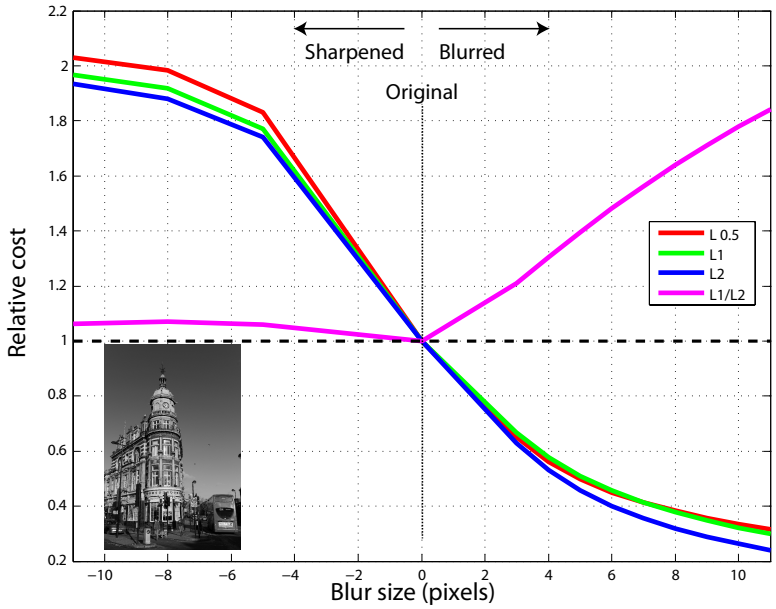


Figure 5.1: A comparison of our novel image regularizer to existing approaches. For a typical real world scene I (inset), we add Gaussian blur b ranging from 0 to 11 pixels in size and measure the cost: $\|(\nabla_x(I \otimes b))\|_\alpha + \|(\nabla_y(I \otimes b))\|_\alpha$ where ∇_x, ∇_y are the x and y derivatives respectively. Existing image regularizers use $\alpha = 0.5$ (red), 1 (green), 2 (blue). Our image regularizer $\frac{\|[\nabla_x(I \otimes b), \nabla_y(I \otimes b)]\|_1}{\|[\nabla_x(I \otimes b), \nabla_y(I \otimes b)]\|_2}$ is shown in magenta. The y-axis shows cost relative to that of the sharp image. A negative blur size corresponds to an unsharp mask filter. Note that the existing priors incorrectly have a *lower* cost for blurry images than sharp ones. By contrast, our regularizer correctly gives lowest cost to the original image.

The ℓ_1/ℓ_2 function is one of a number of sparsity measures in the literature [71] but is relatively rare, being previously used for matrix factorization [70, 119]. In [71], different sparsity measures are compared using 6 heuristic criteria and the ℓ_1/ℓ_2 function satisfies all them. Following our paper, the ℓ_1/ℓ_2 function has been used with a wavelet transform instead of the derivative transform in [73] with similar justifications, although they use a different optimization algorithm. The ℓ_1/ℓ_2 function may be generalized to ℓ_p/ℓ_q with $p < q$.

Sparsity has a natural interpretation using an ℓ_0 measure. However, ℓ_0 is difficult to optimize because of the lack of gradient information everywhere. It is therefore convenient to use a convex measure such as ℓ_1 instead [23]. But, as illustrated in Figure 5.2, ℓ_1 has a very different shape to ℓ_0 and it is also not scale invariant. The ℓ_1/ℓ_2 function, on the other hand, is scale invariant and has minima along the axes, just as ℓ_0 does. It also has the advantage of gradient information which can be exploited to give a tractable optimization algorithm.

The ℓ_1/ℓ_2 function does have several drawbacks. First, it is non-convex, unlike ℓ_1 , thus there are multiple local-minima. Second, it is hard to optimize directly but we introduce approximate methods that can overcome this. Finally, it cannot be expressed as a probabilistic prior as $\int exp(-\|x\|_1/\|x\|_2)dx = \infty$, due to the scale invariance. This is in direct contrast to ℓ_p norms $(0.7 \leq p \leq 1)$ which correspond to probabilistic models of image gradients, having a (hyper)-

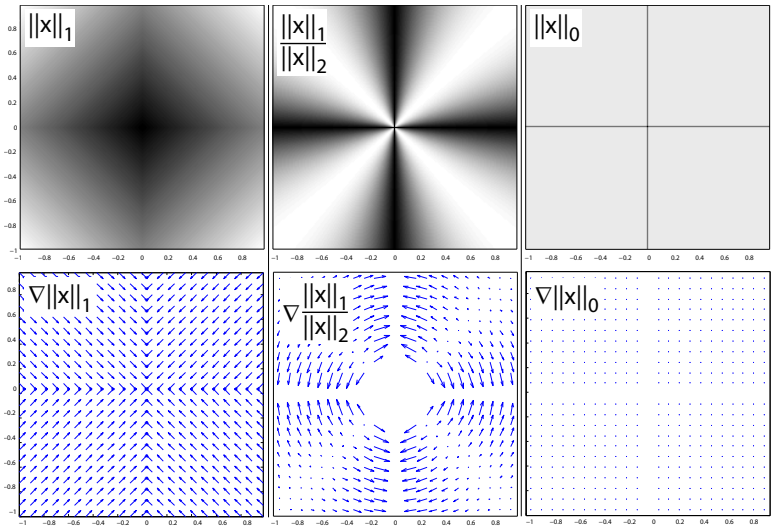


Figure 5.2: A visualization of ℓ_1 , ℓ_1/ℓ_2 and ℓ_0 functions (top row) and their gradient fields (bottom row) for a two dimensional vector (dark corresponds to a lower value). The l_1 function is smallest at the origin and the gradient field uniformly points towards the origin. The ℓ_1/ℓ_2 function has minima along the axes, with smoothly increasing cost in the diagonal directions. Its gradient field is purely radial: starting at arbitrary location the gradient leads to the nearest axis, preserving distance from the origin. Note that the minima structure of ℓ_1/ℓ_2 is very similar that ℓ_0 . However, the ℓ_0 norm is difficult to use, having zero gradient everywhere, except near the axes where it is infinite.

Laplacian form. However, since we intend to use a non-probabilistic framework, it does not matter that the energy surface is not normalized.

5.3 Approach

We assume the formation model of a sharp image u blurred by a matrix K along with the addition of Gaussian i.i.d noise N:

$$g = Ku + N \tag{5.1}$$

We observe the resulting blurry image g and our goal is to recover the unknown sharp image u and the blurring matrix K. Algorithm 2 outlines our approach.

In Section 5.3.1, we first consider the case when the blur is spatially constant. In this case, the matrix K reduces to a 2-dimensional convolution operation with a kernel k. We then show how our algorithm can be extended to the case of pure in-plane rotation in Section 5.3.4. Finally, we consider the case of general 3-D rotations of the camera in Section 5.3.5. The overall algorithm is implemented in a multiscale framework, described in Section 5.3.1.

Algorithm 2 : Overall Algorithm

Require: Observed blurry image g, Maximum kernel size h.
Apply derivative filters to g, creating a high-freq. image y.
1. Blind estimation of blur matrix K (Section 5.3.1) from y.
Loop over coarse-to-fine levels:
Alternate:
- Update sharp high-frequency image x (Section 5.3.1) using l_1/l_2 regularization.
- Update blurring matrix K (Section 5.3.1).
Interpolate solution to finer level as initialization.
2.Image recovery using non-blind algorithm of [86] (Section 5.3.2).
- Deblur g using K to give sharp image u.
return Sharp image u.

5.3.1 Blind Kernel Estimation

Our kernel estimation is performed on the high frequencies of the image. Given the blurry and noisy input g, we use discrete filters $\nabla_x = [1, -1]$ and $\nabla_y = [1, -1]^T$ to generate a high-frequency version $y = [\nabla_x g, \nabla_y g]$[1]. The cost function for spatially invariant blurring is:

$$\min_{x,k} \lambda \|x \otimes k - y\|_2^2 + \frac{\|x\|_1}{\|x\|_2} + \psi \|k\|_1 \tag{5.2}$$

subject to the contraints that $k \geq 0, \sum_i k_i = 1$. Here x is the unknown sharp image in the high-frequency space, k is the unknown blurring kernel (k_i are individual elements) and $\otimes$ is the 2D convolution operator.

Eq. 5.2 consists of 3 terms. The first term is the likelihood that takes into account the formation model Eq. 5.1. The second term is the new l_1/l_2 regularizer on x which encourages scale-invariant sparsity in the reconstruction. To reduce noise in the kernel, we add ℓ_1 regularization on k. The constraints on k (sum-to-1 and non-negativity) follow from the physical principles of blur formation. The scalar weights λ and ψ control the relative strength of the kernel and image regularization terms.

Eq. 5.2 is nonconvex. The standard approach to optimizing such a problem is to start with an initialization on x and k, and then alternate between x and k updates [55]. To make consistent progress along each of the unknowns and avoid local minima as far as possible, only a few iterations are performed in each update.

x Update

The x sub-problem is given by:

$$\min_{x} \lambda \|x \otimes k - y\|_2^2 + \frac{\|x\|_1}{\|x\|_2} \tag{5.3}$$

[1] y is a concatenation of the two gradient images $\nabla_x g$ and $\nabla_y g$.

This sub-problem is non-convex due to the presence of the new regularization term $\frac{\|x\|_1}{\|x\|_2}$. However, if one fixes the denominator of the regularizer from the previous iterate, the problem then becomes a convex l_1-regularized problem. Fast algorithms to solve l_1-regularized problems are well-known in the compressed sensing literature [9, 184]. One such algorithm is the iterative shrinkage-thresholding algorithm (ISTA) [9]. ISTA, detailed in Algorithm 3, is a fast method to solve general linear inverse problems of the form:

$$\min_x \lambda \|Kx - y\|_2^2 + \|x\|_1 \tag{5.4}$$

In our application K is the blurring matrix.

Algorithm 3 : Iterative Shrinkage-Thresholding Algorithm (ISTA)

Require: Operator K, regularization parameter λ
Require: Initial iterate x^0, observed image y
Require: Threshold t, maximum iterations N
1: **for** $j = 0$ to $N - 1$ **do**
2: $\quad v = y - tK^T(Kx^j - y)$
3: $\quad x^{j+1} = S_{t\lambda}(v)$
4: **end for**
5: **return** Output image x^N

Here, the operator S is the soft shrinkage operation on a vector. It shrinks each component of the input vector towards zero:

$$S_\alpha(x)_i = \max(|x_i| - \alpha, 0)\text{sign}(x_i) \tag{5.5}$$

ISTA is very simple and fast, involving only multiplications of the matrix K with vector x, followed by the component-wise shrinkage operation.

We use the ISTA step as the inner iteration in our x-update algorithm. The outer loop then simply re-estimates the weighting on the likelihood term in Eq. 5.3 by updating the denominator $\|x\|_2$. The overall x-update algorithm is as follows:

Algorithm 4 : x Update

Require: Blur kernel k from previous k update
Require: Image x^0 from previous x update
Require: Regularization parameter $\lambda = 20$
Require: Maximum outer iterations $M = 2$, inner its. $N = 2$
Require: ISTA threshold $t = 0.001$
1: **for** $j = 0$ to $M - 1$ **do**
2: $\quad \lambda' = \lambda\|x^j\|_2$
3: $\quad x^{j+1} = \text{ISTA}(k, \lambda', x^j, t, N)$
4: **end for**
5: **return** Updated image x^M.

Despite the non-convexity of the problem, in practice this inner-outer iteration is effective in reducing the cost function in Eq. 5.3. After an x-update step, we update the kernel estimate k.

k Update

The kernel update sub-problem is given by:

$$\min_k \lambda \|x \otimes k - y\|_2^2 + \psi \|k\|_1 \tag{5.6}$$

subject to the constraints $k \geq 0, \sum_i k_i = 1$. We use unconstrained iterative re-weighted least squares (IRLS) [96] followed by a projection of the resulting k onto the contraints (setting negative elements to 0, and renormalizing). During the iterations we run IRLS for just 1 iteration, with the weights being computed from the kernel of the previous k update. We solve the inner IRLS system to a low level of accuracy, using 5 conjugate gradient (CG) iterations.

An important practical point is that after recovering the kernel at the finest level, we threshold small elements of the kernel to zero, thereby increasing robustness to noise. This is similar to other blind deconvolution methods [55, 173].

Multiscale Implementation

For large kernels, an excessive number of x and k updates may be required to converge to a reasonable solution. To mitigate this problem, we perform multiscale estimation of the kernel using a coarse-to-fine pyramid of image resolutions, in a similar manner as in [55]. We always use 4 levels with a size ratio of $\sqrt{2}$ between them (in each dimension). We downsample the input blurry image and then take discrete gradients to form the input y each level.

At each scale level we perform 100 alternating updates of x and k. Once a kernel estimate k and sharp gradient image x are computed, they are upsampled to act as the initialization of the kernel and sharp image at the next finer level. All of the resizing operations are done using bilinear interpolation.

5.3.2 Image recovery

Once the kernel k for the finest level has been estimated, we can use a variety of non-blind deconvolution methods to recover the full-spectrum sharp image u from g. The simplest is Richardson-Lucy (RL). The disadvantage of RL is that this method is sensitive to a wrong kernel estimate, which results in ringing artifacts in u. Therefore, we choose to use the non-blind deconvolution method from Chapter 4, since it is fast and robust to small kernel errors. This algorithm uses a continuation method to solve the following cost function:

$$\min_u \lambda \|u \otimes k - g\|_2^2 + \|\nabla_x g\|_\alpha + \|\nabla_y g\|_\alpha \tag{5.7}$$

where ∇_x and ∇_y are the same derivative filters used in Section 5.3.1. We use $\lambda = 3000, \alpha = 0.8$ for all results.

A natural question is why we do not use the ℓ_1/ℓ_2 prior itself as a regularizer for the non-blind deconvolution. Non-blind deconvolution has fewer unknowns than the blind problem; as such it is less sensitive to the choice of prior. Secondly, in the non-blind step, we need to work in the image domain where the ℓ_1/ℓ_2 prior is defined on the gradient domain. So we have to deal with the ratios of ℓ_1 and ℓ_2 norms of *gradients*. This is a considerably more difficult optimization problem and we have no efficient means of solution. Due to this we choose to use a simpler regularizer for the image recovery step.

5.3.3 Speed and robustness

Our cost function Eq. 5.2 is of a simple form. The x and k update steps involve a few matrix-vector (or convolution) operations. As a result, our algorithm is quite fast as compared to existing algorithms such as [55]. For a 255×255 pixel image, and when estimating a 35×35 size kernel, our algorithm takes 3 minutes, compared to 6 minutes for the method of [55].[2] Our method is amenable to speedups such as GPU acceleration and the use of the Intel IPP libraries.

In our experiments, we find the algorithm to be robust to the choice of parameters and use the same settings for all results reported in this paper. The ψ parameter depends on the user-specified kernel size h, according to the formula: $\psi = \frac{3}{13}h$. This robustness is a major advantage of our algorithm over existing schemes that require parameter adjustment for different input images.

5.3.4 Extension to in-plane rotation

We extend the cost function of Eq. 5.2 to the case where the blurring process arises purely from rotation of the camera around the Z-axis (axis perpendicular to the sensor plane), thus is no longer spatially constant. Similar to the non-uniform blurring model developed in [173], we assume that the blurred image arises form a linear combination of discretely sampled rotations of the sharp image:

$$g = \sum_{\theta \in \omega} k_\theta R_\theta(u) + N \tag{5.8}$$

where ω is a discrete set of angles, and the operator $R_\theta(u)$ rotates the image u by angle θ. For this formation model our blur kernel is the vector k with entries k_θ. The minimization problem is then a modified version of Eq. 5.2, given by:

$$\min_{x,k} \lambda \| \sum_{\theta \in \omega} k_\theta R_\theta(x) - y \|_2^2 + \frac{\|x\|_1}{\|x\|_2} + \psi \|k\|_1 \tag{5.9}$$

subject to the constraints that $k \geq 0$ and $\sum_{\theta \in \omega} k_\theta = 1$. We can now adopt exactly the same approach as for the spatially invariant case, except replacing 2-dimensional convolutional operations with sums of rotations. The use of rotations significantly increases the computational cost of the overall processing. This can be somewhat mitigated by using a coarsely sampled set ω.

[2] Using a single-threaded Matlab on a 2.66Ghz CPU.

5.3.5 Extension to 3-D rotations

By extending the set of rotations ω to the X and Y axes, our model can be generalized to the full non-uniform blur model. It has been demonstrated in [173] that rotations of the camera about X and Y cause more significant blur than translations. Accordingly, we adopt the model proposed in [173] that samples discretely from all 3 planes of rotation. The final formation model and cost function formulation are analogous to Eq. 5.8 and Eq. 5.9:

$$\min_{x,k} \lambda \| \sum_{\theta_{xyz} \in \omega} k_\theta R_{\theta_{xyz}}(x) - y \|_2^2 + \frac{\|x\|_1}{\|x\|_2} + \psi \|k\|_1 \tag{5.10}$$

where θ_{xyz} refers to a particular combination of rotations around all 3 axes. The approach in [173] uses a multi-scale method to determine the form of the kernel. Surprisingly, we find that just using a single scale with very few iterations allows us to recover the rotational kernel very accurately. This greatly increases the speed of our method as compared to [173] which can take hours for even a small image. In contrast, our method takes less than 10 minutes for the kernel estimation process for a 255×255 image.

ω is a set of discrete angles of rotation about each axis. For spatially varying kernels arising from camera rotations, the non-blind deconvolution algorithm must be modified to take into account the different formation model. We extend the non-blind deconvolution algorithm of [86] accordingly.

Gupta et al. [67] present a blur formation model based on in-plane rotations combined with translations. They also use a set of discretized basis functions. By replacing the basis functions $R_{\theta_{xyz}}$ in Eq. 5.10 with their basis functions, we can support their model with our new regularizer.

5.4 Experiments

In this section, we present results of our algorithm and compare it to the algorithms of [55],[142] and [173]. We consider spatially invariant, pure in-plane rotation and 3-D rotational blurring models. We show results on both synthetic and real-world examples. The images are best viewed on screen.

5.4.1 Spatially invariant kernel

Synthetic data

We first test the algorithm on the spatially invariant kernels from the dataset in Levin et al. [101]. This dataset consists of 4 images of size 255×255 and 8 different kernels ranging in size from 13×13 to 27×27 to give a total of 32 blurred images. The blurred data, ground truth data and ground truth kernels are provided. We compare our kernel estimation results with the blind

deconvolution algorithms of Fergus et al. [55] and Shan et al. [142]. For kernels estimated by the 3 different methods, we perform non-blind deconvolution using the algorithm of [86] with the same parameter settings. The error metric used is the same as [101][3].

In Figure 5.3, we plot the cumulative histograms of the error ratios for the 3 algorithms. The performance of our algorithm is similar to that of Fergus et al. [55]. Both these algorithms significantly out perform that of Shan et al. [142]. Additionally, there are a few examples in the dataset (see [102] for details) when the algorithm of [55] fails dramatically, whereas our algorithm is still able to recover a reasonable kernel. In addition, our algorithm is considerably simpler and faster than that of [55].

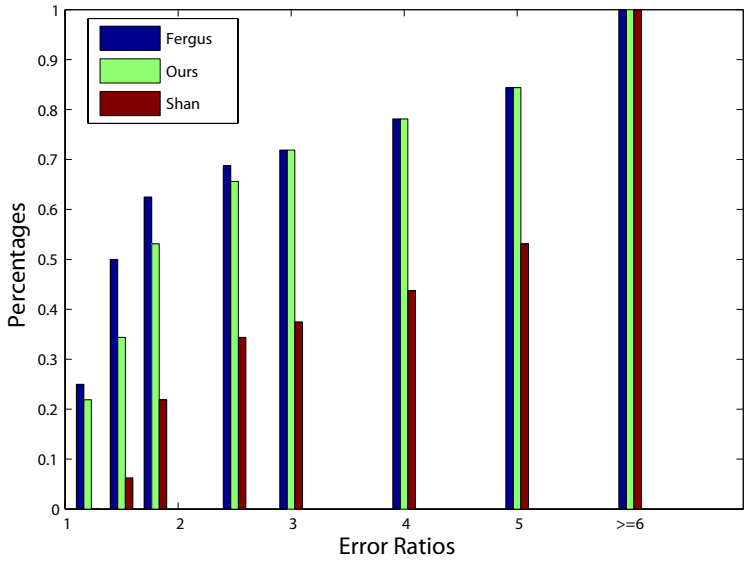


Figure 5.3: Cumulative histograms of the error ratios across the dataset of Levin et al. [101]. See text for details.

Figure 5.4 shows results with a kernel size 27×27. In Figure 5.5, we show a failure case for both [55] and [142]. By constrast, our algorithm is able to recover a reasonable kernel. This illustrates the robustness provided by our new l_1/l_2 regularization function.

Real data

Next we compare the performance of our algorithm on some real-world examples presented in different blind deconvolution papers. As before, we use the different blind algorithms to estimate the kernel and then use the same non-blind algorithm (that of [86]) with identical parameter settings to perform the deconvolution. Figure 5.6 shows the kernel recovery and reconstruction for an image provided in the online code of Fergus et al. [55], along with their result. The output of [55] shows artefacts, for example on the right cheek of the statue. Figure 5.7 compares kernel

[3]The measure is the *ratio* of SSD (sum of squared differences) error between: (i) the deconvolved output and the known ground truth image and (ii) deconvolved output using the ground truth kernel and the ground truth image.

Figure 5.4: Recovery of a 27×27 kernel. Top-left: original; top-right: blurred; middle-left: deblurred with ground truth kernel; middle-right: deblurred with our estimated kernel; bottom-left: deblurred with kernel of [55]; bottom-right: deblurred with kernel of [142]. Corresponding kernels are shown as insets at the bottom left of each image.

recovery and reconstruction of another image from [55]. Our results were obtained using the same parameter settings as for the synthetic data.

5.4.2 In-Plane rotation

We now take a synthetic example presented in [173] of pure in-plane rotation of the camera. We use the model of Eq. 5.8 to perform the kernel estimation. In this case, we use only a single scale of processing (the finest scale) to speed up the computation. The other settings such as the number of iterations, and regularization parameter λ remain unchanged from the spatially invariant setting. We deblur the image (with 2% noise added) with both our algorithm and that of [173]. We use the online code provided by [173] to generate their result. The results are shown in Figure 5.8. The method of [173] took over 3 hours for the entire processing, whereas our method takes 10 minutes on the same CPU.

Figure 5.5: Recovery of a 27×27 kernel. Top-left: original; top-right: blurred; middle-left: deblurred with ground truth kernel; middle-right: deblurred with our estimated kernel; bottom-left: deblurred with kernel of [55]; bottom-right: deblurred with kernel of [142].

5.4.3 3-D Rotation

Finally, we compare our kernel estimation on a real world example given in [173]. The ground truth kernel is unknown in this case. We use the model of Eq. 5.10 with the same discretized set of 3D rotation angles as used in the online code of [173]. Our results, shown in Figure 5.9, are very similar to those of [173].

5.5 Connections with the blind equalization literature

The ℓ_1/ℓ_2 prior has interesting connections with some of the classical blind equalization literature, pointed out to us by Professor Yair Weiss. In this section, we explore these connections. Blind equalization is the one-dimensional equivalent of blind deconvolution. In digital data communications, it is also called channel equalization. The problem is illustrated in Figure 5.10. Here x is the input 1-D signal, assumed to consist of zero-mean IID samples. The only restriction on

Figure 5.6: Recovery of a real-world kernel. Left: Input blurry image; middle: deblurred with kernel of [55]; right: deblurred with our estimated kernel.

Figure 5.7: Recovery of a real-world kernel. Left: Input blurry image; Middle: Deblurred with kernel of [55]; Right: Deblurred with our estimated kernel.

the distribution from which x is drawn is that it is is non-Gaussian. k is the impulse response of the communication channel, the equivalent of a blur kernel. y is the observed blurred signal such that $y = x \oplus k$. The purpose of channel equalization is to process y to recover an *equalized* signal z, which is close to x. One of the most successful methods for blind deconvolution is the algorithm introduced by Shalvi and Weinstein [140, 141]. Variants of their technique are still used in millions of wireless devices today for blind equalization. Here we present a simplified version of the algorithm developed by Shalvi and Weinstein. We start with the definition of the kurtosis of a zero-mean IID signal x:

$$\kappa = \sum_i \left(\frac{x_i}{std(x)} \right)^4 \tag{5.11}$$

Figure 5.8: In-plane rotational deblurring. Top-left: input sharp image; top-right: blurry image with pure in-plane rotation; bottom-left: deblurred with code of [173]; bottom-right: our deblurred result.

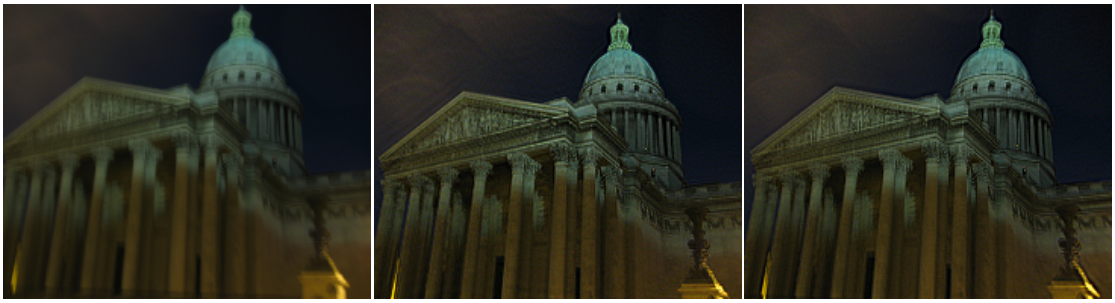


Figure 5.9: 3D rotational deblurring. Left: Input blurry image; middle: deblurred with algorithm of [173]; right: deblurred with our algorithm.

where x_i is element i of the vector x and $std(x)$ is the standard deviation of x. It is well-known [178] that a Gaussian signal has the value $\kappa = 3$. Sub-Gaussian signals are defined as those signals with $\kappa < 3$ and super-Gaussian those with $\kappa > 3$. Sub-Gaussian signals are wider than the Gaussian around the origin and super-Gaussian signals have heavier tails then a Gaussian. Examples of these signals are given in Figure 5.11. Image gradient distributions are super-Gaussian.

The Central Limit Theorem [176] states that as IID variables are averaged together, they become

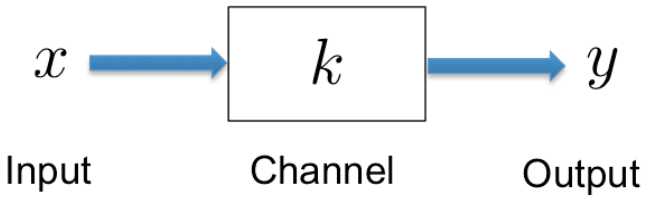


Figure 5.10: Blind Equalization Model. x is the input, k the channel impulse respone and y the output.

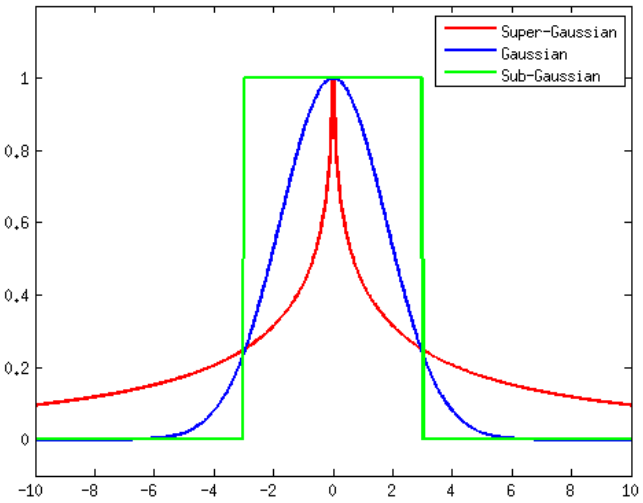


Figure 5.11: Gaussian, Sub-Gaussian and Super-Gaussian signals. It is seen that the super-Gaussian signal has a heavier tail than the Gaussian. Image gradient distributions follow super-Gaussian distributions.

more Gaussian-like. This means that blur makes a signal more Gaussian. In Figure 5.12, we see that a sub-Gaussian signal becomes more Gaussian like as it is blurred i.e. it's kurtosis increases towards 3. In Figure 5.13, the kurtosis of a super-Gaussian signal reduces towards 3, making it more Gaussian. In both cases, a simple Gaussian filter was used to blur the signal although this behavior holds with any other kind of filter.

Shalvi and Weinstein put together these key observations into a novel algorithm for blind equalization. Assuming that the impulse response k was invertible, their idea is to search for an *equalizing filter* e which is the inverse of k, such that $e \oplus k = \delta$, where δ is the delta kernel. The observed signal y is then convolved with the equalizing filter e to give the recovered signal z, which is a phase-shifted version of x. Figure 5.14 illustrates the overall approach.

The Shalvi-Weinstein algorithm is based on maximizing the non-Gaussanity of the reconstructed signal z. Based on the above, this leads to a simple criterion for the blind equalization in the

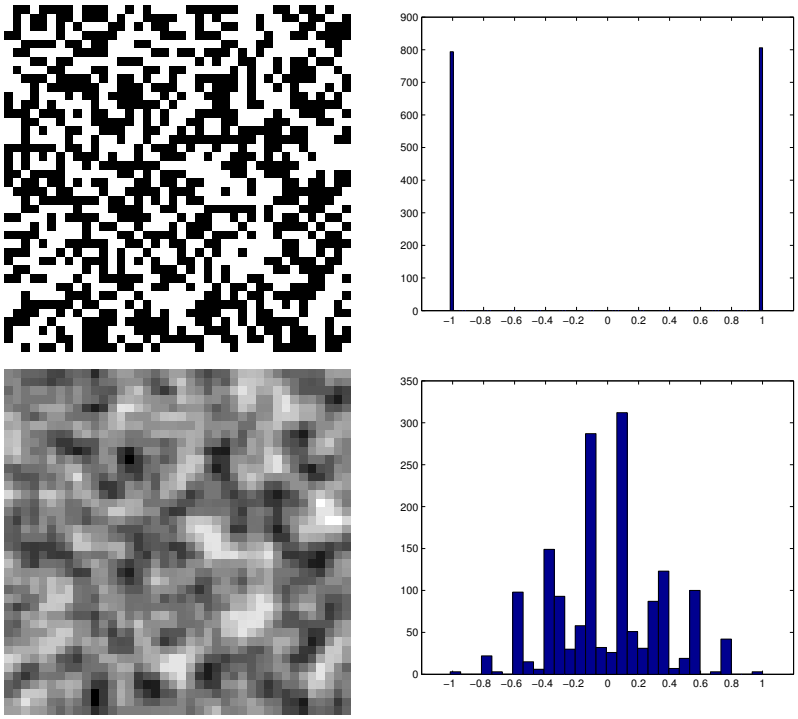


Figure 5.12: Blurring a sub-Gaussian signal increases it's kurtosis towards 3. Top-left: original IID signal ($\kappa = 1$); Top-right: the histogram of the signal; Bottom-left: the blurred version of the signal ($\kappa = 2.78$); Bottom-right: histogram of the blurred signal. The signals are all 1D but shown as images for clarity.

case of super-Gaussian x:

$$e^* = \arg\max_e \kappa(y \oplus e) \tag{5.12}$$

In the case of a sub-Gaussian x, the max in Eq. 5.12 is replaced by a min. The only inputs to the algorithm are whether the unknown distribution is sub-Gaussian or super-Gaussian, and the power (ℓ_2 norm) of x. After recovering e^*, z may be recovered as $z = y \oplus e^*$.

Under the assumptions that k has an inverse filter e, and that x is comprised on IID non-Gaussian measurements, there is a simple and elegant proof of the global optimality of the Shalvi-Weinstein algorithm. The simplified version of the proof is now presented. It relies on the joint cumulants of a signal x. The second-order and fourth-order joint cumulants of a signal x are given as:

$$C_2^x = E(x^2) C_4^x = E(x^4) - 3E(x^2)^2 \tag{5.13}$$

where $E(x)$ is the expectation of x. Joint cumulants enjoy the following properties [177]: linearity i.e. $C^(ax)_p = a^p C^(x)_2$; and the joint cumulant of independent variables is 0. A key property of the kurtosis of a zero-mean IID signal is that it can be expressed in terms of cumulants:

$$\kappa(x) = \frac{C_4^x}{(C_2^x)^2} \tag{5.14}$$

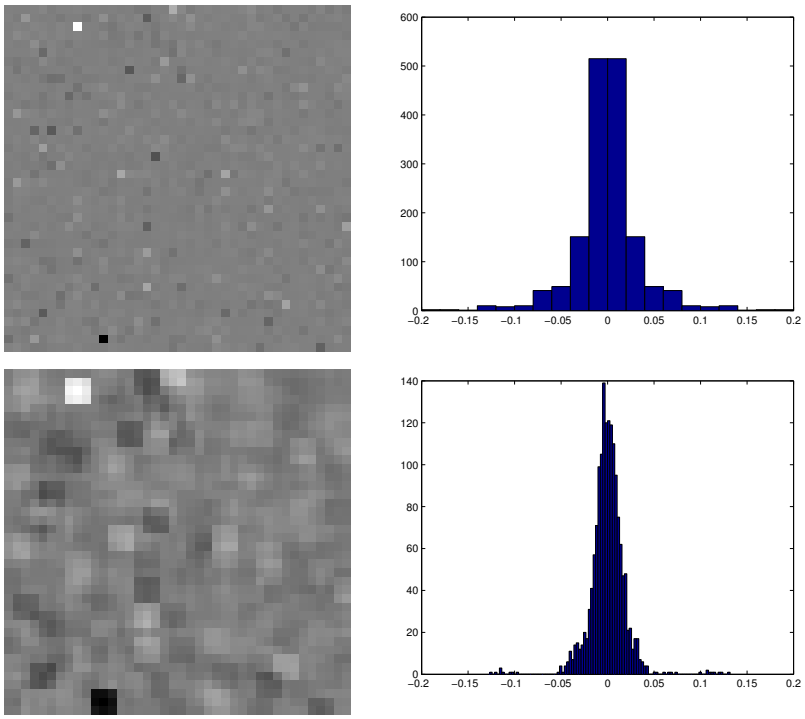


Figure 5.13: Blurring a super-Gaussian signal decreases it's kurtosis towards 3. Top-left: original IID signal with ($\kappa = 116.2$); Top-right: the histogram of the signal; Bottom-left: the blurred version of the signal ($\kappa = 14.7$); Bottom-right: histogram of the blurred signal. The signals are all 1D but shown as images for clarity.

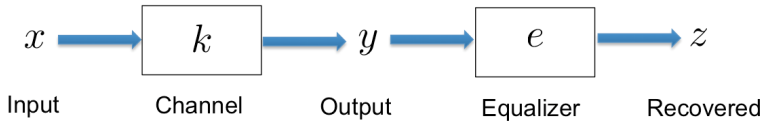


Figure 5.14: Illustration of the Shalvi-Weinstein algorithm: e is an inverse filter to k; and z is the recovered signal. $y = x \oplus k$ and $z = y \oplus e$.

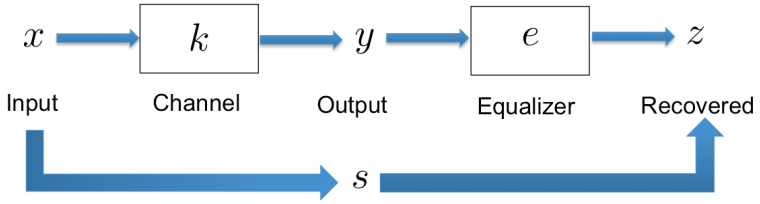


Figure 5.15: We look for a filter s such that $s = k \oplus e$ and $z = x \oplus s$. s is therefore a phase-shifted version of the δ.

The proof of global optimality for the super-Gaussian case, starts by introducing a filter s such that $s = k \oplus e$; see Figure 5.15. Hence $z = y \oplus e = y \oplus e \oplus k = x \oplus s$. We want $s = \delta$. Now, $z = x \oplus s$. By the properties of cumulants, this means that $C_4^z = C_4^x \sum_i s_i^4$, where s_i are the

taps of filter s. Simiarly $C_2^z = C_2^x \sum_i s_i^2$. Now if we restrict the power of z to be equal to the power of x, then $C_2^z = C_x^2$, which implies that $\sum_i s_i^2 = 1$. Now maximizing the kurtosis of z is, by Eq. 5.14, equivalent to the constrained optimization problem:

$$\begin{aligned} &\max C_4^z \\ \text{s.t. } &C_2^z = C_2^x \end{aligned}$$

By the above, this is equivalent to

$$\begin{aligned} &\max \sum_i s_i^4 \\ \text{s.t. } &\sum_i s_i^2 = 1 \end{aligned}$$

which has clearly only one maximum which is $s = \delta$. It can be shown using Lagrange multipliers that all other extrema of this problem are unstable saddle points [141]. Furthermore, this algorithm in s can be converted to an algorithm in terms of e. This algorithm converges extremely fast and is robust to noise. Other variants of this algorithm and more general proofs are given in [141].

Using the Shalvi-Weinstein directly for image deconvolution is problematic. The assumption of x being IID is clearly untrue. Furthermore, many image blur kernels do not have compactly supported inverses. Kurtosis is a measure of non-Gaussanity of the recovered signal. From Eq. 5.11 we see that computing the kurtosis required taking the fourth power. If instead, we use a power of 1, we recover our ℓ_1/ℓ_2 prior (for zero-mean signals). The ℓ_1/ℓ_2 prior is therefore another way of measuring the non-Gaussanity of a signal. Kurtosis is extremely sensitive to outliers. An example of this sensitivity is shown in Figure 5.16. A single value in the original signal (blue) is perturbed to give the signal in red. We see that the kurtosis measure changes dramatically, but the ℓ_1/ℓ_2 measure is much more stable and therefore useful in practice.

5.6 Discussion

Our approach to blind deconvolution is motivated by a re-analysis of the interaction between image regularizers and the effects of blur on the high frequencies in an image. The crucial component of our algorithm is the introduction of a novel scale-invariant regularizer that compensates for the attenuation of high frequencies and therefore greatly stabilizes the kernel estimation process. However, since this regularizer is non-convex, we introduce a minimization scheme that amounts to solving a series of l_1 problems with different regularization parameters. The resulting algorithm is applicable to different models of blur formation and is fast and robust to the choice of parameters. All Matlab code for our algorithm and the experiments in this paper are available at `www.cs.nyu.edu/~dilip/research/blind-deconvolution/`.

Our measure has been used in [63] as a relative measure of the amount of blur in a signal.

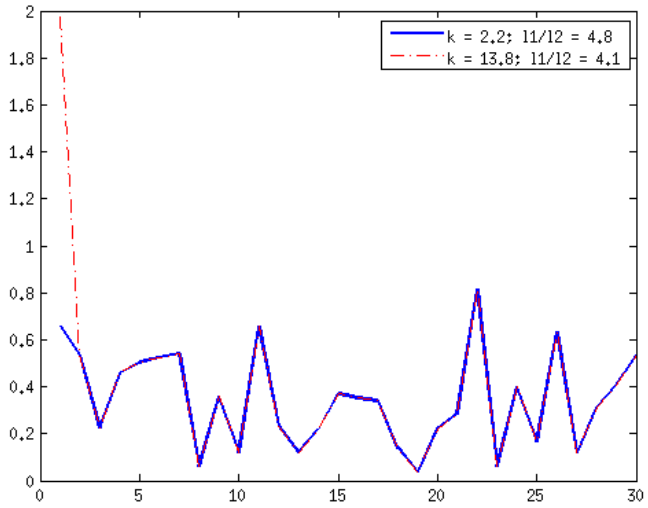


Figure 5.16: The Kurtosis measure is much more sensitive than the ℓ_1/ℓ_2 measure to outliers.

However, our regularizer is by no means the final word in blind deconvolution. While it works well for images with highly sparse gradients, such as city and indoor scenes, it tends to fail when textures dominate the image. The gradients of textured images have a kurtosis very close to that of Gaussian. Therefore their gradients are not sparse. For such images, the ℓ_1/ℓ_2 measure behaves unpredictably - with the cost of blurred images being lower than that of sharp ones. As a result, ℓ_1/ℓ_2 is not robust enough for practical applications. This was numerically demonstrated in [165] over a large range of images. The workaround proposed in [165] is not a new regularizer but a combination of the variational method of [103] and [32]. A regularizer that works well for all types of natural images is therefore still an open research problem.

Chapter 6

Removing Localized Corruption from Natural Images

The work described in this chapter is done jointly with David Eigen and Rob Fergus, and an extension of this work is currently under review.

Traditional approaches to removing image corruption such as blur or noise combine a natural image prior with a reconstruction term. The latter relies on a good generative model of the corruption — which may not exist for many distortions encountered in the real world. In this paper we explore approaches for learning a direct mapping from the corrupt input image to the clean image, obviating the need for any kind of generative model. We evaluate the approaches on several types of synthetic corruption, finding that neural-network based models perform the best. Our techniques can be used for many types of localized corruption. We demonstrate this using photographs of real-world scenes taken behind a pane of glass with water droplets, akin to a rainy window. Our model removes most of the raindrops without significant blur, the first such demonstration of this application.

6.1 Introduction

Natural image priors (as seen in earlier chapters) are an important tool, allowing the recovery of images in a range of under-constrained problems, such as denoising, deblurring and super-resolution. In these problems, the input is a corrupted image y and the goal is to estimate the original uncorrupted image x. Knowledge about the corruption process is embodied in a formation function $K(x)$ that applies the corruption to the estimate of x, which can then be compared to the observed image y. In non-blind problems $K(.)$ is given, but in blind problems (e.g. blind deconvolution) it may have latent variables that must also be inferred. Using a

probabilistic interpretation, the overall problem can be written as:

$$p(x|y, K) \propto p(y|x, K)p(x)p(K) \tag{6.1}$$

The likelihood term $p(y|x, K)$ imposes the formation constraint and $p(x)$ is a prior on natural images. Since these image restoration problems are under-determined, many values of x have a high likelihood and $p(x)$ selects the most natural-looking solution. One attraction of this approach is that a image prior can be built off-line from many examples of natural images and then applied to a wide range of problems, each having a different formation model K.

But what if the image contains a type of corruption for which we cannot write down an accurate K? For example, consider the example shown in Figure 6.3(top left), which shows a photo taken through glass with rain drops on it. An accurate formation model would require latent variables for each rain drop (position, size, orientation) that would have to be inferred – impractical for a real image with hundreds or thousands of rain drops. Without a viable K, the likelihood term in Eq. 6.1 cannot be formulated and the framework above cannot be used.

In this chapter we explore a different approach that directly models the posterior $p(x|y)$ in Eq. 6.1 (so K is not needed). This can be thought of as a conditional image prior: the probability of a clean image x, given the corrupted input y. If a point estimate of x will suffice, then this reduces to a function that maps the noisy image to the clean one. This function must be learned from pairs of clean/noisy patches and so adapts to the specific type of corruption (unlike the $p(x)$ prior in Eq. 6.1). The function must have high capacity, since it needs to capture the the joint space between all possible natural image structures and all possible types of corruption. In this paper we explore four different options for the mapping, showing results on both synthetic and real-world examples of corruption that cannot effectively be removed with other approaches. A literature survey was given in Section 2.7.

6.2 Approach

Given a noisy image y, our goal is to recover the true clean image x^*. From pairs of clean/noisy images, we learn a conditional model $p(x|y)$ such that $\arg\max_x p(x|y)$ is close to x^*. We evaluate four different models for $p(x|y)$: (i) a Gaussian mixture model; (ii) joint sparse-coding; (iii) a convolutional neural network and (iv) a mean-covariance restricted Boltzmann machine (mcRBM).

Given that most of the inter-pixel dependencies in natural images are local in nature, we follow many other approaches and represent the image as a set of small overlapping 8×8 patches, which are assumed to be independent to one another. A linear *patchification* operator P converts an image y into a set of patches $\{y_i\}$. The corresponding *unpatchification* operator P^T converts the set of patches back into an image (effectively averaging all patches at a given location). Although operating on patches makes our approach tractable, it limits the size of the corruption that can be removed since it must fit within a single patch.

6.2.1 Gaussian mixture model

Directly modeling the conditional patch distribution $p(x_i|y_i)$ with a Gaussian mixture would require that the parameters of each component be functions of y_i. However, noting that (i) only $\arg\max_x$ is required, not the full distribution and (ii) y_i is observed with zero uncertainty, we can instead model the joint $p(x_i, y_i)$ since the denominator $p(y_i)$ is a constant for each patch. We thus model the joint space of clean/noisy patches with a 128 dimensional Gaussian mixture on raw pixel values:

$$p(x|y) \propto \prod_i p(x_i, y_i) = \prod_i \sum_c \pi_c \mathcal{N}(x_i, y_i|\mu_c, \Sigma_c) \tag{6.2}$$

where π_c and μ_c are the component weights and means respectively and Σ_c are full covariance matrices. We train this model using EM, initializing π_c, μ_c using k-means. One important issue is the preponderance of low contrast patches in natural images – to prevent the model from concentrating on them, all patches with a variance less than 0.001 are removed from the training set.

At test time, we initialize each x_i to y_i and maximize $\frac{\partial \log p(x|y)}{\partial x}$ (as given in Eq. 6.2) using conjugate gradient descent. As we use large models with 70 components, each having full covariance matrices, there are a large number of parameters that must be estimated in training. Despite training on millions of patch pairs, some components may only see a small portion of the training data, hence we find that Laplace smoothing is beneficial at test time. We implement this by adding ϵI to the covariance of each component, ϵ being determined on validation data (values given in Table 6.2).

6.2.2 Joint sparse coding

We adapt the joint sparse coding framework, introduced by Yang et al.[182], to compute a clean patch x_i, given the input noisy patch y_i. In this approach, the idea is to have a latent sparse feature vector α_i that reconstructs *both* the clean and noisy patches, via separate over-complete dictionaries. In training the goal is to learn the clean and noisy dictionaries D_x and D_y:

$$\min_{D,\alpha_i} \sum_i \|D\alpha_i - z_i\|^2 + \lambda\|\alpha_i\|_1 \tag{6.3}$$

where $D = \begin{bmatrix} D_x \\ D_y \end{bmatrix}$, $z_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ and λ controls the sparsity of the common feature vector.

Evaluation of a new image y proceeds in three stages: (i) For each patch the optimal sparse vector α_i^* is inferred using the noisy dictionary D_y:

$$\alpha_i^* = \arg\min_{\alpha_i} \|D_y\alpha_i - y_i\|^2 + \lambda\|\alpha_i\|_1 \tag{6.4}$$

(ii) the estimated clean patch is directly computed: $x_i = D_x\alpha_i^*$. (iii) the clean image x is

recovered from the set of patches $\{x_i\}$ using P^T. In all our experiments, we use $\lambda = 0.15$ and 4096 element dictionaries for both D_x and D_y.

6.2.3 Neural network

In this approach, we directly model the prediction function $f(y) = \arg\max_x p(x|y)$ using a neural network for f. Specifically, we train a multi-layer perceptron with hyperbolic tangent nonlinearities and a linear output layer:

$$\begin{aligned} f_0(y) &= y \\ f_l(y) &= \tanh(W_l f_{l-1}(y) + b_l), \quad l = 1, ..., n-1 \\ f(y) &= W_n f_{n-1}(y) + b_n \end{aligned}$$

where W_l and b_l are weights and biases for each layer. Since these weights are shared between all patches in the image, this constitutes a Convolutional Neural Network [92]. In our application, we use two hidden layers (i.e. $n = 3$), with 512 units each. We train the network using stochastic gradient descent to minimize the mean squared error loss,

$$L(W_1, ..., W_n, b_1, ..., b_n) = \frac{1}{2|D|} \sum_{i \in D} L_i(y_i) = \frac{1}{2|D|} \sum_{i \in D} ||f(y_i) - x_i^*||^2$$

where y_i iterates over each corrupted patch and x_i^* is the corresponding clean patch.

To cover the full linear range of the hyperbolic tangent, we transform the data by subtracting the mean and dividing by the standard deviation computed across all pixels in a sample of the training set. We initialize the weights in each layer l by sampling from a uniform distribution on $(-m_{l-1}^{-1/2}, m_{l-1}^{-1/2})$, where m_{l-1} is the number of units in the previous layer [93].

6.2.4 Mean-Covariance RBM

While the neural network is simple and direct in modeling the prediction function, different shifts of pixel intensities need to be modeled separately, leading to duplication of weights for each filter. The Mean-Covariance Restricted Boltzmann Machine (mcRBM) [132] overcomes this limitation by splitting the hidden layer into two types of units: mean units, which model the pixel values on an absolute scale, and covariance units, which model the interactions between pixels. This is defined by an energy function that augments the regular RBM energy for Gaussian visible units x and binary mean hiddens h^m with binary covariance hidden units h^c:

$$E_{mcRBM}(x, h^m, h^c) = -h^{mT} W^T x + \frac{1}{2} x^T C \text{diag}(h^c) C^T x + \frac{1}{2} x^T x - b^{xT} x - b^{mT} h^m$$

Here, W is a $R \times M$ matrix that connects pixels and mean hiddens, and C is a $R \times F$ matrix that connects pixels and covariance *factors*. Each factor C_f produces a rank-1 modification $h_f^c C_f C_f^T$

of the inverse covariance for the patch: same-sign entries have positive partial correlation, while opposite-sign entries are anti-correlated.

Given the hidden units, the conditional distribution of the visibles is a normal distribution; given the visibles, the hidden units are Bernoulli variables:

$$p(x|h^m, h^c) = \mathcal{N}(\Sigma(Wh^m + b^x), \Sigma), \quad \text{where} \quad \Sigma^{-1} = C\text{diag}(h^c)C^T + I$$

$$p(h_i^m = 1|x) = \sigma(W_i^T x + b_i^m), \quad p(h_f^c = 1|x) = \sigma(-\frac{1}{2}(C_f^T x)^2)$$

where $\sigma(t) = e^t/(1 + e^t)$ is the sigmoid function.

In our denoising application, we use two mcRBM's placed back-to-back: one that connects the noisy patch input with the hidden layer, and one between the hidden layer and cleaned-up patch output. The new energy function then becomes

$$\begin{aligned} E_{mcRBM2}(x, y, h^m, h^c) &= -h^{mT}W_1^T x - h^{mT}W_2^T y - b^{xT}x - b^{yT}y - b^{mT}h^m \\ &\quad + \frac{1}{2}x^T C_1 \text{diag}(h^c)C_1^T x + \frac{1}{2}y^T C_2 \text{diag}(h^c)C_2^T y \\ &\quad + \frac{1}{2}x^T x + \frac{1}{2}y^T y \end{aligned} \tag{6.5}$$

To infer a clean x given a noisy y, we marginalize out the hidden units to obtain the free energy of just x and y, and use CG to minimize $E(x, y)$ as a function of x. Using a binomial factorization of the possible combinations of mean and covariance unit states, the free energy is

$$\begin{aligned} E(x, y) &= -\log \sum_{h^m \in \{0,1\}^M} \sum_{h^c \in \{0,1\}^F} \exp(-E_{mcRBM2}(x, y, h^m, h^c)) \\ &= -b^{xT}x - b^{yT}y - \sum_{i=1}^{M} \log(1 + \exp(W_{1,i}^T x + W_{2,i}^T y + b_i^m)) \\ &\quad + \frac{1}{2}x^T x + \frac{1}{2}y^T y - \sum_{f=1}^{F} \log(1 + \exp(-\frac{1}{2}((C_{1,f}^T x)^2 + (C_{2,f}^T y)^2)) \end{aligned} \tag{6.6}$$

We train the model as a structured perceptron with stochastic gradient descent [36, 94]. The loss function is

$$L(\theta) = \sum_{i \in D} E(x_i^*, y_i) - E(x_i, y_i)$$

For each training pair (x_i^*, y_i), we first perform inference to produce a predicted clean patch x_i from the noisy y_i. Then, we update the weights with a gradient step,

$$\theta_{t+1} \leftarrow \theta_t - \eta_t \left(\frac{\partial E}{\partial \theta}(x_i^*, y_i) - \frac{\partial E}{\partial \theta}(x_i, y_i) \right)$$

where $\theta = (W_1, W_2, C_1, C_2, b^x, b^y, b^m)$ is the combined weights. For the final weight estimate, we use the average of the weights in the last iteration over all the datapoints. That is, $\tilde{\theta} = \frac{1}{|D|}\sum_{t=t_0}^{t_0+|D|} \theta_t$ where t_0 is the last epoch start time. We found this averaging was needed to

reduce undue influence of the most recent examples.

The network we use has 512 covariance and 512 mean units. To initialize the network, we sample the covariance filters from a normal distribution, $C_f \sim US^{-1/2}\mathcal{N}(0, I)$, where USU^T is the singular value decomposition of the clean training patches' covariance. This distribution is spherical in the PCA-whitened space of the data, and hence in pixel space it has relatively large low-frequency components. The two layers of covariance filters are set initially to be the same, $C_1 = C_2$ at $t = 0$.

The mean filters are initialized by training a 2-layer neural network with the same connections as the mean half of the mcRBM. This is motivated by the fact that if one ignores the feedback influences of the output x, inference is solved by feed-forward propagation. To see this, consider Eq. 6.6: by setting its derivative equal to zero and ignoring $W_{1,i}^T x$ and $C_{1,i}^T x$ (i.e. zeroing them), this (modified) function is minimized at $x = b^x + \sum_{i=1}^{M} W_{1,i}^T \sigma(W_{2,i}^T y + b_i^m)$—a feed-forward network.

6.3 Datasets

6.3.1 Synthetic corruption

We tested each of the above models on five different types of synthetic noise, examples of which are shown in Figure 6.1:

1. *Snow*: Small white line segments between 5-8 pixels in length, 2-3 pixels in width and one of 10 different orientations are added at random image locations.

2. *Mixed Snow*: A more challenging version of "snow," with transparency chosen uniformly at random between 0 and 1.

3. *Scratches*: Five curves generated using a cubic spline of 6 random points are added to each image. The gray and transparency levels of each curve are chosen uniformly at random between 0 and 1. These are meant to mimic the scratches often found in archival film.

4. 3×3 *Blur*: Each image is blurred using a 3×3 Gaussian kernel. This kernel is typical of those used for super-resolution.

5. 7×7 *Blur*: Each image is blurred using a 7×7 Gaussian kernel.

The snow and scratch corruption are typical of the localized noise that is hard to model generatively and thus challenging for conventional restoration methods. However, the two types of blur can be well modeled by standard deblurring methods based on likelihood + natural image priors. We include them to see how well our approaches fare against existing algorithms. The training set for each corruption consists of 10 million patches chosen randomly from 15000 images from the PASCAL VOC 2011 [48] and the Berkeley Segmentation Database [111]. We filtered the training set so that all patch pairs had at least 1 pixel difference between clean and corrupt

Figure 6.1: Examples of corruptions. L to R: "Snow," "Mixed Snow," "Scratches" and "7×7 Blur"

versions. The test set consists of 10 images drawn at random from PASCAL VOC 2011 (and from which no training patches were taken).

6.3.2 Water droplets dataset

Photographs captured through a rainy window are locally corrupted by water droplets which refract light from the scene. As the position and size of the water droplets are essentially random they are difficult to remove with conventional approaches. For example, one approach would be to try to identify the rain drops and then use a standard inpainting approach to remove them. But since the appearance of the rain drops varies considerably, they are difficult to detect without erroneously including parts of uncorrupted image. The water drops therefore represent the awkward type of real-world corruption for which our techniques are well-suited.

We simulate the rainy window by photographing real-world scenes through a piece of anti-reflective glass onto which water has been sprayed. Removing the glass allows us to obtain a ground truth clean image. We gathered a dataset of 49 real-world clean/noisy image pairs, of which 3 were held out for testing. From the training images, we generated 3.5 million 8×8 patch pairs for learning our models.

6.4 Results

Table 6.1 shows the performance of each of the four approaches, applied to the five different types of synthetic corruption. For each model/corruption combination we train a separate model on the corresponding training set and then compute the PSNR between the output of each approach and ground truth image, averaged across the 10 image test set. While all methods improve the input image, the neural network model outperforms the others, except on the small 3×3 blur where the GMM dominates.

The performance of the GMM and neural net on the synthetic blur images compares favorably with leading deblurring methods. The algorithm of Krishnan et al. [86] gives the following mean PSNRs on the test set – 3x3 blur: 32.79 (+7.08) and 7x7 blur: 25.96 (+2.24). These results are worse than the +8.53 and +2.78 obtained by the GMM for 3x3 blur and neural net for 7x7 blur

	Corrupt	GMM	JointSC	NN	mcRBM
Time(s)	-	1.2×10^4	2.0×10^3	37.4	849.7
Snow	23.43	27.83 (+4.40)	26.74 (+4.03)	**32.59 (+9.16)**	28.78 (+5.35)
Scratches	24.21	-	24.81 (+0.41)	**28.63 (+4.42)**	25.44 (+1.23)
Mixed Snow	27.54	-	27.80 (+0.60)	**32.26 (+4.72)**	28.69 (+1.15)
3×3 Blur	25.73	**34.26 (+8.53)**	26.07 (+0.36)	30.79 (+5.07)	28.80 (+3.08)
7×7 Blur	23.29	25.47 (+2.19)	23.91 (-1.81)	**26.07 (+2.78)**	24.54 (+1.26)

Table 6.1: Mean PSNR over the 10 test images for different combinations of corruption type and model. Numbers in parentheses show the relative PSNR gain over the corrupt image. The first row gives the time taken in seconds to test the model on a single image of size 500×350 pixels on a single Xeon 2.67GHz core.

respectively. This shows that directly modeling the posterior in Eq. 6.1 is a viable approach, even when a good generative model for the corruption exists.

Figure 6.2 shows a qualitative comparison between models for a single image with the "snow" corruption. Most methods preserve the clean patches in the image well, while removing the corruptions. The neural network result is the cleanest, followed by the GMM. The mcRBM produces ringing artifacts around the edges, while the joint sparse coding model leaves blurry regions. Further comparison images for the other corruption types can be found in the supplementary material.

For the "scratch" and "mixed snow" corruption, the GMM model did not move from the initial corrupt image. On closer examination, we found that for these two types of corruption, which are less distinctive than others, the probability of a clean/noisy (c, n) patch pair under the trained model was *lower* than that of a noisy/noisy (n, n) pair[1]. Since the latter is used as an initialization, there is no way the model can improve the patch and in practice it does not move from the initialization. Table 6.2 compares $\log p(c, n)$ to $\log p(n, n)$ for GMM models trained on the different corruption types. Since the GMM focuses on modeling $p(x, y)$ rather than ensuring $\arg\max_x p(x, y)$ is close to x^*, it is unable to distinguish less noticeable forms of corruption from clean patches.

The mcRBM removes much of the corruption, but often leaves more artifacts than the neural net. Although the NN has stacked hidden layers, we found that the NN performance with a single 1024-unit hidden layer was between those of the stacked NN and mcRBM. A reason for this may be that the MSE loss used by the NN modulates its gradient on a per-pixel basis, while the perceptron loss we use for the mcRBM works at a per-datapoint level and does not target errors in specific pixels.

6.4.1 Water droplet removal

Figure 6.3 shows two test cases of water droplet removal. A 5×5 median filter is shown as a baseline, but this significantly blurs the image. We also tried a bilateral filter, but this produced

[1]This was the case for all values of the smoothing parameter ϵ.

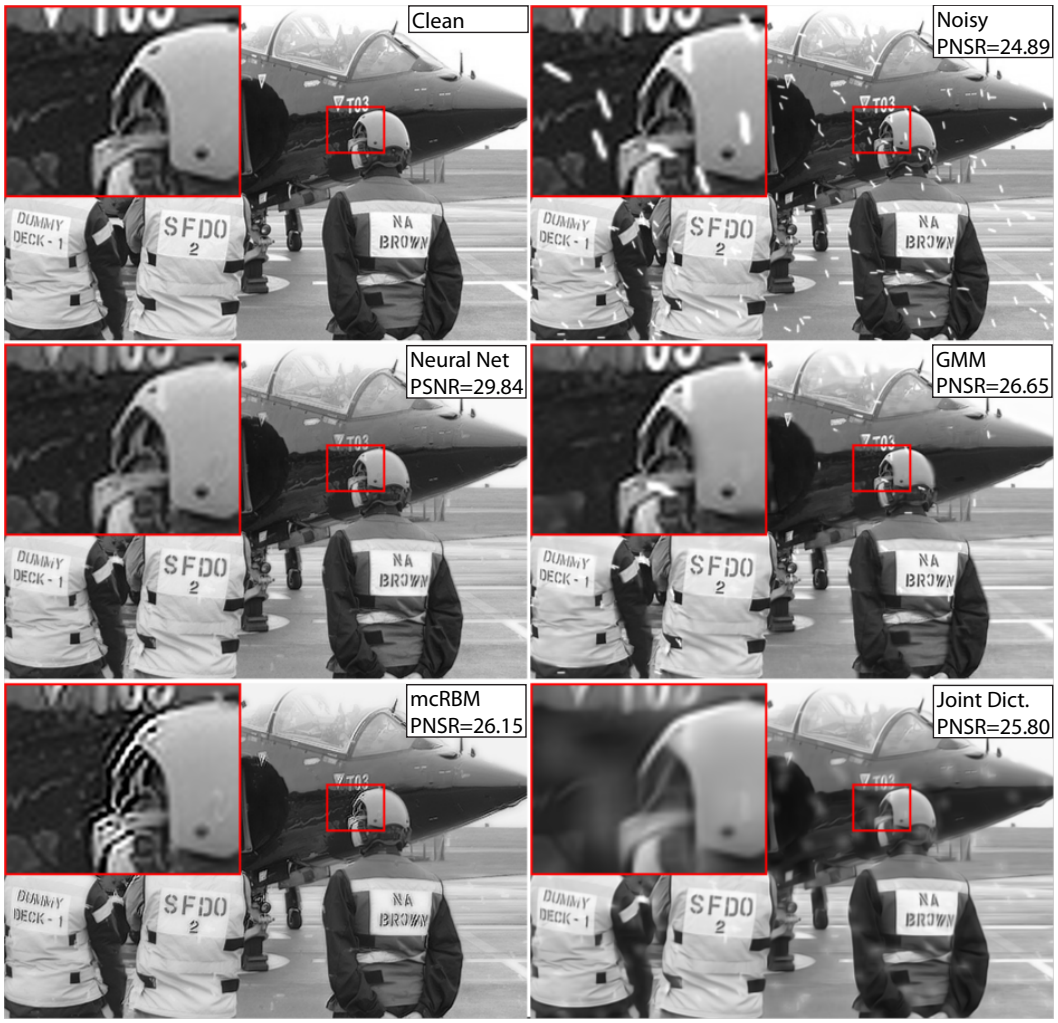


Figure 6.2: Comparison of approaches on an image corrupted with synthetic "snow".

even worse results. Our neural-network model does a good job of removing the small droplets, although some artifacts remain from larger drops. Full size versions of these images, and further examples can be found in the supplementary material.

6.5 Discussion

We have explored four different approaches that attempt to learn a direct mapping from a corrupted input image to a clean output. This allows us to remove corruption that cannot be addressed by existing methods. Results on synthetic forms of corruption surprisingly show that a neural network model consistently outperforms other approaches. We then demonstrate that the neural network approach can be used to remove water droplets from a real image, a novel application that has considerable practical relevance (e.g. for cameras mounted outdoors).

However, our approach has several drawbacks: (i) we must collect training images with and

	Snow	Scratches	Mixed Snow	Blur (3×3)	Blur (7×7)
ϵ	10^{-4}	-	-	0	0
$\log p(c, n)$	**279.5**	296.3	376.5	**406.2**	**408.2**
$\log p(n, n)$	273.3	**320.4**	**385.1**	-2364.4	-57.4

Table 6.2: Log probability for each GMM model for the 5 different corruption types, averaged over the training set. (c, n) = clean/noisy patch pair; (n, n) = noisy/noisy patch pair. The larger the gap between $\log p(c, n)$ and $\log p(n, n)$, the better the model performs. The reversal for "scratches" and "mixed snow" explains why the GMM model does not work in these cases. The first row shows the optimal value of ϵ, selected on a validation set.

without the corruption, which might be difficult in some practical applications; (ii) the model is trained for one type of corruption, so may not be robust to changes in the corruption encountered in test images and (iii) we can only remove corruption that is localized and cannot handle, for example, large blurs.

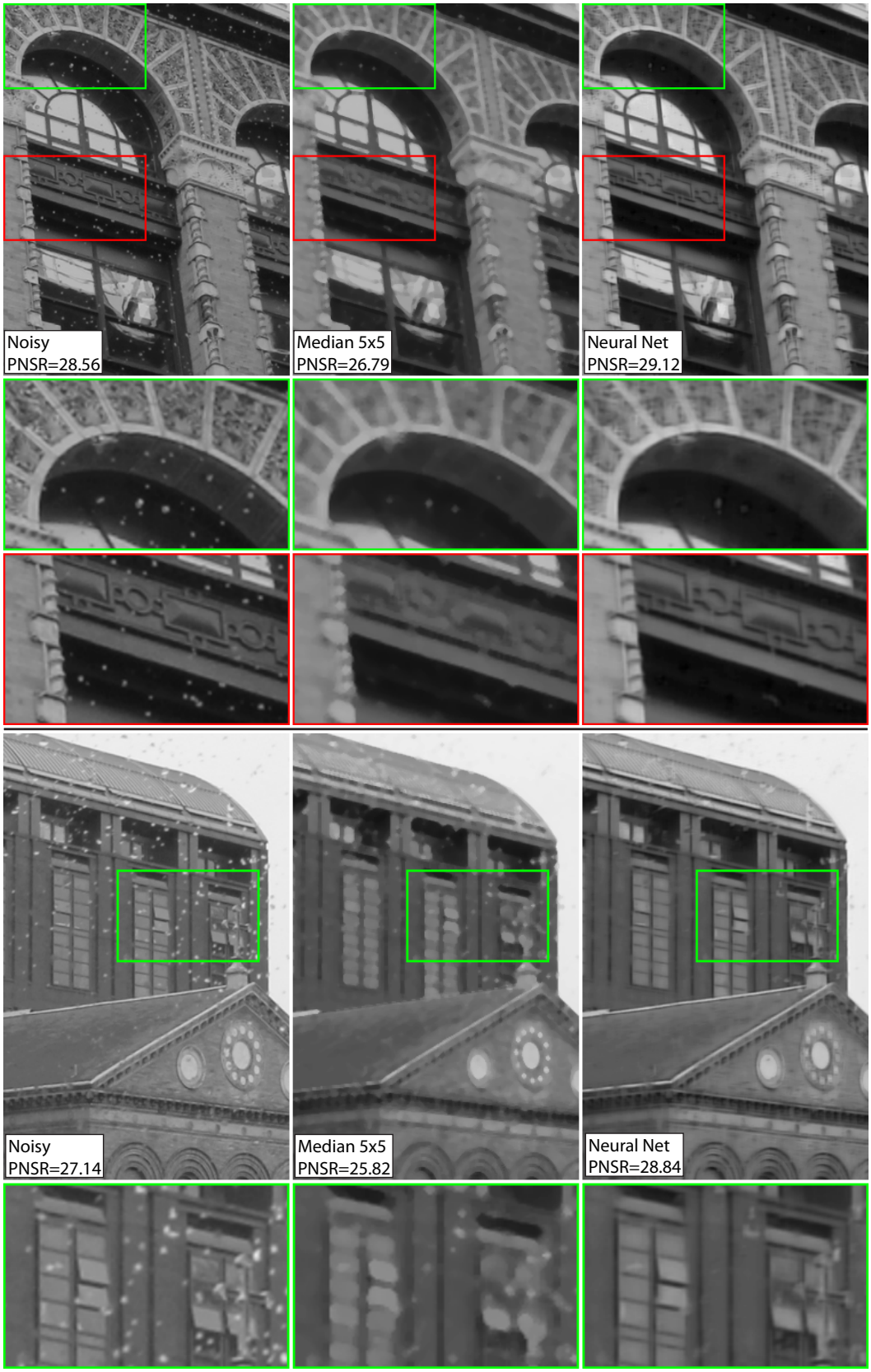


Figure 6.3: Two test images corrupted by real rain drops. A 5×5 median filter removes the small drops but significantly blurs the image, decreasing the PSNR. Our neural network removes most of the rain with minimal blurring. This figure is best viewed in electronic form.

Chapter 7

Efficient Preconditioning of Laplacian Matrices for Computer Graphics

7.1 Introduction

The work described in this chapter is done jointly with Rick Szeliski and Raanan Fattal. We develop two preconditioners for linear systems involving Laplacian matrices. The first solver was published in SIGGRAPH Asia 2011 [89] and the second solver in SIGGRAPH 2013 [84].

A large number of problems in computer graphics and computational photography are formulated as norms over gradients and solved using discrete Poisson equations. Examples in computational photography include gradient-domain tone mapping [53], Poisson blending [127], alpha matting [151], image colorization [97], tonal adjustment [106], edge-preserving smoothing [51], and image relighting and non-photorealistic rendering [12]. Figure 7.1 illustrates some common 2D problems.

Three-dimensional geometric processing applications include mesh segmentation [107] and geodesic distance computation [38]. Examples are shown in Figure 7.3 and Figure 7.2. While the Poisson equation approach excels in terms of quality and mathematical conciseness, it comes at a considerable computational cost, as it requires solving very large and poorly-conditioned linear systems.

The matrices in these linear systems are Laplacians. The connections between Laplacians and graphs has been given in Section 2.8. These Laplacians are M-matrices (non-positive off-diagonals and positive diagonal elements), and so they are symmetric and positive-semidefinite (SPD). When only local interactions in the image or mesh are modeled, they are sparse and banded as well. By sparsity, we mean that the maximum number of non-zero elements in each row of the

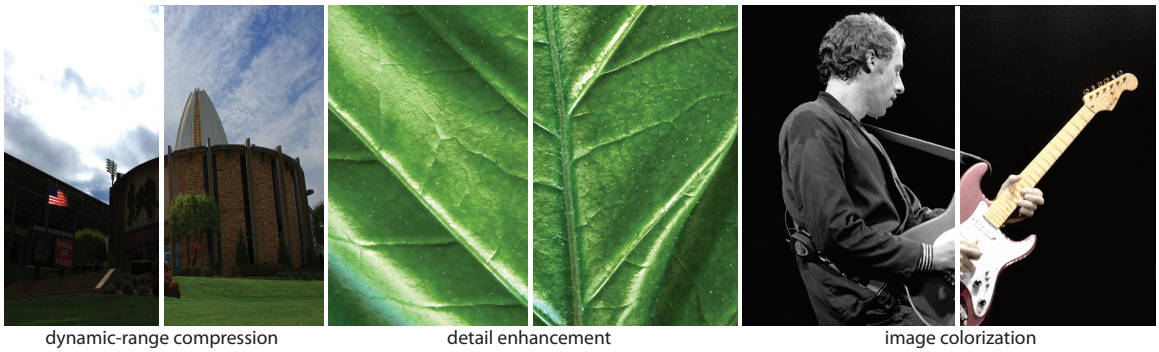


Figure 7.1: Examples of two-dimensional problems that involve the solution of discrete Poisson equations. From left to right are: dynamic range compression [53], detail enhancement [51] and image colorization [97].

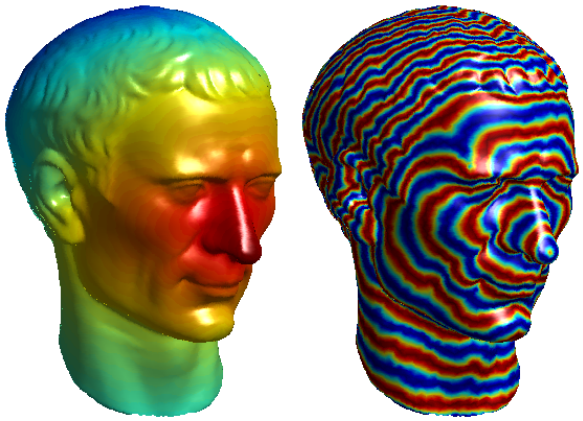


Figure 7.2: An example of geodesic distance computation [38]. A point on the nose of the 3D model is marked as source and distances to it are computed using heat diffusion. The left figure shows the distance to the source point (red is closer, blue is further) and the right figure shows isoline contours.The overall algorithm requires two Poisson equation solves involving the same Laplacian with different diagonal shifts and different right-hand sides.

matrix is a fixed, small constant much less than the dimension n of the Laplacian. When the problem is defined uniformly in space, or equivalently, when the rows (and columns) of the matrix consist of the same values, just shifted, the resulting Laplacian is homogenous and is otherwise called *inhomogenous*.

In most computer graphics and computational photography applications, the size of the Laplacian n is very large. As explained in Section 2.8, an exact solution is not necessary and iterative solvers are typically used to approximate the solution. In the case of sparse matrices, each iteration consists of $O(n)$ operations. The problem is that the number of iterations needed to achieve a particular accuracy depends on the condition number of the matrix and in the case of Laplacian matrices, this number grows with n. Inhomogeneous Laplacian matrices often have substantially higher condition numbers and require more iterations to solve. While we have shown in Section 2.8 how Laplacians are related to graphs, we provide a more direct definition

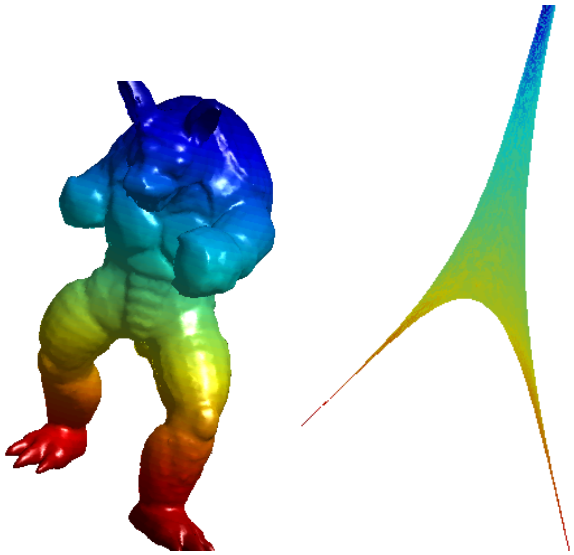


Figure 7.3: Mesh Segmentation using mesh planar embedding [107]. The 3D mesh on the left is embedded into a planar shape by projecting it onto the two lowest non-zero eigenvectors. The planar shape is then contour analyzed to determine segmentation. Note how the two legs of the creature get mapped to the two spikes in the planar region, thereby making segmentation easier.

of Laplacian matrices is Section 7.2 which sheds light on how the Poisson problems in graphics usually arise.

Our new preconditioners are highly efficient and have an overall linear O(n) construction time and memory complexity. Our experiments show that one of the methods outperforms or equals other state of the art methods, both in terms of operation count and wall-clock time. This speedup results from the new methods ability to dramatically reduce the matrix condition number. Our experiments also show that condition numbers of orders of 10^6 due to severe spatial irregularities are reduced to less than 10.

In the case of homogenous problems, our algorithms reduce to the GMG method and hence achieve optimal performance. For linear or tree-like inhomogeneous regions, our algorithm devolves to linear time cyclic reduction [65] and (parallel) tree-based preconditioners. Performing well on these extreme cases makes our method well suited for mixed systems that contain large uniform regions separated by strong discontinuities, which often arise in graphics applications, such as edge-preserving smoothing [51]. Our optimized MATLAB/Mex code is available for download at `www.cs.nyu.edu/~dilip/hsc/`.

[143] develop a multigrid solver to handle mesh deformation problems. They show significant speedup over direct solvers for meshes of upto 3 million vertices. The HSC solver we present in this chapter is faster in wall-clock time. For example, the solver in [143] takes 2.8 seconds to process a mesh with 800K vertices on a Pentium 4, 3.8GHz. We process a mesh of the same size in about 0.5s on a single-core Xeon 2.7GHz. [14] present a GPU-based multigrid solver. However, they restrict their numerical experiments to small grids with less than 200K vertices whereas we consider problems upto tens of millions of vertices.

7.2 Mathematical background

In this section, we review Laplacian matrices and their connection to quadratic regularization problems. The connections between Laplacians and graphs have already been made previously, in Section 7.1. This will help us to establish notations as well as define the scope of computer graphics applications that we are considering. We use bold letters to denote vectors, e.g., $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{R}^n$, capital letters to denote matrices, and calligraphic letters for sets. We use n as the number of variables and denote the set of indices by $\mathcal{I} = \{1..n\}$.

7.2.1 Laplacian matrices

Laplacian matrices result from minimizing objective functions of the form

$$F(\mathbf{x}) = \sum_{i \in \mathcal{I}} \left[u_i (x_i - y_i)^2 + \sum_{j \in \mathcal{N}_i} w_{ij} (x_i - x_j - z_{ij})^2 \right]. \tag{7.1}$$

The first sum contains *data terms* that measure the proximity of $\mathbf{x}$ to a given input data vector $\mathbf{y}$. The second sum contains *smoothness terms* that measure the derivatives (pairwise differences) between every variable x_i and its neighbors x_j, $j \in \mathcal{N}_i$, with respect to (potentially zero) input derivatives z_{ij}. As we describe later, in the applications we are interested in, each set of neighbors $\mathcal{N}_i$ consists of a small number of variables that are geometrically close to x_i. Typical choices in the case of two-dimensional regular arrays of pixels are the four- or eight-nearest pixels. The weights u_i and w_{ij} define the cost for deviating from the data and smoothness objectives respectively and are non-negative. The problem becomes spatially homogeneous when u_i and w_{ij} are constant and is considered spatially inhomogeneous otherwise.

The objective $F(\mathbf{x})$ can be expressed in matrix-vector form as

$$F(\mathbf{x}) = (\mathbf{x} - \mathbf{y})^\top U (\mathbf{x} - \mathbf{y}) + (D\mathbf{x} - \mathbf{z})^\top W (D\mathbf{x} - \mathbf{z}), \tag{7.2}$$

where the matrix D is the discrete derivative operator whose rows correspond to pairs of neighboring variables i and $j \in \mathcal{N}_i$. The weights matrix W is diagonal and contains w_{ij} on the row that corresponds to the interaction between the i-th and j-th elements. The data weights matrix U is an n-by-n diagonal matrix with $U_{ii} = u_i$.

The minimum of this discrete quadratic form is obtained by setting $dF/d\mathbf{x} = \mathbf{0}$, which amounts to solving the following linear system

$$L\mathbf{x} = U\mathbf{y} + WD^\top \mathbf{z}, \tag{7.3}$$

where L is the discrete Laplacian matrix given by

$$L = U + D^\top W D. \tag{7.4}$$

The matrix L is, by construction, symmetric and positive semi-definite since $x^\top Lx = x^\top Ux + x^\top D^\top WDx = x^\top Ux + (Dx)^\top W(Dx)$ and both U and W are diagonal with non-negative values. The off-diagonal elements of L are all non-positive and given by $L_{ij} = -w_{ij}$. The diagonal entries are given by $L_{ii} = u_i + \sum_{j \in \mathcal{N}_i} w_{ij}$ and are hence non-negative. These Laplacian matrices are associated with a graph whose vertices are the variables i and whose edges are weighted by w_{ij}. The data terms u_i can be considered as weights of edges connecting the vertices with a set of auxiliary variables.

The solvers we describe in this chapter applies for the general family of Laplacian matrices described by Eq. 7.1. This family includes a large portion of Laplacian matrices used in computer graphics and computational photography applications. For example. the image colorization of Levin et al. [97] uses $u_i = 1$ at pixels i containing user input colors y_i and $u_i = 0$ elsewhere. The weights w_{ij} depend inversely on the difference between the i and j pixels gray-level values. The reference gradient field is set to zero, $z_{ij} = 0$. The weighted least squares (WLS) edge-preserving smoothing of Farbman et al. [51] uses a similar definition for the smoothness weights but sets *all* data terms $u_i = 1$ and provides the input image as the data $\mathbf{y}$. The dynamic range compression algorithm of Fattal et al. [53] and Poisson blending of Perez et al. [127] use $u_i = 0$ and set z_{ij} to the manipulated gradient field being integrated. This problem is homogeneous, i.e., $w_{ij} = 1$.

In 3D geometry processing applications, the Laplacian occurs in such problems as mesh segmentation [107] and geodesic distance computation [38]. The Laplacian can either be homogeneous or based on the local curvature or geometry (inter-vertex distances and angles) in the 3D mesh. Some of these Laplacian variants, e.g., the co-tangent Laplacian [38], may result in negative weights w_{ij}.

Given that most AMG methods, as well as our methods, are applicable to Laplacian matrices with strictly negative off-diagonal (*M-matrices*), we restrict the formulation of 3D processing problems to this class of matrices. In many cases, defining such matrices while preserving the same qualitative nature of the operator is possible. For example, Crane et al. [38] describe their algorithm for arbitrary discretization of the Lapacian operator and then provide two alternatives. In Section 7.4, we show a particular discretization that results in an M-matrix and at the same time achieves the desired result.

In computer vision applications, problems of the form Eq. 7.1 arise from Gauss-Markov MRF models. The data terms are known as unary potentials and the smoothness terms as binary (or pairwise) potentials. Example application of such models is optical flow regularization [7].

Energy Function. The Laplacian matrix L assigns an *energy* value to each vector $\mathbf{x}$, defined by the Rayleigh quotient

$$E_L(\mathbf{x}) = (\mathbf{x}^\top L\mathbf{x})/(\mathbf{x}^\top \mathbf{x}). \tag{7.5}$$

The energy values are always non-negative due to the positive semi-definiteness of L. The eigenvalues of L are the energy values assigned to their corresponding eigenvectors, since if $L\mathbf{x} = \lambda\mathbf{x}$, $(\mathbf{x}^\top L\mathbf{x})/(\mathbf{x}^\top \mathbf{x}) = (\mathbf{x}^\top \lambda\mathbf{x})/(\mathbf{x}^\top \mathbf{x}) = \lambda$.

The input data values y_i and derivatives z_{ij} contribute only to the right-hand side of Eq. 7.3 and

hence do not affect the properties of L. The energy function E_L is therefore an intrinsic function of the Laplacian matrix and it is closely related to the solvability of Eq. 7.3.

7.2.2 Hierarchical preconditioning

The definition of condition number and preconditioners was given in Eq. 2.17 and Section 2.8. Hierarchical preconditioners are constructed by formulating a smaller version of the original problem and using its solution as the approximate inverse for the original problem [160, 138, 153]. Geometric multigrid techniques use a regular set of decimation rules (e.g., full octave or half-octave decimation) and standard interpolation operators as their basis, and are particularly well suited for homogeneous problems [160]; an example is shown in Figure 2.15. Algebraic multigrid techniques use both adaptive coarsening strategies and adaptive interpolation weights, which make them better suited for inhomogeneous problems. Unfortunately, the elimination of fine-level variables results in an increase the in matrix bandwidth, or, alternatively, a sub-exponential decrease in the matrix size (Figure 7.4). In order to reduce the bandwith growth (fill-in) in coarser problems, AMG techniques drop small off-diagonal terms.

Adaptive basis functions, introduced in [153] perform the sparsification (element elimination) *before* creating the coarser-level (smaller) problem, which allows them to *compensate* for these eliminations by increasing nearby connections. Our first preconditioner (which we call ABF) [89] is an extension of the adaptive basis functions solver developed in [153].

In Section 7.3.2, we derive an alternative compensation strategy that is based on an analysis of the spaces spanned by the fine and coarse variables and hence produces a better preconditioner (slower growth in condition number). This leads to our second preconditioner, which we call Hierarchical Sparsify and Compensate (HSC).

For both ABF and HSC, once the Laplacian matrix has been sparsified and compensated, we divide the variables into coarse $\mathcal{C}$ and fine $\mathcal{F}$ sets. ABF is non-adaptive in it's selection, and uses a fixed half-octave scheme to select the coarse and fine variable sets. On the other hand, HSC is adaptive in its selection procedure. The coarse variables are chosen to encode the low-frequency modes in the solutions, i.e., the modes that are not well solved by local smoothing or relaxation, and the fine level variables have no remaining connections between each other. The exact method for selecting these variables, which in HSC is interleaved with the sparsification step, is described in Section Section 3.4.

Once the selection of coarse and fine variables has been done, we rearrange the indices in $\mathcal{I}$ such that the $\mathcal{C}$ come first followed by the $\mathcal{F}$. Under this permutation, the matrix L becomes

$$L = \begin{bmatrix} L_{\mathcal{CC}} & L_{\mathcal{CF}} \\ L_{\mathcal{FC}} & L_{\mathcal{FF}} \end{bmatrix}, \tag{7.6}$$

where $L_{\mathcal{CC}}$ contains only the connections between the coarse variables, $L_{\mathcal{FC}} = L_{\mathcal{CF}}^{\top}$ contains the connections between coarse and fine variables, and $L_{\mathcal{FF}}$ among the fine variables. Since the fine

variables are uncoupled, $L_{\mathcal{FF}}$ is diagonal. The elimination of the fine variables is obtained by computing the Schur complement using the transformation matrix

$$P = \begin{bmatrix} I_{\mathcal{CC}} & 0 \\ -L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}} & I_{\mathcal{FF}} \end{bmatrix}, \tag{7.7}$$

which is applied to L on both sides,

$$P^\top L P = \begin{bmatrix} L_{\mathcal{CC}} - L_{\mathcal{CF}} L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}} & 0 \\ 0^\top & L_{\mathcal{FF}} \end{bmatrix}. \tag{7.8}$$

Note that the matrix $L_{\mathcal{FF}}^{-1}$ is an inverse of a diagonal matrix and is trivial to compute, and that $S = L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}}$ is the *interpolation* (or *prolongation*) matrix used in hierarchically preconditioned conjugate gradient [89, Algorithm 1]. Since L is a sparse matrix, so is P, and this elimination step is computed in linear time.

The resulting two-block matrix in Eq. 7.8 describes two systems that are solved independently. The fine system, $L_{\mathcal{FF}}$, is diagonal and solved exactly. In Appendix A, we show that the coarser system, $L_{\mathcal{CC}} - L_{\mathcal{CF}} L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}}$, is a Laplacian matrix like the original matrix L. In the multigrid literature, the operation $P^\top L P$ is also known as the *Galerkin* step. This allows us to apply the same procedure again over this system, compute a Schur matrix, and repeat this process recursively over the resulting coarse system. At each level, we do not operate on the fine variables eliminated in the previous levels and hence obtain a sequence of of n-by-n transfer matrices $P^1, P^2, ..$ that contain the current level's prolongation matrix as the top-left block and are identity over the remaining coordinates (corresponding to the fine variables of all previous levels). The recursive elimination process is terminated once the number of coarse variables falls below some threshold (e.g., 1024, although changing this to 512 or 2048 does not affect our performance), since the direct solution of such small systems using Cholesky decomposition takes a negligible amount of time compared to finer-level operations.

7.3 Sparsification and coloring

In the previous section, we presented a general framework encompasing previously developed hierarchical preconditioning algorithms for the solution of sparse Poisson equations. How do our new approaches, ABF and HSC, differ from these other techniques?

The adaptive hierarchical basis function algorithm of [153] relies on fixed half-octave coarsening. At each level, half the variables are chosen as coarse and the other half as fine, following a red-black scheme. In [153], the next step is to sparsity "diagonal" connections between pairs of coarse and pairs of fine variables. These diagonal connections correspond to nodes which are furthest neighbors in a clique. After a connection is dropped, compensation is performed on adjacent edges (Section 7.3.2. Figure 7.5 shows the sparsification process. The process in [153] leads to a 5-band sparsified Laplacian matrix if one starts with a 9-band Laplacian matrix

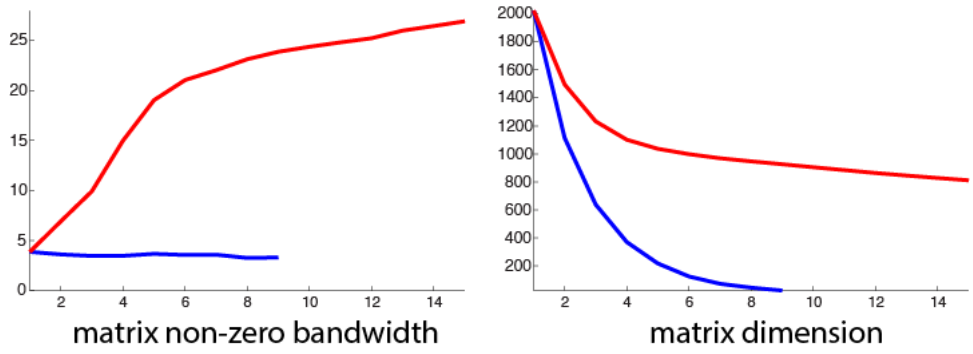


Figure 7.4: Effect of Sparsification. The left graph shows how our sparsification step maintains an average bandwidth below 4 (blue curve) in all the hierarchy levels. Without the sparsification the (red curve) bandwidth grows until it reaches the matrix dimension. The right graph shows the scheme's ability to achieve an exponential reduction in the number of variables (blue curve), where without sparsification, fewer variables can be marked as fine and get eliminated (red curve). We used an inhomogeneous two-dimensional regular five-point Laplacian matrix in this example. Note that the sparsified hierarchy has fewer levels, as it more quickly reaches target coarse level size.

(for 2D nearest-neighbor problems). ABF is a simple extension of this sparsification scheme. It is not necessary to drop connections between coarse variables, since they are re-introduced at the next level anyway. This leads to a lower approximation error between the unsparsified and sparsified Laplacians at the cost of slightly increased bandwidth (on average, 7 instead of 5, for a 9-point Laplacian). As seen in Section 7.4, ABF performs extremely well on homogenous and close-to-homogenous problems.

The ABF coarsening scheme is fixed in nature. There are two drawbacks to a fixed coarsening/sparsification scheme: it restricts the use of the preconditioner to 2D grids. Applying a red-black coarsening to irregular grids (such as 3D surface meshes) leads to a very slow rate of coarsening, with a resulting increase in memory and computational costs. The second drawback is that for inhomogenous problems, a fixed sparsification scheme can lead to very poor approximations. We give an example of this in Section 7.3.2.

To overcome these problems, we introduce the HSC algorithm. Rather than using a fixed sparsification and coarsening scheme as in ABF, we adaptively select which edges to sparsify and which nodes to select as coarse and fine variables. (This process is often called *coloring* [160].) The extensions in HSC allow the ABF algorithm, which already performs well on a wide variety of computational photography applications, to now also perform well on more inhomogeneous problems as well as unstructured meshes. Compared to AMG and CMG, our technique does a better job of creating a hierarchy of smaller approximate problems (because of our use of adaptive interpolants and compensations steps), and hence has better convergence and run-time properties.

The sparsification and coloring algorithm in HSC tries to simultaneously satisfy two somewhat conflicting goals. The first is to produce a large number of disconnected fine variables, since the smaller we can make the coarse system, the easier it is to solve or invert. The second is to only

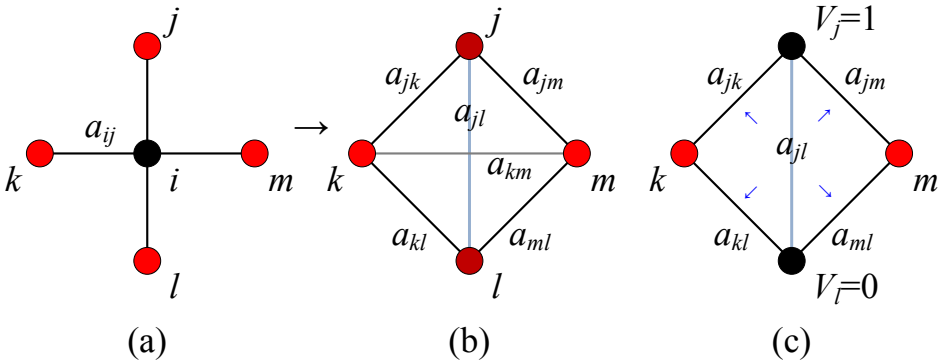


Figure 7.5: (a) A fixed and, and the sparsification scheme of (b) [153] and (c) ABF. In both these algorithms, fixed half-octave coarsening is performed. The black nodes correspond to fine nodes, which are eliminated at each level. Red nodes correspond to coarse variables. In [153], diagonal (geometrically longest) connections are dropped between pairs of coarse and fine nodes. Therefore the $j-l$ and $k-m$ connections are dropped. In ABF only the connections between fine nodes $k-m$ is dropped and the weight of that edge a_{km} is spread to the 4 edges on the quadrilateral $a_{jk}, a_{kl}, a_{ml}, a_{jm}$. Coarse-coarse connections a_{jl} remain unchanged.

sparsify connections that are already quite weak compared to the neighboring compensation paths. We solve this tension by developing a greedy algorithm that visits fine and unmarked variables and searches for connections to other variables that can be sparsified, which then enables these variables to be also colored as fine. The exact algorithm is described in Section Section 7.3.3. Before we get there, however, we first describe how to find good connections to sparsify and how to compensate for such operations.

7.3.1 Matrix sparsification for the HSC preconditioner

We now explain how we avoid the growth in matrix bandwidth and increase the number of fine variables by carefully dropping off-diagonal elements at every level of the hierarchy, before executing the elimination. Unfortunately, eliminating a large number of off-diagonal elements introduces approximation errors, meaning that it cannot be used for computing the exact inverse of L. As we explained earlier, we use the approximated inverse for matrix preconditioning and accelerating different iterative solvers. In order to achieve low condition numbers, we need to keep the sparsified matrix 'close' to the original Laplacian. In this section we explain how we carefully chose the off-diagonals to drop from L to produce $\tilde{L}$ based on its effect on the condition number $\kappa(L\tilde{L}^{-1})$. Let us first define more precisely what we mean by dropping connections in the matrix.

There are two properties that we need to preserve when sparsifying a matrix. We need to preserve its nature as a Laplacian matrix so that the following levels of the hierarchy are constructed in the same manner. To preserve its symmetry when setting L_{ij} to zero we also set L_{ji} to zero. However this alone does not preserve the type of the problem, for example, in the case of data-less problems, where $u_i = 0$, the row and column sums of L are zero. Setting off-diagonals elements

to zero results in a matrix with positive row and column sums which corresponds to a problem *with* data terms.

We avoid this problem by replacing L_{ii} by $L_{ii}+L_{ij}$ and L_{jj} by $L_{jj}+L_{ij}$ when we drop the i,j element. From now on, we interchangeably say that we drop L_{ij} or the weight w_{ij} and in both cases refer to the same procedure where the diagonal elements are modified.

In order to decide which off-diagonal elements are dropped, we refer to the condition described in Section Section 2.8 which guarantees $\kappa(L\tilde{L}^{-1}) \leq b/a$, given lower- and upper energy bounds b and a obeying Eq. 2.18. If the energies assigned by the sparsified matrix $\tilde{L}$ are close enough to those assigned by the original matrix L, the sparsified matrix will provide a good preconditioning. As we discussed in Section 7.2.1, the energy $E(\mathbf{x})$ does not depend on the norm of the vector $\|\mathbf{x}\|$ and hence we can restrict the discussion to unit vectors $\mathbf{x}$ in which case the energy function coincides with the objective function in Eq. 7.2. Dropping a weight w_{ij} means that the energy function ceases to account for changes between x_i and x_j. However, in case there is a third coordinate k such that both $w_{ik} > 0$ and $w_{jk} > 0$, any difference between x_i and x_j must be accompanied by a difference between x_i and x_k or between x_j and x_k. Thus, for example if $w_{ij} \ll w_{ik}, w_{jk}$, the penalty term $w_{ij}(x_i - x_j)^2$ is dominated and negligible compared to either $w_{ik}(x_i - x_k)^2$ or $w_{jk}(x_i - x_k)^2$. In Appendix B we prove that in such cases of triangular connectivity, when dropping the weakest weight, we get $a = 1$ and $b \leq 3$ and hence $\kappa(L\tilde{L}^{-1}) \leq 3$.

In view of these observations and the fact that we need to remove many connections to ensure rapid coarsening, we apply the following sparsification procedure. At each level of the hierarchy, we search for triplets of variables that form a triangle and remove the weakest edge in each triangle. This procedure is applied by scanning the matrix elements until no more triangles are found.

Figure 7.4 shows the non-zero matrix bandwidth and the matrix dimension at each hierarchy level with and without applying sparsification. In this experiment our strategy selects and eliminates between $40\% - 50\%$ of the variables at each level and avoids a growth in bandwidth. These actions are highly local, at the scale of three coupled variables, and hence run in time linear in the number of variables and connections in the matrix. As we explain next, we further improve the energy preservation by adjusting the two remaining weights of every sparsified triangle. Note that the ABF solver always eliminates 50% of the variables at each level.

7.3.2 Compensation for ABF and HSC

When computing a single level of the hierarchy, low condition numbers are achieved even though multiple overlapping triangle are processed and the theoretical bound becomes $\kappa \leq 3^r$ where r is a bound on the number of times the same edge participates in a triangle. However, when applying this strategy at every hierarchy level, the sparsification errors of the different levels accumulate. For example, there could be a vector $\mathbf{x}$ whose energy drops by a factor of 3 due to the sparsification that takes place at each level. Thus, after the expected number of $m = \log(n)$ hierarchy levels, its energy drops by a factor of $3^{\log(n)} = O(n)$, meaning that the condition number

$\kappa(LQ^{-1}) = O(n)$ where Q represents the approximate operator obtained by the entire hierarchy. As we discussed earlier, this dependency of κ on n is observed in non-preconditioned Laplacian matrices.

The key observation that allows us to cope with this shortcoming is the fact that at each hierarchy level, about half the variables are eliminated and these variables define a linear subspace that *does not* experience the sparsification taking place in the following levels. This means that there is a hierarchy of subspaces that undergo a different number of sparsification steps. Similarly to Szeliski [153], we alter the remaining matrix elements in order to compensate for the loss of those elements dropped during sparsification. In the case of the ABF preconditioner, we follow the same compensation as [153]. Referring to Figure 7.5, the rule for distributing the weight of a matrix entry L_{jl} to it's neighbors is:

$$L_{jk} \leftarrow L_{jk} + 2L_{jk}L_{jl}/S \tag{7.9}$$

where $S = L_{jk} + L_{kl} + L_{jm} + L_{lm}$ is the sum of the edge weights adjacent to the entry L_{jl}. L_{jk} and L_{kj} are set to 0. The diagonal entries L_{jj}, L_{kk}, L_{ll} and L_{mm} are accordingly adjusted.

However, ABF's sparsification and compensation scheme can lead to poor condition numbers for highly inhomogenous problems. Consider the case when the value of $|a_{jl}|$ (corresponding to L_{jl} in the matrix) is much larger than all of the adjacent edge values. In this case, sparsifying that edge and the resulting compensation of the neighboring edges leads to a very poor approximation by the sparsified matrix.

The HSC compensation formula overcomes this problem by choosing the weakest edge in each triangle. Moreover, the compensation is based on an analysis that characterizes vectors based on the number of sparsification steps they undergo. Thus, we both extend Szeliski's approach to an adaptive coarsening schemes as well as improve it by establishing the sense at which $\tilde{L}$ should best approximates L.

In Appendix C, we show that the linear subspace spanned by the columns of $P^1P^2..P^l$, which correspond to the fine variables eliminated in the l-th hierarchy level, is affected by the sparsification steps in the first l hierarchy levels. We further show that this subspace is characterized by having low energy values, since the elimination steps that define it correspond to the minimization of the energy over the eliminated variables. In view of this relation between the number of sparsification steps and low-energy vectors, whenever a triangle is sparsified, we compensate by adjusting its two remaining weights such that the triangle's energy contribution remains unchanged over low-energy vectors.

Assuming that w_{ij} is the weakest weight of a triangle which is set to zero, our compensation procedure consists of finding the sparsified matrix weights $\tilde{w}_{ik}$ and $\tilde{w}_{jk}$ of $\tilde{L}$ that satisfy

$$\begin{aligned} E_L(\mathbf{u}) &= w_{ij}(u_i - u_j)^2 + w_{ik}(u_i - u_k)^2 + w_{jk}(u_j - u_k)^2 + E' \\ &= \tilde{w}_{ik}(u_i - u_k)^2 + \tilde{w}_{jk}(u_j - u_k)^2 + E' = E_{\tilde{L}}(\mathbf{u}), \end{aligned} \tag{7.10}$$

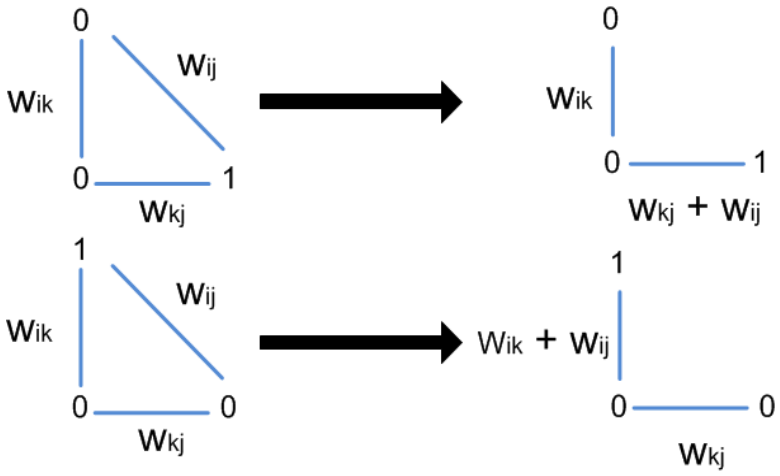


Figure 7.6: Sparsification and Compensation in a Triangle. Left shows how the two vectors **u** and **v** are aligned with respect to the triangle and at the right we see the sparsified and compensated triangle weights which preserve the energy of **u** and **v**.

where **u** is one of the two low-energy vectors. (The same equation is defined over the second vector **v**.) The scalar E' accounts for the total cost of the rest of the energy terms in Eq. 7.2 that are unrelated to w_{ij}, w_{ik}, and w_{jk} and can therefore be omitted from Eq. 7.10. Once this is done, except for the i, j and k coordinates, the values of **u** and **v** cease to effect the compensation and hence we set them to zero. Finally, since Eq. 7.10 and its counterpart for **v** consist only of differences between the variables, they are invariant to the addition of any constant to u_i, u_j, u_k and v_i, v_j, v_k.

These invariants leave us with only a single degree of freedom in **u** and **v**, which we need to determine in order to model the shape of low-energy vectors at the coordinates i, j and k. We use the following rationale to explicitly obtain these two model low-energy vectors. Consider the scenario where w_{ik} is much greater than w_{jk} (and $w_{ik} > w_{ij}$). In this case, differences $(u_i - u_k)^2$ lead to a higher energy than $(u_i - u_j)^2$ and hence the former is expected to be much lower in the case of low-energy **u**. Low-energy vectors will therefore attain similar values at i and k and acquire a different values at u_j. Given the invariants we discussed, this situation can be modeled by choosing $[u_i, u_j, u_k] = [0, 1, 0]$. The reverse scenario where the j-th and k-th variables are strongly coupled and both are weakly connected to the i-th, can be modeled by the second vector $[v_i, v_j, v_k] = [1, 0, 0]$. These scenarios are often encountered in the two-dimensional computer graphics applications that we are interested in, where the weights are determined by pixel differences of some reference image. The two constellations result from strong edges in the reference image that pass through the i, j, k triangle and separate its pixels into two sets of distinct colors.

In this choice of **u** and **v**, we dismiss a third scenario, where $w_{ik} \approx w_{jk}$ and both of them are much higher than w_{ij}. This scenario is less common in the applications that we are interested in since there is no assignment of pixel values that will lead to such weights. Furthermore, in such situations, the cost of $w_{ij}(u_i - u_j)^2$ will be penalized by either $w_{ik}(u_i - u_k)^2$ or $w_{jk}(u_j - u_k)^2$

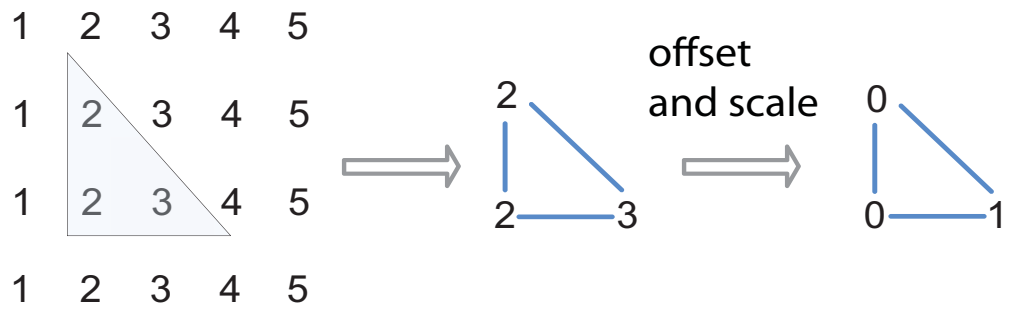


Figure 7.7: Modeling Fourier Modes. Illustration shows how our choice of $\mathbf{u}$ models locally the shape of horizonal low-frequency Fourier modes. The transposed diagram applies for $\mathbf{v}$.

since both w_{ik} and w_{jk} are assumed large. Lastly, the low-energy vectors are expected be close to unform at such triangle and, as we shall see below, we handle these situations properly with our current choice of $\mathbf{u}$ and $\mathbf{v}$.

Solving Eq. 7.10 for $\mathbf{u}$ and its analog for $\mathbf{v}$ boils down to a simple and efficient update formulas for $\tilde{w}_{ik}$ and $\tilde{w}_{jk}$. The energy of $[u_i, u_j, u_k] = [0, 1, 0]$ before sparsification is given by $w_{ij} + w_{jk}$. After dropping w_{ij}, the energy becomes $\tilde{w}_{ik}$ and hence to preserve this energy, we need

$$\tilde{w}_{ik} = w_{ik} + w_{ij}, \tag{7.11}$$

and similarly, for $[v_i, v_j, v_k] = [1, 0, 0]$, we get

$$\tilde{w}_{jk} = w_{jk} + w_{ij}. \tag{7.12}$$

This action is illustrated in Figure 7.6.

The LAMG method in [108] also corrects the matrices with respect to low-energy vectors. However, these vectors are computed globally by applying an iterative method to reduce the energy of random vectors. In order to achieve very low-energy vectors as our analysis suggests, many such iterations are needed for larger systems. Our method avoids this additional cost through its simple local operation.

Data connections. The derivation of both the sparsification and compensation accounts only for the smoothness terms in Eq. 7.2. Non-zero data terms, u_i in Eq. 7.1 can be viewed as weighting differences with auxiliary variables of fixed values, namely y_i in Eq. 7.1. Since each of these auxiliary variables is connected to only a *single* variable x_i, it never participates in a triangle. Hence, these connections need not be involved in any sparsification and compensation steps. In practice, this means that the sparsification and compensation steps are applied on L after removing its excess diagonal values so that its rows and columns become zero-sum. Once these steps have been applied, the excess diagonal values are added back to $\tilde{L}$.

Homogeneous systems. In the case of homogeneous Laplacian matrices, such as the spatially invariant Poisson used in [53, 127, 151], the eigenvectors of L are the Fourier modes [160].

In the case of rectangular domains the eigenvalues are given by $\lambda_{ij} = (4 - 2\cos(2\pi i/n_x) - 2\cos(2\pi j/n_y))$ and their corresponding eigenvectors are $\mathbf{v}^{i,j}_{x,y} = \cos(ix\pi/n_x)\cos(iy\pi/n_y)$ where $0 = i < n_x$ and $0 = j < n_y$ are the wavelengths and $0 \leq x < n_x$ and $0 \leq y < n_y$ are the spatial coordinates in which case $n = n_x n_y$. The lowest eigenvalue $\lambda_{00} = 0$ and corresponds to the null space of L which are the constant vectors $\mathbf{v}^{00}_{xy} = 1$. The next eigenvalue has a multiplicity of two $\lambda_{10} = \lambda_{01} = (2 - 2\cos(2\pi/l)) = O(l^2) = O(n)$ and hence the $\kappa = O(n)$ that we mentioned earlier for these matrices (the maximal eigenvalue is 4 and does not depend on n). The corresponding modes are $\cos(x\pi/l)$ and $\cos(y\pi/l)$, which are two very smooth functions in space.

The two lowest energy (non-constant) eigenvectors are low-frequency horizontal and vertical sinusoids. From the perspective of three variables forming a triangle in the grid, these functions appear as two linear ramps. As shown in Figure 7.7, our compensation function preserves the energy in these Fourier modes.

If we make sure that our selection of coarse and fine variables as we drop matrix elements maintains spatial homogeneity, our construction boils down to a geometrical multigrid method for homogenous problems.

The interpolation matrix we obtain corresponds to prolongation matrices that interpolate first order polynomials exactly. It is well-known in the multigrid literature [160] that such schemes, along with smoothing iterations (which we explain below), lead to a property known as *h-independence* which means that the condition number is independent of n and hence the our scheme, along with the GMG, achieve an optimal complexity of $O(n)$ running time. This behavior is validated by our experiments, which we report in Section 7.4. The CMG algorithm [83] on the other hand, uses constant interpolation matrices as it is an agglomerative method. These matrices interpolate only constants (zero order). As a result, the performance of CMG on homogenous problem is quite poor, as seen in Section 7.4.

Our scheme is also important in the case of inhomogeneous problems, which often contain large homogeneous regions, e.g., due to nearly constant regions in the reference image. In the next section we explain how we make sure our scheme maintains spatial homogeneity in such regions.

Table 7.1 shows the effect that careful compensation has on condition numbers for homogenous Poisson matrices of increasing size. We compare our method with a sparsification-only version, where we drop the weakest edges in each triangle but do not perform any compensation. our method with compensation has the property of h-independence, while without compensation, the condition numbers increase with problem size.

In Figure 7.8, we show how the HSC algorithm works on a small sample problem, choosing weak edges to drop and compensate. As might be intuitively expected, the low eigenvectors of this problem lie on manifolds along the arms of the spiral. Therefore dropping connections *across* the spiral arms and strengthening them *along* the spiral arms is the best strategy to preserve low-frequency energies.

A natural question that arises in the use of more than 2 vectors during compensation. Instead of leading to a simple compensation equation as in Eq. 7.11 and Eq. 7.12, compensating with

Problem Size	Original CN	CN with compensation	CN with no compensation
1024	423	1.2	1.6
4096	1676	1.2	2.4
16384	6674	1.3	3.7
65536	26629	1.4	5.6
262144	106380	1.5	8.6
1048576	452500	1.5	13

Table 7.1: Comparison of condition numbers with and without compensation. First column gives the dimension of the homogenous Laplacian. Second column gives the unpreconditioned condition number. Third and fourth columns give the preconditioned condition number with and without compensation, respectively.

other vectors leads in general to a 2×2 least squares system. We have experimented with a number of other vectors for the compensation. All resulted in inferior results to the scheme in Eq. 7.11-Eq. 7.12. We believe that the reason for this is that using other vectors reduces the accuracy of compensating for the linear ramp vectors, which tend to be the lowest frequency vectors in Laplacians arising from natural images. This is certainly the case for homogenous problems. Nevertheless, it may be possible to improve the accuracy of inhomogenous problems with other vectors. We leave this for future work.

7.3.3 Coloring algorithm

With our sparsification criteria and compensation steps in place, we need to decide how to color the original variables in the Laplacian as coarse $\mathcal{C}$ or fine $\mathcal{F}$ and how to interleave the process of sparsification/compensation into this procedure. For the ABF algorithm, as mentioned previously, simple red-black coloring [153] is used.

The HSC algorithm we have developed is based on the observation that we want to produce a large number of fine variables (to create smaller problems), but that it is best to cut weak edges to limit the growth in condition number.

Our algorithm (Algorithm 5) visits nodes in their lexicographic order (how they were given in the original problems), ignoring nodes already marked as coarse, since we do not need to cut connections between these nodes and their neighbors. For each unmarked or fine node, we search its neighbors and neighbors' neighbors, looking for triangles where an edge emanating from the current node can be cut because it is no larger than the other edges in the triangle. When we find such an edge, we eliminate it, compensate the other two edges in the triangle, and mark both endpoints of the eliminated edge as fine. At the end of this procedure, we label any unmarked variables as fine or coarse (depending on their neighbors), fix fine-fine connections by marking one endpoint as coarse, and check for any coarse variables surrounded by coarse variables, which can be set to fine.

An alternative compensation strategy may be to compensate all triangles that are common to an edge, as is done in [153]. This strategy is advantageous when all triangles are similar in weight.

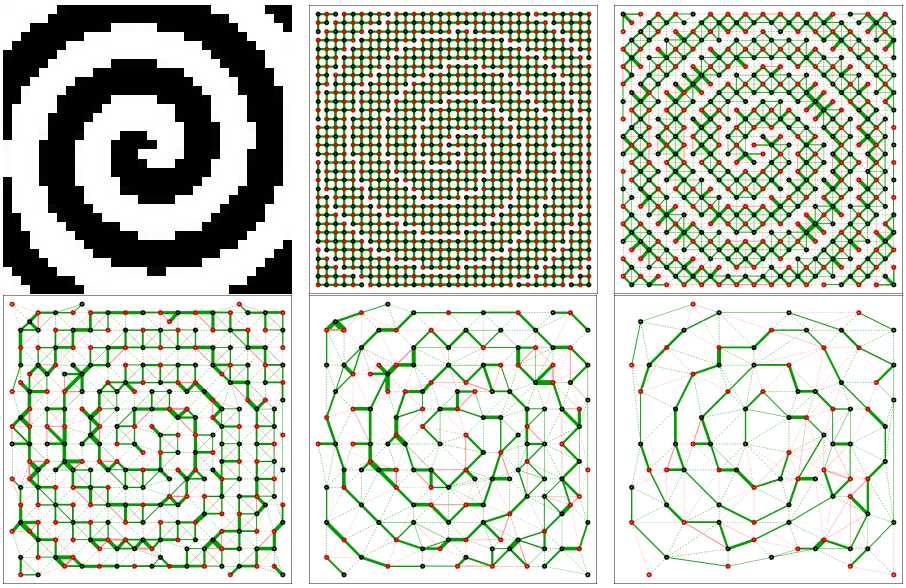


Figure 7.8: Scheme Progression. We show the connectivity graphs at each level for a highly discontinuous EPS problem, starting from a four-point Laplacian matrix defined by the top-left image. Red circles show fine (eliminated) variables and black show the coarse one. Green lines show the connections kept between variables and red show the ones removed during sparsification. The thickness of the lines indicates the corresponding strength of the weights.

However, for highly inhomogenous problems, it could lead to overcompensation of weak edges by stronger ones, leading to a deterioration of performance. In future work, we hope to study such compensation schemes in greater depth.

In homogeneous regions where we have additional geometric information, i.e., the (x, y) locations of variables, we modify the above edge selection and coloring procedure to produce a regular red/black coloring. To determine if a variable is in a homogenous region, we take the difference between the strongest and weakest connections of the variable and divide this difference by the strongest connection. We then find the mean of these ratios over all variables. Any variable whose ratio is below or equal to the mean ratio is marked as geometric. When sparsifying triangles, if all three vertices are marked as geometric, we cut the longest edge in the triangle based on its geometric distance. We also mark vertices according to a global red/black coarsening scheme, so that fine variables are disconnected from each other. These steps ensure that our HSC scheme reduces to geometric multigrid for smoothly varying 2D problems. In this approach, homogeneous regions in inhomogeneous problems are also processed using geometric coarsening. For 3D problems, we do not perform any geometric coarsening since we do not have geometric information.

Once the HSC preconditioner has been constructed, we use it in combination with conjugate gradients to precondition the solution of $L\mathbf{x} = \mathbf{b}$ using the multilevel V-cycle described in Algorithm 6. Algorithm 6 is a generalized description of a unified multilevel multigrid preconditioner, incorporating diagonal preconditioning, pre-smoothing and post-smoothing steps.

Algorithm 5 Sparsify and color (HSC)

input: Laplacian matrix L and optionally the coordinates of the mesh vertices;
output: Fine $\mathcal{F}$ and coarse $\mathcal{C}$ variable indices and the sparsified matrix $\tilde{L}$,
which, according to Eq. 7.7, determine the prolongation matrix P.

1. Remove the excess diagonals from L and store them in E
2. Set all vertices as unmarked except for the first one, which we mark as fine
3. Flag variables as geometric or non-geometric
4. Cycle through vertices i in the matrix
   if i is marked as coarse, skip to the next vertex
   for each triangle in which vertex i participates
   if all three vertices are flagged as geometric
   Cut longest edge in the triangle and compensate
   Mark the vertices in this triangle as coarse or fine
   according to the global red/black settings
   else
   Cut the weakest edge in the triangle and compensate
   If vertices on weakest edge are unmarked, mark them as fine
   endif
   Set unmarked neighbors of i as coarse
   endfor
5. Set unmarked vertices that have fine neighbors to coarse, else fine
6. For any fine-fine connections, set one of the endpoints to coarse
7. Set any coarse variables connected only to coarse variables as fine.
8. Add the excess diagonal in E back to L to produce $\tilde{L}$

At the finest level, the current residual is computed (line 1 in Algorithm 6). This residual is then transferred to the coarse level using the transpose of the interpolation operator P^0 at this level. This restriction of the residual is recursively continued until the coarsest level, at which point, the residual is exactly inverted. The resulting correction is then propagated up the hierarchy using the interpolation operators P^l at each level. The coarse-level correction gives an (approximate) solution to the coarse variables at each level. The correction to the fine level variables is computed using a diagonal preconditioning of only the residuals at the variables marked as fine at each level. This is represented by the *DiagPrecond* function in Algorithm 6. Together, these two steps correspond to the approximate inversion of the matrix defined in Eq. 7.8. The accuracy of the preconditioning is further improved by the use of a single iteration of Gauss-Seidel post-smoothing [138] at each level. This is represented by the *Smooth* function call. Pre-smoothing may also be optionally added, although in our numerical experiments we do not find it to improve performance and therefore we do not use pre-smoothing. An interesting direction for future work is to better characterize the relationships between the sub-spaces of the solution which are affected by diagonal preconditioning and by smoothing.

In the experiments we report in this paper, we use a single step of post-smoothing, $\nu^{post} = 1$, and no pre-smoothing, $\nu^{pre} = 0$ and a V-cycle ($\gamma = 1$) since this resulted, on average, in the fastest running algorithms. For our smoothing algorithm, we use lexicographic-order Gauss-Seidel. These settings were used for both the HSC and ABF preconditioners.

Algorithm 6 Unified multigrid/multilevel algorithm

$[e_l] = MGCYC(l, r_l, \{L_1 \dots L_m\}, \{P_1 \dots P_m\}, m, \omega, \nu^{pre}, \nu^{post}, \gamma, d)$
INPUT: Current level l, residual r_l, per-level Laplacians L_l,
per-level interpolation matrices $\hat{P}_l$, number of levels m, damping factor ω,
pre- and post-smoothing iterations ν^{pre} and ν^{post}, number of cycles γ,
flag for optional diagonal preconditioning d
OUTPUT: Correction at level l e_l

1. $e_l^{pre} = Smooth\ (0, A_l, r_l, \omega, \nu^{pre})$ // Pre-smoothing correction
2. $\bar{r}_l \quad = r_l - L_l e_l^{pre}$ // Update the residual
3. $\bar{r}_{l+1} = P_l^T \bar{r}_l$ // Restrict residual to coarse level
4. **if** $l = m - 1$
5. $\quad e_{l+1} = L_m^{-1} \bar{r}_{l+1}$
6. **else**
7. $\quad e_{l+1} = 0$
8. $\quad$ **for** $j = 1, \dots \gamma$ //$\gamma = 1$ is V-cycle; $\gamma = 2$ is W-cycle
9. $\quad\quad \hat{r}_{l+1} = \bar{r}_{l+1} - L_{l+1} \hat{e}_{l+1}$ // Update the residual
10. $\quad\quad \tilde{e}_{l+1} = MGCYC(l+1, \hat{r}_{l+1}, \{L_1 \dots L_m\}, \{P_1 \dots P_m\}, m, \omega, \nu^{pre}, \nu^{post}, \gamma, d)$
11. $\quad\quad e_{l+1} \leftarrow e_{l+1} + \tilde{e}_{l+1}$ // Add up corrections over the cycles
12. $\quad$ **endfor**
13. **endif**
14. $e_l^{cgc} = P_l e_{l+1}$ // Prolong coarse-grid correction
15. **if** $(d = 1)$ // Optional fine-level diagonal preconditioning
16. $\quad e_l^d = \text{DiagPrecond}(\bar{r}_l,\ L_l)$ // Precondition with inverse diagonal of A_l
17. **else**
18. $\quad e_l^d = 0$ // No diagonal preconditioning
19. **endif**
20. $e_l^{sum} = e_l^{pre} + e_l^{cgc} + e_l^d$ // Add up corrections
21. $e_l = Smooth(e_l^{sum}, L_l, r_l, \omega, \nu^{post})$ // Post-smoothing

7.3.4 Updating the HSC preconditioner for diagonal shifts

In a number of applications [51] and [38], diagonally shifted versions of the original laplacian L are considered. These perturbations result in Laplacians of the form $L+tI$, where I is the identity matrix and t is a non-negative scalar. For example, in edge-preserving decomposition [51], these perturbations give rise to a multi-scale decomposition—larger values of t correspond to coarser scale versions of the original image. In smoothed geodesic computation on meshes, [38], the scalar t controls the smoothness of geodesic distances on a mesh. The offset t is often not known in advance and needs experimental determination.

In order to avoid the expensive steps of sparsification, compensation, and coloring, we developed the following approximate method for computing a good preconditioner for $L + tI$ given our already constructed hierarchical preconditioner for L. We define the *excess diagonal* at the finest level as $E^0 = tI$. The finest-level Laplacian is modified from L^0 to $L^0 + E^0$. The excess diagonal for the next coarser level is then computed as $E^1 = {P^1}^T E^0$, where P^1 is the Schur matrix at the next level as defined in Algorithm 5. The Laplacian at level 1 is then changed from L^1 to $L^1 + E^1$. This process is continued all the way through the hierarchy, and the modified Cholesky decomposition of the coarsest level Laplacian is also recomputed accordingly.

This update process does not require any new sparsification, compensation or Galerkin recomputation of the coarse level matrices, and is hence extremely fast, typically an order of magnitude faster than recomputing the preconditioner. This updated preconditioner is also very accurate, usually only requiring only one extra CG iterations, as compared to using the recomputed hierarchy for $L+tI$. In Section 7.4, we give timings for an edge-preserving sharpening application on a multi-megapixel image. In future work, we intend to better quantify bounds on the accuracy of this updated preconditioner.

7.3.5 Efficient multilevel eigensolver

A number of applications such as mesh segmentation [107] and spectral matting [96] require the computation of a few lowest eigenvectors of a Laplacian. In mesh segmentation, a 3D mesh is spectrally projected onto the plane by projecting the mesh coordinates on the lowest two non-zero eigenvectors of the mesh Laplacian. Contour analysis is then performed on the projected planar shape (Figure 7.9). Two types of Laplacians are used in this application: a graph (homogenous) Laplacian and a geometric Laplacian designed to enhance concavity of the resulting planar shape.

Since our hierarchy explicitly preserves low energies, a natural strategy is to use a multi scale approach to compute the lowest eigenvectors. First, we compute the exact lowest few eigenvectors at a coarse level (using MATLAB's *eigs* function). We then interpolate these eigenvectors to the finest level using the prolongation matrices. This interpolation already gives us a very good approximation to the true lowest eigenvectors; where for a candidate eigenvector v, we compute the approximate eigenvalue as $\lambda = (v^\top Lv)/(v^\top v)$, and the error as $\|Lv - \lambda v\|$. To refine the vectors further, we perform a few iterations of block Davidson smoothing [4] at the finest level. Typically,

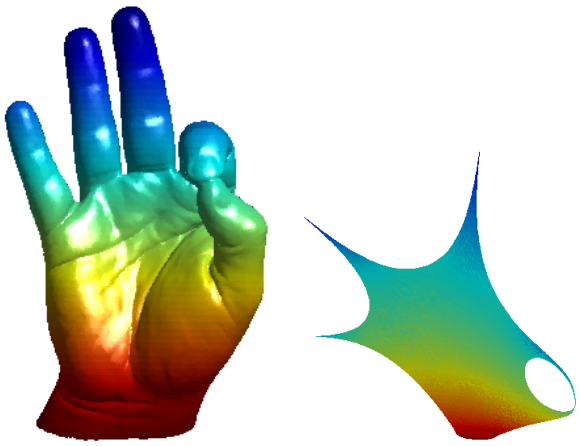


Figure 7.9: Mesh Segmentation using Spectral Embedding: The 3D mesh is embedded into the 2D planar shape. Contour analysis is performed on this planar shape to determine segments in the original mesh. The fingers of the mesh are mapped to the spiky parts of the planar graph. Our solver is used to compute the lowest eigenvectors of the mesh Laplacian, to perform the spectral embedding.

less than a dozen iterations of block Davidson smoothing gives us a very accurate estimation with a relative error of 10^{-4}. This level of accuracy suffices for most graphics applications. The resulting multilevel eigensolver is twice as fast as MATLAB's built-in eigensolver for meshes with a million or more vertices, which uses the state of the art Lanczos methods to determine eigenvectors. It also uses significantly less memory. In Section 7.4, we give timings for the eigensolver for meshes of different sizes.

Our multilevel eigensolver algorithm is given in Algorithm 7. The algorithm starts by computing the HSC hierarchy for the given Laplacian L (line 1). Then the exact eigenvectors are computed at a coarser level which has 30% the number of variables as the finest level (lines 2 and 3). This threshold of 30% was chosen heuristically and is a tradeoff between time taken at the coarsest level and overall number of smoothing iterations. Next, the coarse-level vectors are interpolated to the finest level using the interpolation matrices P_l computed during the preconditioner construction (lines 4 through 9). As we proceed up the hierarchy, the vectors lose their accuracy. To recover some of the accuracy, we perform a Rayleigh-Ritz correction at every level. The idea of the Rayleigh-Ritz correction is briefly given as follows. Suppose we are given an orthonormal set of vectors V which span a subspace $\mathcal{K}$. The Rayleigh-Ritz procedure gives us a rotation of the set V which is the *optimal* approximation to the eigenvectors of L in the subspace $\mathcal{K}$. The rotation is found by finding the eigenvectors V_r of $V^\top LV$, and then setting $V \leftarrow VV_r$. More details and proofs are given in [5]. We perform the Rayleigh-Ritz procedure at every step of the hierarchy during the coarse-to-fine prolongation of the vectors (lines 7 and 8 of Algorithm 7).

Once a candidate set of vectors V are computed from coarse-to-fine prolongation, we perform a series of smoothing iterations to improve the accuracy of the vectors. Given a basis set V, we first compute the residuals of the set (line 12). We check for convergence of vectors using the user-specified tolerance Tol (line 13). If the norm of a column of R drops below Tol, the corresponding

vector in V is considered converged and does not change in further iterations. This is known as *deflation* of the set V. If the number of converged vectors exceeds the number N, we are done (line 13). Otherwise, the residuals are preconditioned using the HSC preconditioner (line 14) and the set of vectors V is augmented with the preconditioned residuals W (line 16), after the set W is orthonormalized with respect to V. To improve the accuracy, Rayleigh-Ritz rotation is performed (lines 17 and 18). This procedure is repeated until convergence is reached. This can cause the set V to keep increasing and greatly slow down the speed. Therefore in practice, all methods to compute eigenvalues are *restarted* methods, whereby the set V is truncated to a pre-defined maximum size to keep memory and computation costs under control. This is done by only keeping the vectors V corresponding to the smallest Rayleigh-Ritz eigenvalues (of course, while always keeping previously converged vectors). More details on deflation, restarting and other techniques are given in [3].

Algorithm 7 Multilevel Eigensolver

$[V, D] = EIGENSOLVER$(L,N,Tol)
INPUT: Laplacian L, number of eigenvectors N, accuracy level Tol
OUTPUT: Orthogonal matrix of eigenvectors V and diagonal matrix of eigenvalues D such that $LV \approx VD$

1. [Fun Hier] = HSCSetup(L) // Set up HSC hierarchy and preconditioner
2. Choose coarse level C with 30% the number of variables as L
3. $[V_C D_C]$ = eigs($Hier.L_C$) // Compute N exact eigenvectors at coarsest level
4. **for** $l = C-1, \ldots 1$ //Work up the levels interpolating and rotating
5. $V_l = Hier.P_l V_{l+1}$ // Interpolate to next level
6. $V_l \leftarrow GramSchmidt(V_l)$ // Make V_l an orthonormal set
7. $[V_r, D_r] = Ritz(V_l, Hier.L_l)$ // Rayleigh-Ritz rotation of V_l
8. $V_l \leftarrow V_l V_r$ // Rotate vectors
9. **endfor**
10. $V \leftarrow V_1$, $D \leftarrow D_1$
11. **for** $it = 1, \ldots, P$ // smoothing iterations
12. $R = LV - VD$ // Compute residual
13. Check for convergence and fix those converged; if number exceeds N quit loop
14. $W = Fun(R)$ // Precondition the residuals
15. $W \leftarrow Orthogonalize(W, V)$ // Orthonormalize W w.r.t V
16. $V \leftarrow [V; W]$ // Augment
17. $[V_r, D] = Ritz(V, L)$ // Rayleigh-Ritz rotation of V
18. $V_l \leftarrow V V_r$ // Rotate vectors
19. **endfor**

7.4 Results

In this section, we compare the performance of the ABF and HSC preconditioning schemes against a number of other preconditioners, as well as MATLAB's direct solver, for a range of 2D and 3D problems. We embed the different preconditioners in the preconditioned conjugate gradient (PCG) method described in Algorithm 6 with $\nu^{pre} = 0$, $\nu^{post} = 1$, and $\gamma = 1$. We

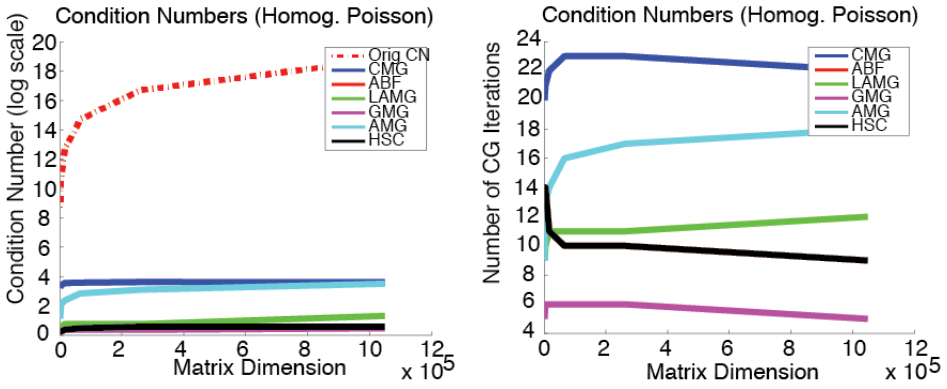


Figure 7.10: Independence to System Size. Left plot shows the condition number of the preconditioned systems achieved by various methods with respect to the matrix dimension n. Similarly, the right plot shows the number of CG iterations needed to achieve a fixed error or 10^{-4} with each method. Both the geometric multigrid method and our preconditioners (ABF and HSC) show invariance to scale.

implement our solvers in MATLAB with Mex acceleration.

In our experiments, we measure the convergence of a solver for $Lx = b$ using two metrics: error with respect to the true solution computed by a direct solver, and relative residual, given by $\|Lx - b\|/\|b\|$. The relative residual is the measure used for terminating PCG iterations.

We compare the performance of our preconditioner against five other state of the art preconditioners. The first is combinatorial multigrid (CMG) [83]. This is an agglomerative-based multigrid method that clusters strongly coupled variables into a single coarse-level variable. The interpolation operators simply copy the value of the coarse variable into all fine variables that belong to the cluster. The second is another variant of this approach that takes the classic AMG approach in [19] and truncates the prolongation matrix to a single variable at every row (which makes it aggregative as well) in order to avoid the growth in the matrix bandwidth. The difference in the CMG and AMG algorithms lie in the manner in which the aggregation of variables is performed. The third method is the lean algebraic multigrid (LAMG) [108], which was described in Chapter 2. LAMG is also an agglomerative method. However, it uses adaptive error smoothing to improve the quality of the interpolation matrices. Low-degree nodes are also eliminated at every level to improve the coarsening rate without sacrificing quality. These additional steps, while increasing accuracy, also greatly increase the setup time for LAMG. This makes it ill-suited for problems where the matrix is used only once for setup and solve. The fourth is a standard geometric multigrid (GMG) method that employs second-order prolongation and restriction matrices [160]. Except for AMG and GMG, we used code provided by the authors. For all these methods, we compare the reduction in error with respect number of iterations and floating-point operations (flops) and the decrease in relative residual with respect to iterations. We also report the wall-clock running times needed to achieve a sensible accuracy threshold. These latter (tabular) results include running time comparisons with MATLAB's direct solver, which is highly optimized.

Problem	Size	Direct	CMG	HSC	LAMG	AMG	ABF	GMG
HDR comp.	4.2M	31.1	29.8	**15.3**	107.4	24.5	**14.8**	19.9
Colorization	5.0M	118.9	19.7	**12.8**	129.7	26.1	**12.6**	27.3
EPD compress (3 scales)	4.2M	140.4	88.8	**70.4**	-	-	-	-
EPD sharpen (5 scales)	4.2M	125.0	97.7	**55.7**	-	-	75.8	-

Table 7.2: Total wall-clock time taken (Setup + Solve) in seconds for problems arising on 2D grids. For each problem, winners are highlighted in bold. Timings within 15% of each other are considered a tie. A '-' means the iterative method did not converge to the target relative residual (10^{-6}) within 30 CG iterations. In all cases, our HSC method is faster than the direct solver by factors ranging from 1.2 to 9.3. Our ABF method is fastest for Laplacians that are homogenous or close to homogenous. The first column gives the number of unknowns. Our methods perform the best over a range of problems with different levels of continuity.

Problem	Size	Unpreconditioned	CMG	HSC	LAMG	AMG	ABF	GMG
HDR comp.	4.2M	1.7×10^6	12.4	**1.5**	5.7	14.0	**1.5**	**1.4**
Colorization	5.0M	2.2×10^7	11.6	**2.2**	**2.0**	19.9	**2.3**	20.8
EPD compress	4.2M	1.0×10^6	9.0	**5.9**	-	-	-	-
EPD sharpen	4.2M	6.7×10^5	10.7	**6.6**	-	**6.9**	-	-

Table 7.3: Condition numbers achieved by the solvers for different 2D problems. The third column gives the condition number of the unpreconditioned Laplacian. The fourth column onwards list the condition number achieved by each method for that problem. For the EPD problems, the Laplacian at the finest level was used to compute the condition numbers. A '-' means that the method did not converge for a problem.

7.4.1 2D Problems

We evaluate our algorithms on three kinds of 2D computational photography problems: homogeneous Poisson equations, non-homogeneous Poisson equations that arise in image colorization [97], and those that arise in EPD problems [51].

Homogenous Poisson Problem. Homogenous Poisson matrices are used for various problems that require the integration of a manipulated gradient image field, e.g., tone mapping [53] and Poisson blending [127]. In Figure 7.10, we verify h-independence we discussed in 7.3.2 by tone mapping the same image at various sizes. The plots show that both HSC and ABF, similarly to GMG, do not depend on the problem size. There are more efficient methods for solving this problem [50]. However, ensuring that the h-independence property holds for our technique allows our method to scale well on problems with mixed coefficients that have large uniform regions within them.

Edge-Preserving Decomposition and Image Colorization. In Figure 7.11 we compare the methods on the highly irregular matrices arising from the use of EPD for dynamic range compression and detail enhancement [51]. For image colorization, HSC and ABF perform the best. For edge-preserving decomposition, HSC and CMG perform the best. For EPD, geometric methods such as GMG and ABF perform poorly because they use regular subsampling grids that fail to preserve oriented or thin regions at coarser levels. As described in Section 7.3.4, our preconditioner can be efficiently updated for problems such as Edge-Preserving decomposition,

where a series of diagonally shifted Laplacians are generated.

In Table 7.2, we show the timing results for EPD compression and EPD sharpening for multiple such shifts. Neither CMG nor the direct solver have such an update, so their performance suffers compared to HSC. CMG is competitive with our solver for EPD-like problems if only a single solve is required. However, for multiple solves, our performance greatly improves due to our efficient preconditioner update. We tried the same heuristic preconditioner update for CMG, but the resulting approximate preconditioner performed poorly for diagonal shifts, thus forcing CMG to recompute the preconditioner for every diagonal shift.

Figure 7.11 also compares the methods on matrices arising from colorization problem [97] of a very large image. This problem uses less irregular weights and hence the piecewise constant basis functions that CMG use undermine its performance.

Finally, in Table 7.2, we summarize wall-clock running times of all the methods on different 2D problems. We see that while ABF works well on homogenous problems, it often fails to converge (in reasonable time) on more heterogenous problem such as EPD.

7.4.2 3D Meshes

We now consider applications of Laplacians arising in 3D surface mesh processing. The first is mesh segmentation, the second is geodesic distance computation on meshes, and the third is mesh denoising. The ABF and GMG preconditioners cannot be used in 3D problems owing to the lack of an underlying grid structure for the regular coarsening.

Mesh Segmentation. As explained in Section 7.3.5 and shown in Figure 7.9 and Figure 7.2, the mesh segmentation method developed in [107] uses at its core an eigensolver to compute the three lowest eigenvectors of a Laplacian defined over the 3D mesh. Two types of Laplacians are described in the paper: a homogenous graph Laplacian and a geometric Laplacian. In Table 7.4, we compare the timings taken by our eigensolver (using HSC), AMG, and MATLAB's built-in eigensolver, *eigs*, to compute the three lowest eigenvectors of homogenous Laplacians defined over 3D meshes of different sizes. For larger mesh sizes, we are between two and three times faster than MATLAB's eigensolver. We observe similar performance for the non-homogenous Laplacians defined in [107]. The CMG solver does not have an explicit preservation of low eigenvectors across the hierarchy. As a result, a multi scale initialization works poorly for eigenvector computation.

Geodesic Distance Computation. In [38], a heat kernel is used to compute geodesic distance between points on a 3D mesh. The method introduced in their paper involves the solution of two linear systems. The Laplacians in these two systems are related to each other by a diagonal shift. Hence, our preconditioner update scheme Section 7.3.4 helps to reduce the overall computation time. In Table 7.5, we compare HSC and MATLAB's direct solver on homogenous Laplacians defined over meshes of different sizes. (AMG and CMG results are not reported, since they failed to converge in an acceptable time.) HSC provides a significant speedup over the direct solver. Figure 7.2 (right) shows the visualization of geodesic distances computed on the Ceasar

Mesh	Vertices	MATLAB *eigs*	AMG (Speedup)	HSC (Speedup)
Hand	50K	0.9	0.9 (1x)	0.9 (1x)
Lion	150K	3.9	3.0 (1.3x)	2.6 (1.5x)
Lago.	800K	27.1	16.1 (1.7x)	10 (2.7x)
Neptune	2M	60.9	35.6 (1.7x)	30.8 (2x)
Statuette	5M	154.8	82.3 (1.9x)	66.5 (2.3x)

Table 7.4:]
Wall clock time (in seconds) to compute the lowest three eigenvectors of homogenous Laplacian defined on a 3D mesh, comparing our eigensolver, agglomerative AMG, and MATLAB's *eigs*. The speedup of our solver and AMG over the direct solver is given in parentheses.

Mesh	Vertices	Direct Solver	HSC Setup/Solve (Speedup)
Lion	150K	7.5	1/0.5 (15.6x)
Lago.	800K	36.2	2.6/1.7 (21.3x)
Neptune	2M	104.4	11.3/6.2 (16.8x)
Statuette	5M	202	29.1/13.3 (15.2x)

Table 7.5:]
Wall clock time (in seconds) to compute geodesic distances on 3D meshes of different sizes. We compare our solver and speedup over MATLAB's direct solver. The agglomerative methods AMG and CMG failed to converge within 50 PCG iterations to an accurate solution for these Laplacians. The speedup ratios are given for the solve phase over the direct solver.

mesh. Here, the distances are computed from a point on Ceasar's nose (red is closer to the source point, blue is farther). The isolines of the resulting distance function are also shown. In geodesic distance computation, often the distance of points to different initialization need to be re-computed. Therefore, the solve phase is repeatedly performed with different right hand sides.

Mesh Smoothing. We perform Laplacian-based smoothing of noisy meshes, using the inverse Euclidean distance measure between vertices , $w_{ij} = \|\mathbf{v}_i - \mathbf{v}_j\|^{-2}$, as the entries in the Laplacian. Given the noisy vertices V_n of the original mesh, we compute smoothed vertices V_s, by solving the smoothing equation $(I + tL)V_s = V_n$. For comparison, on a mesh with 150K vertices, our method takes less than 1 second for the solve phase, whereas the bilateral filtering approach in [58] takes about 24 seconds on a mesh with 100K vertices (on results reported in 2003). Laplacians based on bilateral filtering-based similarity measures may also be used for denoising. In Figure 7.12, we give an example of mesh smoothing, and we report timing results in Table 7.6.

7.5 Discussion

We have presented two efficient and effective multi-level matrix preconditioning schemes that apply to a large class of Laplacian matrices used in computer graphics, including inhomogeneous computational photography problems. The second of the two schemes also works on 3-D mesh processing problems. The new preconditioners operate in a very localized manner and avoid the issue of growth in non-zero matrix bandwidth and hence runs in linear time. The first

Mesh	Vertices	Direct Solver	HSC Setup/Solve (Speedup)
Lion	150K	2.5	0.7/0.9 (2.8x)
Lago.	800K	26.3	2.1/0.5 (52x)
Neptune	2M	87.1	8.2/6.7 (13x)
Statuette	5M	176.2	18.9/50.4 (3.5x)
Lucy	14M	1246	80.6/76.9 (16.2x)

Table 7.6: Wall clock time (in seconds) to to smooth noisy meshes of different sizes. We compare our solver and speedup over MATLAB's direct solver. The agglomerative methods AMG and CMG failed to converge within 50 PCG iterations to an accurate solution for these inhomogenous Laplacians. The speedup ratios are given for our solve phase over the direct solver.

scheme, ABF, use fixed coarsening and sparsification. The second scheme, HSC, adaptively selects variables that are eliminated and the connections that are dropped and compensated, guided by principles that aim to maximize the preconditioning effectiveness. The derivation of HSC is based on a formal analysis that ties the algorithmic decisions with their effect over the condition number of the preconditioned system. We derive a new compensation scheme based on this analysis that considers the interplay between levels in the hierarchy. This compensation helps to decouple the resulting condition number from the system size n.

The experiments we report show that our new preconditioners outperform or equal other state-of-the-art iterative and direct methods in all scenarios, both in terms of operation count and wall-clock time. The ability to perform well on all applications makes HSC a more useful technique than specialized solvers such as GMG, CMG, or ABF, which only work well under certain conditions. Our performance results from our ability to reduce the condition number of highly irregular Laplacian matrices as well as our use of geometric coarsening in homogeneous regions, i.e., a mixed strategy well-suited for many computer graphics applications.

Our approach, like most multilvel methods, is inherently limited to Laplacian matrices that have non-positive off-diagonals entries. As future work, we plan to generalize our approach to wider families of matrices. We also plan to extend our approach to 3D volumetric applications in computer graphics and simulation, and to develop parallel and GPU-based implementations of our algorithms. Finally, we plan to study the theoretical relationship between our approach and existing algorithms such as GMG, AMG, and CMG, to see if we can derive formal proofs on the condition number and scaling properties of our approach.

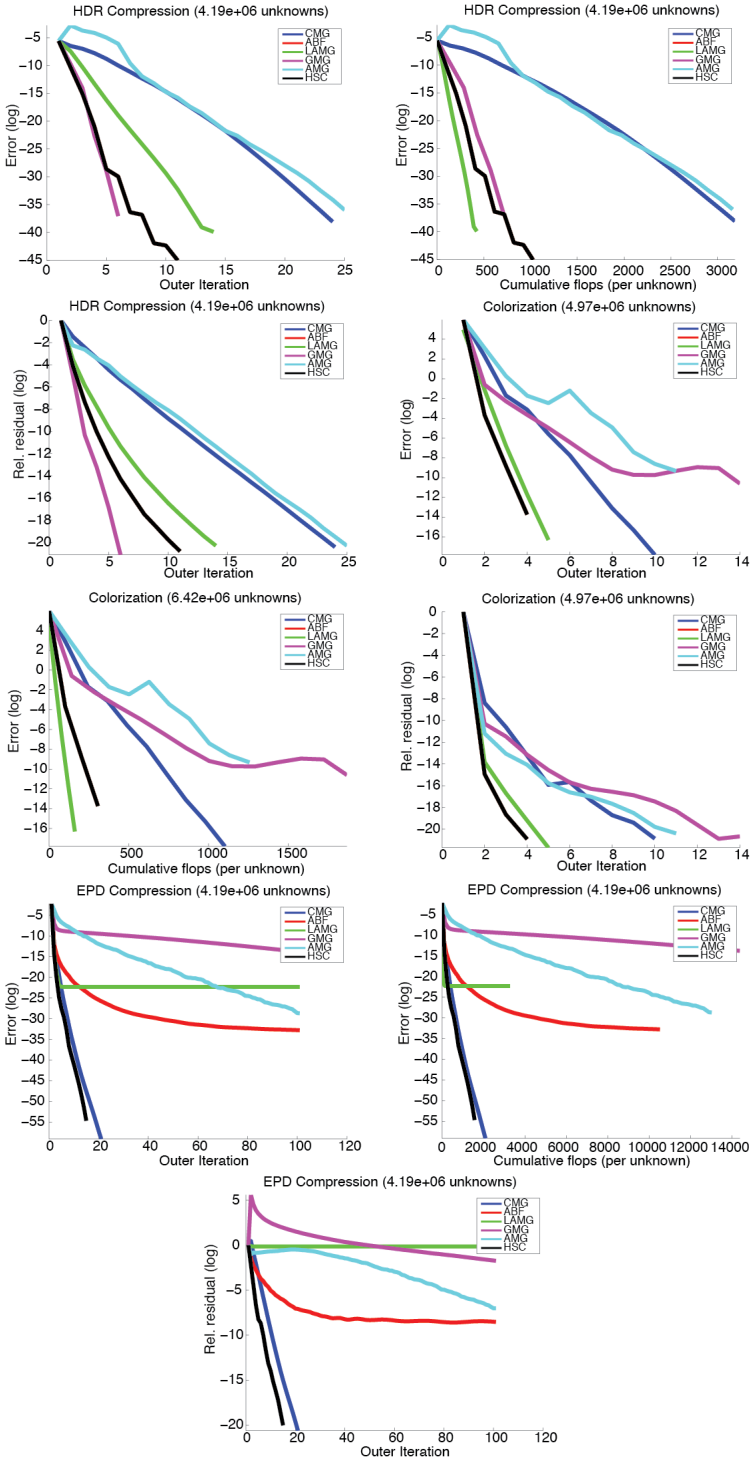


Figure 7.11: Performance comparison for 2D problems with varying degrees of homogeneity: top: HDR Compression on a 2048×2048 image; middle: image colorization on a 1962×2533 sizes image; and Edge-Preserving Dynamic range compression on a 2048×2048 image (single level). Our method consistently ranks at or near the top on all the metrics considered: error with respect to iterations and flop count; and relative residual.

Figure 7.12: Example of mesh smoothing using an inhomogenous Laplacian based on an inverse distance measure between vertices. Left: original mesh; Middle: noisy mesh; Right: denoised mesh with our solver with smoothing parameter $t = 0.1$.

Chapter 8

Conclusions

In this thesis, we have proposed new image priors and explored their use in computational photography applications. We have developed a novel mechanism to enable low-light photography. We have developed an extremely fast algorithm for non-blind deconvolution. One of the new priors, based on a sparsity measure, has proved useful in regularizing blind deconvolution problems. Finally, we have developed novel preconditioners for linear systems involving Laplacians. Such Laplacians arise in computer graphics and computational photography applications. In developing these preconditioners, we have addressed fundamental problems of bandwidth growth and adaptability which exist in multigrid methods.

The work we have explored can be extended in a number of future directions. Recent research [98] has indicated that image denoising using non-parametric methods has reached near-optimal performance. Further progress will require parametric methods and therefore more sophisticated image priors. The spectral prior we introduced in Chapter 3 is simple and does not take into account inter-spectral relationships beyond the correlation of edge positions. A more careful study could result in better priors that overcomes the shortcomings we discussed in Chapter 3. A more powerful prior would be useful in applications such as the fusion of far-IR and visible wavelength images, which would be useful in car driver assistance systems.

The ℓ_1 / ℓ_2 prior we developed in Chapter 5 for blind deconvolution works well for indoor scenes. However, for complex textures and outdoor scenes, this prior can often fail, as pointed out in [165]. Finding a more robust measure to distinguish blurred and sharp images remains an outstanding problem. Such a measure would be necessary to tackle the much tougher challenge of spatially varying deconvolution, where there are still not satisfactory solutions. A proof of optimality of such a measure would also be of interest. Unfortunately, the proofs from the blind equalization literature [140, 141] cannot be directly used due to their assumption on the IID nature of the latent signal. Images do not enjoy such independence properties.

Our Laplacian preconditioners only work with M-matrices, which have negative off-diagonal entries. Many Laplacians in computer graphics, however, result in non-M matrices. Examples of

these are cotangent and biharmonic Laplacians. Cotangent Laplacians arise because weights on an edge correspond to (sums of) cotangents of angles opposite that edge. Clearly, depending on the angles, these can be positive or negative. Biharmonic Laplacians are of the form $L^T L$ where L is a Laplacian. These arise from reparametrization problems [122]. These are commonly used in computer graphics and at present, only Cholesky-based direct solvers such those in MATLAB or TAUCS [158] provide fast code to solve such problems. A preconditioner for such problems would be of interest to the computer graphics community.

Appendix A

Coarse Level matrices are Laplacian

Lemma 1. *Let L be a Laplacian matrix with the following characteristics: it is symmetric, diagonally dominant and has non-positive off-diagonals. Let $\mathcal{C}$ and $\mathcal{F}$ be disjoint variables index sets such that no two variables in $\mathcal{F}$ are connected to each other in L. Then the coarse level system $L^1 = L_{\mathcal{CC}} - L_{\mathcal{CF}} L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}}$ is also a Laplacian matrix which shares the same properties as L.*

Proof. Since no variable in $\mathcal{F}$ is connected to any other variables, $L_{\mathcal{FF}}$, is a diagonal matrix with positive diagonal entries (which was the case in L). Since the off-diagonal entries in L are all non-positive, all the entries in $L_{\mathcal{CF}}$ are also non-positive. Hence all the entries in $L_{\mathcal{CF}} L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}}$ are positive. The matrix $L_{\mathcal{CC}}$ also has non-positive off-diagonals since it is a sub-matrix of L. Hence all the off-diagonal values of L^1 are non-positive. The diagonal elements in Laplace matrices are greater or equal than the sum of the corresponding columns (without the diagonal elements). Hence $L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}}$ consists of negative values above -1. Similarly, the row sums of $-L_{\mathcal{CF}}$ is smaller than the correspond diagonal elements in $L_{\mathcal{CC}}$. Hence, all the diagonal elements in $L_{\mathcal{CF}} L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}}$ (which are positive) are smaller than those in $L_{\mathcal{CC}}$ and therefore the diagonal elements of L^1 are all positive. □

Appendix B

Bounds on Energy Deviation after Sparisification

Lemma 2. *Let T be a Laplacian matrix of three variables. Let w_{12}, w_{13} and w_{23} be $-T_{12}, -T_{13}$ and $-T_{23}$ respectively, where T_{ij} are the entries of the matrix T. Assume these weights are positive and $w_{12}, w_{13} \geq w_{23}$. The matrix $\tilde{T}$ produced by dropping w_{23} will obey*

$$E_{\tilde{T}} \leq E_T \leq bE_{\tilde{T}}, \tag{B.1}$$

with $b \leq 3$, where the energies E_T and $E_{\tilde{T}}$ are with respect to any vector x.

Proof. The definition of E_T in terms of the weights is given in Eq. C.2. $E_{\tilde{T}} \leq E_T$ follows immediately since $E_{\tilde{T}}$ has one less term, $w_{23}(x_2 - x_3)^2$, than E_T. Without loss of generality let us assume $\|\mathbf{x}\| = 1$ in which case $E_{\tilde{T}} = w_{12}(x_2 - x_1)^2 + w_{13}(x_1 - x_3)^2$. However, since $w_{12}, w_{13} \geq w_{23}$ we get $E_{\tilde{T}} \geq w_{23}\big((x_2 - x_1)^2 + (x_1 - x_3)^2\big)$ and since $s^2 + t^2 \geq (s+t)^2/2$ for any s and t, we get $E_{\tilde{T}} \geq w_{23}(x_2 - x_3)^2/2$. Thus, $3E_{\tilde{T}} \geq w_{23}(x_2 - x_3)^2 + E_{\tilde{T}} = E_T$ and therefore $b = 3$ satisfies the upper bound. □

Appendix C

Characterization of Sparsified Spaces and Compensation

As we explained in Section Section 7.3.2 the errors introduced by the sparsification done in each hierarchy level can add up and hurt the preconditioning such that $\kappa(LQ^{-1}) = O(n)$ as in the case of a non-preconditioned system.

The key observation that allows us to cope with this shortcoming is the that at each hierarchy level about half the variables are eliminated and these variables define a linear subspace that *does not* experience the sparsification taking place in the following levels. This means that there is a hierarchy of subspaces that undergo a different number of sparsification steps. Hence we should focus the compensation step to minimize the error due to sparsification in the subspaces that are affected by the largest number of sparsification steps. Therefore we need to establish the sense at which $\tilde{L}$ should best approximates L.

The sparsification that takes place at the finest level, over the input matrix L, is likely to affect most of $\mathbb{R}^n$. The second sparsification which takes place at the second hierarchy level operates over the sub-matrix that corresponds to the coarse variables selected at the finest level after nearly half of the variables were eliminated. This gives raise to two linear spaces one which is affected by this second sparsifiication step and one which is *not*. To characterize these vector spaces let us ignore the sparsification done at the finest level and consider the system resulting after computing the first level of the hierarchy,

$$L\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad (P^1)^\top L P^1 \mathbf{y}^1 = L^1 \mathbf{y}^1 = (P^1)^\top \mathbf{b}, \tag{C.1}$$

The solution $\mathbf{y}$ to this problem provides an exact solution, $\mathbf{x} = P^1\mathbf{y}$, for the original system. However, once we sparsify L^1, and get $\tilde{L}^1$, this procedure computes an approximate solution for $\mathbf{x}$. Equivalently, the matrix $P^1(\tilde{L}^1)^{-1}(P^1)^\top$ is the preconditioning matrix Q^{-1} that approximates L^{-1}in this one-level construction. As we see in (7.8), the resulting matrices L^1 and $\tilde{L}^1$ are block diagonal, meaning that the coarse and fine coordinates of $\mathbf{y}^1 = [\mathbf{y}^1_{\mathcal{C}}, \mathbf{y}^1_{\mathcal{F}}]^\top \in \mathbb{R}^n$ are uncoupled.

Furthermore, since the sparsification at that level (producing $\tilde{L}^1$ from L^1) operates only over the top-left block that corresponds to the coarse variables $\mathbf{y}^1_{\mathcal{C}}$ and the fine coordinates $\mathbf{y}^1_{\mathcal{F}}$ are not affected by it. At the original coordinates $\mathbf{x}$, the latter subspace is given by $\{\mathbf{x} : ((P^1)^{-1}\mathbf{x})_{\mathcal{C}} = \mathbf{0}\}$ and the one affected by the sparsification by $\{\mathbf{x} : ((P^1)^{-1}\mathbf{x})_{\mathcal{F}} = \mathbf{0}\}$. In fact, according to the definition of an inverse matrix, these subspaces are given more explicitly by $\{P^1[\mathbf{0}, \mathbf{y}^1_{\mathcal{F}}]^\top : \forall \mathbf{y}^1_{\mathcal{F}}\}$ and $\{P^1[\mathbf{y}^1_{\mathcal{C}}, \mathbf{0}]^\top : \forall \mathbf{y}^1_{\mathcal{C}}\}$ respectively.

This rationale can be applied at any hierarchy level l where we get that $\{P^1P^2..P^l[\mathbf{y}^l_{\mathcal{C}}, \mathbf{0}]^\top : \forall \mathbf{y}^l_{\mathcal{F}}\}$ is affected by the sparsification steps of the first $l+1$ levels (and the following levels computed), whereas $\{P^1P^2..P^l[\mathbf{0}, \mathbf{y}^l_{\mathcal{F}}, \mathbf{0}]^\top : \forall \mathbf{y}^l_{\mathcal{F}}\}$ by the sparsification of only the first l levels. In fact, by starting this analysis from an arbitrary level l and get that $\{P^lP^{l+1}..P^{l'}[\mathbf{0}, \mathbf{y}^{l'}_{\mathcal{F}}, \mathbf{0}]^\top : \forall \mathbf{y}^{l'}_{\mathcal{F}}\}$ are vectors, given in the coordinates of the l-th level, that undergo the following $l' - l$ sparsification steps.

This implies that in order to avoid excessive energy mismatch over vectors that undergo multiple sparsification steps, at every hierarchy level l the compensation should improve the accuracy of the sparsified matrix over the column vectors of $P^lP^{l+1}..p^m$ which correspond to coarser levels. The problem is, however, that at the time we construct the l-th level, the matrix $P^lP^{l+1}..P^m$ is not yet defined and *depends* on our operations at the l-th level, including the compensation itself. Therefore, a more qualitative description of the coarse spaces is needed.

To do so, let us still consider a single level of the hierarchy with the sparsification done at the finest level neglected. The column vectors of P^1 which correspond to the coarse variables describe an interpolation from the coarser system $\mathbf{y}^1_{\mathcal{C}}$ to the original grid, i.e., $\mathbf{y}^1_{\mathcal{C}} \mapsto P^1[\mathbf{y}^1_{\mathcal{C}}, \mathbf{0}]^\top \in \mathbb{R}^n$. According to its definition in (7.7) , the matrix P^1 has $I_{\mathcal{C}\mathcal{C}}$ at its top-left block and hence the coarse variables in the original grid receive the same values they have at the coarser system $\mathbf{y}^1_{\mathcal{C}}$. The fine variables in the original grid are also determined by $\mathbf{y}^1_{\mathcal{C}}$ and, according to (7.7), they given by $-(L^{-1}_{\mathcal{F}\mathcal{F}}L_{\mathcal{F}\mathcal{C}})\mathbf{y}^1_{\mathcal{C}}$.

On the other hand in the case of null input values and derivatives ($y_i, z_{ij} = 0$) the energy is related to the objective function in Eq. 7.2 by

$$E_L(\mathbf{x}) = F(x)/(\mathbf{x}^\top\mathbf{x}) = \sum_{i\in\mathcal{I}} \big(u_i x_i^2 + w_{ij}(x_i - x_j)^2\big)/(\mathbf{x}^\top\mathbf{x}), \tag{C.2}$$

and the two coincide on unit-norm vectors, $\mathbf{x}^\top\mathbf{x} = \|\mathbf{x}\|^2 = 1$, and both are given by $\mathbf{x}^\top L\mathbf{x}$. Let us constrain the coarse coordinates of $\mathbf{x}_{\mathcal{C}}$ to be equal to $\mathbf{y}^1_{\mathcal{C}}$ and minimize the functional with respect to the remaining variables $\mathbf{x}_{\mathcal{F}}$. This constrained optimization is computed by $dF(\mathbf{x})/d\mathbf{x}_{\mathcal{F}} = d(x^\top L\mathbf{x})/d\mathbf{x}_{\mathcal{F}} = \mathbf{0}$ given $\mathbf{x}_{\mathcal{C}}$, and gives

$$\begin{bmatrix} I_{\mathcal{C}\mathcal{C}} & 0 \\ L_{\mathcal{F}\mathcal{C}} & L_{\mathcal{F}\mathcal{F}} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{y}^1_{\mathcal{C}} \\ \mathbf{0} \end{bmatrix} \;\Rightarrow\; \begin{array}{c} \mathbf{x}_{\mathcal{C}} = \mathbf{y}^1_{\mathcal{C}} \\ L_{\mathcal{F}\mathcal{C}}\mathbf{x}_{\mathcal{C}} + L_{\mathcal{F}\mathcal{F}}\mathbf{x}_{\mathcal{F}} = \mathbf{0}, \end{array} \tag{C.3}$$

implying that

$$\mathbf{x}_{\mathcal{F}} = -\big(L^{-1}_{\mathcal{F}\mathcal{F}}L_{\mathcal{F}\mathcal{C}}\big)\mathbf{y}^1_{\mathcal{C}}, \tag{C.4}$$

which is what $P^1\mathbf{y}_{\mathcal{C}}^1$ produces at the fine variables of $\mathbf{x}$.

Thus, $P^1[\mathbf{y}_{\mathcal{C}}^1, \mathbf{0}]^\top$ are the minimal-cost vectors given the assignment $\mathbf{y}_{\mathcal{C}}^1$ over its coarse coordinates. If we assume no sparsification (and compensation) steps are applied throughout the hierarchy, computing multiple elimination steps in the hierarchy is equivalent eliminating these variables at once. Hence, $P^1P^2..P^l[\mathbf{y}_{\mathcal{C}}^l, \mathbf{0}]^\top$ are the minimal-cost vectors given the assignment $\mathbf{y}_{\mathcal{C}}^l$ over its coarse coordinates. However, since there are far less variables in the l-th level as l grows, this optimization has fewer constraints and hence it is expected to achieve lower energies.

Based on this observation we associate variables of coarser levels with vectors of low energy and focus the compensation step to preserve the energy of such vectors. As explained in Section (7.3.2), whenever a triangle is sparsified we compensate by adjusting its two remaining weights such that the triangle's energy contribution remains unchanged over low-energy vectors. This requires modeling the profile of low-energy vectors locally, over three variables in a triangular connectivity but it does not requires knowing P^l of coarser levels. In Section (7.3.2) we explain how the low-energy vectors are modeled.

Finally, this reasoning did not take into account the sparsification (and compensation) applied to the matrices; the low-energy vectors which we use in this process are predicted based on a sparsified hierarchy. However, assuming we succeed in preserving the energy of low-energy vectors, in each step, the predictions we make in the next level will be reasonably accurate.

Bibliography

[1] A. Agrawal, R. Raskar, S. Nayar, and Y. Li. Removing photography artifacts using gradient projection and flash-exposure sampling. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 24, pages 828–835, 2005.

[2] M. Aharon, M. Elad, and A. Bruckstein. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Processing*, 54(11):4311–4322, November 2006.

[3] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq, and R. S. Tuminaro. A comparison of eigensolvers for large-scale 3d modal analysis using amg-preconditioned iterative methods. *International journal for numerical methods in engineering*, 64(2):204–236, 2005.

[4] P. Arbenz, U. Hetmanuik, R. Lehoucq, and R. Tuminaro. A comparison of eigensolvers for large-scale 3d modal analysis using amg-preconditioned iterative methods. *Int. Journal for Numerical Methods in Engg.*, 1, 2003.

[5] Z. Bai. Krylov subspace projection methods. `http://www.cs.ucdavis.edu/~bai/Winter09/krylov.pdf`.

[6] S. Baker, R. Gross, and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:221–255, 2004.

[7] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *Int. J. Comput. Vision*, 92:1–31, March 2011.

[8] P. Barnum, S. G. Narasimhan, and T. Kanade. Analysis of rain and snow in frequency space. *IJCV*, 2008.

[9] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. In *SIAM J. on Img. Sciences*, pages 183–202, 2009.

[10] W. Bell, L. Olson, and J. Schroder. Pyamg: Algebraic multigrid solvers in python v2. 0, 2011. *URL http://www. pyamg. org. Release*, 2, 2011.

[11] E. Bennett, J. Mason, and L. McMillan. Multispectral bilateral video fusion. *IEEE Trans. Image Processing*, 16(5):1185–1194, 2007.

[12] P. Bhat, C. L. Zitnick, M. Cohen, and B. Curless. Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Trans. Graph.*, 29:10:1–10:14, April 2010.

[13] J. Biemond, A. M. Tekalp, and R. L. Lagendijk. Maximum likelihood image and blur identification: a unifying approach. *Optical Engineering*, 29(5):422–435, 1990.

[14] J. Bolz, I. Farmer, E. Grinspun, and P. Schröoder. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 917–924, New York, NY, USA, 2003. ACM.

[15] G. Boman, E. and B. Hendrickson. On spanning tree preconditioners. *Sandia National Labs*, 2001.

[16] G. Boman, E. and B. Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, (3):694–717, 2003.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[18] A. Brandt. Multi-level adaptive technique (mlat) for fast numerical solution to boundary value problems. In *Proc. Conf. on Numerical Methods in Fluid Mechanics*, volume 18 of *Lecture Notes in Physics*, pages 82–89. Springer Berlin / Heidelberg, 1973.

[19] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(14):23 – 56, 1986.

[20] B. Brooksby, S. Srinivasan, D. Jiang, H. Dehghani, B. Pogue, K. Paulsen, J. Weaver, C. Kogel, and S. Poplack. Spectral priors improve near-infrared diffuse tomography more than spatial priors. *Optics Letters*, 30:1968–1970, 2005.

[21] 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*, volume 2, pages 60–65. IEEE, 2005.

[22] J.-F. Cai, H. Ji, C. Liu, and Z. Shen. Blind motion deblurring from a single image using sparse approximation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 104–111. IEEE, 2009.

[23] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52:489–509, 2004.

[24] E. J. Candes and D. L. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise c2 singularities. *Communications on pure and applied mathematics*, 57(2):219–266, 2003.

[25] A. Chakrabarti, K. Hirakawa, and T. Zickler. Color constancy beyond bags of pixels. In *CVPR*, pages 1–6, 2008.

[26] A. Chakrabarti and T. Zickler. Statistics of real-world hyperspectral images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 193–200. IEEE, 2011.

[27] A. Chakrabarti and T. Zickler. Depth and deblurring from a spectrally-varying depth-of-field. In *Proc. ECCV*, 2012.

[28] G. Chantas, N. Galatsanos, A. Likas, and M. Saunders. Variational bayesian image restoration based on a product of t-distributions image prior. *IEEE Trans. Image Process*, 17(10):1795–1805, 2008.

[29] R. Chartrand. Fast algorithms for nonconvex compressive sensing: Mri reconstruction from very few data. In *IEEE International Symposium on Biomedical Imaging (ISBI)*, 2009.

[30] R. Chartrand and V. Staneva. Restricted isometry properties and nonconvex compressive sensing. *Inverse Problems*, 24:1–14, 2008.

[31] R. Chartrand and W. Yin. Iteratively reweighted algorithms for compressive sensing. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 3869–3872. IEEE, 2008.

[32] S. Cho and S. Lee. Fast motion deblurring. *SIGGRAPH ASIA 2009*, 28(5):article no. 145, 2009.

[33] S. Cho, J. Wang, and S. Lee. Handling outliers in non-blind image deconvolution. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 495–502. IEEE, 2011.

[34] T. S. Cho, C. L. Zitnick, N. Joshi, S. B. Kang, R. Szeliski, and W. T. Freeman. Image restoration by matching gradient distributions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):683–694, 2012.

[35] J. Christian and F. Zapata. Noise Ninja, Photoshop denoising plugin. `http://www.picturecode.com/`, 2008.

[36] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8, 2002.

[37] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212, 2011.

[38] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in Heat. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 31(4), July 2012.

[39] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095, 2007.

[40] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *Image Processing, IEEE Transactions on*, 16(8):2080–2095, 2007.

[41] A. Danielyan, V. Katkovnik, and K. Egiazarian. Bm3d frames and variational image deblurring. *Image Processing, IEEE Transactions on*, 21(4):1715–1728, 2012.

[42] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2009.

[43] T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, 2006.

[44] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 31(3):369–378, 1997.

[45] B. Dong, H. Ji, J. Li, Z. Shen, and Y. Xu. Wavelet frame based blind image inpainting. *Applied and Computational Harmonic Analysis*, 32(2):268–279, 2011.

[46] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 23, pages 673–678, 2004.

[47] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, 2006.

[48] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[49] R. Falgout and U. Yang. hypre: A library of high performance preconditioners. *Computational ScienceICCS 2002*, pages 632–641, 2002.

[50] Z. Farbman, R. Fattal, and D. Lischinski. Convolution pyramids. *ACM Trans. Graph.*, 30:175:1–175:8, December 2011.

[51] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3), August 2008.

[52] R. Fattal. Edge-avoiding wavelets and their applications. In *ACM SIGGRAPH papers*, pages 22:1–22:10, New York, NY, USA, 2009. ACM.

[53] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. *ACM Transactions on Graphics*, pages 249–256, 2002.

[54] R. Fergus and D. Krishnan. Methods, computer-accessible medium and systems for facilitating dark flash photography, July 16 2010. WO Patent 2,010,081,010.

[55] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25:787–794, 2006.

[56] D. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.

[57] D. J. Field et al. Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A*, 4(12):2379–2394, 1987.

[58] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral Mssh Denoising. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3), 2003.

[59] K. Garg and S. Nayar. Detection and removal of rain from videos. In *CVPR*, pages 528–535, 2004.

[60] K. Garg and S. Nayar. When Does a Camera See Rain? In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1067–1074, Oct 2005.

[61] D. Geman and G. Reynolds. Constrained restoration and recovery of discontinuities. *PAMI*, 14(3):367–383, 1992.

[62] D. Geman and C. Yang. Nonlinear image recovery with half-quadratic regularization. *PAMI*, 4:932–946, 1995.

[63] A. Goldstein and R. Fattal. Blur-kernel estimation from spectral irregularities. In *European Conference on Computer Vision (ECCV)*, pages 622–635. Springer, 2012.

[64] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[65] G. Golub and C. F. Van Loan. *Matrix Computation, third edition*. The John Hopkins University Press, Baltimore and London, 1996.

[66] J. Gu, R. Ramamoorthi, P. Belhumeur, and S. Nayar. Removing Image Artifacts Due to Dirty Camera Lenses and Thin Occluders. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, Dec 2009.

[67] A. Gupta, N. Joshi, L. Zitnick, M. Cohen, and B. Curless. Single image deblurring using motion density functions. In *ECCV '10*, 2010.

[68] S. Harmeling, M. Hirsch, and B. Schölkopf. Space-variant single-image blind deconvolution for removing camera shake. *Advances in Neural Inform. Processing Syst*, 2010.

[69] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Scholkopf. Fast removal of non-uniform camera shake. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 463–470. IEEE, 2011.

[70] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *JMLR*, 5:1457–1469, 2004.

[71] N. Hurley and S. Rickard. Comparing measures of sparsity. *EUSIPCO*, 2008.

[72] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. *European Conf. Computer Vision (ECCV)*, 2012.

[73] H. Ji, J. Li, Z. Shen, and K. Wang. Image deconvolution using a characterization of sharp images in wavelet domain. *Applied and Computational Harmonic Analysis*, 2011.

[74] J. Jia. Single image motion deblurring using transparency. In *CVPR*, 2007.

[75] J. Jiaya. Single image motion deblurring using transparency. In *CVPR*, pages 1–8, 2007.

[76] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski. Image deblurring using inertial measurement sensors. *ACM Transactions on Graphics (TOG)*, 29(4):30, 2010.

[77] M. Kazhdan and H. Hoppe. Streaming multigrid for gradient-domain operations on large images. *ACM Trans. Graph.*, 27(3):21:1–21:10, 2008.

[78] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. *arXiv preprint arXiv:1301.6628*, 2013.

[79] D. Kincaid and W. Cheney. *Numerical analysis: mathematics of scientific computing*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 1991.

[80] J. T. Klosowski and S. Krishnan. Real-time image deconvolution on the gpu. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7872, page 16, 2011.

[81] I. Koutis, A. Levin, and R. Peng. Improved spectral sparsification and numerical algorithms for sdd matrices. *STACS*, 2012.

[82] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving sdd linear systems. *Proceedings of FOCS 2010*, 2010.

[83] I. Koutis, G. L. Miller, and D. Tolliver. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. *Computer Vision and Image Understanding*, pages 1638–1646, 2011.

[84] D. Krishnan, R. Fattal, and R. Szeliski. Efficient preconditioning of laplacian matrices for computer graphics. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 32(4), 2013.

[85] D. Krishnan and R. Fergus. Dark flash photography. In *ACM Transactions on Graphics, SIGGRAPH 2009 Conference Proceedings*, volume 28, 2009.

[86] D. Krishnan and R. Fergus. Dark flash photography. *ACM Transactions on Graphics, SIGGRAPH 2009 Conference Proceedings*, 2009.

[87] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1033–1041. 2009.

[88] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. 2009.

[89] D. Krishnan and R. Szeliski. Multigrid and multilevel preconditioners for computational photography. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, (5), 2011.

[90] D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 233–240. IEEE, 2011.

[91] D. Kundur and D. Hatzinakos. Blind image deconvolution. *Signal Processing Magazine, IEEE*, 13(3):43–64, 1996.

[92] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[93] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and M. K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[94] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[95] A. Levin. Blind motion deblurring using image statistics. In *NIPS*, 2006.

[96] A. Levin, R. Fergus, F. Durand, and W. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3):70, 2007.

[97] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, pages 689–694, 2004.

[98] A. Levin and B. Nadler. Natural image denoising: Optimality and inherent bounds. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2833–2840. IEEE, 2011.

[99] A. Levin, B. Nadler, F. Durand, and W. T. Freeman. Patch complexity, finite pixel correlations and optimal denoising. 2012.

[100] A. Levin and Y. Weiss. User assisted separation of reflections from a single image using a sparsity prior. *PAMI*, 29(9):1647–1654, Sept 2007.

[101] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *CVPR*, 2009.

[102] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *Extended Technical Report*, 2009.

[103] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2657–2664. IEEE, 2011.

[104] A. C. Likas and N. P. Galatsanos. A variational approach for bayesian blind image deconvolution. *Signal Processing, IEEE Transactions on*, 52(8):2222–2233, 2004.

[105] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16:346–358, 1979.

[106] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski. Interactive local adjustment of tonal values. In *ACM SIGGRAPH Papers*, pages 646–653, New York, NY, USA, 2006. ACM.

[107] R. Liu and H. Zhang. Mesh segmentation via spectral embedding and contour analysis. *Eurographics*, 26(3), 2007.

[108] O. Livne and A. Brandt. Lean algebraic multigrid (LAMG): Fast graph laplacian solver. *arXiv:1108.0123v1*, 2011.

[109] L. Lucy. An iterative technique for the rectification of observed distributions. *The astronomical journal*, 79:745, 1974.

[110] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on*, 17(1):53–69, 2008.

[111] 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 *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[112] J. J. McAuley, T. S. Caetano, A. J. Smola, and M. O. Franz. Learning high-order MRF priors of color images. In *ICML '06*, pages 617–624, 2006.

[113] J. Miskin and D. J. MacKay. Ensemble learning for blind image separation and deconvolution. 2000.

[114] J. Miskin and D. J. C. MacKay. Ensemble Learning for Blind Image Separation and Deconvolution. In M. Girolani, editor, *Adv. in Independent Component Analysis*. Springer-Verlag, 2000.

[115] A. Mohan, R. Raskar, and J. Tumblin. Agile spectrum imaging: Programmable wavelength modulation for cameras and projectors. *Computer Graphics Forum*, 27(2):709–717, 2008.

[116] R. Molina, A. K. Katsaggelos, and J. Mateos. Bayesian and regularization methods for hyperparameter estimation in image restoration. *Image Processing, IEEE Transactions on*, 8(2):231–246, 1999.

[117] J. Money and S. Kang. Total variation minimizing blind deconvolution with shock filter reference. In *Image and Vision Computing*, number 2, pages 302–314, 2008.

[118] N. Morris, S. Avidan, W. Matusik, and H. Pfister. Statistics of infrared images. In *CVPR*, pages 1–7, 2007.

[119] M. Mørup, K. H. Madsen, and L. K. Hansen. Approximate l_0 constrained non-negative matrix and tensor factorization. In *ISCAS 2008 special session on Non-negative Matrix and Tensor Factorization and Related Problems*, 2008.

[120] J. D. Moulton, J. E. Dendy, Jr., and J. M. Hyman. The black box multigrid numerical homogenization algorithm. *J. Comput. Phys.*, 142:80–108, May 1998.

[121] R. M. Neal et al. Probabilistic inference using markov chain monte carlo methods. 1993.

[122] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389. ACM, 2006.

[123] S. Osher and L. I. Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis*, 27(4):919–940, 1990.

[124] S. Osindero, M. Welling, and G. Hinton. Topographic product models applied to natural scene statistics. *Neural Computation*, 1995.

[125] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *ECCV*, pages IV: 568–580, 2006.

[126] J. Park, M. Lee, M. D. Grossberg, and S. K. Nayar. Multispectral Imaging Using Multiplexed Illumination. In *ICCV*, pages 1–8, Oct 2007.

[127] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):313–318, 2003.

[128] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 23(3):664–672, 2004.

[129] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using a scale mixture of Gaussians in the wavelet domain. *IEEE Trans. Image Processing*, 12(11):1338–1351, November 2003.

[130] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge University Press, 2007.

[131] A. Raj and R. Zabih. A graph-cut problem for general deconvolution problems. In *CVPR*, 2005.

[132] M. Ranzato and G. E. Hinton. Modeling pixel means and covariances using factored third-order boltzmann machines. In *CVPR*, 2010.

[133] W. Richardson. Bayesian-based iterative method of image restoration. *JOSA*, 62:55–59, 1972.

[134] W. H. Richardson. Bayesian-based iterative method of image restoration. *JOSA*, 62(1):55–59, 1972.

[135] B. Rorslett, 2008. `http://www.naturfotograf.com/UV_flowers_list.html`.

[136] S. Roth and M. J. Black. Fields of Experts: A Framework for Learning Image Priors. In *CVPR*, volume 2, pages 860–867, 2005.

[137] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[138] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.

[139] U. Schmidt, K. Schelten, and S. Roth. Bayesian deblurring with integrated noise estimation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2625–2632. IEEE, 2011.

[140] O. Shalvi and E. Weinstein. New criteria for blind deconvolution of nonminimum phase systems (channels). *Information Theory, IEEE Transactions on*, 36(2):312–321, 1990.

[141] O. Shalvi and E. Weinstein. Super-exponential methods for blind deconvolution. *Information Theory, IEEE Transactions on*, 39(2):504–519, 1993.

[142] Q. Shan, J. Jia, and A. Agarwala. High quality motion deblurring from a single image. *SIGGRAPH*, 27, 2008.

[143] L. Shi, Y. Yu, N. Bell, and W.-W. Feng. A fast multigrid algorithm for mesh deformation. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1108–1117, New York, NY, USA, 2006. ACM.

[144] E. Simoncelli and E. H. Adelson. Noise removal via bayesian wavelet coring. In *ICIP*, pages 379–382, 1996.

[145] B. Singh, W. T. Freeman, and D. H. Brainard. Exploiting spatial and spectral image regularities for color constancy. In *Workshop on Statistical and Computational Theories of Vision*, 2003.

[146] A. Spielman, D. Algorithms, graph theory and linear equations in laplacian matrices. *Proceedings of the International Congress of Mathematicians (ICM)*, 2010.

[147] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *arXiv:0803.0929v4*, 2009.

[148] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2006.

[149] J.-L. Starck, M. Elad, and D. L. Donoho. Image decomposition via the combination of sparse representations and a variational approach. *Image Processing, IEEE Transactions on*, 14(10):1570–1582, 2005.

[150] C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Reviews*, 41(3):513–537, Sept. 1999.

[151] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. Poisson matting. In *ACM SIGGRAPH Papers*, pages 315–321, New York, NY, USA, 2004. ACM.

[152] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(6):513 –528, jun 1990.

[153] R. Szeliski. Locally adapted hierarchical preconditioning. *Proceedings of ACM SIGGRAPH*, 2006.

[154] Y.-W. Tai, H. Du, M. S. Brown, and S. Lin. Image/video deblurring using a hybrid camera. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[155] M. F. Tappen, B. C. Russell, and W. T. Freeman. Exploiting the sparse derivative prior for super-resolution and image demosaicing. In *SCTV*, 2003.

[156] J. Telleen, A. Sullivan, J. Yee, O. Wang, P. Gunawardane, I. Collins, and J. Davis. Synthetic shutter speed imaging. *Computer Graphics Forum*, 26(3):591–598, Sept. 2007.

[157] TLVs. *TLVs and BEIs: threshold limit values for chemical substances and physical agents.* American Conference of Governmental Industrial Hygienists, 2001.

[158] S. Toledo, D. Chen, and V. Rotkin. Taucs: A library of sparse linear solvers, 2003.

[159] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

[160] U. Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid.* Academic Press, 2001.

[161] P. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 1990.

[162] P. Vanek. Fast multigrid solver. *Applications of Mathematics*, 40(1):1–20, 1995.

[163] J. Vos. Colorimetric and photometric properties of a 2-deg fundamental observer. *Color Research and Application*, pages 125–128, 1978.

[164] B. A. Wandell. *Foundations of Vision.* Sinauer Associates., 1995.

[165] C. Wang, Y. Yue, F. Dong, Y. Tao, X. Ma, G. Clapworthy, H. Lin, and X. Ye. Nonedge-specific adaptive scheme for highly robust blind motion deblurring of natural imagess. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 22(3), 2013.

[166] O. Wang, J. Davis, E. Chuang, K. Rickard, I. amd de Mesa, and D. Chirag. Video relighting using infrared illumination. *Computer Graphics Forum*, 27, 2008.

[167] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM J. Imaging Sciences*, 1(3):248–272, 2008.

[168] M. Wardetzky, S. Mathur, F. Kalberer, and E. Grinspun. Discrete laplace operators: no free lunch. In *ACM International Conference Proceeding Series*, volume 257, pages 33–37, 2007.

[169] Y. Weiss and W. T. Freeman. What makes a good model of natural images? In *CVPR*, 2007.

[170] E. W. Weisstein. Cubic formula. `http://mathworld.wolfram.com/CubicFormula.html`.

[171] E. W. Weisstein. Quartic equation. `http://mathworld.wolfram.com/QuarticEquation.html`.

[172] O. Whyte, J. Sivic, and A. Zisserman. Deblurring shaken and partially saturated images. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 745–752. IEEE, 2011.

[173] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. In *CVPR*, 2010.

[174] N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series*. 1964.

[175] Wikipedia. Bayes theorem. `http://en.wikipedia.org/wiki/Bayes'_theorem`.

[176] Wikipedia. Central limit theorem. `http://en.wikipedia.org/wiki/Central_limit_theorem`.

[177] Wikipedia. Central limit theorem. `http://en.wikipedia.org/wiki/Cumulant`.

[178] Wikipedia. Kurtosis. `http://en.wikipedia.org/wiki/Kurtosis`.

[179] S. Wright, R. Nowak, and M. Figueredo. Sparse reconstruction by separable approximation. *IEEE Trans. Signal Processing*, page To appear, 2009.

[180] K. Xu, Y. Li, T. Ju, S.-M. Hu, and T.-Q. Liu. Efficient affinity-based edit propagation using k-d tree. In *ACM SIGGRAPH Asia papers*, pages 118:1–118:6, New York, NY, USA, 2009. ACM.

[181] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. *Computer Vision–ECCV 2010*, pages 157–170, 2010.

[182] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution via sparse representation. *PAMI*, page To appear, 2010.

[183] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for l1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168, 2008.

[184] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for l1-minimization with applications to compressed sensing. *SIAM J. on Img. Sciences*, 1(1):143–168, 2008.

[185] Y.-L. You and M. Kaveh. A regularization approach to joint blur identification and image restoration. *Image Processing, IEEE Transactions on*, 5(3):416–428, 1996.

[186] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986. 10.1007/BF01389538.

[187] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum. Image deblurring with blurred/noisy image pairs. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 26, pages 1–10, 2007.

[188] P. D. Zeeuw. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *Journal of Computational and Applied Mathematics*, 33(1):1–27, 1990.

[189] S. C. Zhu and D. Mumford. Prior learning and Gibbs reaction-diffusion. *IEEE Trans. PAMI*, 19(11):1236–1250, 1997.

[190] Y. Zhu, E. Sifakis, J. Teran, and A. Brandt. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph.*, 29:16:1–16:18, April 2010.

[191] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 479–486. IEEE, 2011.