FAST SPACE-VARYING CONVOLUTION IN STRAY LIGHT REDUCTION, FAST MATRIX VECTOR MULTIPLICATION USING THE SPARSE MATRIX TRANSFORM, AND ACTIVATION DETECTION IN FMRI DATA ANALYSIS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Jianing Wei

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2010

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

I would like to thank my advisors Prof. Jan Allebach, Prof. Charles Bouman, Prof. Ilya Pollak, and Dr. Peter Jansson for their support in my graduate research work. Professor Jan Allebach provided great guidance in my research in stray light reduction, advised on my research directions, and showed me general research methodologies that I believe will contribute to my entire career. Professor Charles Bouman motivated me to work on the extremely interesting and impactful problem of fast matrix vector multiplication using the sparse matrix transform, and helped me build a relentless attitude toward solving problems. Professor Ilya Pollak brought me to this great area of image processing, stimulated interesting ideas in fMRI data analysis, and taught me good technical writing skills and presentation skills that I enjoy through my Ph.D. study and will continue to enjoy in my future work. Dr. Peter Jansson shared his expertise in optics which helped a lot in my work on stray light reduction. I would also like to thank my parents Mr. Yena Wei and Ms. Yanqiu Huang for their encouragement in my life. In addition, I thank my girlfriend Wei Di for organizing all the paperwork and providing her support.

TABLE OF CONTENTS

Page

LIST OF FIGURES v

ABSTRACT x

1 Fast Space-varying Convolution in Stray Light Reduction 1

1.1 Introduction 1

1.2 Forward Model Formulation 4

1.3 PSF Parameter Estimation 8

1.3.1 Estimation strategy 8

1.3.2 Estimation result 12

1.4 Stray Light Reduction 14

1.5 Piecewise Isoplanatic Modeling Based on Vector Quantization . . . 16

1.5.1 Isoplanatic partitioning using vector quantization 16

1.5.2 Simulated experiments 19

1.5.3 Real image restoration results 25

1.6 Matrix Source Coding 30

1.6.1 Matrix source coding theory 30

1.6.2 Efficient implementation of matrix source coding 32

1.6.3 Experimental results 37

1.7 Conclusion 41

2 Fast Matrix Vector Multiplication Using the Sparse Matrix Transform . . 44

2.1 Introduction 44

2.2 The SMT for Covariance Estimation and Decorrelation 46

2.3 Fast Matrix-Vector Multiplication using the SMT 49

2.3.1 Matrix Approximation Using the SMT 49

2.3.2 Online computation of matrix-vector product 58

Page

2.4 Experimental Results 58
2.5 Conclusion 60
3 Activation Detection in FMRI Data Analysis 62
3.1 Introduction 62
3.2 Background on FMRI Analysis 65
3.2.1 The GLM method 65
3.2.2 Spatio-temporal analysis 67
3.3 Model Formulation 68
3.4 Activation Detection 69
3.4.1 TV-based image restoration 70
3.4.2 MAP estimation 71
3.5 Model Parameter Estimation 73
3.6 Synthetic Data Experiments 76
3.6.1 HRF Models 76
3.6.2 Simulated Data 78
3.6.3 Synthetic Data Experiment 1: Comparison to Three Other Methods 80
3.6.4 Synthetic Data Experiment 2: Insensitivity to Parameter Variations 87
3.6.5 Synthetic Data Experiment 3: Robustness to the Incorrect HRF Model 87
3.6.6 Synthetic Data Experiment 4: Different HRFs at Different Pixels 89
3.7 Real Data Experiments 93
3.7.1 Real Data Experiment 1: Qualitative Evaluation of Activation Regions 93
3.7.2 Real Data Experiment 2: Insensitivity to Parameter Variations 99
3.8 Conclusion 101
LIST OF REFERENCES 104

LIST OF FIGURES

Figure Page

1.1 Illustration of relation between cross-section of stray light PSF and the location of the corresponding ideal point source. 6

1.2 Geometric illustration of shading effect. 7

1.3 Target locations for obtaining PSF characterization data. 8

1.4 Illustration of how a space-varying PSF is approximated with a locally space-invariant PSF. The center of target region is $(\bar{i}_p, \bar{j}_p)$. We approximate the PSF at pixel (i_p, j_p) by the PSF at pixel $(\bar{i}_p, \bar{j}_p)$. 11

1.5 Horizontal cross sections of model fitting for the model with shading and the model without shading at two position: (a) center position, (c) top right position, and their zoomed in plots on the tails (b) and (d). The thick solid curve is the corrupted signal. The thin solid curve is the fitted signal using the model with shading factor. The dashed curve is the fitted signal using the model without shading factor. The vertical axis is the image intensity normalized by the maximum dynamic range (65535). . 15

1.6 Log magnitude of PSF at different positions in the image: (a) shows the log magnitude of PSF due to a point source at center of the image, (b) shows the log magnitude of PSF due to a point source at corner of the image. 20

1.7 Piecewise isoplanatic patches using rectangular partition and our proposed algorithm. Pixels with the same gray level belong to the same patch. Among the above figures, (a) shows rectangular partition into four patches, (b) shows rectangular partition into eight patches, (c) shows our proposed partition into four patches, (d) shows our proposed partition into eight patches. 21

1.8 Stray light free images. 22

1.9 An example of corrupted image and its restoration results using rigorous space-varying convolution, piecewise isoplanatic approximation with eight patch rectangular partition and proposed eight patch partition. 23

1.10 An example of corrupted image and its restoration results using rigorous space-varying convolution, piecewise isoplanatic approximation with eight patch rectangular partition and proposed eight patch partition. 24

Figure Page

1.11 Horizontal cross sections of corrupted images, restored images, and ideal images at two positions. Dashed lines denote corrupted images. Solid lines denote restored images. Dash dotted lines denote ideal images. The plots in (a) and (b) are for the center position. The plots in (c) and (d) are for the center right position. The vertical axis is the image intensity normalized by the maximum dynamic range (65535). 26

1.12 First example of captured and restored images. 27

1.13 Second example of captured and restored images. 28

1.14 Third example of captured and restored images. 29

1.15 An example of a row of S displayed as an image together with its wavelet transform, and the locations of non-zero entries after quantization: (a) shows the log amplitude map of the row image, (b) shows the log of the absolute value of its wavelet transform, (c) shows the locations of non-zero entries of the corresponding row of $SW_2^t\Lambda_w^{1/2}$. 34

1.16 An illustration of the tree structure of approximation coefficients. Shaded squares indicate that the detail coefficients are not zero at those locations. So we go to the next level recursion. Empty squares indicate that the detail coefficients are zero, and we stop to compute their values. 35

1.17 Pseudo code for fast computation of significant Haar approximation coefficients. Part (a) is the main routine. Part (b) is a subroutine called by (a). The approximation coefficient at level k, location (m, n) is denoted $f_a[k][m][n]$. 36

1.18 Test image for space-varying convolution with stray light PSF. 39

1.19 Experiments demonstrating fast space-varying convolution with stray light PSF: (a) relative distortion versus number of multiplies per output pixel (i.e. rate) using two different matrix source coding strategies: the black solid line shows the curve for our proposed matrix source coding algorithm as described in Eq. (1.23), and the red dashed line shows the curve resulting from direct quantization of the matrix S; (b) comparison of relative distortion versus computation between two different resolutions: the solid black line shows the curve for 256×256, and the dashed blue line shows the curve for 1024×1024. 40

1.20 Example of stray light reduction: (a) shows a captured image, (b) shows the restored image, (c) shows a comparison between captured and restored for a part of the image, (d) shows a comparison between captured and restored for another part of the image. 42

2.1 The structure of a Givens rotation. 47

Figure Page

2.2 An illustration of the SMT operation on input data x. 47

2.3 Flow diagram of approximating matrix-vector multiplication with SMTs and scaling. 50

2.4 An illustration of our greedy optimization approach: we operate on a pair of coordinates in every iteration to minimize the cost function. 55

2.5 Pseudo code for iterative design of second stage SMT decomposition. . 56

2.6 Normalized root mean square error for different amount of computation. 59

2.7 A comparison between the result using our previous method and the algorithm proposed in this work: (a) shows the result from our previous method, (b) shows the result from our currently proposed algorithm, (c) shows the difference image. The convolution results are very close to each other. 60

3.1 Model parameter estimation results. Solid blue lines with diamonds denote true parameter values. Dashed black lines with squares denote their estimates. Error bars are ± one standard deviation of the estimates. . . 76

3.2 Example of TV-based image restoration: (a) Noise and blur free image, (b) Image corrupted with noise and blur, (c) Restoration result. 79

3.3 Performance comparison among different methods on the synthetic data for the Gamma Variate HRF model in Eq. (3.27). Red lines represent ROC curves of proposed algorithm. Black lines represent ROC curves of GLM framework using 5mm FWHM Gaussian kernel for spatial smoothing. Green lines represent ROC curves of GLM framework with 10mm FWHM Gaussian kernel. Blue lines represent ROC curves of GLM framework with 20mm FWHM Gaussian kernel. Error bars on each line indicate ± twice the standard deviation of the correct detection rates for the corresponding method. 81

3.4 Performance comparison among different methods on the synthetic data for the sum of two Gamma functions HRF model in Eq. (3.28). Red lines represent ROC curves of proposed algorithm. Black lines represent ROC curves of GLM framework using 5mm FWHM Gaussian kernel for spatial smoothing. Green lines represent ROC curves of GLM framework with 10mm FWHM Gaussian kernel. Blue lines represent ROC curves of GLM framework with 20mm FWHM Gaussian kernel. Error bars on each line indicate ± twice the standard deviation of the correct detection rates for the corresponding method. 82

Figure Page

3.5 ROC curves of the GLM method for different amount of spatial smoothing. Blue lines represent ROC curves for FWHM=20mm Gaussian kernel. Cyan lines represent ROC curves for FWHM=30mm Gaussian kernel. Magenta lines represent ROC curves for FWHM=40mm Gaussian kernel. Error bars on each line indicate ± twice the standard deviation of the correct detection rates. 84

3.6 Parameter maps produced by our proposed algorithm and the GLM framework with different extent of spatial smoothing: (a) Amplitude map produced by our proposed algorithm, (b) T-statistic map produced by the GLM method with FWHM=5mm Gaussian smoothing kernel, (c) T-statistic map produced by the GLM method with FWHM=10mm Gaussian smoothing kernel, (d) T-statistic map produced by the GLM method with FWHM=20mm Gaussian smoothing kernel. 85

3.7 Activation maps produced by our proposed algorithm and the GLM framework with different extent of spatial smoothing: (a) Activation map produced by our proposed algorithm, (b) Activation map produced by the GLM method with FWHM=5mm Gaussian smoothing kernel, (c) Activation map produced by the GLM method with FWHM=10mm Gaussian smoothing kernel, (d) Activation map produced by the GLM method with FWHM=20mm Gaussian smoothing kernel. The threshold is set such that the number of declared active pixels is equal to the number of truely active pixels. 86

3.8 Comparison of partial ROC curves with different parameter settings in the regularized HRF parameter estimation stage. (a) σ_e is changed. (b) σ_x is changed. (c) σ_δ and σ_τ are changed. (d) q is changed. 88

3.9 Performance comparison on the synthetic data using the Cox waveform as the activation waveform. Gamma Variate HRF model in Eq. (3.27) is used in detection. Red lines represent ROC curves of proposed algorithm. Black lines represent ROC curves of GLM framework using 5mm FWHM Gaussian kernel for spatial smoothing. Green lines represent ROC curves of GLM framework with 10mm FWHM Gaussian kernel. Blue lines represent ROC curves of GLM framework with 20mm FWHM Gaussian kernel. Error bars on each line indicate ± twice the standard deviation of the correct detection rates for the corresponding method. 89

3.10 Two HRFs with different delay time and peak time. 91

3.11 Two frames at the peak times of two HRFs in the synthetic dataset. . . 91

3.12 Comparison of ROC curves between four methods on a synthetic dataset with two activation areas, each with a different HRF. 92

Figure Page

3.13 An example of detection results of the event-related experiment using proposed method and GLM method with comparison to benchmark (block paradigm) result. The threshold is chosen corresponding to a p-value of 0.001. 95

3.14 An example of detection results of the event-related experiment using proposed method and GLM method with comparison to benchmark (block paradigm) result. The threshold for the benchmark experiment is chosen corresponding to a FDR of 0.03. The thresholds for the other methods are chosen such that the number of suprathreshold pixels is the same as that in the benchmark experiment. 98

3.15 An example of detection results of the event-related experiment using proposed method and GLM method with comparison to benchmark (block paradigm) result. The threshold for all methods are automatically determined from k-means clustering. 100

3.16 Detection results for different regularization parameters. In this example, the amplitude parameter map is thresholded corresponding to p-value of 0.001. 101

ABSTRACT

Wei, Jianing Ph.D., Purdue University, May 2010. Fast Space-varying Convolution in Stray Light Reduction, Fast Matrix Vector Multiplication Using the Sparse Matrix Transform, and Activation Detection in FMRI Data Analysis . Major Professor: Jan P. Allebach, Ilya Pollak, and Charles A. Bouman.

In this dissertation, I will address three interesting problems in the area of image processing: fast space-varying convolution in stray light reduction, fast matrix vector multiplication using the sparse matrix transform, and activation detection in functional Magnetic Resonance Imaging (fMRI) data analysis.

In the first topic, we study the problem of space-varying convolution which often arises in the modeling or restoration of images captured by optical imaging systems. Specifically, in the application of stray light reduction, where the stray light point spread function varies across the field of view, accurate restoration requires the use of space-varying convolution. While space-invariant convolution can be efficiently implemented with the Fast Fourier Transform (FFT), space-varying convolution requires direct implementation of the convolution operation, which can be very computationally expensive when the convolution kernel is large.

In this work, we developed two general approaches for the efficient implementation of space-varying convolution through the use of piecewise isoplanatic approximation and matrix source coding techniques. In the piecewise isoplanatic approximation approach, we partition the image into isoplanatic patches based on vector quantization, and use a piecewise isoplanatic model to approximate the fully space-varying model. Then the space-varying convolution can be efficiently computed by using FFT to compute space-invariant convolution of each patch and adding all pieces together. In the matrix source coding approach, we dramatically reduce computation by approximately factoring the dense space-varying convolution operator into a product of

sparse transforms. This approach leads to a trade-off between the accuracy and speed of the operation. The experimental results show that our algorithms can achieve a dramatic reduction in computation while achieving high accuracy.

In the second topic, we aim at developing a fast algorithm for computing matrix-vector products which are often required to solve problems in linear systems. If the matrix size is $P \times P$ and is dense, then the complexity for computing this matrix-vector product is generally $O(P^2)$. So as the dimension P goes up, the computation and storage required can increase dramatically, in some cases making the operation infeasible to compute. When the matrix is Toeplitz, the product can be efficiently computed in $O(P \log P)$ by using the fast Fourier transform (FFT). However, in the general problems the matrix is not Toeplitz, which makes the computation difficult.

In this work, we adopt the concept of the recently introduced sparse matrix transform (SMT) and propose an algorithm to approximately decompose the matrix into the product of a series of sparse matrices and a diagonal matrix. In fact, we show that this decomposition forms an approximate singular value decomposition of a matrix. With our decomposition, the matrix-vector product can be computed with order $O(P)$. Our approach is analogous to the way in which an FFT can be used for fast convolution; but importantly, the proposed SMT decomposition can be applied to matrices that represent space or time-varying operations.

We test the effectiveness of our algorithm on a space-varying convolution example. The particular application we consider is stray-light reduction in digital cameras, which requires the convolution of each new image with a large space-varying stray light point spread function (PSF). We demonstrate that our proposed method can dramatically reduce the computation required to accurately compute the required convolution of an image with the space-varying PSF.

In the third topic, we study the problem of fMRI activation detection, which is to detect which region of the brain is activated when the subject is presented with a specific stimulus. FMRI data is subject to severe noise and some extent of blur. So

recovering the signal and successfully identifying the activated voxels are challenging inverse problems.

We propose a new model for event-related fMRI which explicitly incorporates the spatial correlation introduced by the scanner, and develop a new set of tools for activation detection. We propose simple, efficient algorithms to estimate model parameters. We develop an activation detection algorithm which consists of two parts: image restoration and maximum a posteriori (MAP) estimation. During the image restoration stage, a total-variation (TV) based approach is employed to restore each data slice, for each time index. At the MAP estimation stage, we estimate parameters of a parametric hemodynamic response function (HRF) model for each pixel from the restored data. We employ the generalized Gaussian Markov random field (GGMRF) model to enforce spatial regularity when we compute the MAP estimate of the HRF parameters. We then threshold the amplitude parameter map to obtain the final activation map. Through comparison with the widely used general linear model method, in synthetic and real data experiments, we demonstrate the promise and advantage of our algorithm.

1. FAST SPACE-VARYING CONVOLUTION IN STRAY LIGHT REDUCTION

1.1 Introduction

Space-varying convolution often arises in the modeling or restoration of images captured by optical imaging systems. This is due to the fact that the point spread function (PSF) and/or distortion introduced by a lens typically varies across the field of view [1], so accurate restoration also requires the use of space-varying convolution.

In this work, we will consider the problem of stray light reduction in digital photography. In all optical imaging systems, a small portion of the entering light flux is misdirected to undesired locations in the image plane. Possible causes for this phenomena include but are not limited to: (1) Inter-reflections between optical-element surfaces; (2) scattering from surface imperfections on lens elements; (3) scattering from air bubbles within lens elements; (4) scattering from dust or other particles; (5) diffraction at aperture edges; (6) lens aberrations. The portion due mainly to scattering and undesired reflections is stray light, or lens flare, or veiling glare [2]. Stray light degrades both the image contrast and the color accuracy [3]. The extent of stray light contamination mainly depends on the quality of the lens or in general the camera optical imaging system, the nature of the subject and its surroundings [2]. Stray light due to a strong light source outside the field of view can be efficiently reduced by a lens hood, which shields the lens from glare sources, such as the sun [2]. However, it is difficult to treat the stray light which arises from high luminance areas within the field of view. For example, in photography, back-lighted scenes such as portraits that contain a darker foreground object suffer from poor contrast and reduced detail of the foreground object. Contrast stretching is a traditional approach to reduce the

impact of stray light subjectively. However, it does not correct the underlying flaw of the image and sometimes introduces additional errors.

We solve this problem with a point spread function (PSF) model based approach following [3–7]. Because the optical characteristics of lenses vary across the image field [8], and scattering varies with the incidence angle [9], the stray light PSF must necessarily vary to some degree with the location across the image field. We therefore adopt a space-varying PSF model [6] for the imaging system, estimate the model parameters, and then use a deconvolution algorithm to invert this model. This framework is proved to be effective by both synthetic data results [3] and real data results [6, 7]. Compared to contrast stretching, this method is a systematic way of removing the contamination. Another feature of the stray light PSF is that it has a big 2D support. That is because unlike diffraction and aberration, scattering misdirects light to further locations on the image plane, and its distribution has heavy tails. In addition, in optical imaging systems, illumination at off-axis image points is usually lower than image points on the optical axis. This is usually referred to as shading. Shading can be described by a cosine fourth law [1]. So we incorporate this shading effect inherent in any digital camera optical system into our forward model. Our forward model can be written in matrix form as

$$y = Ax, \tag{1.1}$$

where x is a $P \times 1$ vector representing the ideal image to be restored, y is a $P \times 1$ vector representing the captured image, A is a $P \times P$ matrix representing our digital camera imaging system model which incorporates stray light, diffraction and aberration, and shading, and P is the size of the image in pixels. Thus inverting the forward model, i.e. deconvolution, corrects both the stray light and the shading in digital images. We use Van Cittert's method [10] as a deconvolution algorithm following [5–7]. Van Cittert's method is an iterative restoration algorithm which can be described as

$$\hat{x}^{(k+1)} = \hat{x}^{(k)} + y - A\hat{x}^{(k)}, \tag{1.2}$$

where $\hat{x}^{(k)}$ represents the restored image at the k^{th} iteration. However, due to the large support of the stray light PSF, matrix A is a dense matrix. If A implements space-invariant convolution, i.e. A is a Toeplitz matrix, then $A\hat{x}^{(k)}$ can be computed with order $O(P\log(P))$ using FFT. However, in this case, A implements space-varying convolution. So we need P^2 multiplies to compute $A\hat{x}^{(k)}$, which takes tremendous amount of time for large images. For example, for a 6 million pixel image, we need 3.6×10^{12} multiplies to get the result. This expensive computation makes the stray light reduction algorithm not feasible for real digital camera architecture.

In this dissertation, we propose two algorithms to achieve a significant speed up of space-varying convolution with only a small amount of error. In our first algorithm, We approximate the fully space-varying model with a piecewise isoplanatic model as discussed in [11, 12], and compute the convolution of each patch of the input image with its corresponding PSF, which can be implemented using fast Fourier transform (FFT). The final approximation is obtained by superimposing the convolution results from each patch together. However, the issue of how to find a good 2D partition and the convolution kernel for each patch was not fully addressed in [11, 12]. We propose a locally optimal way to determine the isoplanatic patches and corresponding convolution kernels based on vector quantization. This technique reduces the computation from $O(P^2)$ to $O(P\log(P))$ with a small amount of error.

In our second algorithm, we introduce a novel approach for efficient computation of space-varying convolution, based on the theory of matrix source coding [13]; and we demonstrate how the technique can be used to efficiently implement the stray light reduction for digital cameras. The matrix source coding technique uses the method of lossy source coding which makes a dense matrix sparse. This is done by first decorrelating the rows and columns of a matrix and then quantizing the resulting compacted matrix so that most of its entries become zero. By making the matrix A sparse, we not only save storage, but we also dramatically reduce the computation of the required matrix-vector product.

Our experimental results indicate that, by using this approach, we can reduce the computational complexity of space-varying convolution from $O(P^2)$ to $O(P)$. For digital images exceeding 1 million pixels in size, this makes deconvolution practically possible, and results in deconvolution algorithms that can be computed with 5 to 10 multiplies per output pixel.

1.2 Forward Model Formulation

Our forward model for the digital camera imaging system in Eq. (1.1) is designed to account for the following contributions: (1) diffractive spreading owing to the finite aperture of the imaging device; (2) aberrations arising from non-ideal aspects of device design and device fabrication materials; (3) stray light due to scattering and undesired reflections; (4) decrease of luminance toward the periphery of the image due to shading. Therefore, we rewrite our forward model as follows:

$$\begin{aligned} y &= Ax \\ &= \left((1-\beta)G + \beta S\right)\Lambda x, \end{aligned} \tag{1.3}$$

where G accounts for diffraction and aberration, S accounts for stray light, β represents the weight factor of stray light, Λ is a diagonal matrix that accounts for shading. We now describe our models for G, S, and Λ in the rest of this section.

Following [6], we model the effect of diffraction and aberration as a 2-D Gaussian blur. So the entries of G are described by

$$\begin{aligned} G_{q,p} &= g(i_q, j_q; i_p, j_p) \\ &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i_q - i_p)^2 + (j_q - j_p)^2}{2\sigma^2}\right), \end{aligned} \tag{1.4}$$

where (i_p, j_p) and (i_q, j_q) are the 2D positions of the input and output pixel locations, measured relative to the center of the camera's field of view, and σ is the width of the Gaussian kernel.

We model the stray light contamination as convolution with a point spread function (PSF). So each column of matrix S is a PSF due to the corresponding point

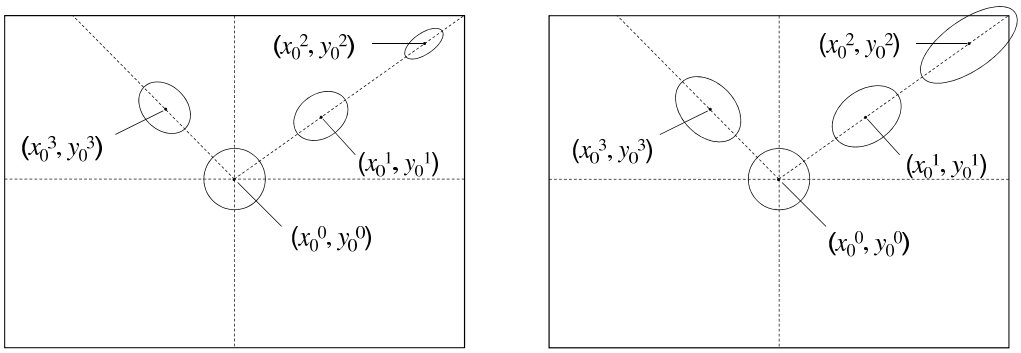
source. To construct a stray light PSF model suited for application to digital cameras, as in [6, 7], we want a normalized rotationally invariant two-dimensional functional form that exhibits smooth behavior over the entire field of view of the camera and has elliptical cross sections when arbitrary slices are taken of it. Owing to symmetry considerations, the elliptical cross section should become circular when it is located on the optical axis. In addition, real world digital camera PSF must accommodate multiple scattering. The distribution of power for single scattering can be approximated by Harvey-Shack approximation [9]. This distribution was further approximated by hyperbolic functions for paraxial rays [6], whose variance is infinite. The resulting PSF, due to multiple scattering, is the convolution of the distribution of each single scattering. From central limit theorem [14], the convolution of distributions with infinite variance approaches Cauchy distribution. So the stray light PSF should have the form of Cauchy function as in [6]. However, the integral of the stray light PSF in [6] is not constant as the location of the point source changes over the field of view. This behavior is due to a limitation of the model. In fact, the luminance level in the image will decrease toward the periphery of the field of view due to shading. But this should be explicitly accounted for in the model. Thus, we believe that a better approach is to first normalize the PSF and then apply the shading correction using matrix Λ to the normalized result. Taking all the criteria mentioned above, the entries of S are expressed as

$$\begin{aligned} S_{q,p} &= s(i_q, j_q; i_p, j_p) \\ &= \frac{1}{\kappa} \frac{1}{\left(1 + \frac{1}{i_p^2+j_p^2}\left(\frac{(i_q i_p + j_q j_p - i_p^2 - j_p^2)^2}{(c+a(i_p^2+j_p^2))^2} + \frac{(-i_q j_p + j_q i_p)^2}{(c+b(i_p^2+j_p^2))^2}\right)\right)^{\alpha}}, \end{aligned} \tag{1.5}$$

where $s(i_q, j_q; i_p, j_p)$ represents our stray light PSF model, (a, b, c, α) are parameters of this PSF model, κ is a normalizing constant that makes $\int_{i_q}\int_{j_q} s(i_q, j_q; i_p, j_p) di_q dj_q = 1$. An analytic expression for κ is

$$\kappa = \frac{\pi}{\alpha - 1}\left(c + a(i_p^2 + j_p^2)\right)\left(c + b(i_p^2 + j_p^2)\right).$$

Among the model parameters, a and b govern the rate of change of the respective elliptical cross-section widths with respect to the distance from the optical axis center,



(a) $a < 0$, $b < 0$ (b) $a > 0$, $b > 0$

Fig. 1.1. Illustration of relation between cross-section of stray light PSF and the location of the corresponding ideal point source.

c governs the circular diameter of the PSF when it is centered at the optical axis, and α controls the rate of decrease. When a and b are both zero, the cross-section of the PSF becomes circular and this space-varying model becomes space-invariant. Figure 1.1 shows how the cross section of the stray light PSF behaves as a function of the angle and distance of the ideal point source from the origin.

In any simple optical imaging system, illumination falls off toward the periphery of the image according to the cosine fourth law [1]. We name this shading. Figure 1.2 shows a geometric illustration of the shading effect. We incorporate this shading effect into our model by multiplying the input image x with a diagonal matrix Λ. The entries of the matrix Λ are expressed as

$$\Lambda_{p,p} = \frac{D^4}{(i_p^2 + j_p^2 + D^2)^2}, \tag{1.6}$$

where D is the distance from the exit pupil to the image plane. This completes our model of the digital camera imaging system described in Eq. (1.3). We have all together seven parameters $(a, b, c, \alpha, \sigma, \beta, D)$ in the entire model. Thus, if we want to remove the stray light and shading from the captured image, we have to estimate these parameters and invert this model to recover the ideal scene. We will discuss parameter estimation in the next section.

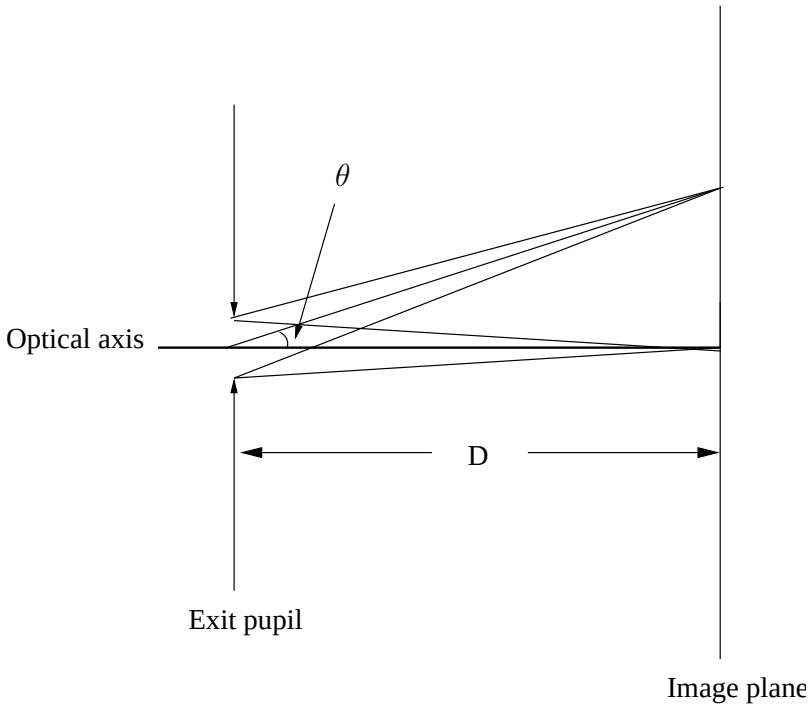


Fig. 1.2. Geometric illustration of shading effect.

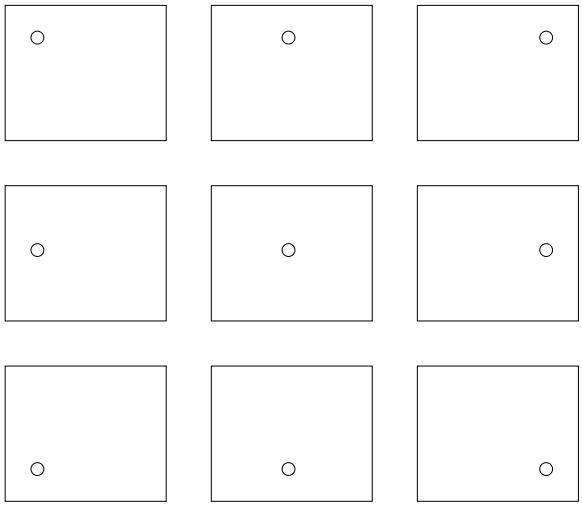


Fig. 1.3. Target locations for obtaining PSF characterization data.

1.3 PSF Parameter Estimation

1.3.1 Estimation strategy

We use the apparatus described by Bitlis *et. al.* [6] to obtain the PSF characterization data in a dark room. We use a digital camera to acquire images from a uniform light source located at nine positions in the object plane (shown in Fig. 1.3), thereby capturing the space-varying property of our PSF. At each position, we take raw images at two different exposure settings, i.e. normal exposure and over exposure. In saturated areas of the overexposed image, we use the pixel values from the normally exposed image for the corrupted image. In other areas, we scale the overexposed image by the factor of the ratio between the two exposure times to obtain pixel values for the corrupted image. This strategy enables us to construct a corrupted image scaled to the same level as the normally exposed image but with a lower noise level in dark areas. This scaling is valid since image intensity is proportional to the exposure time. In order to construct the ideal image, we threshold each image with normal exposure at eighty percent of its maximum intensity to obtain a binary image. Then pixels at the boundary of the target are assigned intermediate values to account for

partial pixel illumination [6]. We also scale these constructed images with the same factor for all positions such that at the center position the image's total energy is the same as the corrupted image. This forms a set of ideal images, which we consider as the ground truth. After obtaining the corrupted images and ideal images, we can estimate the PSF parameters from these data. Note that the image data is in linear space, not gamma corrected space.

Our strategy for estimating PSF parameters is to minimize a cost function

$$\begin{aligned} \xi &= \sum_{k=1}^{K} (\hat{y}_k - y_k)^2 \\ &= \sum_{k=1}^{K} (A(a, b, c, \alpha, \sigma, \beta, D)x_k - y_k)^2, \end{aligned} \tag{1.7}$$

where K is the number of positions used for estimation ($K = 9$ in our case), y_k is the k^{th} observed image, x_k is the k^{th} ideal image, $A(a, b, c, \alpha, \sigma, \beta, D)$ is the matrix which characterizes the digital camera imaging system. So this cost function represents the difference between the actual observed images and the ones computed using our forward model. Note that since stray light contamination is more observable outside the object, we do not take the entire image when computing the cost. Instead, we mainly consider pixels outside the target and only take a few pixels inside the target region to account for shading. Our final estimates of parameters are computed as follows:

$$(\hat{a}, \hat{b}, \hat{c}, \hat{\alpha}, \hat{\sigma}, \hat{\beta}, \hat{D}) = \arg \min_{(a,b,c,\alpha,\sigma,\beta,D)} \xi. \tag{1.8}$$

However, since our PSF is space-varying, directly computing the cost function using Eq. (1.7) is quite expensive and inefficient: evaluating the stray light PSF at one output pixel for one input pixel using Eq. (1.5) requires at least 6 real multiplies and 1 power operation, not to mention the other terms. If we denote M to be the number of real operations required for evaluating the PSF at one output pixel for one input pixel, N_{non0} to be the number of nonzero pixels in an ideal image, N_{out} to be the number of pixels on which to compute the cost, then we need $MN_{non0}N_{out}$ real operations to compute the cost for one position among a total of 9 positions.

In this work, we compute the cost function over the entire region through approximation. Since the target region is relatively small compared to the size of the image, we can assume that the stray light PSF $s(i_q, j_q; i_p, j_p)$ within the target region is space-invariant at each individual location and is equal to the PSF at the center of the target region. As is illustrated in Fig. 1.4, at pixel (i_p, j_p) instead of using the PSF indicated by the solid ellipse, we use the PSF depicted by the dashed ellipse, which is a shifted copy of the PSF at pixel $(\bar{i}_p, \bar{j}_p)$. In fact, this is equivalent to the approximating the space-varying convolution of stray light PSF with the ideal image with a space-invariant convolution:

$$\begin{aligned}
\hat{y}^{STR} &= \iint_{i_q, j_q} s(i_q, j_q; i_p, j_p) x(i_p, j_p) di_q dj_q \\
&= \iint_{i_q, j_q} \frac{1}{z} \frac{1}{\left(1 + \frac{1}{i_p^2 + j_p^2}\left(\frac{((i_q - i_p)i_p + (j_q - j_p)j_p)^2}{(c + a(i_p^2 + j_p^2))^2} + \frac{((j_q - j_p)i_p - (i_q - i_p)j_p)^2}{(c + b(i_p^2 + j_p^2))^2}\right)\right)^{\alpha}} x(i_p, j_p) di_q dj_p \\
&\approx \iint_{i_q, j_q} \frac{1}{z} \frac{1}{\left(1 + \frac{1}{i_p^2 + j_p^2}\left(\frac{((i_q - i_p)\bar{i}_p + (j_q - j_p)\bar{j}_p)^2}{(c + a(i_p^2 + j_p^2))^2} + \frac{((j_q - j_p)\bar{i}_p - (i_q - i_p)\bar{j}_p)^2}{(c + b(i_p^2 + j_p^2))^2}\right)\right)^{\alpha}} x(i_p, j_p) di_q dj_p
\end{aligned} \tag{1.10}$$

where $\bar{i}_p$ and $\bar{j}_p$ are both constants, so the last line is in the form of space-invariant convolution. Once the PSFs at our target region are space-invariant, we can use fast Fourier transform (FFT) to compute the convolution [15]. Then we need only $MN_{out} + 2N_{out} \log N_{out}$ real operations to compute the cost for one position. So the ratio of number of real operations between exact computation and our approximation approach is $\frac{MN_{non0}}{(M + 2 \log N_{out})}$. In a typical PSF characterization experiment, N_{non0} is on the order of 10^4 and $\log N_{out}$ is on the order of 10. So this approximation leads to a significant reduction in computation on the order of 10^4. Besides, since our target is relatively small compared to the size of the image, our locally space-invariant approximation has high accuracy. If we let $\hat{y}_k^{STR}$ be the exact space-varying convolution of the kth ideal image with the stray light PSF, and let $\tilde{y}_k^{STR}$ be the corresponding result due to locally space-invariant approximation, then we can measure the accuracy of

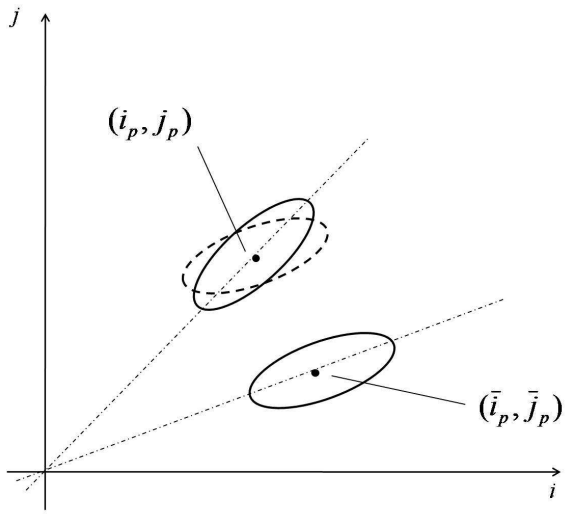


Fig. 1.4. Illustration of how a space-varying PSF is approximated with a locally space-invariant PSF. The center of target region is $(\bar{i}_p, \bar{j}_p)$. We approximate the PSF at pixel (i_p, j_p) by the PSF at pixel $(\bar{i}_p, \bar{j}_p)$.

Table 1.1
NRMSE of locally shift invariant approximation

Position	center	top left	top center	top right	center right
NRMSE	0.0001	0.0014	0.0010	0.0015	0.0011
Position	center left	bottom left	bottom center	bottom right	
NRMSE	0.0009	0.0011	0.0007	0.0013	

this locally shift invariant approximation using normalized root mean square error (NRMSE) defined by the following equation:

$$\text{NRMSE}_k = \sqrt{\frac{\|\hat{y}_k^{STR} - \tilde{y}_k^{STR}\|_2^2}{\|\hat{y}_k^{STR}\|}}. \tag{1.10}$$

For a typical set of stray light PSF parameter, $a = -2.68 \times 10^{-5}$, $b = -8.04 \times 10^{-5}$, $c = 2.61 \times 10^{-3}$, $\alpha = 1.3$, the NRMSE at 9 positions in our dark room experiments in Fig. 1.3 is shown in Table 1.1. We can see that the largest error is only 0.15%, which means that we have an accurate approximation. We use this locally space-invariant approximation to compute the cost function in Eq. (1.7) for all 9 positions and estimate PSF parameters by minimizing the cost. The minimization is accomplished with a subspace trust region method [16] using MATLAB optimization toolbox [17].

1.3.2 Estimation result

We conduct experiments with a camera model available on the market: Olympus SP-510 UZ [1]. The camera settings are: aperture f8.0, ISO 50, normal exposure time 1/25s, over exposure time 2s. The estimated parameters for the model with shading factor and without shading factor are listed in Table 1.2. Note that σ and D are both in units of pixels. The entire image size is 2310×3088 for Olympus SP-510 UZ. From the estimation results, we have the following observations: (1) The diffraction-aberration PSF is very sharp considering the size of the image (σ is 3 pixels); (2) The

[1] Olympus America Inc., PA 18034

Table 1.2
PSF parameters.

Parameter	a (mm^{-1})	b (mm^{-1})	c (mm)	α	σ (mm)	β	D (mm)
shading	-1.65×10^{-5}	-5.35×10^{-5}	1.76×10^{-3}	1.31	5.71×10^{-3}	0.39	17.26
no shading	-2.68×10^{-5}	-6.43×10^{-5}	1.64×10^{-3}	1.30	5.76×10^{-3}	0.40	

shading factor $D^4/(x^2+y^2+D^2)^2$ ranges from 0.9184 to 1, so generally it is very close to 1; (3) The stray light component is relatively small (39%).

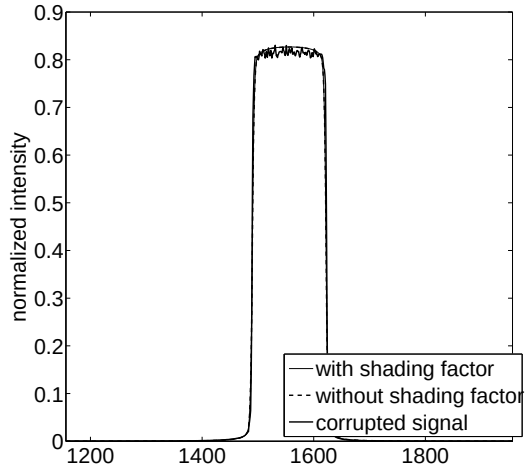
We show the cross-sections of model fitting at two positions for two different models, with shading factor and without shading factor, in Fig. 1.5. The fitted curves are obtained by substituting the estimated parameters into the PSF and computing the corrupted image by rigorous superposition using Eq. (1.3). The thick solid curve is the corrupted signal. The thin solid curve is the fitted signal using the model with shading factor. The dashed curve is the fitted signal using the model without shading factor. We can clearly see that at the top right position the model with shading factor fits much better than the model without shading factor. We define the root mean square (RMS) error between fitted signal and corrupted signal at the jth target position as

$$\mathrm{RMS}_k = \sqrt{\frac{1}{P_k}\|A(\hat{a},\hat{b},\hat{c},\hat{\alpha},\hat{\sigma},\hat{\beta},\hat{D})x_k - y_k\|_2^2}, \tag{1.11}$$

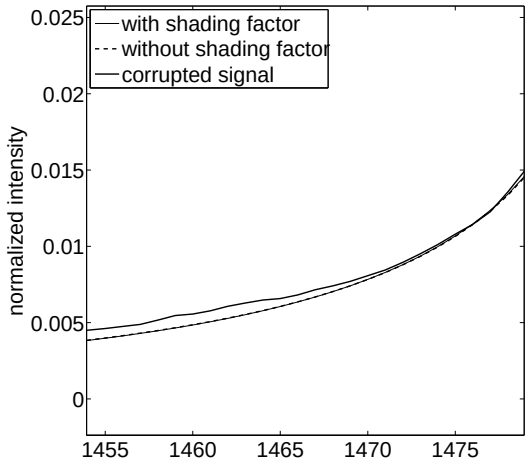
where P_k is the number of pixels used to compute the cost at position k. At the top right position, the RMS error is 0.2980×10^{-3} for the model with shading factor, and is 0.3295×10^{-3} for the model without shading factor. This difference in RMS error is primarily due to the fitting at the center of the target, since from Fig. 1.5 (d) we can see that the tails of the fitted signal for the model with shading and without shading sit on top of each other. The difference in RMS error is not very large, because only a small number of pixels are chosen at the center of the target when computing the cost function. At the center position, the RMS error is 0.1814×10^{-3} for the model with shading factor and 0.1817×10^{-3} for the model without shading factor. The difference in RMS error is very small at the center, because the center position has the least amount of shading.

1.4 Stray Light Reduction

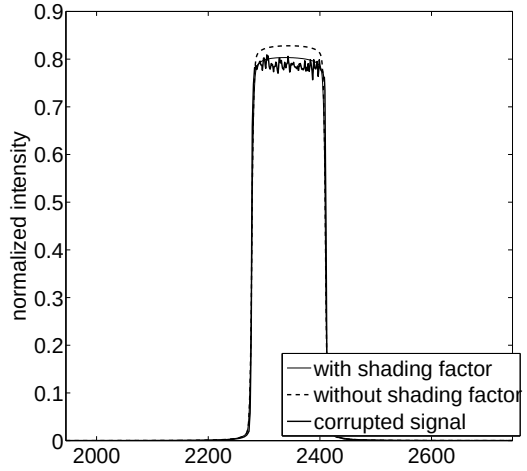
After obtaining parameters for the forward model, we apply Van Cittert's iterative image restoration method [10, 18] in Eq. (1.2) to recover the original image x from



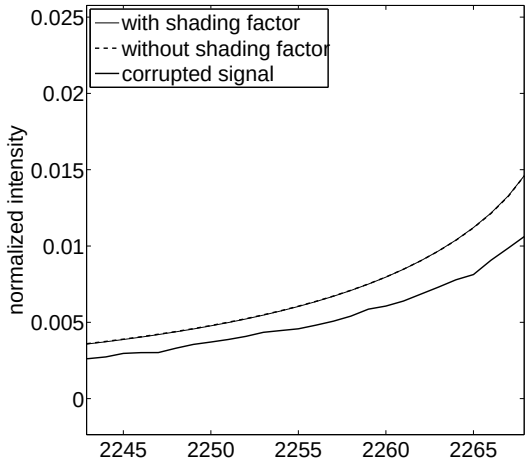
(a) Center position.



(b) Zoomed in plot at center position.



(c) Top right position.



(d) Zoomed in plot at top right position.

Fig. 1.5. Horizontal cross sections of model fitting for the model with shading and the model without shading at two position: (a) center position, (c) top right position, and their zoomed in plots on the tails (b) and (d). The thick solid curve is the corrupted signal. The thin solid curve is the fitted signal using the model with shading factor. The dashed curve is the fitted signal using the model without shading factor. The vertical axis is the image intensity normalized by the maximum dynamic range (65535).

the corrupted image y. We initialize $\hat{x}^{(0)}$ to be the observed (corrupted) image y, and only apply one iteration of Van Cittert's formula. So the restoration formula becomes

$$\hat{x} = 2y - Ay. \tag{1.12}$$

In addition, from our estimation result in Sec. 1.3.2, we can see that the diffraction-aberration PSF is very sharp considering the image size and can be approximated by a delta function. Thus our restoration strategy is given by

$$\hat{x} \approx 2y - (1-\beta)\Lambda y - \beta S\Lambda y \tag{1.13}$$

We show in Appendix 3.8 that the result of our restoration formula is a good estimate of the original image by using the facts that the shading factor is close to 1 and that the stray light part represents only a small portion of the entire PSF.

Note that in order to restore the image using Eq. (1.13), we need to compute $S\Lambda y$. Remember that Λ is a diagonal matrix, so computing Λy is easily implemented as scaling on each pixel of the image y. So we first compute $\tilde{y} = \Lambda y$. However, matrix S is a large and dense matrix. In addition, it is not a Toeplitz matrix, so FFT cannot be used to compute the matrix-vector product. Therefore exact computation of $S\tilde{y}$ needs P^2 real multiplies, which is too expensive for implementation on real camera architecture. In the next two sections, we are going to present two algorithms for efficiently computing

$$\tilde{x} = S\tilde{y}. \tag{1.14}$$

1.5 Piecewise Isoplanatic Modeling Based on Vector Quantization

1.5.1 Isoplanatic partitioning using vector quantization

The idea of our algorithm is to partition the input image into isoplanatic patches and compute the convolution of each patch with a space-invariant PSF, that we call the representative PSF for this patch, using FFT. Then the convolution results from all the patches are superimposed together to form the result. This idea is very similar

to the overlap-and-add method [19,20], except that the overlap and the size of output corresponding to each input patch is the same as the size of entire image, because the patches may have irregular shapes and the stray light PSF has a large 2D support. In addition, the overlap-and-add method presented in [19, 20] fixed the patch shape to be rectangular, whereas in our work, we don't have this constraint. This piecewise isoplanatic approximation of space-varying systems was discussed in [11,12]. However, the algorithm for isoplanatic partitioning provided in [11] only handles one dimensional scenario. In this research, we adopt a method widely used in vector quantization for determining the isoplanatic partition and computing representative convolution kernels in two dimensional cases. With the piecewise isoplanatic approximation, Eq. (1.14) can be written as follows:

$$\begin{aligned} \tilde{x}(i_q, j_q) &= \sum_{i_p, j_p} s(i_q, j_q; i_p, j_p)\tilde{y}(i_p, j_p) \\ &\approx \sum_{l=1}^{L} \sum_{(i_p, j_p)\in R_l} s_l(i_q - i_p, j_q - j_p)\tilde{y}(i_p, j_p), \end{aligned} \tag{1.15}$$

where L is the total number patches, s_l is the representative PSF used for the lth patch, R_l denotes the lth patch, and it satisfies $\bigcup_{l=1}^{L} R_l = \Omega$, with Ω being the entire image area, and $R_{l_1} \cap R_{l_2} = \phi$ for $l_1 \neq l_2$. Note that the last line of Eq. (1.15) can now be computed using FFT, since for each patch l, it is a space-invariant convolution. So the computation of space-varying convolution is reduced from P^2 real multiplies to $2LP \log P$ real multiplies. For a 256×256 image with 8 patch partition, it results in a 256 fold reduction in computation. For a 1024×1024 image with 8 patch partition, it results in a 3277 fold reduction in computation. As a trade-off, we need to store the partition result and the representative PSF for each patch. We now show how to partition the image and how to determine the representative PSF for a particular patch.

Assuming the same convolution kernel for all pixels in an isoplanatic region is equivalent to quantizing the convolution kernels at every pixel location in that region to the representative convolution kernel. So it is a vector quantization problem.

However, the distinction is that the centroid or codeword of a patch may not be represented by any pixel in that patch, but can be fully described by a convolution kernel. In addition, the appropriate definition of distance between two pixels in the quantization process is the mean square difference between their corresponding convolution kernels, since we want to achieve the minimum distortion in PSF. If we let $\boldsymbol{s}_{l,m}$ denote the vectorized convolution kernel at the mth pixel in the lth region, and let $\boldsymbol{s}_l$ denote the vectorized representative convolution kernel for the lth region, then we can define the distortion of quantization at the lth region as

$$e_l = \frac{1}{N_l} \sum_{m=1}^{N_l} \|\boldsymbol{s}_{l,m} - \boldsymbol{s}_l\|_2^2, \tag{1.16}$$

where N_l is the number of pixels in the lth region. For an image of P pixels, the objective is to minimize the average distortion defined by

$$\begin{aligned} d &= \frac{1}{P} \sum_{l=1}^{L} e_l N_l \\ &= \frac{1}{P} \sum_{l=1}^{L} \sum_{m=1}^{N_l} \|\boldsymbol{s}_{l,m} - \boldsymbol{s}_l\|_2^2. \end{aligned} \tag{1.17}$$

The average distortion d reaches minimum, when $\boldsymbol{s}_l = \frac{1}{N_l} \sum_m^{N_l} \boldsymbol{s}_{l,m}$ for $l = 1, \ldots, L$. Thus once the partition is determined, the optimal representative convolution kernel for the lth region is the average convolution kernel for all pixels in that region.

After defining the distortion metric, we now use vector quantization to find a good partition. A number of methods have been proposed for vector quantization, including median split algorithm [21], pairwise nearest neighbor algorithm [22], LBG algorithm [23], and some other algorithms designed to achieve global optimal solution [24] and [25]. We adopt LBG algorithm because it produces good locally optimal solutions and it is a widely used technique [26]. Our algorithm is described as follows:

1. Initialization: Set the number of isoplanatic patches L, and a distortion threshold $\epsilon \geq 0$. Partition the image into uniform rectangular regions as an initial

partition. Compute the initial representative convolution kernels (codewords) by averaging the convolution kernels in each patch. .

2. For each pixel in the image, classify it into the region with the closest codeword to that pixel based on the optimum nearest-neighbor rule [23].

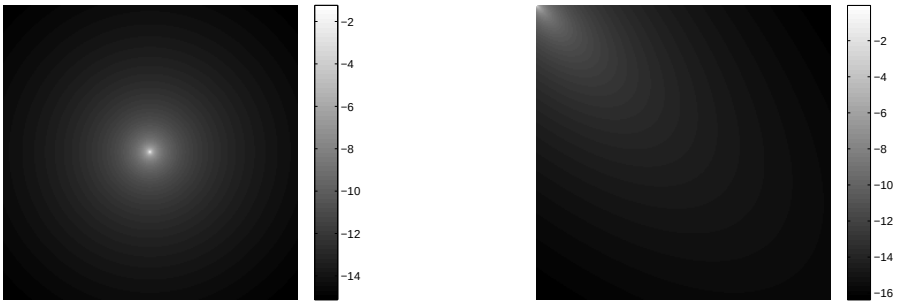
3. Compute the average distortion d_i in the current iteration i. If $(d_{i-1}-d_i)/d_i \leq \epsilon$, stop. Otherwise, continue.

4. Update the codeword of each patch by the average convolution kernel in that region, since this achieves minimum distortion. Update i with $i+1$. Go to step 2.

1.5.2 Simulated experiments

We conduct simulation experiments over images of size 256×256. We follow our forward model in Eq. (1.3) to construct stray light contaminated images. In order to emphasize our piecewise isoplanatic partition strategy, we ignore the diffraction-aberration term and the shading factor, which means that matrices G and Λ are both identity matrices in this simulation. We choose the weight of stray light to be $\beta = 0.3$. We choose the parameters of the stray light PSF to be: $a = -10^{-4}\text{mm}^{-1}$, $b = -4 \times 10^{-4}\text{mm}^{-1}$, $c = 5.76 \times 10^{-3}\text{mm}$, and $\alpha = 1.1063$. The log magnitude of the PSFs due to a point source at the image center and a point source at the corner of the image are shown in Fig. 1.6(a) and (b) respectively. Space-varying behavior of the PSF can be observed from this figure. We partition the image into 4 and 8 patches with rectangular partition and our proposed approach respectively and show the results in Fig. 1.7. Pixels with the same gray level belong to the same isoplanatic patch. Note that pixels symmetric to the origin (the center of the image) have exactly the same convolution kernel. So the partitioning is symmetric to the origin.

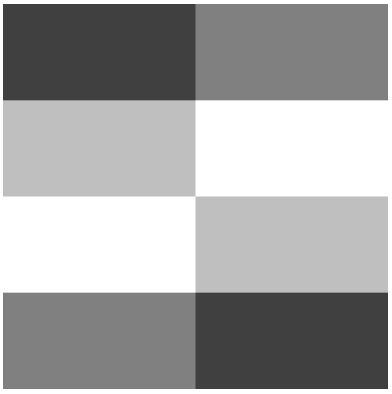
In our experiment, we take natural images as the ideal, stray light free images and apply our forward model to construct the captured images. Figure 1.8 shows



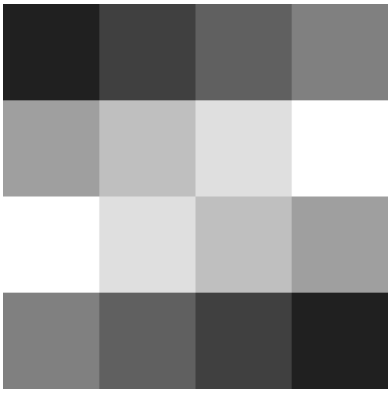
(a) PSF at center of the image.

(b) PSF at corner of the image.

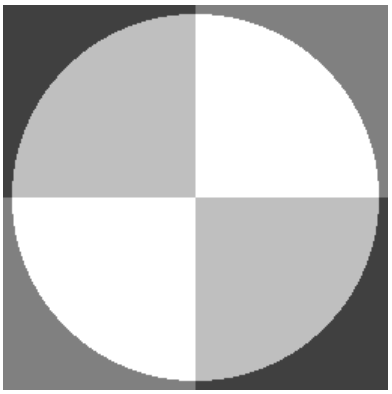
Fig. 1.6. Log magnitude of PSF at different positions in the image: (a) shows the log magnitude of PSF due to a point source at center of the image, (b) shows the log magnitude of PSF due to a point source at corner of the image.



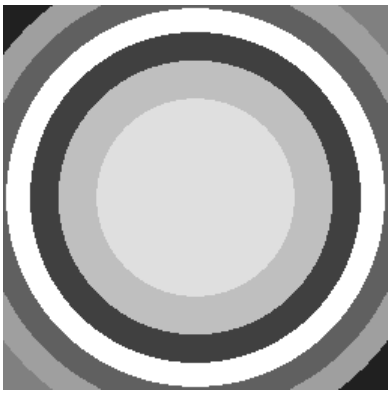
(a) Four patches with rectangular partition.



(b) Eight patches with rectangular partition.



(c) Four patches with proposed partition.



(d) Eight patches with proposed partition.

Fig. 1.7. Piecewise isoplanatic patches using rectangular partition and our proposed algorithm. Pixels with the same gray level belong to the same patch. Among the above figures, (a) shows rectangular partition into four patches, (b) shows rectangular partition into eight patches, (c) shows our proposed partition into four patches, (d) shows our proposed partition into eight patches.



(a) Ideal image 1. (b) Ideal image 2.

Fig. 1.8. Stray light free images.

the ideal images. Figures 1.9 and 1.10 show the corrupted images, and the corresponding restoration results using rigorous computation of space-varying convolution (Fig. 1.9(b) and Fig. 1.10(b)), piecewise isoplanatic approximation with eight patch rectangular partition (Fig. 1.9(c) and Fig. 1.10(c)), and piecewise isoplanatic approximation with our proposed eight patch partition (Fig. 1.9(d) and Fig. 1.10(d)). Note that in our restoration experiments, we assume that the model parameters $\{a, b, c, \alpha, \beta\}$ are known or accurately estimated. We compare the performance of the regular rectangular partition and our proposed partition by computing the signal to noise ratio (SNR) as

$$\text{SNR} = 10 \log_{10} \left(\frac{\|x\|_2^2}{\|x - \hat{x}\|_2^2} \right).$$

From the results in Fig. 1.9, we can see that our proposed approach increases the SNR from 24.86dB in the corrupted image to 41.39dB in the restored image, and is close to the SNR=43.52dB produced by rigorous computation, whereas rectangular partition produces an SNR=35.90dB, which is more than 5dB lower than our proposed approach. Similar results are observed in Fig. 1.10. Our proposed algorithm produces an SNR that is about 8dB higher than rectangular partition.



(a) Corrupted image
SNR=24.86dB



(b) Rigorous restoration
SNR=43.52dB



(c) Restoration with rectangular partition
SNR=35.90dB



(d) Restoration with proposed partition
SNR=41.39dB

Fig. 1.9. An example of corrupted image and its restoration results using rigorous space-varying convolution, piecewise isoplanatic approximation with eight patch rectangular partition and proposed eight patch partition.



(a) Corrupted image.
SNR=26.04dB



(b) Rigorous restoration.
SNR=47.09dB



(c) Restoration with rectangular partition.
SNR=36.39dB



(d) Restoration with proposed partition.
SNR=44.10dB

Fig. 1.10. An example of corrupted image and its restoration results using rigorous space-varying convolution, piecewise isoplanatic approximation with eight patch rectangular partition and proposed eight patch partition.

1.5.3 Real image restoration results

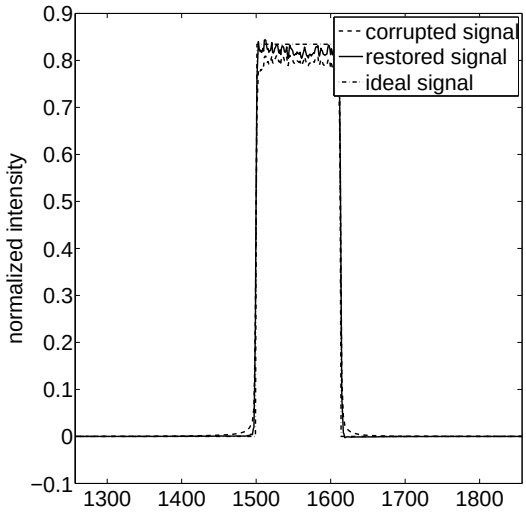
Darkroom image restoration results

Similar to the experiments described in Sec. 1.3, we took another set of darkroom images to test our stray light and shading reduction algorithm. Again, we construct ideal and corrupted images according to the method in Sec. 1.3. We show in Fig. 1.11 a horizontal cross section of the restoration results of corrupted images at the center position and center right position.

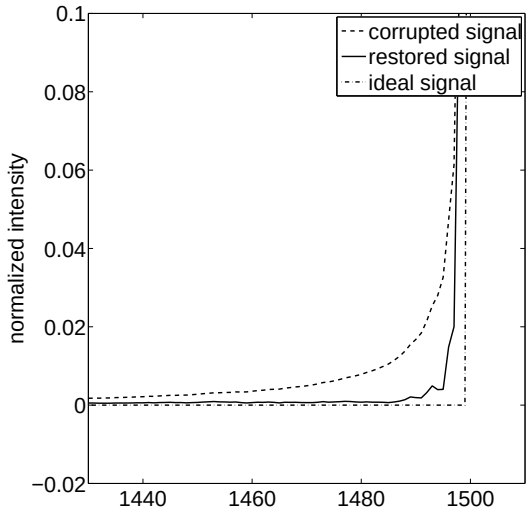
From Fig. 1.11 (c), we can clearly observe the reduction of shading. From the zoomed in plots in Fig. 1.11 (b) and (d), we can see that the stray light at the wings outside the target has been effectively reduced.

Outdoor image restoration results

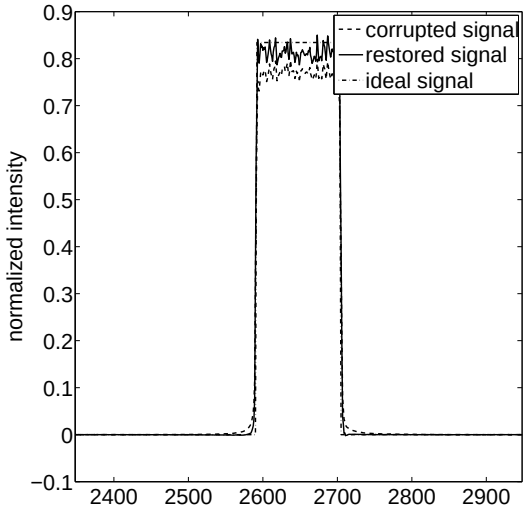
We took outdoor pictures with an Olympus SP-510 UZ camera. The settings were kept the same as in the PSF estimation experiment, except for the shutter speed. We show three examples of the captured images by the Olympus camera and corresponding restoration results in Figs. 1.12, 1.13, and 1.14. We also zoom in part of the images (enclosed by the black box) to better display the effect of stray light reduction. In Fig. 1.12, the glare source is the road which has a relatively large amount of light reflection. From the zoomed in view of the grass, we can see that the saturation of the grass is increased after restoration, i.e. the grass becomes greener, and more details of the grass is revealed. In Fig. 1.13, we observe a veil on top of the building and trees before restoration. After restoration, that veil is removed, and the grass color becomes more saturated. In Fig. 1.14, the glare source is the cloud in the background. From the zoomed in view, it is clear that stray light reduces the contrast on the tree branches. Our restoration algorithm recovers the lost details and reduced contrast on the tree branches. These real-world digital image examples show the capability of our algorithm to reduce the amount of stray light.



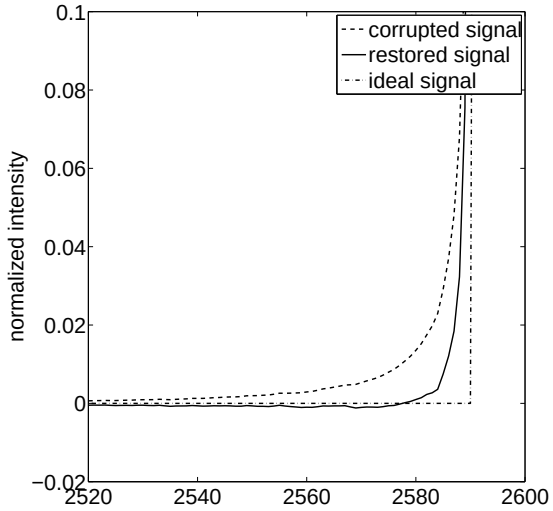
(a) Restoration result at the center position.



(b) Zoomed in plots at the center position.



(c) Restoration result at the center right position.



(d) Zoomed in plots at the center right position.

Fig. 1.11. Horizontal cross sections of corrupted images, restored images, and ideal images at two positions. Dashed lines denote corrupted images. Solid lines denote restored images. Dash dotted lines denote ideal images. The plots in (a) and (b) are for the center position. The plots in (c) and (d) are for the center right position. The vertical axis is the image intensity normalized by the maximum dynamic range (65535).



(a) Captured image.



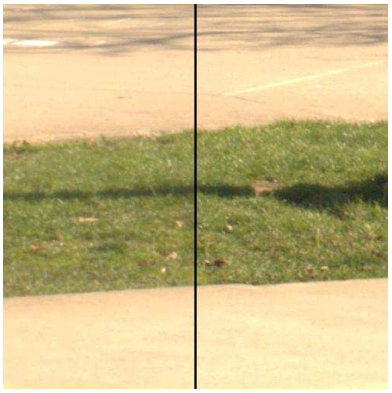
(b) Restored image.



(c) Part of the captured image.



(d) Part of the restored image.



(e) Captured | restored.

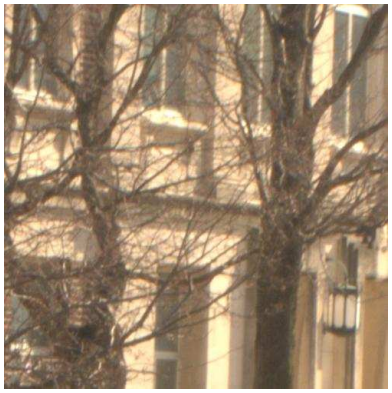
Fig. 1.12. First example of captured and restored images.



(a) Captured image.



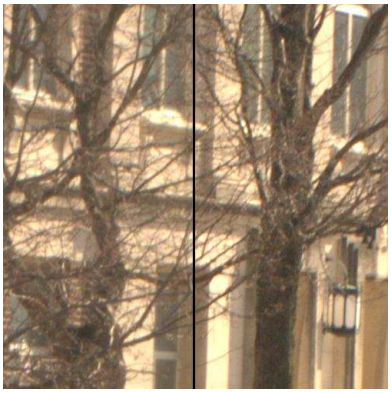
(b) Restored image.



(c) Part of the captured image.



(d) Part of the restored image.



(e) Captured | restored.

Fig. 1.13. Second example of captured and restored images.



(a) Captured image.



(b) Restored image.



(c) Part of the captured image.



(d) Part of the restored image.



(e) Restored \ captured.

Fig. 1.14. Third example of captured and restored images.

1.6 Matrix Source Coding

1.6.1 Matrix source coding theory

Our strategy for speeding up the computation of Eq. (1.14) is to compress the matrix S, such that it becomes sparse. In order to find a sparse representation of the matrix S, we use the techniques of lossy source coding. Let $[S]$ represent the quantized version of S. Then $S = [S] + \delta S$, where δS is the quantization error. This error δS results in some distortion in $\tilde{x}$ given by

$$\delta\tilde{x} = \delta S\tilde{y}. \tag{1.18}$$

A conventional distortion metric for the source coding of the matrix S is $\|\delta S\|^2$. However, this may be very different from the squared error distortion in the matrix-vector product result which is given by $\|\delta\tilde{x}\|^2$. Therefore, we would like to relate the distortion metric for S to the quantity $\|\delta\tilde{x}\|^2$.

It has been shown that if the image $\tilde{y}$ and the quantization error δS are independent, then the distortion in $\tilde{x}$ is given by [13, 27]

$$\mathrm{E}\left[\|\delta\tilde{x}\|^2|\delta S\right] = \|\delta S\|^2_{R_{\tilde{y}}} = \mathrm{trace}\{\delta S R_{\tilde{y}} \delta S^t\}, \tag{1.19}$$

where $R_{\tilde{y}} = \mathrm{E}[\tilde{y}\tilde{y}^t]$. So if the data vector $\tilde{y}$ is white, then $R_{\tilde{y}} = I$, and

$$\mathrm{E}\left[\|\delta\tilde{x}\|^2|\delta S\right] = \|\delta S\|^2. \tag{1.20}$$

In other words, when the data are white, minimizing the squared error distortion of the matrix S is equivalent to minimizing the expected value of the squared error distortion for the restored image $\hat{x}$.

So our strategy for source coding of matrix S is to simultaneously:

- Whiten the image $\tilde{y}$, so that minimizing the distortion in the coded matrix is equivalent to minimizing the distortion in the restored image.
- Decorrelate the columns of S, so that they will code more efficiently after quantization.

- Decorrelate the rows of S, so that they will code more efficiently after quantization.

The above goals can be achieved by applying the following transformation

$$\check{S} = W_1 S T^{-1} \tag{1.21}$$

$$\check{y} = T\tilde{y}, \tag{1.22}$$

where T is a matrix that simultaneously whitens the components of $\tilde{y}$ and decorrelates the columns of S, and W_1 is a matrix that approximately decorrelates the rows of S. In our case, W_1 is a wavelet transform, since wavelet transforms are known to be an approximation to the Karhunen-Loeve transform for stationary sources, and are commonly used as decorrelating transforms [28]. We form the matrix T by applying a wavelet transform to $\tilde{y}$ followed by gain factors designed to normalize the variance of the wavelet coefficients. Specifically, $T = \Lambda_w^{-1/2} W_2$ and $T^{-1} = W_2^{-1} \Lambda_w^{1/2}$, where W_2 is also a wavelet transform, and $\Lambda_w^{-1/2}$ is a diagonal matrix of gain factors designed so that

$$\Lambda_w = \mathrm{E}[\mathrm{diag}(W_2 \tilde{y} \tilde{y}^t W_2^t)].$$

This matrix T approximately whitens and decorrelates the image $\tilde{y}$. The diagonal matrix Λ_w can be estimated from training images. We take the wavelet transform W_2 of these training images, compute the variance of wavelet coefficients in each band, and average over all images. In this way, we only have a single gain factor for each band of the wavelet transform.

Using the above matrix source coding technique, the computation of the space-varying convolution now becomes

$$\tilde{x} \approx W_1^{-1}[\check{S}]\check{y}, \tag{1.23}$$

where $[\check{S}]$ is the quantized version of $\check{S}$. Therefore, the on-line computation consists of two wavelet transforms and a sparse matrix vector multiply. We will show later that this technique results in huge savings in computation of space-varying convolution.

1.6.2 Efficient implementation of matrix source coding

In order to implement the fast space-varying convolution method of Eq. (1.23), it is first necessary to compute the source coded matrix $[\check{S}]$. We will refer to the computation of $[\check{S}]$ as the off-line portion of the computation. Once the sparse matrix $[\check{S}]$ is available, then the space-varying convolution may be efficiently computed using on-line computation shown in Eq. (1.23).

However, even though the computation of $[\check{S}]$ is performed off-line, rigorous evaluation of this sparse matrix is still too large for practical problems. For example, if the image is 16 Megapixel in size, then temporary storage of S requires approximately 1 Petabyte of memory. The following section describes an efficient approach to compute $[\check{S}]$ which eliminates the need for any such temporary storage.

In our implementation, we use the Haar wavelet transform [29] for both W_2 and W_1. The Haar wavelet transform is an orthonormal transform, so we have that

$$\begin{aligned} \check{S} &= W_1 S T^{-1} \\ &= W_1 S W_2^t \Lambda_w^{1/2}. \end{aligned}$$

Therefore, the precomputation consists of a wavelet transform W_2 along the rows of S, scaling of the matrix entries, and a wavelet transform W_1 along the columns. Unfortunately, the computational complexity of these two operations is $O(P^2)$, where P is the number of pixels in the image, because the operation requires that each entry of S be touched. In order to reduce this computation to order $O(P)$, we will use a two stage quantization procedure combined with a recursive top-down algorithm for computing the Haar wavelet transform coefficients.

The first stage of this procedure is to reduce the computation of the wavelet transform W_1. To do this, we first compute the wavelet transform of each row using W_2, and then we perform an initial quantization step. This procedure is expressed mathematically as

$$[\check{S}] \approx [W_1 [S W_2^t \Lambda_w^{1/2}]], \tag{1.24}$$

where the notation $[\cdots]$ denotes quantization. After the initial quantization, the resulting matrix $[SW_2^t\Lambda_w^{1/2}]$ is quite sparse. If the number of remaining nonzero entries in each row is K, then the computational complexity of the second wavelet transform W_1 is reduced to order $O(KP)$ rather than $O(P^2)$.

However, the problem still remains as to how to efficiently implement the first wavelet transform W_2, since naive application of W_2 to every row still requires $O(P^2)$ operations. The next section describes an approach for efficiently computing this first wavelet transform by only evaluating the significant coefficients. This can be done by using a top-down recursive method to only evaluate the wavelet coefficients that are likely to have absolute values substantially larger than zero.

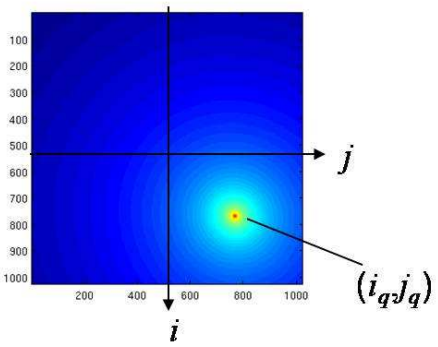
Determining the location of significant wavelet coefficients

As discussed above, we need a method for computing only the significant wavelet coefficients in order to reduce the complexity of the wavelet transform W_2. To do this, we will first predict the location of the largest K entries in every row of the matrix $SW_2^t\Lambda_w^{1/2}$. Then we will present an approach for computing only those coefficients, without the need to compute all coefficients in the transform.

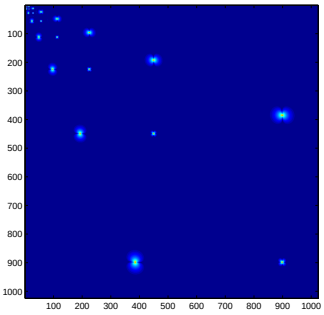
Since each row of S is actually an image, we will use the 2D index (i_q, j_q) to index a row S, and (i_p, j_p) to index a column of S. Using this notation, $S_{(i_q,j_q),(i_p,j_p)}$ represents the effect of the point source at position (i_p, j_p) on the output pixel location (i_q, j_q).

Figure 1.15(a) shows an example of the row of S displayed as an image, together with its wavelet transform in Fig. 1.15(b). The wavelet transform is then scaled by gain factors $\Lambda_w^{-1/2}$, and represented by the corresponding row of $SW_2^t\Lambda_w^{1/2}$. Finally, Fig. 1.15(c) shows the locations of the wavelet coefficients that are non-zero after quantization, and a circle in each band that is used to indicate the region containing all non-zero entries.

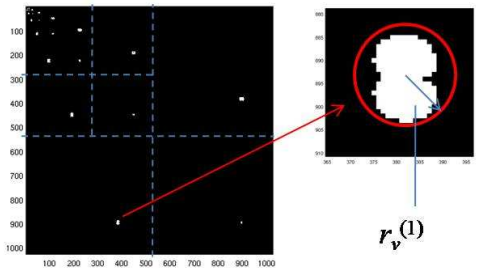
By identifying the regions containing non-zero wavelet coefficients, we can reduce computation. In practice, we determine these regions by randomly selecting rows of



(a) An example of the row image.



(b) The corresponding wavelet transform.



(c) Location of non-zero wavelet coefficients after quantization.

Fig. 1.15. An example of a row of S displayed as an image together with its wavelet transform, and the locations of non-zero entries after quantization: (a) shows the log amplitude map of the row image, (b) shows the log of the absolute value of its wavelet transform, (c) shows the locations of non-zero entries of the corresponding row of $SW_2^t \Lambda_w^{1/2}$.

the matrix and determining the center and maximum radius necessary to capture all non-zero coefficients.

A top-down approach for computing sparse haar wavelet coefficients

In order to efficiently compute sparse Haar wavelet coefficients, we compute only the necessary approximation coefficients at each level using a top-down tree-based approach. The tree structure is illustrated in Fig. 1.16. Since the approximation co-

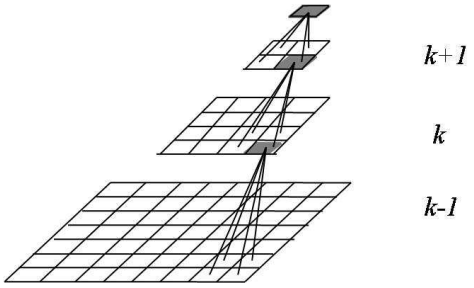


Fig. 1.16. An illustration of the tree structure of approximation coefficients. Shaded squares indicate that the detail coefficients are not zero at those locations. So we go to the next level recursion. Empty squares indicate that the detail coefficients are zero, and we stop to compute their values.

efficients at different scales form a quad-tree, the value of each node can be computed from its children as

$$\begin{aligned} f_a[k][m][n] &= \frac{1}{2}(f_a[k-1][2m][2n] + f_a[k-1][2m+1][2n] \\ &\quad + f_a[k-1][2m][2n+1] + f_a[k-1][2m+1][2n+1]), \end{aligned} \tag{1.25}$$

where $f_a[k][m][n]$ represents an approximation coefficient at level k and location (m, n). Here level $k = 0$ is the lowest level corresponding to the full resolution image. A recursive algorithm can be easily formulated from Eq. (1.25). We know that when the corresponding detail coefficients of an approximation coefficient are all zero, it means that the approximation is perfect. So in this case, for the nodes whose corresponding detail coefficients are all zeroed out after quantization, we stop and compute the value of the approximation coefficient by

$$f_a[k][m][n] = 2^k f_a[0][2^k m][2^k n]. \tag{1.26}$$

Note that $f_a[0][2^k m][2^k n]$ can be directly computed from the expression of the PSF. Another stopping point occurs when the leaves of the tree are reached, i.e. $k = 0$. Therefore, we have a recursive algorithm for computing the significant Haar wavelet approximation coefficients as shown in Fig. 1.17.

```
float FastApproxCoef(k, m, n) {
    float value = 0;
    if (k==0) {
        value = s(i_q, j_q; m, n);
        f_a[k][m][n] = value;
        return value;
    }
    if (ZeroDetailCoef(k,m,n)) {
        value = 2^k s(i_q, j_q; 2^k m, 2^k n);
        f_a[k][m][n] = value;
        return value;
    }
    value += FastApproxCoef(k-1, 2m, 2n);
    value += FastApproxCoef(k-1, 2m, 2n+1);
    value += FastApproxCoef(k-1, 2m+1, 2n);
    value += FastApproxCoef(k-1, 2m+1, 2n+1);
    value = value/2;
    f_a[k][m][n] = value;
    return value;
}
```

(a) Primary function.

```
int ZeroDetailCoef(k,m,n) {
    if (d((m,n),(m_c^k, n_c^k)) > max{r_h^(k), r_v^(k), r_d^(k)}) {
    /* (m_c^k, n_c^k) is the center of the circle at level k */
    /* d((m,n),(m_c^k, n_c^k)) is the distance from (m,n) to (m_c^k, n_c^k)
*/
        return 1;
    }
    else {
        return 0;
    }
}
```

(b) Subroutine.

Fig. 1.17. Pseudo code for fast computation of significant Haar approximation coefficients. Part (a) is the main routine. Part (b) is a subroutine called by (a). The approximation coefficient at level k, location (m, n) is denoted $f_a[k][m][n]$.

After obtaining the significant approximation coefficients, we can compute the necessary wavelet coefficients [29] using the following equation:

$$\begin{aligned} f_d^h[k][m][n] &= \frac{1}{2}(f_a[k-1][2m][2n] - f_a[k-1][2m][2n+1] \\ &\quad + f_a[k-1][2m+1][2n] - f_a[k-1][2m+1][2n+1]) \\ f_d^v[k][m][n] &= \frac{1}{2}(f_a[k-1][2m][2n] + f_a[k-1][2m][2n+1] \\ &\quad - f_a[k-1][2m+1][2n] - f_a[k-1][2m+1][2n+1]) \\ f_d^d[k][m][n] &= \frac{1}{2}(f_a[k-1][2m][2n] - f_a[k-1][2m][2n+1] \\ &\quad - f_a[k-1][2m+1][2n] + f_a[k-1][2m+1][2n+1]), \end{aligned} \quad (1.27)$$

where $f_d^h[k][m][n]$, $f_d^v[k][m][n]$, and $f_d^d[k][m][n]$ denote the horizontal, vertical, and diagonal detail coefficients respectively, at level k, location (m, n). We can show that the complexity of our algorithm for computing sparse Haar wavelet coefficients of an P-pixel image is $O(K)$, where K represents the number of nonzero entries after quantization. The number of nodes that we visit in the tree of approximation coefficients is at most $4K$. Computing these $4K$ nodes requires at most $16K$ additions. So the complexity of computing the approximation tree is $O(K)$. Once the necessary approximation coefficients are computed, computing the detail coefficients only requires $O(K)$ operations. Thus the complexity of our algorithm is $O(K)$ for one row of S. Therefore, we reduce the complexity of precomputation of $[\check{S}]$ from $O(P^2)$ to $O(KP)$.

1.6.3 Experimental results

Experiments on space-varying convolution with stray light PSF

In order to test the performance of our algorithm in computing $\tilde{x} = S\tilde{y}$, we use the test image shown in Fig. 1.18 for $\tilde{y}$, and a realistic stray light PSF for S. The parameters of this PSF are: $a = -1.65 \times 10^{-5}$ mm^{-1}, $b = -5.35 \times 10^{-5}$ mm^{-1}, $\alpha = 1.31$, $c = 1.76 \times 10^{-3}$ mm. The parameters of this PSF were estimated for an Olympus SP-510 UZ[2] camera. We picked three natural images as training images to

[2]Olympus America Inc., Center Valley, PA 18034

obtain the scaling matrix Λ_w. We use normalized root mean square error (NRMSE) as the distortion metric:

$$\text{NRMSE} = \sqrt{\frac{\|\tilde{x} - W_1^{-1}[\check{S}]\check{y}\|_2^2}{\|\tilde{x}\|_2^2}} \ . \tag{1.28}$$

Our measure of computation is the average number of multiplies per output pixel (rate), which is given by the average number of non-zero entries in the sparse matrix $[\check{S}]$ divided by the total number of pixels P.

In order to better understand the computational savings, we used two different image sizes of 256×256 and 1024×1024 for our experiments. In the 256×256 case, we used an eight level Haar wavelet transform for W_2. In the 1024×1024 case, we used a ten level Haar wavelet transform for W_2. In both cases, W_1 was chosen to be a three level CDF 5-3 wavelet transform [30]. In addition, the wavelet transform W_1 was computed using a 64×64 block structure so as to minimize the size of temporary storage used in the off-line computation. Using a block structure enables us to compute the rows of $[SW_2^t\Lambda_w^{1/2}]$ corresponding to one block, take the wavelet transform along the columns, then quantize and save, and then go to the next block.

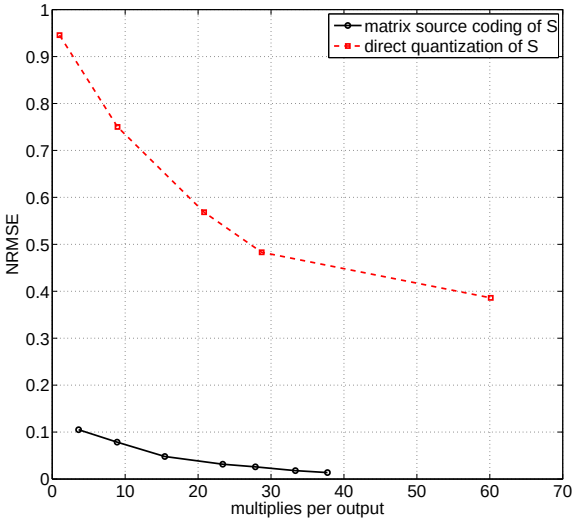
Figure 1.19(a) shows the distortion rate curves for a 256×256 image using two different techniques, i.e. simple quantization of S and matrix source coding of S as described by Eq. (1.23). Note that direct multiplication by S without quantization requires $65,536$ multiplies per output pixel.

Figure 1.19(b) compares the distortion rate curves of our matrix source coding algorithm on two different image sizes — 256×256 and 1024×1024. Notice that the computation per output pixel actually *decreases* with the image resolution. This is interesting since the computation of direct multiplication by S *increases* linearly with the size of the image P.

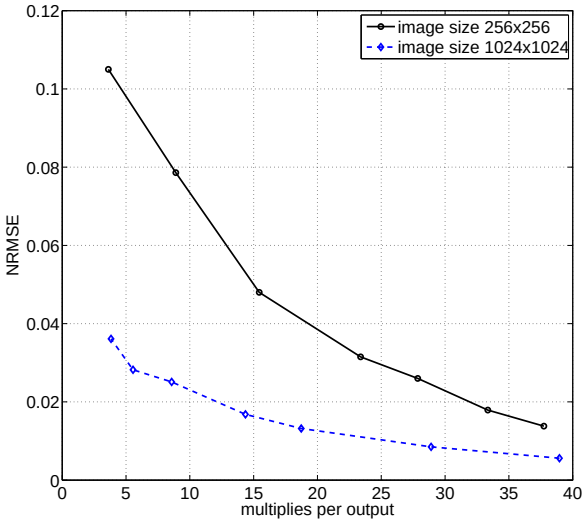
At 1024×1024 resolution, with only 2% distortion, our algorithm reduces the computation from $1048576 = 1024^2$ multiplies per output pixel to 12 multiplies per output pixel; and that is a $87,381:1$ reduction in computation.



Fig. 1.18. Test image for space-varying convolution with stray light PSF.



(a) Distortion vs. computation for two methods on image size 256×256.



(b) Distortion vs. computation for two image sizes using matrix source coding of S.

Fig. 1.19. Experiments demonstrating fast space-varying convolution with stray light PSF: (a) relative distortion versus number of multiplies per output pixel (i.e. rate) using two different matrix source coding strategies: the black solid line shows the curve for our proposed matrix source coding algorithm as described in Eq. (1.23), and the red dashed line shows the curve resulting from direct quantization of the matrix S; (b) comparison of relative distortion versus computation between two different resolutions: the solid black line shows the curve for 256×256, and the dashed blue line shows the curve for 1024×1024.

Experiment on stray light reduction

We use Eq. (1.13) to perform stray light reduction. The space-varying convolution Sy is efficiently computed using our matrix source coding approach. We take outdoor images with an Olympus SP-510UZ camera. The stray light PSF parameters were shown in the previous subsection. The other parameter $\beta = 0.3937$. We use an eight level Haar wavelet for W_2, and a three level CDF 5-3 wavelet [30] for W_1, with block structure of block size 64×64.

Figure 1.6.3 shows an example of stray light reduction. Figure 1.6.3(a) is the captured image. Figure 1.6.3(b) is the restored image. Figures 1.6.3(c-d) shows a comparison between captured and restored versions for different parts of the image. From this example, we can see that the stray light reduction algorithm increases the contrast and recovers more details of the original scene.

1.7 Conclusion

In this chapter, we proposed two approaches for efficiently computing space-varying convolution with a small amount of error. In one approach, we reduce the computation by using a piecewise isoplanatic model to approximate the fully space-varying model. In the other approach, we reduce the computation by using matrix source coding theory to factor a dense matrix into the product of sparse matrices.

Our major contribution in the piecewise isoplanatic approximation approach is that we developed an algorithm to determine a good partition of the image into isoplanatic patches based on vector quantization. Then we used FFT to compute the space-invariant convolution of each patch with an optimal PSF and added the results from all patches together to obtain the final result. The computational complexity is reduced from $O(P^2)$ to $O(P \log(P))$ if the image contains P pixels. We showed results from simulated and real data experiments to demonstrate the effectiveness of our algorithm.



(a) Captured image.



(b) Restored image.



(c) Captured | restored.



(d) Captured | restored.

Fig. 1.20. Example of stray light reduction: (a) shows a captured image, (b) shows the restored image, (c) shows a comparison between captured and restored for a part of the image, (d) shows a comparison between captured and restored for another part of the image.

Our matrix source coding approach reduced the computation from $O(P^2)$ to $O(P)$ by factor the dense matrix S into a product of sparse matrices. The factorization involves using wavelet transforms to decorrelate the rows of columns of the matrix and quantizing the result. We also developed a fast algorithm for computing sparse entries of Haar wavelet transform to accelerate our offline computation, which involves precomputation of quantized wavelet transforms along the rows and columns of the convolution matrix S. Our algorithm for computing the Haar wavelet transform uses a top-down recursive approach, rather than the bottom-up approach used by a conventional filter-bank implementation. The value of this algorithm is to obtain the sparse entries of Haar wavelet coefficients with $O(K)$ complexity, where K is the number of sparse entries to compute. We showed experimental results of space-varying convolution which demonstrated the promise of our algorithm. We also showed a stray light reduction example using our fast algorithm.

2. FAST MATRIX VECTOR MULTIPLICATION USING THE SPARSE MATRIX TRANSFORM

2.1 Introduction

Solutions to problems in linear systems can often be described by the following matrix-vector product

$$y = Ax, \tag{2.1}$$

where x and y are of size $P \times 1$, A is of size $P \times P$. This kind of formulation arises in many applications, for example, image restoration and reconstruction [26], stray light reduction [7, 31], and the evaluation of large-scale electromagnetic integral equations [32]. If A is Toeplitz, then Ax may be computed in $O(P\log P)$ time using the fast Fourier transform (FFT) [33]. However, in the general case, the evaluation of matrix-vector products requires $O(P^2)$ complexity, since each element of the matrix must be processed.[1]

In practice, $O(P^2)$ complexity can render applications intractable when the input is high dimensional. For example, in the problem of stray-light reduction in digital photography, it is necessary to convolve the digital image with a very large point-spread function (PSF) in order to remove the effects of defects in camera optics. However, if an image contains 1 million pixels, then the associated space-varying convolution matrix contains 10^{12} or 1 Tera-word of entries, which is not practical for storage or computation in consumer systems.

Our basic strategy for fast computation of Ax is to factor the matrix A into a series of two sparse matrix transforms (SMT) [35] and a diagonal matrix. Since each SMT is an orthonormal transformation, this decomposition approximates the singular value decomposition (SVD) of the original matrix A. The SMTs are formed by the

[1] Fast algorithms do exist for general matrix-matrix products [34].

product of K Givens rotations, each of which performs a simple rotation on a pair of coordinates. Due to its sparse structure, each Givens rotation requires only 2 multiplies [36]; so they are very fast to implement. In fact, the SMT can be viewed as a generalization of the FFT, with each Givens rotation playing a role analogous to a "butterfly" in an FFT. Once A has been decomposed in this form, then the required matrix-vector product can be approximately computed by first transforming the vector with an initial SMT, then applying a diagonal matrix of scaling factors, and finally transforming the result with a second SMT. Together these operations are of order $O(K)$, where K is the number of rotations used in the SMTs. If we let $r = K/P$ quantify the number of rotations per pixel, then in physical applications, it is typically possible to have accurate approximation with r between 1 and 5. So in practice this SMT decomposition can dramatically reduce computation. However, a challenge with the method is the decomposition of very large matrices. If P is very large, then it may not be possible to store A, let alone compute its SMT decomposition.

This chapter presents an approach to SMT decomposition which is efficient in both computation and memory for very large matrices. In order to make this off-line computation practical, we propose a novel algorithm for factoring large matrices based on the use of training vectors and a cost minimization approach. Using this approach, we show that the off-line computational complexity required to factor the matrix is $O(P \log P)$ once the training pairs are available. However, we note that one limitation for our method is that there must be a method for computing or experimentally measuring output vectors $Y = AX$ for sample input vectors X, and output vectors $Z = A^t O$ for sample input vectors O.

Our algorithm also provides an efficient method for approximately computing the SVD of large dense matrices. The conventional method for computing the SVD is $O(P^3)$ [37]. However, our approach reduces the complexity to $O(P \log P)$ once the training vector pairs are given. The SVD form of a matrix has many advantages, for example, making the approximate inversion of A simple and computation of $A^{-1}y$ fast, for general input vectors y.

We test the effectiveness of our algorithm on the space-varying convolution required in stray-light reduction for digital photography. We demonstrate that our proposed method can dramatically reduce the computation required to accurately compute the convolution of an image with the space-varying PSF, while maintaining high accuracy in the result.

2.2 The SMT for Covariance Estimation and Decorrelation

Sparse matrix transform (SMT) of order K is an orthonormal transform consisting of the product of a series of K sparse transformation $\Pi_{k=1}^{K} E_k$ where each matrix E_k is a Givens rotation [38]. A Givens rotation, E_k is an orthonormal rotation operating on two coordinates, i_k and j_k, which has the form

$$E_k = I + \Theta(i_k, j_k, \theta_k), \tag{2.2}$$

where $\Theta(i_k, j_k, \theta_k)$ is defined as

$$[\Theta]_{ij} = \begin{cases} \cos(\theta_k) - 1 & \text{if } i = j = i_k \text{ or } i = j = j_k \\ \sin(\theta_k) & \text{if } i = i_k \text{ and } j = j_k \\ -\sin(\theta_k) & \text{if } i = j_k \text{ and } j = i_k \\ 0 & \text{otherwise} \end{cases} . \tag{2.3}$$

Figure 2.1 illustrates the structure of a Givens rotation E_k. Figure 2.2 illustrates the SMT operation on input data vector x.

Recently techniques have been developed to estimate the covariance of N sample data vectors using the SMT [35, 39]. More specifically, let $Y = [y_1, \ldots, y_N]$ be a $P \times N$ matrix of zero-mean i.i.d. Gaussian random vectors with covariance $R = E\Lambda E^t$, where Λ is the diagonal matrix of eigenvalues, and E is the orthonormal matrix of eigenvectors. Given this structure, we can compute the constrained maximum log-likelihood estimate of E and Λ as [35]

$$\hat{E} = \arg \min_{E \in \Omega_K} \{|\text{diag}(E^t Y Y^t E)|\} \tag{2.4}$$

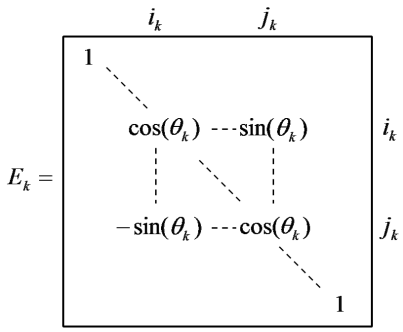


Fig. 2.1. The structure of a Givens rotation.

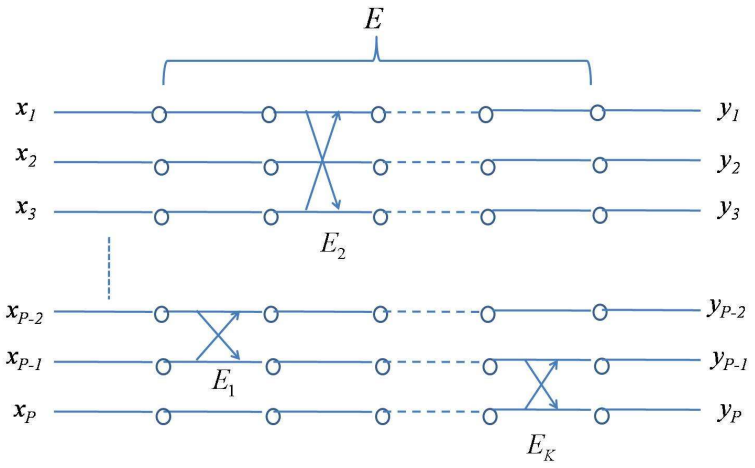


Fig. 2.2. An illustration of the SMT operation on input data x.

$$\hat{\Lambda} = \frac{1}{N}\text{diag}(\hat{E}^t Y Y^t \hat{E}), \tag{2.5}$$

where Ω_K is the set of SMTs of order K. We will often express the order of the SMT as $r = \frac{K}{P}$, where r is the average number of rotations per coordinate. This parameterization is useful because in physical problems, r is typically a small number, for example, less than 5.

The transform E is a very useful approximation to the Karhunen-Loeve (KL) transform [26]. Applying the exact KL transform is a matrix-vector multiplication, so it requires $O(P^2)$ operations for a P dimensional data vector. However, applying the SMT, E, only requires $O(P)$ operations if r is a constant. So E represents a fast method for decorrelating data vectors.

The optimization of Eq. (2.4) is of course the key step in the design of the SMT $\hat{E}$. This can be done using greedy optimization techniques by searching for the best coordinate pair, (i_k, j_k), for the each new Givens rotation [35]. However, the resulting SMT design algorithm is then $O(P^3)$ because each Givens rotation requires the search over all coordinate pairs. Most recently, an algorithm has been published which results in an $O(P \log P)$ complexity SMT design algorithm, through the incorporation of a graphical neighborhood constraint in the coordinate [39].

So in using the methods of [35,39], the $\hat{E}$ can be computed through the application of an algorithm with the following form:

$$\hat{E} \leftarrow \text{FastSMTDesign}(Y).$$

Once $\hat{E}$ is computed, then the estimate of $\hat{\Lambda}$ is easily computed from Eq. (2.5), and we assume that the columns of $\hat{E}$ returned by the FastSMTDesign are arranged so that the diagonal entries of $\hat{\Lambda}$ are in descending order. The transform $\hat{E}$ is very valuable both for computing an estimate of the covariance, $\hat{R} = \hat{E}\hat{\Lambda}\hat{E}^t$, and for fast decorrelation of data vectors

$$\tilde{y} = \hat{E}^t y,$$

when $y \sim N(0, R)$.

2.3 Fast Matrix-Vector Multiplication using the SMT

Our approach for fast computation of the matrix-vector product of Eq. (2.1) is to approximate the linear system represented by the large dense matrix A with the product of two low order SMTs and a diagonal matrix. If the order of the SMTs is $K = rP$, then the total number of multiplies required for computing the approximate matrix-vector product is then $4r + 1$, which for typical values of $r \approx 4$ is much less than P.

2.3.1 Matrix Approximation Using the SMT

Our objective is to represent the matrix A in the form

$$A \approx F\Sigma T^t, \tag{2.6}$$

where F and T are orthonormal SMTs and Σ is a diagonal matrix. Notice that Eq. (2.6) is essentially an SVD of the matrix A, with F and T serving as the approximate left and right singular vectors. With this decomposition, we can efficiently evaluate the required matrix-vector product of Eq. (2.1) as

$$Ax \approx F\Sigma \left(T^t x\right), \tag{2.7}$$

where the multiplication by each SMT is only of order $O(P)$. Intuitively, the method of Eq. (2.7) is analogous to the use of an FFT for fast convolution, with T^t serving as the fast transform, Σ serving as the transfer function, and F serving as the fast inverse transform. Figure 2.3 shows the flow diagram of approximating matrix-vector product using our scheme.

Once the decomposition of Eq. (2.6) is computed, then the on-line computation of Eq. (2.7) is straightforward and fast. However, the difficult step is the off-line computation of the SMT decomposition, particularly when P is very large and A is dense. In this case, even storing the matrix A may not be practical. So we will need clever methods for computing T, Σ, and F without explicitly storing or even evaluating A.

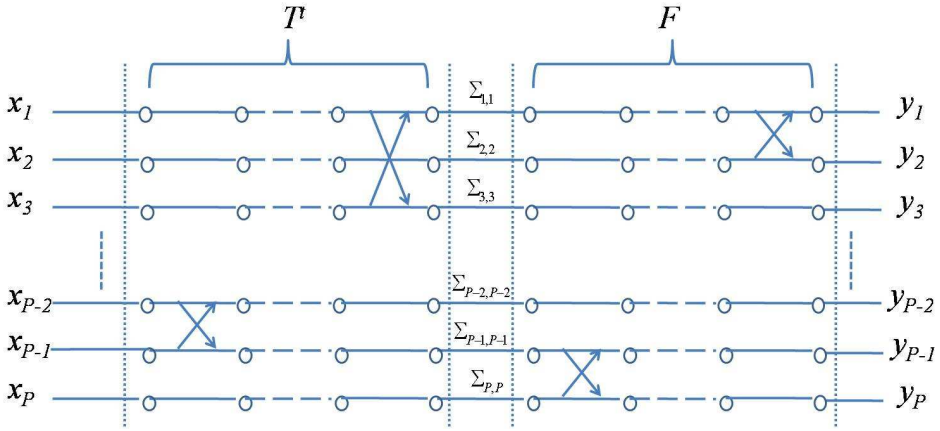


Fig. 2.3. Flow diagram of approximating matrix-vector multiplication with SMTs and scaling.

In this chapter, we will assume that while A is too large to store, it is practical (but perhaps very costly) to compute the application of A, and its transpose A^t, to training inputs. This is possible, for example, if application of A can be implicitly expressed as the solution of a differential equation, or if there is some other method for applying A which is computationally expensive but tractable. In our example application of stray light reduction, we have previously proposed a method for making the required matrix-vector product computationally tractable [31].

Using this assumption, we will be able to compute the following two sets of input/output training pairs,

$$Y = AX \tag{2.8}$$

$$Z = A^t O, \tag{2.9}$$

where X and O are $P \times N$ training input matrices, and Y and Z are corresponding $P \times N$ outputs. Notice that (X, Y) represents the input/output pairs for the system, and (O, Z) represents the output for the system's adjoint. We have found that by using training data from both the system and its adjoint, we can improve the accuracy of the training process.

Therefore, by combining the relations of Eqs. (2.7), (2.8), and (2.9), we can derive the following L_2 cost function for the design of the SMT Decomposition.

$$Cost(F, T, \Sigma) = \|Y - F\Sigma T^t X\|_{\mathcal{F}}^2 + \|Z - T\Sigma F^t O\|_{\mathcal{F}}^2, \tag{2.10}$$

where $\|\cdot\|_{\mathcal{F}}$ represents Frobenius norm.

In the following Subsections 2.3.1 and 2.3.1, we present a two-stage approach to the design of the SMTs F and T. In the first stage, we compute the SMTs that decorrelate the rows and columns of A using an existing method [39]. This first stage approach is reasonable since we know that the ideal left and right singular transforms of the exact SVD do in fact decorrelate the rows and columns of A, respectively. In the second stage, we apply the SMTs obtained from the first stage to input and output training vectors, and then iteratively search for SMTs that minimize the cost function in Eq. (2.10).

First Stage SMT Decomposition Design

We know that, ideally, the SMT F^t should decorrelate the rows of the matrix A. Therefore we can use the method of Section 2.2 to estimate the matrix F from the matrix A:

$$F_0 \leftarrow \text{FastSMTDesign}(A). \tag{2.11}$$

However, this approach is not practical because the matrix A is too large. So in order to reduce computation, we reduce the number of columns in the matrix A to form a new $P \times N$ matrix

$$A_c = AW,$$

where W is a $P \times N$ matrix with i.i.d. $\mathcal{N}(0,1)$ entries. So the columns of A_c are formed by adding randomly weighted columns of A. We then compute the estimate of the decorrelating transform as

$$F_0 \leftarrow \text{FastSMTDesign}(A_c)\ . \tag{2.12}$$

Notice that, formed in this manner, the columns of the matrix A_c are in fact i.i.d. Gaussian random vectors with mean zero and covariance

$$E\left[\frac{1}{P}A_c A_c^t\right] = AE\left[\frac{1}{P}WW^t\right]A^t = AA^t\ .$$

So the basic assumptions of the covariance estimation algorithm are met, and F_0^t should be a good estimate of the decorrelating transform for the rows of A.

Similarly, we compute

$$T_0 \leftarrow \text{FastSMTDesign}(A_r^t), \tag{2.13}$$

where $A_r^t = A^t W$. This completes the first stage of SMT decomposition design.

Second Stage SMT Decomposition Design

Once the initial values of the transforms, F and T, are created using the first stage SMT design, next they can be iteratively improved using the recursion

$$F_{k+1} \leftarrow F_k U_k \tag{2.14}$$

$$T_{k+1} \leftarrow T_k V_k, \tag{2.15}$$

where U_k and V_k are both Givens rotations operating on the same coordinate pair (i_k, j_k). So in this case, F_{k+1} and T_{k+1} are still both SMTs.

With each iteration, the specific Givens rotations, U_k and V_k, are chosen to minimize the cost function of Eq. (2.10) subject to the constraint that the previous values of F_k and T_k remain constant. Substituting the recursions of Eqs. (2.14) and (2.15) into the cost function of Eq. (2.10) results in

$$\begin{aligned} Cost_k(U_k, V_k, \Sigma) &= \|Y - F_k U_k \Sigma V_k^t T_k^t X\|_{\mathcal{F}}^2 + \|Z - T_k V_k \Sigma U_k^t F_k^t O\|_{\mathcal{F}}^2 \\ &= \|F_k^t Y - U_k \Sigma V_k^t T_k^t X\|_{\mathcal{F}}^2 + \|T_k^t Z - V_k \Sigma U_k^t F_k^t O\|_{\mathcal{F}}^2 \\ &= \|Y_k - U_k \Sigma V_k^t X_k\|_{\mathcal{F}}^2 + \|Z_k - V_k \Sigma U_k^t O_k\|_{\mathcal{F}}^2, \end{aligned} \tag{2.16}$$

where Y_k, X_k, Z_k, and O_k can be recursively computed by

$$Y_{k+1} \leftarrow U_k^t Y_k \tag{2.17}$$

$$T_{k+1} \leftarrow V_k^t T_k \tag{2.18}$$

$$Z_{k+1} \leftarrow U_k^t Z_k \tag{2.19}$$

$$O_{k+1} \leftarrow V_k^t O_k \tag{2.20}$$

with initial values $Y_0 = F_0^t Y$, $X_0 = T_0^t X$, $Z_0 = F_0^t Z$, and $O_0 = T_0^t O$. In addition, the diagonal matrix Σ is initialized as

$$\Sigma_{(i,i)} = \frac{\langle Y_{0,(i,\cdot)}, X_{0,(i,\cdot)} \rangle + \langle Z_{0,(i,\cdot)}, O_{0,(i,\cdot)} \rangle}{\|X_{0,(i,\cdot)}\|_2^2 + \|O_{0,(i,\cdot)}\|_2^2}, \tag{2.21}$$

where $X_{0,(i,\cdot)}$, $Y_{0,(i,\cdot)}$, $O_{0,(i,\cdot)}$, and $Z_{0,(i,\cdot)}$ are the i^{th} row of X_0, Y_0, O_0, and Z_0 respectively. The K^{th} update is then computed by solving the optimization problem

$$(U_k, V_k, \Sigma) \leftarrow \arg \min_{U_k, V_k, \Sigma} Cost_k(U_k, V_k, \Sigma) \tag{2.22}$$

subject to the constrains that U_k and V_k are Givens Rotations, and Σ is diagonal.

In order to compute the solution to Eq. (2.22), we will need to determine the coordinates, (i_k, j_k), that yield the best reduction in cost. To do this, we will select the best transformation for each choice of coordinate pairs, and compute the reduction in cost for that choice. This cost reduction is given by

$$\begin{aligned} \Delta c_k(i,j) &= \|Y_k^{(i,j)} - \Sigma X_k^{(i,j)}\|_{\mathcal{F}}^2 + \|Z_k^{(i,j)} - \Sigma O_k^{(i,j)}\|_{\mathcal{F}}^2 \\ &\quad -\|Y_k^{(i,j)} - B X_k^{(i,j)}\|_{\mathcal{F}}^2 - \|Z_k^{(i,j)} - B O_k^{(i,j)}\|_{\mathcal{F}}^2 \end{aligned} \tag{2.23}$$

where B is an unconstrained 2×2 matrix, and $Y_k^{(i,j)}$, $X_k^{(i,j)}$, $Z_k^{(i,j)}$, and $O_k^{(i,j)}$ are $2 \times N$ matrices consisting of the i^{th} and j^{th} rows of Y_k, X_k, Z_k, and O_k, respectively. We find the transform B that makes the maximum amount of cost reduction. Figure 2.4 illustrates this idea. For a specific pair of coordinates (i, j), maximizing the cost reduction in Eq. (2.23) is equivalent to minimizing the cost function. So the best transform for coordinates (i, j) at iteration k is

$$B_k^{(i,j)} = \arg \min_B \left\{ \|Y_k^{(i,j)} - B X_k^{(i,j)}\|_{\mathcal{F}}^2 + \|Z_k^{(i,j)} - B O_k^{(i,j)}\|_{\mathcal{F}}^2 \right\}. \tag{2.24}$$

The solution of the above minimization problem can be obtained by taking the derivative of the cost function with respect to B and letting it be zero. We obtain the following expression of $B_k^{(i,j)}$ for coordinates (i, j)

$$\text{vec}(B_k^{(i,j)}) = \left(I \otimes (O_k^{(i,j)}(O_k^{(i,j)})^t) + (X_k^{(i,j)}(X_k^{(i,j)})^t) \otimes I \right)^{-1} \text{vec}\left(O_k^{(i,j)}(Z_k^{(i,j)})^t + Y_k^{(i,j)}(X_k^{(i,j)})^t \right), \tag{2.25}$$

where $\text{vec}(\cdot)$ represents stacking the columns of a matrix into a column vector, and $\otimes$ represents Kronecker product.

By substituting the expression of $B_k^{(i,j)}$ in Eq. (2.25) for B in Eq. (2.23), we obtain the maximum cost reduction for coordinate pair (i, j). We choose the pair of coordinates (i, j) with the maximum amount of cost reduction and perform singular value decomposition of $B_k^{(i,j)}$ such that

$$B_k^{(i,j)} = U_k^{(i,j)} D_k^{(i,j)} (V_k^{(i,j)})^t,$$

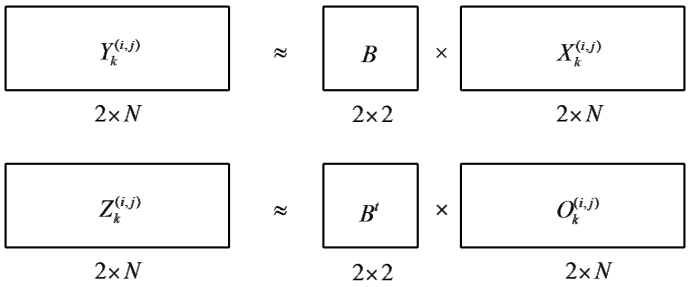


Fig. 2.4. An illustration of our greedy optimization approach: we operate on a pair of coordinates in every iteration to minimize the cost function.

```
[Iarray, Jarray, U, V, Σ] ← SMTSVD(X_0, Y_0, O_0, Z_0, Σ, K) {
    for k = 1:K
        [Iarray(k), Jarray(k)] ← (i, j) with the max cost reduction;
        Compute transform matrix B_k^(i,j) according to Eq. (2.25);
        Compute SVD of B: B_k^(i,j) = U_k^(i,j) D_k^(i,j) (V_k^(i,j))^t;
        U(k) ← U_k^(i,j), V(k) ← V_k^(i,j);
        Update X_k^(i,j), Y_k^(i,j), O_k^(i,j), Z_k^(i,j), and Σ according to Eqs. (2.26)
and (2.27);
    end
}
```

Fig. 2.5. Pseudo code for iterative design of second stage SMT decomposition.

where $U_k^{(i,j)}$ and $V_k^{(i,j)}$ are unitary, and $D_k^{(i,j)}$ is a diagonal matrix. We then update X, Y, O, Z as follows

$$\begin{aligned} X_k^{(i,j)} &= \left(V_k^{(i,j)}\right)^t X_{k-1}^{(i,j)} \\ Y_k^{(i,j)} &= \left(U_k^{(i,j)}\right)^t Y_{k-1}^{(i,j)} \\ O_k^{(i,j)} &= \left(U_k^{(i,j)}\right)^t O_{k-1}^{(i,j)} \\ Z_k^{(i,j)} &= \left(V_k^{(i,j)}\right)^t Z_{k-1}^{(i,j)}. \end{aligned} \tag{2.26}$$

We also update the diagonal matrix Σ as

$$\begin{aligned} \Sigma_{(i,i)} &= D_{k,(1,1)}^{(i,j)} \\ \Sigma_{(j,j)} &= D_{k,(2,2)}^{(i,j)}. \end{aligned} \tag{2.27}$$

We continue this process until the number of iterations K is reached. The pseudo code of our algorithm is shown in Fig. 2.5. For an easier mathematical description, we can substitute the entries (i,i), (i,j), (j,i), (j,j) of an identity matrix of size $P \times P$ with values in $V_k^{(i,j)}$ to form a Givens rotation matrix V_k. Similarly we can

construct a Givens rotation matrix U_k, for $k = 1, \ldots, K$. Then the big dense matrix A is approximated by multiplication of a series of sparse matrices

$$A \approx F_0 U_1 \cdots U_K \Sigma V_K^t \cdots V_1^t T_0^t. \tag{2.28}$$

In fact, the above equation shows that our algorithm performs an approximate singular value decomposition of the big dense matrix A.

Neighborhood based search algorithm

Our singular value decomposition algorithm described by Fig. 2.5 contains a computationally expensive part, which is to find the coordinate pair with the maximum cost reduction. If implemented rigorously, this takes $P(P-1)/2$ comparisons, which makes the complexity of the entire algorithm $O(KP^2)$. However, we can use a neighborhood based approach and a Red-Black tree data structure [34] similar to the graph based algorithm described in Section 2.2 to dramatically reduce the complexity.

When we search for the pair of coordinates with the maximum cost reduction, we only consider coordinates that are neighbors of each other. The neighboring coordinates of coordinate i are defined as $\{i-b, \ldots, i-1, i+1, \ldots, i+b\}$, where $2b$ is the size of neighborhood. We define this neighborhood relation because in the pre-processing stage the estimated eigenvalues Λ_1 and Λ_2 are in descending order. Coordinates with eigenvalues close to each other tend to be correlated. For each coordinate i, we compute the cost reduction of the coordinate pair (i, j) for each neighbor j, and store the maximum. In each iteration, we search for the maximum cost reduction among all coordinates. Since we are using a Red-Black tree to store the maximum cost reduction for each coordinate, we are guaranteed to have $O(\log_2(P))$ complexity of searching for the maximum. After we update the input and output training vectors and Σ, we also have to update the maximum cost reduction of related coordinates and update the Red-Black tree, which is also $O(\log_2(P))$. So the computational complexity is reduced to $O(K\log_2(P))$. Therefore, the total complexity of our off-line SMT design algorithm, including first and second stage, is then $O((K_0 + K)\log_2(P))$.

In real applications, K_0 and K are both on the same order as P. So the complexity is $O(P\log_2(P))$.

2.3.2 Online computation of matrix-vector product

Our algorithm presented in previous sub-sections factors the big dense matrix into the product of SMTs and a diagonal matrix. So our original problem of matrix-vector multiplication in Eq. (2.1) can now be efficiently computed by

$$\hat{y} = F_0 U_1 \cdots U_K \Sigma V_K^t \cdots V_1^t T_0^t x. \tag{2.29}$$

In the above equation, F_0 and T_0 are SMTs of order K_0, $\{U_k\}_k$ and $\{V_k\}_k$ are all Givens rotations. If we define $r = (K + K_0)/P$, then by using our algorithm, we just need $4r + 1$ multiplies per coordinate to compute the matrix-vector product. For real application, r is typically a small number (less than 5), so we achieve a significant amount of reduction in computation from $O(P^2)$ to $O(P)$.

2.4 Experimental Results

In this section, we show some results for computing the convolution of an image with a spac-varying stray light point spread function (PSF) [7]. Stray light PSF features in large spatial support and space-varying property. So in this case, matrix A in Eq. (2.1) is a large dense matrix, with each column being a stray light PSF, and x is an image with P pixels. In the iterative algorithm to correct stray light contamination [7], we need to compute Ax. The entries of A are described by

$$A_{i,j} = s(p_i, q_i; p_j, q_j),$$

where (p_i, q_i) and (p_j, q_j) are the 2D positions of coordinates i and j. Following the previously proposed PSF model [3, 4, 6, 7], we have

$$s(p_i, q_i; p_j, q_j) = \frac{1}{z} \frac{1}{\left(1 + \frac{1}{p_j^2+q_j^2}\left(\frac{(p_i p_j + q_i q_j - p_j^2 - q_j^2)^2}{(c+a(p_j^2+q_j^2))^2} + \frac{(-p_i q_j + q_i p_j)^2}{(c+b(p_j^2+q_j^2))^2}\right)\right)^{\alpha}}, \tag{2.30}$$

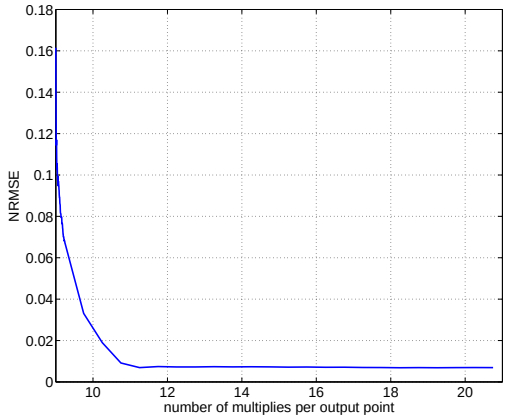


Fig. 2.6. Normalized root mean square error for different amount of computation.

where z is a normalizing constant that makes the integral of the PSF equal to 1. An analytic expression for z is $z = \frac{\pi}{\alpha-1}(c + a(p_j^2 + q_j^2))(c + b(p_j^2 + q_j^2))$. The model parameters are (a, b, c, α). We use estimated parameters [31] for an Olympus SP-510 UZ camera in this experiment: $a = -1.65 \times 10^{-5}$ mm^{-1}, $b = -5.35 \times 10^{-5}$ mm^{-1}, $\alpha = 1.31$, $c = 1.76 \times 10^{-3}$ mm. In our experiments, we use images of resolution 256×256, so $P = 65536$. We set the number of training vectors to be $N = 256$. In the first stage, we use $K_0 = 2P$ for using the SMT covariance estimation algorithm. In addition, since for this example, the matrix A is not too far from symmetry, we use the same transform for F_0 and E_0. In the second stage, we use natural images as training inputs. The training output from the first and second stages are computed using the method proposed in our previous paper [31]. We use normalized root mean square error (NRMSE) as the distortion metric:

$$\text{NRMSE} = \sqrt{\frac{\|\hat{y} - y\|_2^2}{\|y\|_2^2}}. \tag{2.31}$$

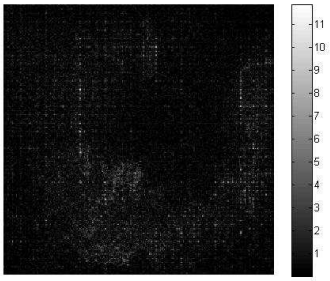
A natural 256×256 image is used as testing input. In Fig. 2.6, we plot the NRMSE against the number of multiplies performed per coordinate. The NRMSE is computed over randomly sampled pixels in the output image. From Fig. 2.6, we can see that the NRMSE goes down to less than 1% with only 12 multiplies per coordinate. If we



(a) Result from very accurate computation.



(b) Result from proposed method.



(c) Difference image.

Fig. 2.7. A comparison between the result using our previous method and the algorithm proposed in this work: (a) shows the result from our previous method, (b) shows the result from our currently proposed algorithm, (c) shows the difference image. The convolution results are very close to each other.

implement this matrix-vector multiply rigorously, then we need $P = 256^2 = 65536$ multiplies per coordinate. Figure 2.7 shows a comparison between the result of our previous method and the algorithm proposed in this work. The convolution results are visually very close to each other. So we can achieve a huge amount of reduction in computation with only a small amount of error.

2.5 Conclusion

In this work, we propose a fast algorithm for computing a big dense matrix-vector product using the SMT decomposition. We achieve a reduction of computational complexity from $O(P^2)$ to $O(P)$. Our algorithm does not assume any structure of the matrix and does not need to construct or store the full matrix. Our experi-

mental results with an example in space-varying convolution with a stray light PSF demonstrate the capability of our algorithm.

In addition, the offline portion of our algorithm performs an approximate singular value decomposition of the big matrix by factoring it into the product of a series of SMTs and a diagonal matrix. We only use several training input and output vectors to obtain the SMTs. Therefore our algorithm solves the storage problem which regular singular value decomposition algorithms have when dealing with extremely large matrices. Plus, we are able to reduce of computational complexity of singular value decomposition from $O(P^3)$ to $O(P\log(P))$ after the training data is given. It is worth noting that both SMT and a diagonal matrix is easy to invert: $A^{-1} \approx T_0 V_1 \cdots V_K \Sigma^{-1} U_K^t \cdots U_1^t F_0^t$. So when we approximate the linear system with SMTs and a diagonal matrix in Eq.(2.29), we provide a way to solve the inverse problem by directly inverting the matrix, without having to resort to iterative methods. This is our future research direction.

3. ACTIVATION DETECTION IN FMRI DATA ANALYSIS

3.1 Introduction

Functional magnetic resonance imaging (fMRI) is a noninvasive method for studying functional brain anatomy. This technique has been recently finding applications in fields as diverse as marketing [40, 41], surgical planning [42], and forensics [43, 44]. It is also being used for describing the neurobiology that underlies decision making [45,46], visual perception [47,48], perception of rewards and risks [49], and various emotions [50, 51].

FMRI images measure the blood oxygenation levels as a function of the location in the brain. When the subject is presented with a stimulus, related areas of the brain exhibit increased neuronal activity and thus a change in blood oxygenation levels. Accurate detection of the regions of the increased activity from fMRI data would allow establishing which areas of the brain are responsible for processing the stimulus. However, since the fMRI measurements suffer from severe noise and other degradations, accurate detection of activated regions is a challenging inverse problem which has been widely studied in the literature ([52–62]).

The focus of our study is event-related fMRI which measures the response to a single short stimulus or a sequence of such stimuli. We develop a framework to detect activations in event-related fMRI. This framework constructs a forward model for fMRI data and solves an inverse problem to estimate activation strength at each pixel, which is then thresholded to produce a binary activation map.

Our forward model requires a parametric model for the hemodynamic response function (HRF) [63]. HRF models a pixel's response when stimulated by an ideal impulse. While our framework is capable of handling any parametric HRF model, we

work with two of them that are widely used in the prior literature: one is Gamma Variate model ([64, 65]), and the other is a sum of two Gamma functions model ([55, 62, 63]).

To capture the temporal correlation of the data, we adopt the autoregressive noise model of [66]. We model the spatial correlation of fMRI data, introduced by the scanner, as the effect of a Gaussian blur. We then develop computationally efficient procedures to estimate the parameters of the noise model and the width of the Gaussian kernel. We show through experiments with synthetic data that this parameter estimation method is accurate.

The parameter estimates are used in the next stage, which is a frame-by-frame image restoration procedure [67] based on minimizing the total variation (TV) of the image. TV-based image restoration techniques [68] have been shown to reduce noise while preserving image edges [69]. These methods have been demonstrated to be robust to significant amount of noise and blurring, and to be especially well suited to the situations where the image to be restored is piecewise constant. These properties make such techniques ideally match our problem of producing a binary activation map based on very noisy, blurred data.

After restoring the images frame by frame, we compute the maximum a posteriori (MAP) estimate [70] of the HRF parameters for each pixel. Then the activation map is produced by thresholding the estimated amplitude parameter map. We employ the generalized Gaussian Markov random field (GGMRF) model [71] for the prior distribution of the HRF parameters. GGMRF prior model tends to be edge-preserving [71], while also imposing spatial regularity on the estimated HRF parameters. Despite enforcing the spatial regularity of the estimates, this method is robust to different shapes of HRF waveforms and to spatially-varying HRFs, as we demonstrate through several experiments. Note that this approach is quite different from previous uses of MRFs in fMRI analysis, such as in [53] where an MRF was used to regularize the estimate of the response at every pixel.

Our framework is tested on both simulated data and real data, and compared with the widely used general linear model (GLM) method [54], which consists of spatial Gaussian smoothing and linear regression analysis. We assess the performance by constructing the empirical receiver operating characteristic (ROC) curves for the synthetic data. We show that our algorithm not only outperforms GLM, but is also much more robust to the change of spatial regularization parameters and the change of underlying activation waveforms. For the real data, the performance is assessed with anatomical analysis and benchmark results from a corresponding block paradigm experiment. In a block paradigm experiment, one or more stimulus categories are presented for extended periods of time, either via a long single or repeated short presentations. In event-related experiments, one or more stimuli are presented for short durations interleaved either with one another or with no stimulus presentation, resulting in an inter-stimulus interval that is on the order of seconds (i.e., an appreciable portion of the duration of a hemodynamic response). Block paradigms provide the best detection power, whereas event-related paradigms are used primarily to facilitate estimation of the time-course of a hemodynamic response. However, event-related paradigm has many advantages (e.g., ability to observe transient cognitive behavior, lack of subject habituation [72]) in spite of their weaker detection power. The results of this work suggest that the detection power of event-related experimentation can be improved to enhance the usefulness of these paradigms.

Our work has four major contributions:

- A forward model for fMRI data which simultaneously characterizes both spatial and temporal dependencies of the data in a novel way.

- An effective algorithm for estimating the model parameters, including the noise parameters and the extent of the blur. These estimates not only facilitate our TV-based image restoration, but also provide us with insight into the nature of the data, i.e., how noisy and blurry it is.

- The introduction of TV-based restoration as a very effective preprocessing tool for fMRI data, to reduce the noise and blurring.
- A novel application of a Markov random field (MRF) model to fMRI analysis which results in spatially regularized, robust estimates of the HRF parameters.

After reviewing prior literature in Section 3.2, we introduce our contributions in the subsequent sections. The forward model is described in Section 3.3; the activation detection algorithm is presented in Section 3.4, its two subsections devoted to the two components of the algorithm—image restoration and MAP estimation; and the model parameter estimation methods are introduced in Section 3.5. Our experimental results are described and discussed in Section 3.6 and Section 3.7.

3.2 Background on FMRI Analysis

3.2.1 The GLM method

The GLM method [54] is the most widely used statistical analysis approach for detecting activations in fMRI experiments. It adopts a multivariate linear regression model for the observations in fMRI experiments. For a particular pixel, the general linear model can be written in matrix form as follows:

$$Y = GX + e. \tag{3.1}$$

In the above equation, if we let T be the number of time points (scans), then Y is the data vector of size $T \times 1$ with one entry for each time point, and G is the design matrix whose columns are models for components contributing to the observed response. The components can be either stimulus related responses or confounding factors such as drift terms. In addition, X is the vector of unknown regression coefficients, and e is the noise term. The elements of e are assumed to be independent and identically distributed Gaussian random variables. One can perform a statistical test on a pixel-by-pixel basis and then threshold the statistical parametric map to obtain the

activation map. For example, if we want to test the activation effect (significance) of component one at a pixel, then we can introduce the contrast vector $c = [1\ 0\ 0\ \ldots]'$ and test whether or not there is strong evidence for this activation effect by t-test [63]. The t-statistic for each pixel is

$$t = \frac{c'\hat{X}}{\sqrt{(Y - G\hat{X})'(Y - G\hat{X})c'(G'G)^{-1}c/r}}, \tag{3.2}$$

where $\hat{X} = (G'G)^{-1}G'Y$ is the ordinary least squares estimate[1] of the regression coefficients, $r = T - \text{rank}(G)$, and the prime denotes transpose of a vector. Note that $c'\hat{X}$ is the estimate of the first regression coefficient and $(Y - G\hat{X})'(Y - G\hat{X})c'(G'G)^{-1}c/r$ is the variance of the estimate. The computed t-statistic follows t distribution with r degrees of freedom under the null hypothesis of no effect [63]. Larger values of t mean more confidence of rejecting the null hypothesis. The activation map is obtained by thresholding this t statistic map. If we want to test the contribution or effect from multiple columns of the matrix G, we can use F-test instead and compute the F statistic [63] for each pixel. The experiments conducted in this chapter only involve testing a single effect, so t statistic is sufficient for simple GLM implementation.

Different design matrices can be used for different applications. For event-related paradigm analysis, the columns of G can be a model for the impulse response of an activated pixel or a combination of this impulse response model and its derivative ([55, 73]) or a Fourier basis [56]. In our experiments, we fit the averaged temporal waveform in the region of interest to a parametric HRF model, and use the normalized version of this fitted signal as the first column of our design matrix. We then orthonormalize the temporal derivative of the HRF with respect to the first column and make it the second column of our design matrix [73]. Similarly, the third column is the orthnormalized version of the dispersion [2] derivative of the HRF with respect to the first two columns. Our contrast vector $c = [1\ 0\ 0\ \ldots]'$, which means that we only test

[1]When the noise is correlated, one can either use feasible generalized least squares estimate instead of ordinary least squares estimate [52], or pre-whiten the data [62]. However, in our implementation of the GLM method, we assume the noise to be uncorrelated.

[2]Here we assume that the HRF model contains a parameter that controls the dispersion of the response. In fact, most widely used HRF models ([55, 62–65, 74]) do contain this parameter.

the significance of the first column. Note that we remove the drift in the preprocessing step of our analysis, so the design matrix doesn't include the drift as a confounding factor. Finally, due to the severe noise in fMRI data, smoothing each frame of the data with a Gaussian kernel is applied prior to the statistical analysis to reduce the amount of noise. This spatial smoothing also plays the role of regularization in the detection process. However, the extent of smoothing (full width at half maximum (FWHM) of the Gaussian kernel) is an unknown parameter and is difficult to estimate. This parameter can much affect the performance of the GLM algorithm.

3.2.2 Spatio-temporal analysis

It is believed that neighboring pixels tend to behave similarly when a stimulus is presented [75]. Linear regression on a pixel-by-pixel basis only uses the temporal features of each pixel, without taking advantage of their underlying spatial correlations. Thus this method tends to declare isolated active pixels. In many cases, isolated active pixels are false alarms. So it is common to perform spatial regularization along with temporal response analysis ([53, 75–78]) in order to exploit both spatial and temporal correlations of the data. Both [75] and [77] employ maximum a posteriori (MAP) estimate of the activation using discrete Markov random field (MRF) as spatial priors. The distinction is that [75] uses binary states of activation, whereas [77] assumes multiple states incorporating the anatomical information. Continuous MRF is used in [53] to perform spatio-temporal restoration of the fMRI data and to produce the parameter map. This spatio-temporal restoration is more appropriate for block paradigm data, since the ideal response of event-related paradigm is not smooth in the temporal domain. A fully Bayesian spatio-temporal model is proposed in [78]. Prior distributions of the signal and noise parameters as well as corresponding hyper-priors are assigned. Spatial regularization is imposed on the noise parameters but not on the signal parameters. Markov-Chain Monte-Carlo (MCMC) simulation is performed to generate samples from the full joint posterior distribution, and thus facilitates the

estimation of the parameters. In [76], a cost function consisting of a weighted combination of the negative Gaussian log-likelihood function and an l_1 norm constraint on the variability of the signal parameters is proposed. In each iteration of the minimizer, soft thresholding is applied in the wavelet domain of the signal parameters in order to approximately perform l_1 regularization.

3.3 Model Formulation

Real fMRI data is 4D: It has three spatial dimensions and one temporal dimension. However, for notational convenience as well as for the ease and speed of our implementations and experiments, we assume 3D data $y_{i,j,t}$ where i and j index two spatial dimensions, and t is a discrete time parameter. This dimension reduction is also due to the fact that the data are generally acquired as 2D slices. If the data come from 3D acquisition, then we should use 4D formulation instead.

In this chapter, we focus on event-related fMRI where the ideal response of a pixel is its impulse response, which is commonly referred to as the hemodynamic response function (HRF). Several parametric models have been proposed for the HRF, for example, the difference of two gamma functions [55], and a simple gamma variate ([64] and [59]). Any HRF model would fit into our framework. We use $h_{i,j,t}(\boldsymbol{\theta_{i,j}})$ to denote the HRF at pixel (i, j) with parameter set $\boldsymbol{\theta_{i,j}}$. One of the parameters in $\boldsymbol{\theta_{i,j}}$ must be an amplitude parameter that characterizes signal intensity.

In addition, each pixel's hemodynamic response is subject to physiological noise $\omega_{i,j,t}$, i.e., noise arising due to the subject's heartbeat, breathing, etc. This noise process is stationary over time [79]. We model this noise as an additive AR(1) process, following ([79] and [66]). In other words, the physiological noise is modeled as the output of the following linear system driven by white noise $\epsilon_{i,j,t}$:

$$\omega_{i,j,t} = \rho\omega_{i,j,t-1} + \epsilon_{i,j,t}. \tag{3.3}$$

We assume that the white noise $\epsilon_{i,j,t}$ is zero-mean. The parameters of this model are therefore ρ (which we assume to be a real number between 0 and 1) and the

standard deviation σ_ϵ of the noise $\epsilon_{i,j,t}$. We assume each of these parameters to be the same for all pixels and all times following the stationarity property of the noise process [79]. Therefore, the noise variance remains constant for each frame. This assumption is used in our total variation formulation and parameter estimation algorithm introduced in the next two sections.

In addition, it is common to assume the presence of an additive baseline m, which is the signal level when no stimulus is presented [63]. However, we assume that an accurate estimate of the baseline may be obtained by averaging the response values measured when no stimulus is applied. This estimate can then be subtracted from the data. Therefore, in the remainder of the chapter we assume zero baseline.

The sum of the HRF and the physiological noise $h_{i,j,t}(\boldsymbol{\theta_{i,j}}) + \omega_{i,j,t}$ is subject to two further types of degradation introduced by the scanner: spatial blurring and noise. The spatial blurring comes from the fact that in any practical MRI experiment, the resolution is limited [80], and it can be modeled as convolution with a point spread function. We use a spatially isotropic Gaussian kernel $g_{i,j}$ with zero mean and standard deviation σ to model this point spread function. We model scanner noise as additive white noise $\eta_{i,j,t}$ with zero mean and standard deviation σ_η which is assumed to be a constant both spatially and temporally. Putting all the pieces together results in the following model for the observed data:

$$y_{i,j,t} = \sum_k \sum_l g_{k,l}(h_{i-k,j-l,t}(\boldsymbol{\theta_{i,j}}) + \omega_{i-k,j-l,t}) + \eta_{i,j,t}. \tag{3.4}$$

3.4 Activation Detection

Our activation detection algorithm first restores the HRF $h_{i,j,t}(\boldsymbol{\theta_{i,j}})$ from the data $y_{i,j,t}$, and then computes the MAP estimate of the parameter set $\boldsymbol{\theta}_{i,j}$ of the HRF. Finally, we generate an activation map through thresholding the parameter map.

The remainder of this section describes the TV-based restoration algorithm and the MAP estimate of the HRF parameters. The image restoration algorithm relies on the estimates of the following model parameters: the standard deviation σ of the

Gaussian blurring kernel, the physiological noise parameters ρ and σ_ϵ, and the scanner noise parameter σ_η. The procedures for estimating these parameters are developed in the next section.

3.4.1 TV-based image restoration

The first part of our activation detection algorithm is accomplished through a total-variation (TV) based restoration algorithm [68] which is applied frame by frame—in other words, the data $\{y_{i,j,t}\}_{i,j}$ is restored for each fixed value of t. We therefore drop the t index for notational convenience. We denote the combined contribution of the physiological and scanner noise to pixel (i, j) in Eq. (3.4) by $v_{i,j}$:

$$v_{i,j} \equiv \sum_k \sum_l g_{k,l}\omega_{i-k,j-l} + \eta_{i,j}. \tag{3.5}$$

With this notation, our model of Eq. (3.4) can be rewritten as follows:

$$y_{i,j} = \sum_k \sum_l g_{k,l}h_{i-k,j-l} + v_{i,j}. \tag{3.6}$$

We formulate the following constrained total variation minimization problem, where we assume that our images are $N \times N$:

$$\begin{aligned} \min_{\{h_{i,j}\}} \quad & \sum_i \sum_j \sqrt{(h_{i+1,j} - h_{i,j})^2 + (h_{i,j+1} - h_{i,j})^2} \\ \text{subject to} \quad & \text{Eq. (3.6)} \\ \text{and to} \quad & \frac{1}{N^2}\sum_i \sum_j v_{i,j}^2 \leq \sigma_v^2. \end{aligned} \tag{3.7}$$

We follow [67] and use second-order cone programming (SOCP) to solve this optimization problem. Our SOCP is almost identical to that in [67], with the exception of the additional blurring term in the constraint of Eq. (3.6). This SOCP is solved using

interior point method through MOSEK software [81]. The constraint parameter σ_v^2 is an estimate of the total noise variance, which can be obtained as follows:

$$\begin{aligned}
\sigma_v^2 &= \mathrm{E}[v_{i,j}^2] - \mathrm{E}[v_{i,j}]^2 \\
&= \mathrm{E}\left[\left(\sum_{k,l} g_{k,l}\omega_{i-k,j-l} + \eta_{i,j}\right)^2\right] \\
&= \sum_{k,l} g_{k,l}^2 \mathrm{E}[\omega_{i-k,j-l}^2] + \mathrm{E}[\eta_{i,j}^2] \\
&= \left(\sum_{k,l} g_{k,l}^2\right) \frac{\sigma_\epsilon^2}{1-\rho^2} + \sigma_\eta^2. \qquad (3.8)
\end{aligned}$$

Section 3.5 shows how to estimate all the parameters used on the right-hand side of the last line in this equation.

The result of performing the above optimization is a set of estimates of HRF, at every pixel location, and for every time t. We denote these estimates by $\hat{h}_{i,j,t}$. The second stage of our activation detection algorithm uses $\hat{h}_{i,j,t}$ in conjunction with the HRF model to produce a MAP estimate of the parameter set $\hat{\boldsymbol{\theta}}_{i,j}$ for every pixel.

3.4.2 MAP estimation

After TV-based image restoration, we obtain restored data $\hat{h}_{i,j,t}$. The ideal case is that $\hat{h}_{i,j,t}$ is free of noise and blur. However, due to errors in estimation and restoration, there are still residual errors after image restoration. We assume the following model for the restored data:

$$\hat{h}_{i,j,t} = h_{i,j,t}(\boldsymbol{\theta}_{i,j}) + e_{i,j,t}. \qquad (3.9)$$

We assume that the noise terms are independent and identically distributed Gaussian random variables with standard deviation σ_e. We let $\hat{\boldsymbol{h}}_{i,j}$ denote the temporal vector of the restored data at pixel (i,j), and let $\boldsymbol{h}_{i,j}(\boldsymbol{\theta}_{i,j})$ denote the HRF at pixel (i,j).

Then the conditional probability density function of the restored data at pixel (i, j) given the parameters is:

$$p(\hat{\boldsymbol{h}}_{i,j}|\boldsymbol{\theta}_{i,j}) = \frac{1}{(2\pi)^{T/2}\sigma_e^T}\exp\left\{-\frac{\|\hat{\boldsymbol{h}}_{i,j} - \boldsymbol{h}_{i,j}(\boldsymbol{\theta}_{i,j})\|_2^2}{2\sigma_e^2}\right\}, \tag{3.10}$$

where T is the number of the time points in the response. If the parameter set $\boldsymbol{\theta}_{i,j}$ for every pixel (i, j) contains R parameters $\theta_{i,j}^{(1)}$, ..., $\theta_{i,j}^{(R)}$, we let $\boldsymbol{\theta}^{(r)}$ be the vector of the r-th parameter values for all pixels, for $r = 1, \ldots, R$. We assume that, given the restored data, the parameters $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(R)}$ are conditionally independent. We model each of them as the following generalized Gaussian Markov random field:

$$p(\boldsymbol{\theta}^{(r)}) = \frac{1}{z_r}\exp\left\{-\frac{1}{q\sigma_r^q}\sum_{\{(i,j),(k,l)\}\in C} K_{i-k,j-l}|\theta_{i,j}^{(r)} - \theta_{k,l}^{(r)}|^q\right\} \quad \text{for } r = 1, \ldots, R, \tag{3.11}$$

where $C = \{\{(i, j), (k, l)\}|(k, l)$ is in the 8-point neighborhood of $(i, j)\}$, z_r is a normalizing constant, σ_r is a regularization parameter. The value of q is usually chosen to be between 1 and 2 to produce regularized estimation that preserves the edges [71]. In most of our experiments, $q = 1.2$; however, as described below, we also tried different values of q to demonstrate that our algorithm is not very sensitive to parameter variations. We choose the spatial weights $K_{i,j}$ to be 1/6 for $i^2 + j^2 = 1$ and 1/12 for $i^2 + j^2 = 2$. Because the R parameters are assumed to be independent, the joint posterior distribution can be written as

$$p(\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(R)}|\hat{\boldsymbol{h}}) = \prod_{(i,j)} p(\hat{\boldsymbol{h}}_{i,j}|\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(R)})p(\boldsymbol{\theta}^{(1)}) \cdot \ldots \cdot p(\boldsymbol{\theta}^{(R)})/p(\hat{\boldsymbol{h}}), \tag{3.12}$$

where $\hat{\boldsymbol{h}}$ is the restored data for all pixels and all time points. The MAP estimates of the HRF parameters are

$$\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(R)}\} = \arg\max_{\boldsymbol{\theta}^{(1)},\ldots,\boldsymbol{\theta}^{(R)}} p(\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(R)}|\hat{\boldsymbol{h}}). \tag{3.13}$$

Maximizing the joint probability distribution is equivalent to minimizing the negative log of it. Therefore the cost function to be minimized is given by the following equation:

$$c(\boldsymbol{\theta}^{(1)},\ldots,\boldsymbol{\theta}^{(R)}) = \frac{1}{2\sigma_e^2}\sum_{(i,j)}\|\hat{\boldsymbol{h}}_{i,j} - \boldsymbol{h}_{i,j}(\boldsymbol{\theta}_{i,j})\|_2^2 + \sum_{\{(i,j),(k,l)\}\in C} K_{i-k,j-l}\sum_{r=1}^{R}\frac{1}{q\sigma_r^q}|\theta_{i,j}^{(r)} - \theta_{k,l}^{(r)}|^q. \tag{3.14}$$

This equation is obtained by taking the negative log of the right-hand side of Eq. (3.12), substituting Eqs. (3.10) and (3.11), and getting rid of the constant terms that do not depend on the parameters $\boldsymbol{\theta}^{(1)},\ldots,\boldsymbol{\theta}^{(R)}$. We use the iterative coordinate descent (ICD) algorithm [82] to perform optimization of this cost function. The ICD algorithm works by iteratively updating parameter estimates at individual pixels to minimize the cost function. After we obtain the MAP estimates of the HRF parameters, we threshold the amplitude parameter of each pixel to get the binary activation map.

3.5 Model Parameter Estimation

The model parameters can be estimated from the fMRI data obtained from regions not anticipated to be active, or obtained from baseline scans conducted in addition to the task scans. This results in the following simplified version of Eq. (3.4) for $t = 1,\ldots,T$:

$$y_{i,j,t} = \sum_k\sum_l g_{k,l}\omega_{i-k,j-l,t} + \eta_{i,j,t}. \tag{3.15}$$

We use these observations to estimate the model parameters in Eq. (3.8): the width σ of the Gaussian blurring kernel, the physiological noise parameters ρ and σ_ϵ, and the scanner noise parameter σ_η. We rasterize the data and the three noise processes, for each time t, to define four vectors $\boldsymbol{y}_t$, $\boldsymbol{\omega}_t$, $\boldsymbol{\epsilon}_t$, and $\boldsymbol{\eta}_t$. For example, $\boldsymbol{y}_t = (y_{1,1,t},\ldots,y_{1,N,t},y_{2,1,t},\ldots,y_{2,N,t},\ldots,y_{N,N,t})'$ for $t = 1,\ldots,T$, where the prime denotes the transpose of a vector. We let A be the Gaussian filtering matrix—i.e.,

the matrix representing the convolution with the spatial Gaussian blurring kernel. Using this notation, Eqs. (3.3) and (3.15) can be rewritten, respectively, as follows:

$$\begin{aligned} \boldsymbol{\omega}_t &= \rho\boldsymbol{\omega}_{t-1} + \boldsymbol{\epsilon}_t, \\ \boldsymbol{y}_t &= A\boldsymbol{\omega}_t + \boldsymbol{\eta}_t. \end{aligned}$$

Our algorithm for estimating the noise parameters ρ, σ_ϵ and σ_η is based on their relationships with the auto-correlation function of $\boldsymbol{y}_t$ which can be easily demonstrated:

$$\mathrm{E}[\boldsymbol{y}_t'\boldsymbol{y}_t] = \frac{\sigma_\epsilon^2}{1-\rho^2}\mathrm{trace}\{AA'\} + N^2\sigma_\eta^2, \tag{3.16}$$

$$\mathrm{E}[\boldsymbol{y}_t'\boldsymbol{y}_{t-1}] = \frac{\rho}{1-\rho^2}\sigma_\epsilon^2\mathrm{trace}\{AA'\}, \tag{3.17}$$

$$\mathrm{E}[\boldsymbol{y}_t'\boldsymbol{y}_{t-2}] = \frac{\rho^2}{1-\rho^2}\sigma_\epsilon^2\mathrm{trace}\{AA'\}. \tag{3.18}$$

We denote $r \triangleq \mathrm{E}[\boldsymbol{y}_t'\boldsymbol{y}_t]$, $r_1 \triangleq \mathrm{E}[\boldsymbol{y}_t'\boldsymbol{y}_{t-1}]$, $r_2 \triangleq \mathrm{E}[\boldsymbol{y}_t'\boldsymbol{y}_{t-2}]$. We use the following standard estimates of the auto-correlations r, r_1, and r_2:

$$\hat{r} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} y_{i,j,t}^2, \tag{3.19}$$

$$\hat{r}_1 = \frac{1}{T-1}\sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} y_{i,j,t}y_{i,j,t-1}, \tag{3.20}$$

$$\hat{r}_2 = \frac{1}{T-2}\sum_{t=3}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} y_{i,j,t}y_{i,j,t-2}. \tag{3.21}$$

If we let the row of A corresponding to pixel (i,j) be $a_{i,j}$, then $\mathrm{trace}\{AA'\} = N^2\|a_{i,j}\|_2^2$. Note that $\|a_{i,j}\|_2^2$ is the same for all rows and is only a function of σ. We denote this function by $\alpha(\sigma)$. Substituting the estimates of Eqs. (3.19-3.21) into Eqs. (3.16-3.18), replacing $\mathrm{trace}\{AA'\}$ with $N^2\alpha(\sigma)$, and solving for the noise parameters, we get the following estimates of the noise parameters:

$$\hat{\rho} = \hat{r}_2/\hat{r}_1, \tag{3.22}$$

$$\hat{\sigma}_\epsilon = \sqrt{\frac{\hat{r}_1(1-\hat{\rho}^2)}{\hat{\rho}\ N^2\alpha(\sigma)}}, \tag{3.23}$$

$$\hat{\sigma}_\eta = \sqrt{\frac{\hat{r} - \hat{r}_1/\hat{\rho}}{N^2}}. \tag{3.24}$$

The estimation of the standard deviation σ of the Gaussian blurring kernel is based on its relationship with the covariance between pixels. Since we are observing non-activated pixels, the expected values of the observations of $y_{i,j,t}$ are 0. Thus the covariance between two pixels which are a distance c apart is equal to $\mathrm{E}[y_{i,j,t}y_{k,l,t}]$ where $(i-k)^2+(j-l)^2=c^2$. It can be easily shown that:

$$\mathrm{E}[y_{i,j,t}y_{k,l,t}] = \frac{\sigma_\epsilon^2\langle a_{i,j}, a_{k,l}\rangle}{1-\rho^2}. \tag{3.25}$$

Note that $\langle a_{i,j}, a_{k,l}\rangle$ is a only a function of σ and of the distance c between pixels (i,j) and (k,l). We denote this function by $\beta_c(\sigma)$. We use the following standard estimate of $\mathrm{E}[y_{i,j,t}y_{k,l,t}]$:

$$\frac{1}{N_c T}\sum_t \sum_{i,j,k,l:\,(i-k)^2+(j-l)^2=c^2} y_{i,j,t}y_{k,l,t}$$

for every pair of pixels (i,j) and (k,l) which are distance c apart. Here, N_c is the total number of such pairs. We assume that the covariance estimates for pairs of pixels that are more than a distance of 2 apart are too noisy to be of use in estimating σ. We form the following sum of squared differences between the theoretical and estimated values of $\mathrm{E}[y_{i,j,t}y_{k,l,t}]$ for $(i-k)^2+(j-l)^2=c^2$ where $c^2=1,2,4$:

$$S = \sum_{c^2=1,2,4}\left(\frac{\sigma_\epsilon^2\beta_c(\sigma)}{1-\rho^2} - \frac{1}{N_c T}\sum_t\sum\nolimits_{i,j,k,l:\,(i-k)^2+(j-l)^2=c^2} y_{i,j,t}y_{k,l,t}\right)^2. \tag{3.26}$$

We then substitute into Eq. (3.26) the estimates of ρ and σ_ϵ from Eqs. (3.22) and (3.23). This makes S a function of σ and the data only. We then estimate the value of σ by minimizing S with respect to σ.

Our parameter estimation strategy is tested on simulated data. The data set consists of 400 Monte Carlo simulations. Each simulation is of size $32\times 32\times 270$, representing image size of 32×32 and 270 time points. The noise realizations are simulated following the noise model stated in Eqs. (3.3,3.5). We construct three datasets, with three different levels of noise. We use $\sigma_\epsilon=\sigma_\eta=0.5$, $\sigma_\epsilon=\sigma_\eta=1$, and $\sigma_\epsilon=\sigma_\eta=2$, resulting in the overall noise standard deviations of $\sigma_v=0.54$, $\sigma_v=1.09$, and $\sigma_v=2.17$, respectively, as computed through Eq. (3.8). The remaining

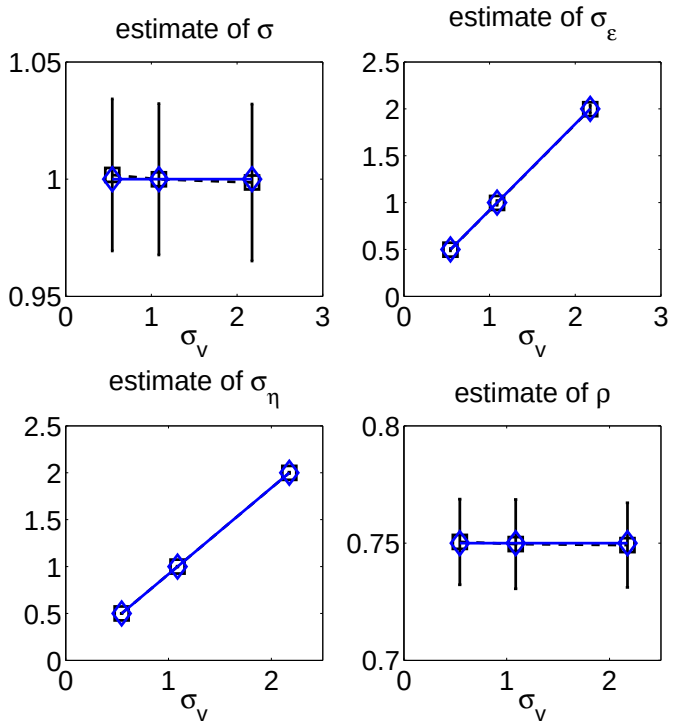


Fig. 3.1. Model parameter estimation results. Solid blue lines with diamonds denote true parameter values. Dashed black lines with squares denote their estimates. Error bars are ± one standard deviation of the estimates.

parameters are fixed at $\rho = 0.75$ and $\sigma = 1$. The parameter estimation results are shown in Fig. 3.1, where the dashed lines with squares are the estimates and the solid lines with diamonds are the true values. The error bars are ± one standard deviation of the estimates. It can be seen that our approach produces estimates very close to the true values, and results in small standard deviations (within 5%) of estimates.

3.6 Synthetic Data Experiments

3.6.1 HRF Models

Our proposed activation detection algorithm can be used in conjunction with any HRF model. Two widely used HRF models are adopted in experimental evaluations

of our algorithm. The first model is a Gamma Variate model [64] which describes the HRF for any activated pixel (i, j) as follows:

$$h_{i,j,t} = \begin{cases} x_{i,j} \left(\dfrac{t - \delta_{i,j}}{\tau_{i,j}} \right)^2 e^{-\frac{t-\delta_{i,j}}{\tau_{i,j}}} & t \geq \delta_{i,j}, \\ 0 & t < \delta_{i,j}, \end{cases} \tag{3.27}$$

where the parameters $x_{i,j}$, $\delta_{i,j}$, and $\tau_{i,j}$ have the following interpretations:

- $4x_{i,j}/e^2$ is the peak amplitude of the response;
- $\delta_{i,j}$ is the delay between the presentation of the stimulus and the onset of the response;
- $\tau_{i,j}$ is the dispersion parameter, and $2\tau_{i,j}$ is the time from the onset of the response to its peak.

Therefore, the parameter set $\boldsymbol{\theta}_{i,j}$ at pixel (i, j) equals $\{x_{i,j}, \delta_{i,j}, \tau_{i,j}\}$. The corresponding regularization parameters in the GGMRF model are σ_x, σ_δ, and σ_τ. Also note that, for every non-activated pixel (i, j), $h_{i,j,t}$ is equal to 0.

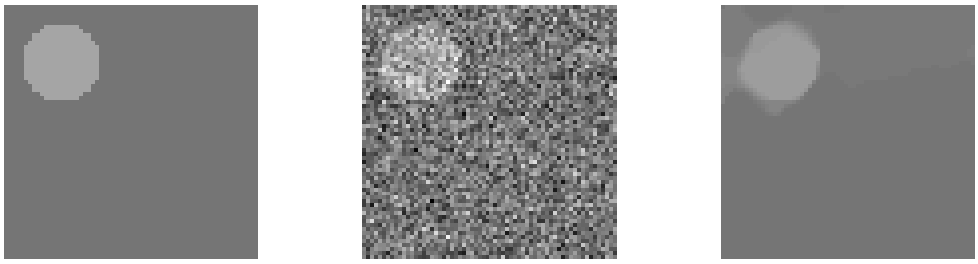
The second model ([55,62,65,83]) is a sum of two Gamma functions and accounts for the undershoot behavior of the hemodynamic response. In this model, the HRF for any activated pixel (i, j) is described as:

$$h_{i,j,t} = x_{i,j} \left(\left(\frac{t}{a_{i,j}^{(1)} b_{i,j}^{(1)}} \right)^{a_{i,j}^{(1)}} e^{-\frac{t - a_{i,j}^{(1)} b_{i,j}^{(1)}}{b_{i,j}^{(1)}}} - c_{i,j} \left(\frac{t}{a_{i,j}^{(2)} b_{i,j}^{(2)}} \right)^{a_{i,j}^{(2)}} e^{-\frac{t - a_{i,j}^{(2)} b_{i,j}^{(2)}}{b_{i,j}^{(2)}}} \right), \tag{3.28}$$

where $x_{i,j}$ controls the amplitude of the response, $c_{i,j}$ is the amount of undershoot, and the remaining parameters $\{a_{i,j}^{(1)}, b_{i,j}^{(1)}, a_{i,j}^{(2)}, b_{i,j}^{(2)}\}$ control the shape of the curve, including the peak time and the undershoot time. Note that $b_{i,j}^{(1)}$ is the dispersion parameter. The parameter set $\boldsymbol{\theta}_{i,j}$ at pixel (i, j) equals $\{x_{i,j}, a_{i,j}^{(1)}, b_{i,j}^{(1)}, a_{i,j}^{(2)}, b_{i,j}^{(2)}, c_{i,j}\}$. The corresponding regularization parameters in the GGMRF model are σ_x, $\sigma_{a^{(1)}}$, $\sigma_{b^{(1)}}$, $\sigma_{a^{(2)}}$, $\sigma_{b^{(2)}}$, σ_c.

3.6.2 Simulated Data

We conduct an experiment to test the performance of the proposed activation detection approach on synthetic data. Our synthetic dataset is of size $64 \times 64 \times 135$, representing 2D images of size 64×64 collected at 9 trials with 15 time points (representing 15 seconds) in each trial. Following our model formulation in Eqs. (3.3,3.4), we generate two sets of simulation data corresponding to the two HRF models described in Eqs. (3.27, 3.28). The correct parameters of the first HRF for each activated pixel (see Eq. (3.27)) are $\delta_{i,j} = 2$, $\tau_{i,j} = 1.5$, and peak amplitude of 1, which corresponds to $x_{i,j} = e^2/4$. The correct parameters of the second HRF for each activated pixel (see Eq. (3.28)) are $x_{i,j} = 1$, $a_{i,j}^{(1)} = 8$, $b_{i,j}^{(1)} = 0.5$, $a_{i,j}^{(2)} = 15$, $b_{i,j}^{(2)} = 0.5$, $c_{i,j} = 0.2$. We simulate two shapes of activation region: one is a 20×20 square, the other is a circle with radius 10. The data is corrupted with a spatial Gaussian blur, following our model formulation in Section 3.3. The width of Gaussian kernel is $\sigma = 1$. The data is also corrupted with AR(1) noise and white Gaussian noise. The contrast-to-noise ratio (CNR) is defined as the ratio between peak signal intensity and the standard deviation of noise which can be computed as the square root of the right-hand side of Eq. (3.8). In our experiment, CNR is set to be 0.5. We first average among the trials to obtain a dataset of size $64 \times 64 \times 15$, then we restore the images frame by frame, and then we compute the MAP estimate of the HRF parameters at each pixel. Finally, we produce the activation map by thresholding the amplitude parameter $x_{i,j}$. Figure 3.2 shows an example of TV-based image restoration for a single image. This image is the frame which contains peak signal intensity. The corrupted image in Fig. 3.2 has a signal-to-noise ratio (SNR) of -8.15 dB, whereas the restored image has an SNR of 9.22 dB.



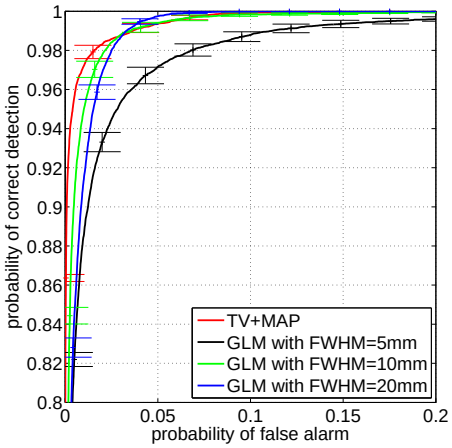
(a) Ideal image. (b) Corrupted image. (c) Restored image.

Fig. 3.2. Example of TV-based image restoration: (a) Noise and blur free image, (b) Image corrupted with noise and blur, (c) Restoration result.

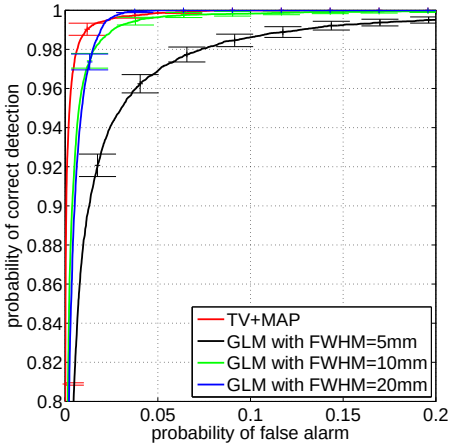
3.6.3 Synthetic Data Experiment 1: Comparison to Three Other Methods

We perform a Monte-Carlo run consisting of 50 simulations, and compare our proposed method, TV-based image restoration followed by MAP estimation of HRF parameters, with the GLM method ([54, 55, 63]) which consists of spatial smoothing followed by linear regression. Note that both methods in this experiment are implemented with knowledge of the correct functional form of the HRF, but no information about the true HRF parameters. For the first HRF model, the regularization parameters in our proposed algorithm are set as follows: $\sigma_e = 1$, $\sigma_x = 3$, $\sigma_\delta = 1$, $\sigma_\tau = 1$, $q = 1.2$. For the second HRF model, the regularization parameters are set as follows: $\sigma_e = 1$, $\sigma_x = 1$, $\sigma_{a^{(1)}} = 10$, $\sigma_{b^{(1)}} = 0.5$, $\sigma_{a^{(2)}} = 10$, $\sigma_{b^{(2)}} = 0.5$, $\sigma_c = 0.3$, $q = 1.2$. We use the same set of regularization parameters across all simulations. In the GLM framework, we try different full width at half maximum (FWHM) of the Gaussian kernel for spatial smoothing. Assuming pixel dimension to be 2.7mm $\times$ 2.7mm, we use FWHM of size 5mm, 10mm, and 20mm respectively. As described in Section 3.2.1, we average the temporal signal in the region of interest and fit it to the HRF model. The normalized version of the least squares fit is assigned as the first column of the design matrix in linear regression. In this case, the region of interest is the quadrant where the activation area lies. In a real situation, the region of interest would be the functional area of the brain related to the stimulus. The second and third columns are the orthonormalized version of the temporal and dispersion derivatives of the fitted HRF.

For the 50 simulations, we generate empirical receiver operating characteristic (ROC) curves for both our proposed algorithm and the GLM method by varying their thresholds for detecting activation. The correct detection rate is plotted against the false alarm rate in Fig. 3.3 for the Gamma Variate HRF model, and in Fig. 3.4 for the sum of two Gamma functions HRF model. The top and bottom panels in each figure represent the results for the square and circular activation regions, respectively.

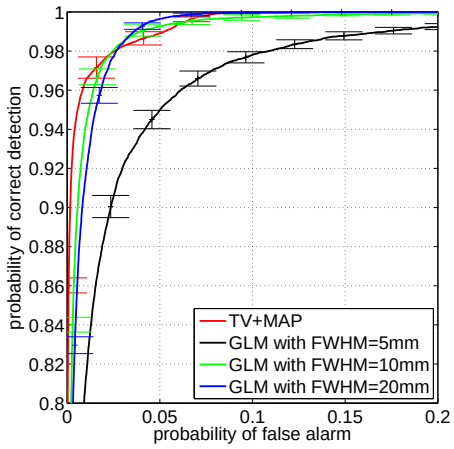


(a) ROC curves for square activation region.

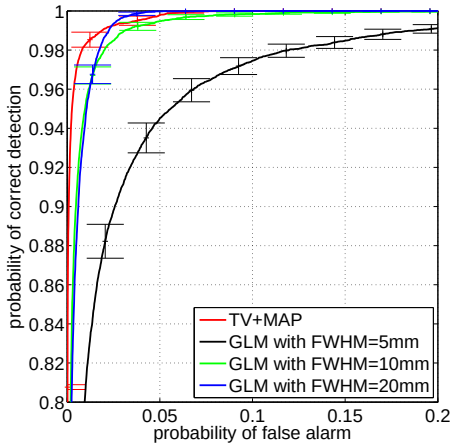


(b) ROC curves for circular activation region.

Fig. 3.3. Performance comparison among different methods on the synthetic data for the Gamma Variate HRF model in Eq. (3.27). Red lines represent ROC curves of proposed algorithm. Black lines represent ROC curves of GLM framework using 5mm FWHM Gaussian kernel for spatial smoothing. Green lines represent ROC curves of GLM framework with 10mm FWHM Gaussian kernel. Blue lines represent ROC curves of GLM framework with 20mm FWHM Gaussian kernel. Error bars on each line indicate $\pm$ twice the standard deviation of the correct detection rates for the corresponding method.



(a) ROC curves for square activation region.



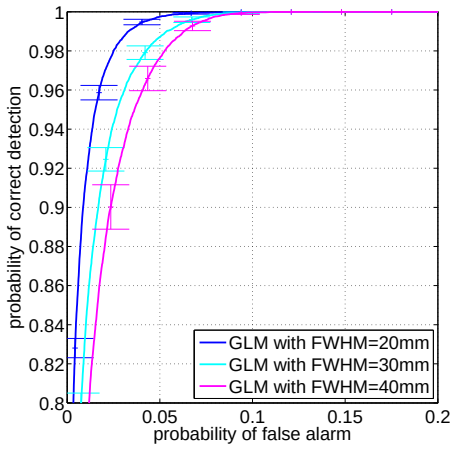
(b) ROC curves for circular activation region.

Fig. 3.4. Performance comparison among different methods on the synthetic data for the sum of two Gamma functions HRF model in Eq. (3.28). Red lines represent ROC curves of proposed algorithm. Black lines represent ROC curves of GLM framework using 5mm FWHM Gaussian kernel for spatial smoothing. Green lines represent ROC curves of GLM framework with 10mm FWHM Gaussian kernel. Blue lines represent ROC curves of GLM framework with 20mm FWHM Gaussian kernel. Error bars on each line indicate $\pm$ twice the standard deviation of the correct detection rates for the corresponding method.

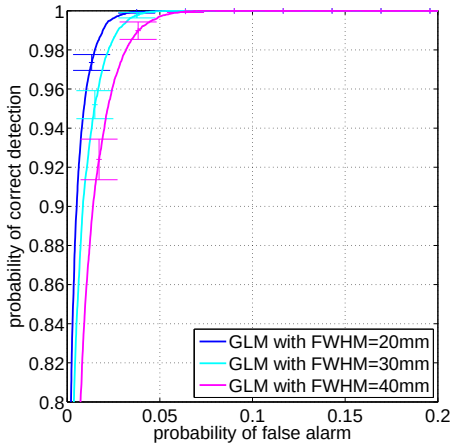
Error bars indicate ± twice the standard deviation of the correct detection rate in 50 Monte-Carlo simulations. In the legend, "TV+MAP" denotes our proposed algorithm, "GLM with FWHM=5mm" denotes the GLM method with 5mm FWHM Gaussian smoothing kernel, "GLM with FWHM=10mm" denotes the GLM method with 10mm FWHM Gaussian smoothing kernel, and "GLM with FWHM=20mm" denotes the GLM method with 20mm FWHM Gaussian smoothing kernel.

From the ROC curves, we can tell that our proposed algorithm outperforms the GLM framework with all three different widths of Gaussian kernel at low false alarm rate. We also try wider smoothing kernels for the GLM method on the same synthetic data as in Fig. 3.3. The performance further degraded as FWHM gets larger than 20mm.

We show an example of the parameter maps before thresholding for both algorithms in Fig. 3.6. The data in this example is one simulation in the synthetic dataset for the Gamma Variate HRF model with circular activation region. Visually, the parameter maps produced by GLM with FWHM=5mm and FWHM=10mm are not as clean as our algorithm, and the parameter map produced by GLM with FWHM=20mm is oversmoothed. Oversmoothed parameter maps tend to produce lower correct detection rate at low false alarm rate, since locality is lost. At high false alarm rate, oversmoothed parameter maps produce high detection rate if the activated pixels are clustered together (as in our case). In order to see which region of the ROC is more meaningful to compare the performance, we need to determine a reasonable threshold to produce the binary activation map. We choose to threshold the parameter maps such that the number of declared active pixels is equal to the truely active pixels, then the binary activation maps are shown in Fig. 3.7. The false alarm rates for this threshold are 0.32%, 0.79%, 0.55%, and 0.66% for our proposed method, GLM with FWHM=5mm, GLM with FWHM=10mm, and GLM with FWHM=20mm respectively. The correct detection rates for this threshold are 96.07%, 90.16%, 93.11%, and 91.80% for our proposed method, GLM with FWHM=5mm, GLM with FWHM=10mm, and GLM with FWHM=20mm respec-

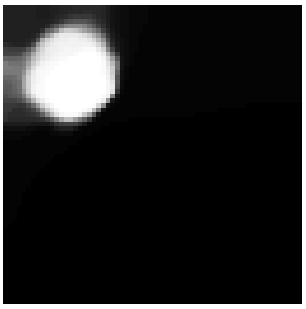


(a) ROC curves for square activation region.

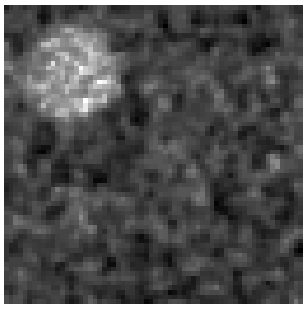


(b) ROC curves for circular activation region.

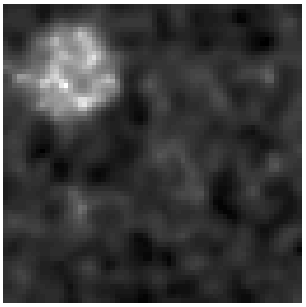
Fig. 3.5. ROC curves of the GLM method for different amount of spatial smoothing. Blue lines represent ROC curves for FWHM=20mm Gaussian kernel. Cyan lines represent ROC curves for FWHM=30mm Gaussian kernel. Magenta lines represent ROC curves for FWHM=40mm Gaussian kernel. Error bars on each line indicate ± twice the standard deviation of the correct detection rates.



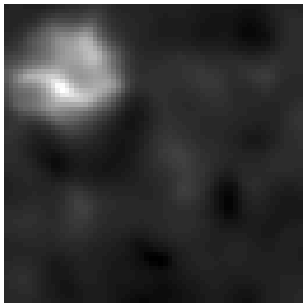
(a) Parameter map produced by TV+MAP.



(b) Parameter map produced by GLM with FWHM=5mm.

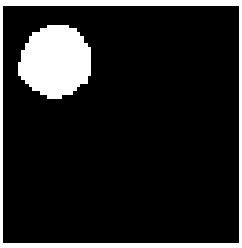


(c) Parameter map produced by GLM with FWHM=10mm.

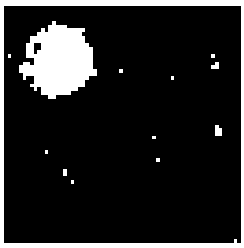


(d) Parameter map produced by GLM with FWHM=20mm.

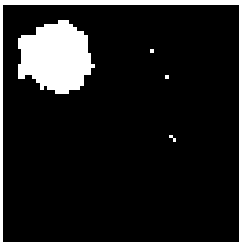
Fig. 3.6. Parameter maps produced by our proposed algorithm and the GLM framework with different extent of spatial smoothing: (a) Amplitude map produced by our proposed algorithm, (b) T-statistic map produced by the GLM method with FWHM=5mm Gaussian smoothing kernel, (c) T-statistic map produced by the GLM method with FWHM=10mm Gaussian smoothing kernel, (d) T-statistic map produced by the GLM method with FWHM=20mm Gaussian smoothing kernel.



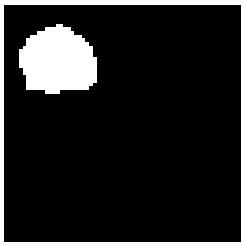
(a) Activation map produced by TV+MAP.



(b) Activation map produced by GLM with FWHM=5mm.



(c) Activation map produced by GLM with FWHM=10mm.



(d) Activation map produced by GLM with FWHM=20mm.

Fig. 3.7. Activation maps produced by our proposed algorithm and the GLM framework with different extent of spatial smoothing: (a) Activation map produced by our proposed algorithm, (b) Activation map produced by the GLM method with FWHM=5mm Gaussian smoothing kernel, (c) Activation map produced by the GLM method with FWHM=10mm Gaussian smoothing kernel, (d) Activation map produced by the GLM method with FWHM=20mm Gaussian smoothing kernel. The threshold is set such that the number of declared active pixels is equal to the number of truely active pixels.

tively. We can see from the ROC curves that our algorithm performs much better than GLM at around 0.5% false alarm rate or around 93% correct detection rate. In real experimental settings, we don't know the number of truely active pixels. So we need to set some threshold after we obtain the parameter map. The thresholding strategy is discussed in Section 3.7.1.

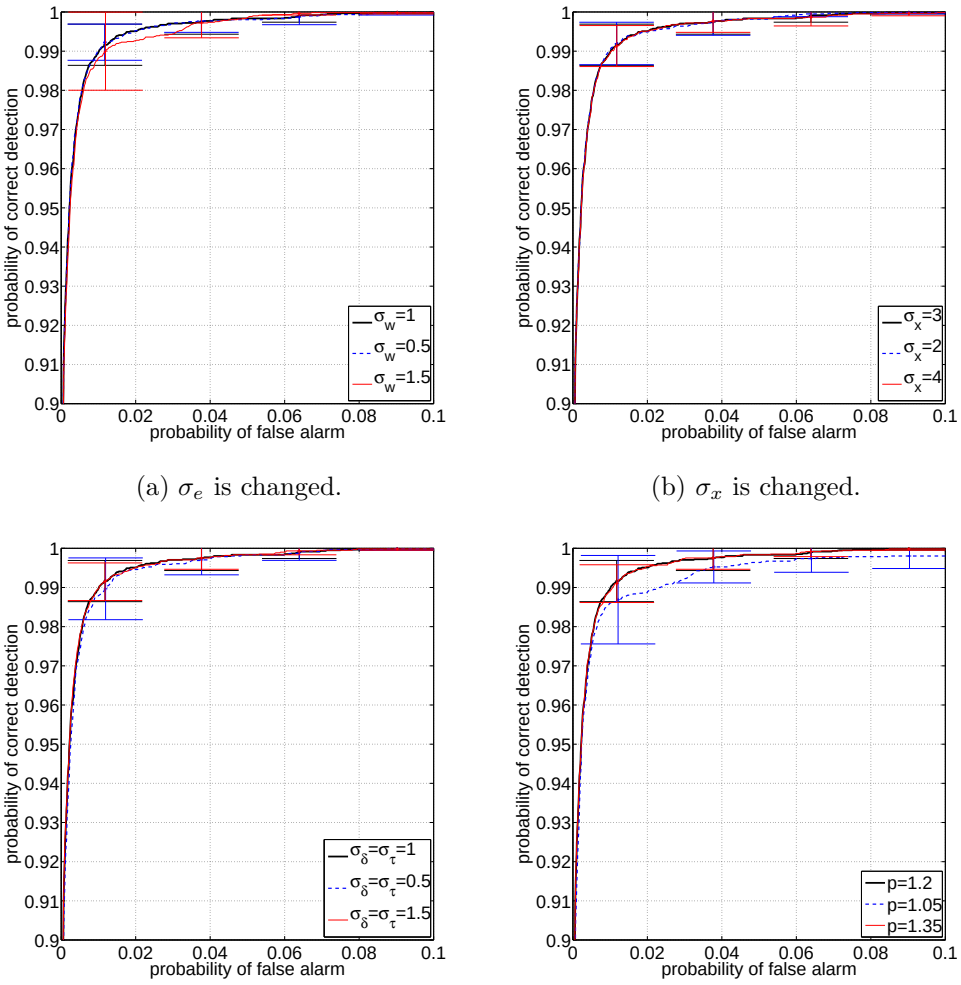
In addition, note that for the GLM framework, the performance degrades a lot when we change the FWHM from 10mm to 5mm. We will show in the next subsection that our algorithm is much more robust to the change of parameters.

3.6.4 Synthetic Data Experiment 2: Insensitivity to Parameter Variations

We now demonstrate that our algorithm is robust to the changes of the regularization parameters. To this end, we vary the regularization parameters σ_e, σ_x, σ_δ, σ_τ, and q in the MAP estimation stage of our algorithm, and recompute the results. We use the dataset with a circular activation region from the previous experiment. We change one or two parameters at a time and keep all others the same as the values previously described. We show the resulting empirical ROC curves in Fig. 3.8. Figure 3.8(a) shows the ROC curves for $\sigma_e = 0.5, 1, 1.5$. Figure 3.8(b) shows the ROC curves for $\sigma_x = 2, 3, 4$. Figure 3.8(c) shows the ROC curves for $\sigma_\delta = \sigma_\tau = 0.5, 1, 1.5$. Figure 3.8(d) shows the ROC curves for $q = 1.05, 1.2, 1.35$. We observe very small variations in ROC curves when the parameters are varied quite substantially. The largest difference in the correct detection rate under the same false alarm rate is less than 1%. None of the differences are statistically significant—i.e., all of them are within the error bars.

3.6.5 Synthetic Data Experiment 3: Robustness to the Incorrect HRF Model

In the first two experiments, the detection algorithms use the same HRF model as the one used for generating the synthetic data. Since in real-data experiments, the "correct" HRF model is not known, activation detection algorithms must not be sensitive to the exact form of the HRF model. In order to demonstrate the robustness of our algorithm, we generate the synthetic data using the Cox waveform [74], and detect the activation with the Gamma Variate HRF model in Eq. (3.27). The simulated dataset we use is the same the one with circular activation region described in Section 3.6.2, except that we replace the true activation waveform with the Cox waveform. The parameter settings are: delay time = 2 seconds, rise time = 3 seconds, fall time = 3 seconds, undershoot = 20%, restore time = 2 seconds, peak amplitude



(a) σ_e is changed.

(b) σ_x is changed.

(c) σ_δ and σ_τ are changed.

(d) q is changed.

Fig. 3.8. Comparison of partial ROC curves with different parameter settings in the regularized HRF parameter estimation stage. (a) σ_e is changed. (b) σ_x is changed. (c) σ_δ and σ_τ are changed. (d) q is changed.

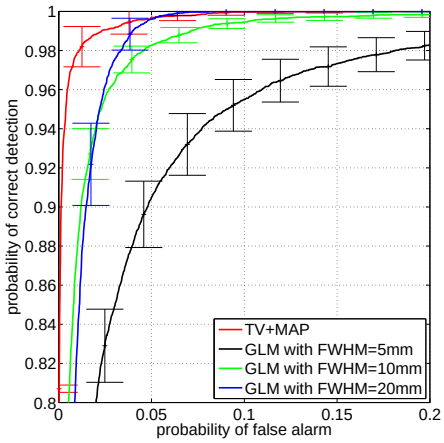


Fig. 3.9. Performance comparison on the synthetic data using the Cox waveform as the activation waveform. Gamma Variate HRF model in Eq. (3.27) is used in detection. Red lines represent ROC curves of proposed algorithm. Black lines represent ROC curves of GLM framework using 5mm FWHM Gaussian kernel for spatial smoothing. Green lines represent ROC curves of GLM framework with 10mm FWHM Gaussian kernel. Blue lines represent ROC curves of GLM framework with 20mm FWHM Gaussian kernel. Error bars on each line indicate $\pm$ twice the standard deviation of the correct detection rates for the corresponding method.

$= 1$. The regularization parameters in our algorithm are the same as those used in Fig. 3.3. The resulting ROC curves are shown in Fig. 3.9. The ROC of our algorithm looks very similar to the one in Fig. 3.3(b), however the ROC curves of the GLM framework is much lower. This means that our algorithm is robust to incorrect modeling of HRF, however the GLM framework suffers a lot from it.

3.6.6 Synthetic Data Experiment 4: Different HRFs at Different Pixels

In the first three experiments, we use the same activation waveform for all activated pixels when generating synthetic data. However, the activation waveform could vary depending on the region of activation [84]. Therefore, assuming spatially constant activation waveform may not be appropriate. In order to test our algorithm

under this scenario, we generate another set of synthetic data similar to the previously used synthetic dataset, but with two activated regions, each with a different HRF. Specifically, the activation regions are two 15×15 squares with the two HRFs shown in Fig. 3.10. The two HRFs are obtained by using different delay and peak times in the Gamma Variate model. We keep all other settings the same as in the previous synthetic dataset. Figure 3.11 shows two frames at the peak times ($t = 3$ and $t = 5$) of the two HRFs in the synthetic dataset averaged over trials. We perform Monte-Carlo simulations and obtain the ROC curves for the four methods mentioned in Section 3.6.2 as in Fig. 3.12. The region of interest for constructing the regressor is the upper half of the image. We can see that the GLM framework performs very poorly in this case, much worse than in the first three experiments. GLM with the best choice of Gaussian kernel width (FWHM=10mm) only achieves 80% correct detection rate under 0.7% false alarm rate, whereas our proposed algorithm is able to achieve 94% correct detection rate under the same false alarm rate. The reason for this is not difficult to see. Incorporating the temporal and dispersion derivatives into the design matrix of GLM helps to capture the variation of delay time and dispersion of activation waveforms. This is in fact equivalent to first order Taylor series approximation. In other words, a small perturbation of the delay time parameter and the dispersion parameter of the HRF can be approximated by first order Taylor series expansion. This linear approximation can be written in matrix form as GX in Eq. 3.1. However, when the true activation waveform is much different from the nominal one, then this approximation is no longer good enough, whereas our parametric model allows enough freedom for the delay time and dispersion parameters to fit the activation waveform. Therefore, the performance of our algorithm is better than GLM when we have different activation waveforms for different pixels.

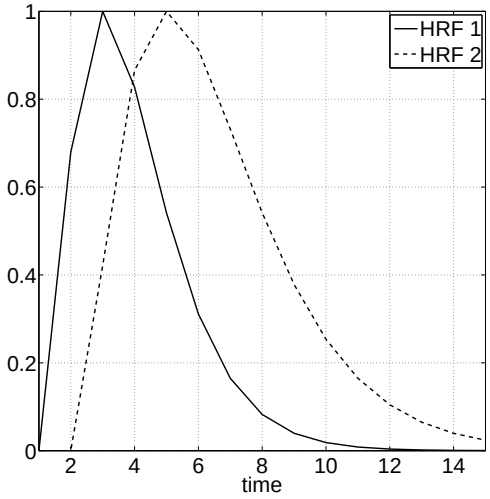
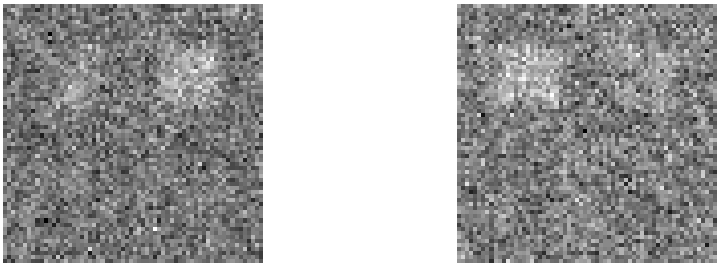


Fig. 3.10. Two HRFs with different delay time and peak time.



(a) $t = 3$ (b) $t = 5$

Fig. 3.11. Two frames at the peak times of two HRFs in the synthetic dataset.

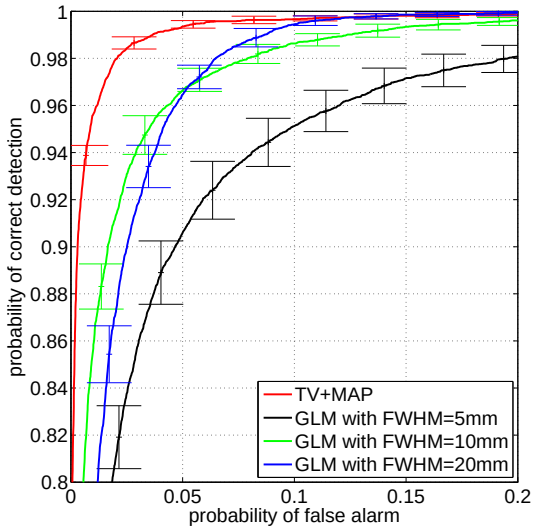


Fig. 3.12. Comparison of ROC curves between four methods on a synthetic dataset with two activation areas, each with a different HRF.

3.7 Real Data Experiments

We use the data from a visual stimulus experiment [59] to test our algorithm. A flashing checkerboard presented to the left and right visual hemifields in an alternating fashion was used as the stimulus. In this event-related paradigm, the stimulus duration was 1s with a 15s inter-stimulus interval. The experiment consisted of nine trials for each type of stimulus, with each trial lasting 15 time points. A block paradigm experiment (30s block duration) was also conducted as a benchmark.

As described in [85] and [59], experiments were conducted at the Indiana University School of Medicine using a 1.5T GE Signa LX Horizon. A visual surface coil (Nova Medical, Inc., Wilmington, MA) was used to obtain images of 10 slices, perpendicular to the Calcarine fissure with slice thickness 3.8mm. A spiral echo-planar imaging (EPI) sequence was used (TR/TE = 1s/40ms, FOV = 24cm^2, matrix = 64×64, flip angle = 90°). Motion correction and second order detrending [63] were conducted using AFNI [74] as preprocessing of the data.

3.7.1 Real Data Experiment 1: Qualitative Evaluation of Activation Regions

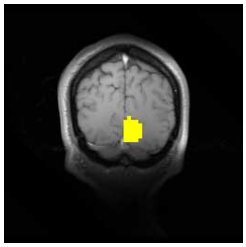
We compare our proposed algorithm and the GLM method using the event-related experimental data. We also use the corresponding block paradigm result obtained with AFNI [74] as benchmark, since the block paradigm maximizes detection power and provides a meaningful ground-truth. In our approach, we use the data corresponding to the right hemifield stimulus as training data to obtain regularization parameters (i.e., we adjust the regularization parameters so that activation is mainly detected in the visual cortex), and apply the same set of parameters on the data corresponding to the left hemifield stimulus. The regularization parameters that we use for the Gamma Variate HRF model in Eq. (3.27) are $\sigma_e = 1$, $\sigma_x = 200$, $\sigma_\delta = 1000$, $\sigma_\tau = 2000$, and $q = 1.2$. The regularization parameters for the two Gamma functions HRF model are $\sigma_e = 0.1$, $\sigma_x = 1$, $\sigma_{a_1} = 200$, $\sigma_{a_2} = 200$, $\sigma_{b_1} = 10$, $\sigma_{b_2} = 10$, $\sigma_c = 5$,

and $q = 1.2$. In order to obtain a binary activation map, we need to determine a reasonable threshold level for all methods under comparison. In the next three subsections, we list three thresholding strategies for our experiment.

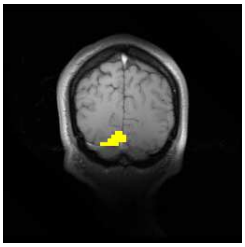
Thresholding p-values

Without any knowledge of the underlying distribution of the amplitude estimates using our approach, we can consider the estimates over the region of no interest and find the threshold corresponding to a certain p-value. In our experiment, we consider the visual cortex as the region of interest. It is assumed that the null hypothesis of no activation is true everywhere inside the region of no interest. Therefore, the threshold on amplitude estimates corresponding to a certain p-value is set such that the proportion of amplitude estimates in this region over this threshold is equal to that p-value. If the region of no interest is not easily identified, then a baseline trial (no stimuli are presented) can be conducted and the threshold can be obtained from that trial. In this experiment, we set the threshold to be p-value of 0.001. Figures. 3.13(a,b) show the detection results of our algorithm using the Gamma Variate model overlayed on top of the 256×256 high-resolution anatomical image. Note that this is just the result for slice no. 6 of the event-related data. We also provide, in Figs. 3.13(c,d), the results for our algorithm when using the sum of two Gamma functions HRF model in Eq. (3.28).

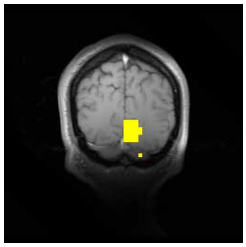
For the GLM method, we use a spatial Gaussian filter to reduce noise as a pre-processing step. The width of the Gaussian kernel is FWHM = 6mm. According to the study in [86], a spatial smoothing of 6mm would work best for cortical activations. The HRF model of Eq. (3.27) is then fitted to the averaged temporal vector over the region of interest (visual cortex), and the fitted curve and its temporal and dispersion derivatives are used as the columns of the design matrix. We compute a t-statistic for each pixel to provide the statistical map. Then using the same method described above, we determine the threshold corresponding to p-value of 0.001. Fig-



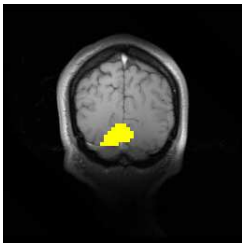
(a) Activation detected by proposed method with Gamma Variate HRF for right hemifield stimulus.



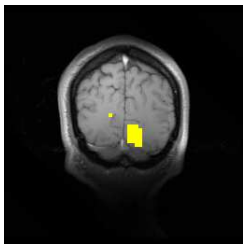
(b) Activation detected by proposed method with Gamma Variate HRF for left hemifield stimulus.



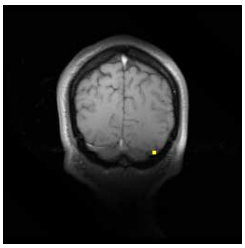
(c) Activation detected by proposed method with sum of two Gamma functions HRF for right hemifield stimulus.



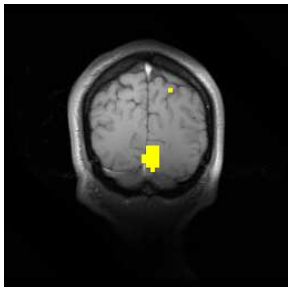
(d) Activation detected by proposed method with sum of two Gamma functions HRF for left hemifield stimulus.



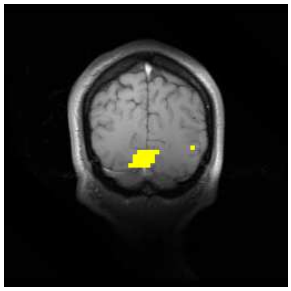
(e) Activation detected by GLM for right hemifield stimulus.



(f) Activation detected by GLM for left hemifield stimulus.



(g) Benchmark result for right hemifield stimulus.



(h) Benchmark result for left hemifield stimulus.

Fig. 3.13. An example of detection results of the event-related experiment using proposed method and GLM method with comparison to benchmark (block paradigm) result. The threshold is chosen corresponding to a p-value of 0.001.

ures. 3.13(e,f) show the results for slice no. 6 of the event-related data overlayed on the high resolution anatomical image.

The benchmark results are obtained using AFNI [74] on the block paradigm data, and are shown in Figs. 3.13(g,h). The design matrix is the Cox waveform [74] with default parameter settings (delay time = 2.0 seconds, rise time = 4.0 seconds, fall time = 6.0 seconds, undershoot = 20%, restore time = 2.0 seconds) convolved with the box-car function corresponding to the paradigm. Note that the same thresholding method is applied to obtain the binary activation map.

Figure 3.13 shows that most of the activations detected by our algorithm is confined to the visual cortex and is close to the benchmark results, whereas the GLM method detects only one active pixel for the left hemifield stimulus as shown in Fig. 3.13(f). Note that the checkerboards presented to the left and right hemifields are spatially contiguous (albeit disjoint), meeting at the center of the subject's visual field. A slight misalignment of the presentation could have produced the observed activation regions that cross the mid-line of the brain.

Thresholding using false discovery rate

Thresholding using false discovery rate (FDR) [87] has been widely used in functional neuroimaging as an approach to the multiple testing problem [88]. FDR based thresholding methods control the expected proportion of false positives among all rejected tests, and they are adaptive to the amount of signal in the data. We follow the method described in [87] to threshold the benchmark results at a FDR of 0.03. The benchmark statistical values at the region of no interest are assumed to follow normal distribution. We compute the sample mean and sample standard deviation of it to obtain a fit to the normal distribution. After obtaining the distribution of the benchmark results under null hypothesis, we can compute the p-value at each pixel location. Note that we do not use the same method as in the previous subsection to compute the empirical p-values, since if we use the same method then any value above

the largest value in the region of no interest will have a p-value of zero, which does not work well for the FDR thresholding strategy. We then follow the steps in [87] to threshold benchmark results using FDR:

1. Sort the p-values in ascending order:

$$p_{(1)} \leq p_{(2)} \leq \cdots p_{(N)},$$

   where N is the number of pixels.

2. Let r be the largest i for which

$$p_{(i)} \leq \frac{i}{N}\frac{f}{C_N},$$

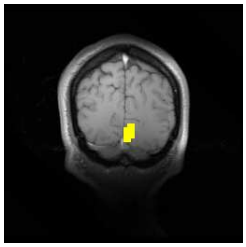
   where $C_N = \sum_{i=1}^{N} \frac{1}{i}$, and f is the FDR we choose to tolerate (0.03 in our case).

3. Reject the null hypothesis of no activation for pixels with p-values less than or equal to $p_{(r)}$.

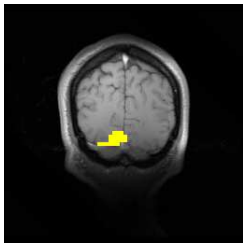
Note that since we do not know the distribution of the amplitude estimates produced by our algorithm, we threshold the our results such that the number of suprathreshold pixels is the same as that in the benchmark experiment. For a fair comparison, we threshold the GLM results in the same fashion. The detection results are shown in Fig. 3.14. Again, we find that compared to the GLM algorithm, our detection results are closer to the benchmark results, and are confined within the primary visual cortex. We can see that with the FDR thresholding strategy, we detect a bit more activated pixels for left hemifield stimulus than right hemifield stimulus. That means the response to the left hemifield stimulus is a little stronger.

Thresholding with k-means clustering

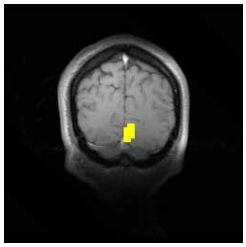
The previous two thresholding methods both need some pre-determined value to differentiate between activated and non-activated pixels. However, we can use k-means clustering [89] to find out the activated pixels. Both GLM method and our



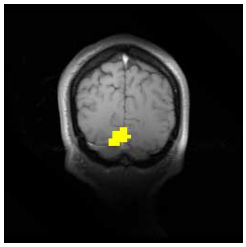
(a) Activation detected by proposed method with Gamma Variate HRF for right hemifield stimulus.



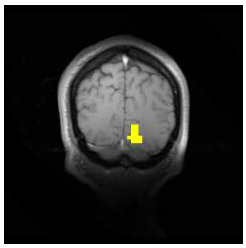
(b) Activation detected by proposed method with Gamma Variate HRF for left hemifield stimulus.



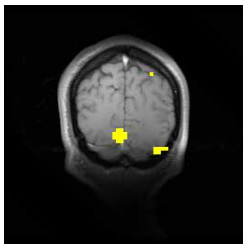
(c) Activation detected by proposed method with sum of two Gamma functions HRF for right hemifield stimulus.



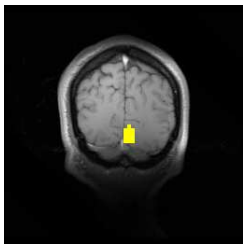
(d) Activation detected by proposed method with sum of two Gamma functions HRF for left hemifield stimulus.



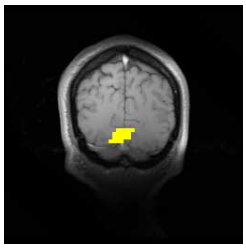
(e) Activation detected by GLM for right hemifield stimulus.



(f) Activation detected by GLM for left hemifield stimulus.



(g) Benchmark result for right hemifield stimulus.



(h) Benchmark result for left hemifield stimulus.

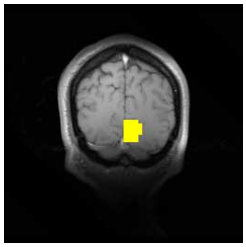
Fig. 3.14. An example of detection results of the event-related experiment using proposed method and GLM method with comparison to benchmark (block paradigm) result. The threshold for the benchmark experiment is chosen corresponding to a FDR of 0.03. The thresholds for the other methods are chosen such that the number of suprathreshold pixels is the same as that in the benchmark experiment.

proposed approach eventually produce a two dimensional statistical map (either the t-statistics map in the GLM method or amplitude estimates $\{\hat{x}_{i,j}\}_{i,j}$ in our algorithm). The larger the pixel value on this map, the more probable that pixel is activated. So we classify pixel values on the statistical map as highly positive, close to zero, and highly negative ($k = 3$ in our case) without using any pre-determined number as threshold level. The highly negative value cluster is to account for the possible negative response from the superior sagittal sinus (the cluster of bright pixels at the top of the brain region close to the inner surface of the frontal bone [90]). We perform k-means clustering on the statistical map obtained from each method, and declare the group of pixels with the largest centroid value as activated pixels. We show the detection results using this strategy in Fig. 3.15.

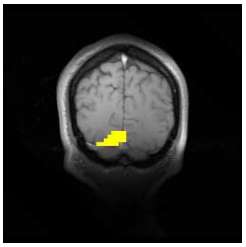
We can see that both our proposed algorithm using the Gamma Variate HRF model and the benchmark experiment produce reasonable detection results using this k-means clustering. Our proposed algorithm using the sum of two Gamma functions HRF model produces a few possible false alarms for the right hemifield stimulus (Fig. 3.15(c)). However, k-means clustering on the GLM results does not provide reasonable activation maps. That is because the GLM method does not produce distinct statistical values at activated pixels. This example also shows the power of our algorithm compared to GLM method. The advantage of using k-means clustering is that the activated pixels are identified without the need of certain pre-determined threshold.

3.7.2 Real Data Experiment 2: Insensitivity to Parameter Variations

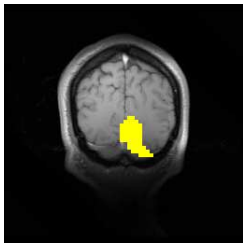
To demonstrate that our algorithm is not very sensitive to the regularization parameters, we vary these parameters over a large range. We find that the detection results remain similar. For example, for the HRF model in Eq. (3.27), we change σ_x to 100 and 10000 and keep all other parameters the same. Figure 3.16 shows the results if we threshold the amplitude estimates according to the p-values. We can



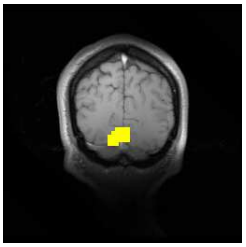
(a) Activation detected by proposed method with Gamma Variate HRF for right hemifield stimulus.



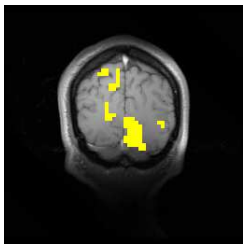
(b) Activation detected by proposed method with Gamma Variate HRF for left hemifield stimulus.



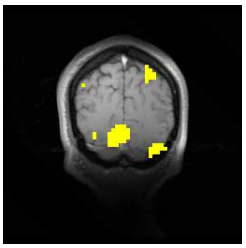
(c) Activation detected by proposed method with sum of two Gamma functions HRF for right hemifield stimulus.



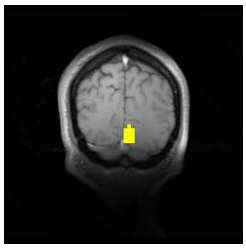
(d) Activation detected by proposed method with sum of two Gamma functions HRF for left hemifield stimulus.



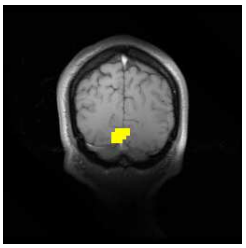
(e) Activation detected by GLM for right hemifield stimulus.



(f) Activation detected by GLM for left hemifield stimulus.

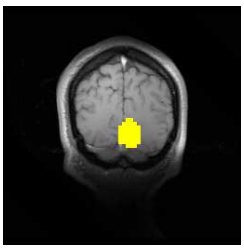


(g) Benchmark result for right hemifield stimulus.

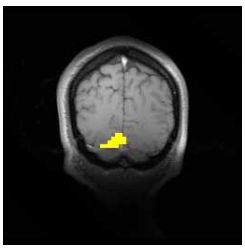


(h) Benchmark result for left hemifield stimulus.

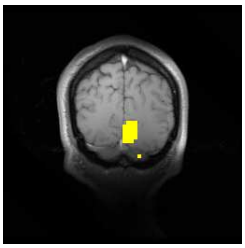
Fig. 3.15. An example of detection results of the event-related experiment using proposed method and GLM method with comparison to benchmark (block paradigm) result. The threshold for all methods are automatically determined from k-means clustering.



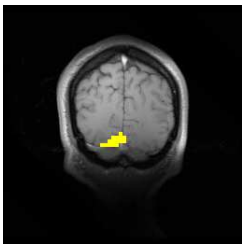
(a) Right hemifield stimulus, $\sigma_x = 100$.



(b) Left hemifield stimulus, $\sigma_x = 100$.



(c) Right hemifield stimulus, $\sigma_x = 10000$.



(d) Left hemifield stimulus, $\sigma_x = 10000$.

Fig. 3.16. Detection results for different regularization parameters. In this example, the amplitude parameter map is thresholded corresponding to p-value of 0.001.

see that the detection results only differ at a few pixel locations. Similar results are obtained when other regularization parameters are varied, i.e., σ_e, σ_δ, σ_τ, and q.

3.8 Conclusion

We have proposed a new forward model for event-related fMRI which explicitly incorporates the effect of blurring introduced by the scanner. We have developed a new activation detection algorithm which uses a total variation minimization algorithm to restore image data and estimates HRF parameters by enforcing spatial regularization through a generalized Gaussian Markov random field model. We have also proposed novel ways of estimating model parameters including the width of the blurring kernel. Our experimental results on synthetic data have shown that our method outperforms GLM by a large margin. Our analysis has shown that our method favorably compares to GLM on real data as well.

APPENDIX

Verification of Stray Light Reduction Algorithm

We show that our restoration formulation in Eq. (1.13) results in a good estimate of the original image x as follows.

Equation (1.13) is in matrix-vector form. We now write the matrix-vector product out in terms of superpositions. So Eq. (1.13) becomes

$$\hat{x}^{(1)}(i_q, j_q) = 2y(i_q, j_q) - (1-\beta)\lambda(i_q, j_q)y(i_q, j_q) - \beta \sum_{i_p} \sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)y(i_p, j_p), \tag{29}$$

where $\lambda(i_q, j_q)$ denotes the shading factor at 2D position (i_q, j_q). According to our assumption that the diffraction-aberration PSF can be approximated by a delta function, so

$$\begin{aligned} y(i_q, j_q) &= \sum_{i_p} \sum_{j_p} ((1-\beta)\delta(i_q - i_p, j_q - j_p) + \beta s(i_q, j_q; i_p, j_p))\lambda(i_p, j_p)x(i_p, j_p), \\ &= (1-\beta)\lambda(i_q, j_q)x(i_q, j_q) + \beta \sum_{i_p} \sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)x(i_p, j_p). \end{aligned} \tag{30}$$

By substituting the above results into Eq. (29), we obtain the following:

$$\begin{aligned} \hat{x}^{(1)}(i_q, j_q) &= 2(1-\beta)\lambda(i_q, j_q)x(i_q, j_q) + 2\beta \sum_{i_p} \sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)x(i_p, j_p) \\ &\quad -(1-\beta)^2\lambda^2(i_q, j_q)x(i_q, j_q) \\ &\quad -(1-\beta)\beta\lambda(i_q, j_q) \sum_{i_p} \sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)x(i_p, j_p) \\ &\quad -\beta(1-\beta) \sum_{i_p} \sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda^2(i_p, j_p)x(i_p, j_p) \\ &\quad -\beta^2 \sum_{i_p} \sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p) \left(\sum_{i'} \sum_{j'} s(i_p, j_p; i', j')\lambda(i', j')x(i', j') \right). \end{aligned}$$

Since stray light only occupies a small portion of the entire PSF, β is very small, $\beta^2 \approx 0$. Thus the above can be approximated as in Eq. (31).

$$\begin{aligned}\hat{x}^{(1)}(i_q, j_q) &\approx (1-\beta)\lambda(i_q, j_q)(2-(1-\beta)\lambda(i_q, j_q))x(i_q, j_q) \\ &\quad +(2\beta-(1-\beta)\beta\lambda(i_q, j_q))\sum_{i_p}\sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)x(i_p, j_p) \\ &\quad -\beta(1-\beta)\sum_{i_p}\sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda^2(i_p, j_p)x(i_p, j_p). \end{aligned} \tag{31}$$

The shading factor is in fact close to 1 for digital cameras, even at the periphery of the image. So $(1-\beta)\lambda(i_q, j_q)$ is close to 1. We can let $(1-\beta)\lambda(i_q, j_q) = 1-\epsilon$, where ϵ is positive and close to 0. Then we have the following:

$$\begin{aligned}\hat{x}^{(1)}(i_q, j_q) &\approx (1-\epsilon^2)x(i_q, j_q) \\ &\quad +(2\beta-(1-\beta)\beta\lambda(i_q, j_q))\sum_{i_p}\sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)x(i_p, j_p) \\ &\quad -\beta(1-\beta)\lambda(i_q, j_q)\sum_{i_p}\sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)x(i_p, j_p) \\ &\quad +\beta(1-\beta)\lambda(i_q, j_q)\sum_{i_p}\sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)x(i_p, j_p) \\ &\quad -\beta(1-\beta)\sum_{i_p}\sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda^2(i_p, j_p)x(i_p, j_p), \\ &= (1-\epsilon^2)x(i_q, j_q) + 2\beta\epsilon\sum_{i_p}\sum_{j_p} s(i_q, j_q; i_p, j_p)\lambda(i_p, j_p)x(i_p, j_p) \\ &\quad +\beta(1-\beta)\sum_{i_p}\sum_{j_p} s(i_q, j_q; i_p, j_p)(\lambda(i_q, j_q)-\lambda(i_p, j_p))\lambda(i_p, j_p)x(i_p, j_p). \end{aligned}$$

Since $|\lambda(i_q, j_q) - \lambda(i_p, j_p)| < \epsilon$, if we drop all the second order small terms with $\beta\epsilon$, we get the final approximation result as in Eq. (32).

$$\hat{x}^{(1)}(i_q, j_q) \approx x(i_q, j_q). \tag{32}$$

LIST OF REFERENCES

LIST OF REFERENCES

[1] W. J. Smith, *Modern Optical Engineering : the Design of Optical Systems*. New York, NY: McGraw Hill, 2000.

[2] S. F. Ray, *Applied Photographic Optics*. Woburn, MA: Focal Press, 2002.

[3] P. A. Jansson and R. P. Breault, "Correcting color-measurement error caused by stray light in image scanners," in *Proceedings of the Sixth Color Imaging Conference: Color Science, Systems, and Applications*, (Scottsdale, AZ), November 1998.

[4] P. A. Jansson, "Method, program, and apparatus for efficiently removing stray-flux effects by selected-ordinate image processing," *US Patent 6,829,393*, 2004.

[5] P. A. Jansson and J. H. Fralinger, "Parallel processing network that corrects for light scattering in image scanners," *US Patent 5,153,926*, 1992.

[6] B. Bitlis, P. A. Jansson, and J. P. Allebach, "Parametric point spread function modeling and reduction of stray light effects in digital still cameras," in *Proceedings of the SPIE/IS&T Conference on Computational Imaging VI*, vol. 6498, (San Jose, CA), January 2007.

[7] J. Wei, B. Bitlis, A. Bernstein, A. Silva, P. A. Jansson, and J. P. Allebach, "Stray light and shading reduction in digital photography – a new model and algorithm," in *Proceedings of the SPIE/IS&T Conference on Digital Photography IV*, vol. 6817, (San Jose, CA), January 2008.

[8] P. Y. Maeda, P. B. Catrysse, and B. A. Wandell, "Integrating lens design with digital camera simulation," in *Proceedings of the SPIE/IS&T Conference on Digital Photography*, vol. 5678, (San Jose, CA), January 2005.

[9] P. Y. Bely, M. Lallo, L. Petro, K. Parrish, K. Mehalick, C. Perrygo, G. Peterson, R. Breault, and R. Burg, *Straylight Analysis of the Yardstick Mission*. Greenbelt, MD: Goddard Space Flight Center, 1999.

[10] P. H. V. Cittert, "Zum einfluss der spaltbreite auf die intensitatswerteilung in spektrallinien," *Z. Physik*, vol. 69, pp. 298–308, 1931.

[11] S. Lin, T. F. Krile, and J. F. Walkup, "Piecewise isoplanatic modeling of space-variant linear systems," *Journal of the Optical Society of America*, vol. 4, no. 3, pp. 481–487, 1987.

[12] R. J. M. II and T. F. Krile, "Holographic representation of space-variant systems: system theory," *Applied Optics*, vol. 15, no. 9, pp. 2241–2245, 1976.

[13] G. Cao, C. A. Bouman, and K. J. Webb, "Fast and efficient stored matrix techniques for optical tomography," in *Proceedings of the 40th Asilomar Conference on Signals, Systems, and Computers*, October 2006.

[14] M. G. Kendall, *Advanced Theory of Statistics*, vol. 1. New York, NY: Halsted Press, 1994.

[15] P. S. R. Diniz, E. A. B. da Silva, and S. L. Netto, *Digital signal processing system analysis and design*. New York, NY: Cambridge University Press, 2002.

[16] T. H. Coleman and Y. Li, "An interior, trust region approach for nonlinear minimization subject to bounds," *SIAM Journal on Optimization*, vol. 6, pp. 418–445, 1996.

[17] *MATLAB R2007a*. Natick, MA: The MathWorks.

[18] P. A. Jansson, *Deconvolution of Images and Spectra*. New York, NY: Academic Press, 1996.

[19] J. G. Nagy and D. P. O'Leary, "Fast iterative image restoration with a spatially varying psf," in *Proceedings of SPIE Conference on Advanced Signal Processing: Algorithms, Architectures, and Implementations VII*, vol. 3162, (San Diego, CA), pp. 388–399, 1997.

[20] J. Bardsley, S. Jefferies, J. Nagy, and R. Plemmons, "A computational method for the restoration of images with an unknown, spatially-varying blur," *Optics Express*, vol. 15, no. 5, pp. 1767–1782, 2006.

[21] P. S. Heckbert, "Color image quantization for frame buffer display," in *Proceedings of SIGGRAPH 1982*, (Boston, MA), pp. 297–307, 1982.

[22] W. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust. Speech Sig. Process.*, vol. 37, pp. 1568–1575, 1989.

[23] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Communications*, vol. 28, pp. 84–95, 1980.

[24] J. Vaisey and A. Gersho, "Simulated annealing and codebook design," in *Int. Conf. on Acoustics, Speech and Signal Processing*, (New York, NY), pp. 1176–1179, April 1988.

[25] K. Zeger, J. Vaisey, and A. Gersho, "Globally optimal vector quantizer design by stochastic relaxation," *IEEE Trans. Signal Process.*, vol. 40, no. 2, pp. 310–322, 1992.

[26] A. Bovik, *Handbook of image and video processing*. San Diego, CA: Academic Press, 2000.

[27] G. Cao, C. A. Bouman, and K. J. Webb, "Results in non-iterative map reconstruction for optical tomography," in *Proceedings of the SPIE/IS&T Conference on Computational Imaging VI*, vol. 6814, (San Jose, CA), January 2008.

[28] D. Tretter and C. A. Bouman, "Optimal transforms for multispectral and multilayer image coding," *IEEE Trans. on Image Processing*, vol. 4, no. 3, pp. 296–308, 1995.

[29] S. Mallat, *A Wavelet Tour of Signal Processing*. San Diego, CA: Academic Press, 1998.

[30] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. on Image Processing*, vol. 1, no. 2, pp. 205–220, 1992.

[31] J. Wei, G. Cao, C. A. Bouman, and J. P. Allebach, "Fast space-varying convolution and its application in stray light reduction," in *Proceedings of the SPIE/IS&T Conference on Computational Imaging VII*, vol. 7246, (San Jose, CA), February 2009.

[32] R. F. Harrington, *Field computation by moment method*. New York, NY: IEEE Press, 1993.

[33] G. Golub and C. V. Loan, *Matrix Computations*. Baltimore, MD: The Johns Hopkins University Press, 1996.

[34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[35] G. Cao, C. A. Bouman, and K. J. Webb, "Noniterative map reconstruction using sparse matrix representations," *IEEE Transactions on Image Processing*, vol. 18, pp. 2085–2099, 2009.

[36] W. M. Gentleman, "Least squares computations by givens transformations without square roots," *IMA Journal of Applied Mathematics*, vol. 12, no. 3, pp. 329–336, 1973.

[37] D. Bau and L. N. Trefethen, *Numerical Linear Algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.

[38] W. Givens, "Computation of plane unitary rotations transforming a general matrix to triangular form," *Journal of the Society for Industrial and Applied Mathematics*, vol. 6, no. 1, pp. 26–50, 1958.

[39] L. R. Bachega, G. Cao, and C. A. Bouman, "Fast signal analysis and decomposition on graphs using the sparse matrix transform," in *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, (Dallas, TX), March 2010.

[40] H. Plassman, J. O'Doherty, B. Shiv, and A. Rangel, "Marketing actions can modulate neural representations of experienced pleasantness," *Proceedings of the National Academy of Sciences*, vol. 105, no. 3, pp. 1050–1054, 2008.

[41] S. M. McClure, J. Li, D. Tomlin, K. S. Cypert, L. M. Montague, and P. R. Montague, "Neural correlates of behavioral preference for culturally familiar drinks," *Neuron*, vol. 44, pp. 379–387, 2004.

[42] C. Stippich, N. Rapps, J. Dreyhaupt, A. Durst, B. Kress, E. Nenning, V. Tronnier, and K. Sartor, "Localizing and lateralizing language in patients with brain tumors: Feasibility of routine preoperative functional MR imaging in 81 consecutive patients," *Radiology*, vol. 243, no. 3, pp. 828–836, 2007.

[43] F. B. Mohamed, S. H. Faro, N. J. Gordon, S. M. Platek, H. Ahmad, and J. M. Williams, "Brain mapping of deception and truth telling about an ecologically valid situation: Functional MRI imaging and polygraph investigation—initial experience," *Radiology*, vol. 238, no. 2, pp. 679–688, 2006.

[44] C. Davatzikos, K. Ruparel, Y. Fan, D. G. Shen, M. Acharyya, J. Loughead, R. Gur, and D.D.Langleben, "Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection," *NeuroImage*, vol. 28, pp. 663–668, 2005.

[45] B. D. Martino, D. Kumaran, B. Seymour, and R. Dolan, "Frames, biases, and rational decision-making in the human brain," *Science*, vol. 313, no. 5787, pp. 684–687, 2006.

[46] J.-D. Haynes, K. Sakai, G. Rees, S. Gilbert, C. Frith, and R. E. Passingham, "Reading hidden intentions in the human brain," *Current Biology*, vol. 17, pp. 323–328, 2007.

[47] K. N. Kay, T. Naselaris, R. J. Prenger, and J. L. Gallant, "Identifying natural images from human brain activity," *Nature*, vol. 452, pp. 352–356, 2008.

[48] D. Hassabis, C. Chu, G. Rees, N. Weiskopf, P. D. Molyneux, and E. A. Maguire, "Decoding neuronal ensembles in the human hippocampus," *Current Biology*, vol. 19, pp. 546–554, 2009.

[49] K. Preuschoff, P. Bossaerts, and S. Quartz, "Neural differentiation of expected reward and risk in human subcortical structures," *Neuron*, vol. 51, pp. 381–390, 2006.

[50] S. Berthoz, J. L. Armony, R. J. R. Blair, and R. J. Dolan, "An fMRI study of intentional and unintentional (embarrassing) violations of social norms," *Brain*, vol. 125, pp. 1696–1708, 2002.

[51] H. Takahashi, M. Matsuura, N. Yahata, M. Koeda, T. Suhara, and Y. Okubo, "Men and women show distinct brain activations during imagery of sexual and emotional infidelity," *NeuroImage*, vol. 32, pp. 1299–1307, 2006.

[52] M. A. Burock and A. M. Dale, "Estimation and detection of event-related fMRI signals with temporally correlated noise: a statistically efficient and unbiased approach," *Human Brain Mapping*, vol. 11, pp. 249–260, 2000.

[53] X. Descombes, F. Kruggel, and D. Y. V. Cramon, "Spatial-temporal fMRI analysis using Markov Random fields," *IEEE Transactions on Medical Imaging*, vol. 17, no. 6, pp. 1028–1039, 1998.

[54] K. J. Friston, A. P. Holmes, K. J. Worsley, J. P. Poline, C. D. Frith, and R. S. J. Frackowiak, "Statistical parametric maps in functional imaging: A general linear approach," *Human Brain Mapping*, vol. 2, pp. 189–210, 1995.

[55] K. J. Friston, P. Fletcher, O. Josephs, A. Holmes, M. D. Rugg, and R. Turner, "Event-related fMRI: Characterizing differential responses," *NeuroImage*, vol. 7, pp. 30–40, 1998.

[56] O. Josephs, R. Turner, and K. J. Friston, "Event-related fMRI," *Human Brain Mapping*, vol. 5, pp. 243–248, 1997.

[57] J. Kim, J. W. F. III, A. Tsai, C. Wible, A. Willsky, and W. M. W. III, "Incorporating spatial priors into an information theoretic approach for fMRI data analysis," in *Proceedings of MICCAI*, pp. 62–71, 2000.

[58] T. Deneux and O. Faugeras, "Using nonlinear models in fMRI data analysis: Model selection and activation detection," *NeuroImage*, vol. 32, pp. 1669–1689, 2006.

[59] A. A. Rao and T. M. Talavage, "Clustering of fMRI data for activation detection using hdr models," in *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, pp. 1876–1879, 2004.

[60] J. Wei, T. M. Talavage, and I. Pollak, "Modeling and activation detection in fMRI data analysis," in *Proceedings of IEEE/SP 14th Workshop on Statistical Signal Processing*, (Madison, WI), August 2007.

[61] J. Wei and I. Pollak, "A new framework for fMRI data analysis: Modeling, image restoration, and activation detection," in *Proceedings of IEEE International Conference on Image Processing*, (San Antonio, TX), September 2007.

[62] K. J. Worsley, C. H. Liao, J. Aston, V. Petre, G. H. Duncan, F. Morales, and A. C. Evans, "A general statistical analysis for fMRI data," *NeuroImage*, vol. 15, pp. 1–15, 2002.

[63] P. Jezzard, P. M. Matthews, and S. M. Smith, *Functional MRI: An introduction to methods*. New York, NY: Oxford University Press, 2001.

[64] A. M. Dale and R. L. Buckner, "Selective averaging of rapidly presented individual trials using fMRI," *Human Brain Mapping*, vol. 5, pp. 329–340, 1997.

[65] G. H. Glover, "Deconvolution of impulse respose in event-related BOLD fMRI," *NeuroImage*, vol. 9, pp. 416–429, 1999.

[66] P. L. Purdon, V. Solo, R. M. Weisskoff, and E. N. Brown, "Locally regularized spatiotemporal modeling and model comparison for functional MRI," *NeuroImage*, vol. 14, pp. 912–923, 2001.

[67] D. Goldfarb and W. Yin, "Second-order cone programming methods for total variation-based image restoration," *SIAM J. Sci. Comput.*, vol. 27, no. 2, pp. 622–645, 2005.

[68] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.

[69] D. Strong and T. Chan, "Edge-preserving and scale-dependent properties of total variation regularization," *Inverse Problems*, vol. 19, pp. S165–S187, 2003.

[70] H. L. V. Trees, *Detection, Estimation, and Modulation Theory, Part I.* John Wiley & Sons, 2001.

[71] C. A. Bouman and K. Sauer, "A generalized Gaussian image model for edge-preserving MAP estimation," *IEEE Trans. on Image Processing*, vol. 2, no. 3, pp. 296–310, 1993.

[72] B. R. Rosen, R. L. Buckner, and A. M. Dale, "Event-related functional MRI: Past, present, and future," in *Proceedings of Natl. Acad. Sci. USA*, vol. 95, pp. 773–780, 1998.

[73] V. D. Calhoun, M. C. Stevens, G. D. Pearlson, and K. A. Kiehl, "fMRI analysis with the general linear model: removal of latency-induced amplitude bias by incorportation of hemodynamic derivative terms," *NeuroImage*, vol. 22, pp. 252–257, 2004.

[74] R. W. Cox, "AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages," *Computers and Biomedical Research*, vol. 29, pp. 162–173, 1996.

[75] E. R. Cosman, J. W. Fisher, and W. M. Wells, "Exact MAP activity detection in fMRI using a GLM with an Ising spatial prior," in *Proceedings of Medical Image Computing and Computer-Assisted Intervention 2004*, pp. 703–710, 2004.

[76] C. Long, E. N. Brown, D. Manoach, and V. Solo, "Spatiotemporal wavelet analysis for functional MRI," *NeuroImage*, vol. 23, pp. 500–516, 2004.

[77] W. Ou and P. Golland, "From spatial regularization to anatomical priors in fMRI analysis," in *Proceedings of Information Processing in Medical Imaging*, vol. 19, pp. 88–100, 2005.

[78] M. W. Woolrich, M. Jenkinson, J. M. Brady, and S. M. Smith, "Fully Bayesian spatio-temporal modeling of fMRI data," *IEEE Transactions on Medical Imaging*, vol. 23, pp. 213–231, 2004.

[79] F. Kruggel and D. Y. von Cramon, "Modeling the hemodynamic response in single-trial functional MRI experiments," *Magnetic Resonance in Medicine*, vol. 42, pp. 787–797, 1999.

[80] Z.-P. Liang and P. C. Lauterbur, *Principles of magnetic resonance imaging: a signal processing perspective*. New York, NY: IEEE Press, 2000.

[81] E. D. Andersen, C. Rois, and T. Terlaky, "On implementing a primal-dual interior-point method for conic quadratic optimization," *Math. Program*, vol. 95, no. 2, pp. 249–277, 2003.

[82] C. A. Bouman and K. Sauer, "A unified approach to statistical tomography using coordinate descent optimization," *IEEE Trans. on Image Processing*, vol. 5, no. 3, pp. 480–492, 1996.

[83] Y. Lu, T. Jiang, and Y. Zang, "Single-trial variable model for event-related fMRI data analysis," *IEEE Transactions on Medical Imaging*, vol. 24, no. 2, pp. 236–245, 2005.

[84] R. M. Birn, P. A. Bandettini, R. W. Cox, and R. Shaker, "Event-related fMRI of tasks involving brief motion," *Human Brain Mapping*, vol. 7, pp. 106–114, 1999.

[85] L. Liu, A. A. Rao, and T. M. Talavage, "Regional approach to fMRI data analysis using hemodynamic response modeling," in *Proceedings of the SPIE/IS&T Conference on Computational Imaging VI*, vol. 6498, (San Jose, CA), January 2007.

[86] J. B. Hopfinger, C. Buchel, A. P. Holmes, and K. J. Friston, "A study of analysis parameters that influence the sensitivity of event-related fmri analyses," *NeuroImage*, vol. 11, pp. 326–333, 2000.

[87] C. R. Genovese, N. A. Lazar, and T. E. Nichols, "Thresholding of statistical maps in functional neuroimaging using the false discovery rate," *NeuroImage*, vol. 15, pp. 870–878, 2002.

[88] T. Nichols and S. Hayasaka, "Controlling the familywise error rate in functional neuroimaging: a comparative review," *Statistical Methods in Medical Research*, vol. 12, no. 5, pp. 419–446, 2003.

[89] D. J. C. MacKay, *Information theory, inference, and learning algorithms*. New York, NY: Cambridge University Press, 2003.

[90] T. Scarabino, U. Salvolini, and T. Scarabino, *Atlas of morphology and functional anatomy of the brain*. New York, NY: Springer-Verlag, 2006.