Gradient Based Mrf Learning For Image Restoration And Segmentation

Kegan Samuel
University of Central Florida

GRADIENT BASED MRF LEARNING FOR IMAGE RESOTORATION AND SEGMENTATION

by

KEGAN G. G. SAMUEL
B.S. Morehouse College
M.S. Rensselaer Polytechnic Institute
M.S. University of Central Florida

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2012

Major Professor: Marshall F. Tappen

ABSTRACT

The undirected graphical model or Markov Random Field (MRF) is one of the more popular models used in computer vision and is the type of model with which this work is concerned. Models based on these methods have proven to be particularly useful in low-level vision systems and have led to state-of-the-art results for MRF-based systems. The research presented will describe a new discriminative training algorithm and its implementation.

The MRF model will be trained by optimizing its parameters so that the minimum energy solution of the model is as similar as possible to the ground-truth. While previous work has relied on time-consuming iterative approximations or stochastic approximations, this work will demonstrate how implicit differentiation can be used to analytically differentiate the overall training loss with respect to the MRF parameters. This framework leads to an efficient, flexible learning algorithm that can be applied to a number of different models.

The effectiveness of the proposed learning method will then be demonstrated by learning the parameters of two related models applied to the task of denoising images. The experimental results will demonstrate that the proposed learning algorithm is comparable and, at times, better than previous training methods applied to the same tasks.

A new segmentation model will also be introduced and trained using the proposed learning method. The proposed segmentation model is based on an energy minimization framework that is novel in how it incorporates priors on the size of the segments in a way that is straightforward to implement. While other methods, such as normalized cuts, tend to produce segmentations of similar sizes, this method is able to overcome that problem and produce more realistic segmentations.

ACKNOWLEDGMENTS

First and foremost, I would like to God for giving me this opportunity and the strength to complete this journey successfully.

I am extremely grateful to my advisor, Dr. Marshall F. Tappen for his guidance, support and patience(especially during the last two years). I can sincerely say that none of this would have been possible without his assistance. Thanks to my committee members, Dr. Hassan Foroosh, Dr. Niels Da Vitoria Lobo and Dr. Xin Li for their time and feedback to this dissertation.

I would also like to thank all my friends I have met in college and graduate school, including Tyron, Yogya, Kingsley, Daryn, Zain and the many others who I have worked with. Thanks to all my teachers at the Augier Combined School and Vieux-Fort Comprehensive Secondary School.

Finally, the people I reserve my greatest thanks. I would like to thank my family. My wife, Cassandra, for her love, support and patience. My son, Isaac, for his pleasant and joyful distractions. I would like to thank my parents Giles and Ramona Samuel for their prayers, support and encouragement over the years. I would also like to thank my siblings: Javed, Cedric, Gilna and Kesan. A special thanks goes to my uncle, Stanley. Without his sacrifice things would have turned out completely different. I am also grateful to my parents-in-law and my extended family for their support.

TABLE OF CONTENTS

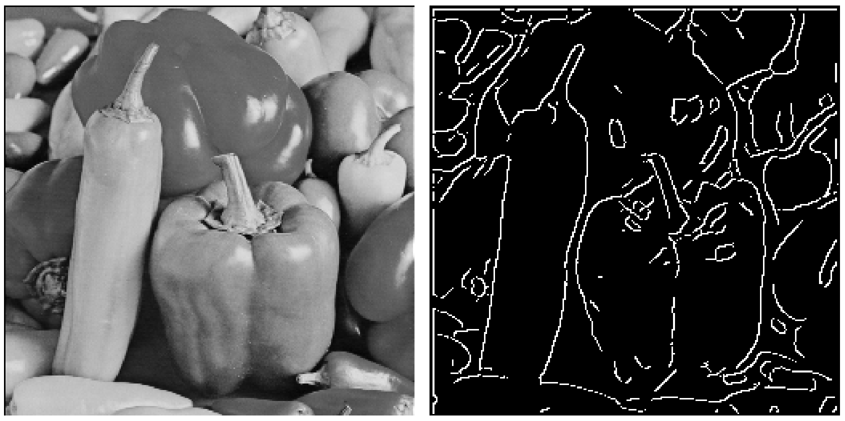
LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The human visual system can perceive and understand the world around with apparently very little difficulty. In the field of computer vision , researchers attempt process and understand digital representation of images in a similar manner using various computer algorithms or models [Mar82]. While much has been accomplished in the field, there is still a significant amount of work to be done before a system is developed which could be able to interpret images at a level remotely close to that of humans.

The problems in the computer vision field can be divided into two areas: low-level and high-level vision. In low-level vision one is concerned with understanding an image at a pixel level. Some examples of low-level vision tasks may include image segmentation and edge detection. On the other hand, high-level vision is concerned with providing a richer description and understanding of an image. One example of high-level vision is scene and object recognition. High-level vision applications typically use the low-level results to provide this semantic description of what is ’seen’ in the image. As shown in Figure 1.1, a low-level task might be to find the edges in the image. A high-level vision task, on the other hand, might be to classify the image, that is, to determine that it is an images of peppers.



(a) (b)

Figure 1.1: A low-level vision task might be to find the edges in (a). The edges found using the Canny algorithm are shown in (b). A high-level vision task, on the other hand, might be to determine what is actually in the image, that is, it is an image of peppers.

The focus of this work will lie in the area of low-level vision. One of the tools that has a wide range of applications in low-level computer vision is the graphical model. These probabilistic models fall into two categories: directed(Bayesian networks) or undirected(Markov Random Fields). The undirected graphical model or Markov Random Field (MRF) is one of the more popular models used in computer vision and is the model with which this work is concerned. Models based on these methods have proven to be particularly useful in low-level vision systems and have led to state-of-the-art results for MRF-based systems. These models depend on various parameters whose values have to be determined manually or by training the model over some test data. One of the main contributions of this work will be a new approach for discriminatively training MRF

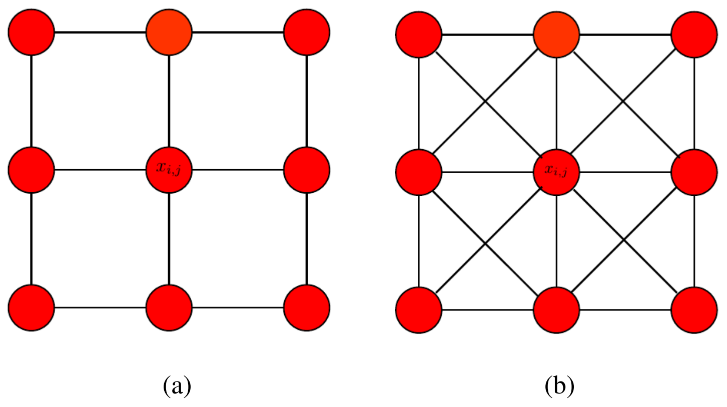


Figure 1.2: This figure shows (a) a simple 4 edge connected setup and (b) a more complex 8 edge connected setup between pixels. The number of edges are based on the number of pixels connected to the center node $x_{i,j}$.

parameters. It will be shown that this proposed learning algorithm is flexible and, at the same time, efficient in learning parameters for the MRF.

1.1 Motivation

A Markov Random Field consists of a set of nodes which correspond to variables. These nodes are connected to each other with undirected links. In vision applications, the MRF nodes can conveniently correspond to pixels or groups of pixels. Figure 1.2 shows two examples of possible setups with pixel $x_{i,j}$ as the center node. Figure 1.2a the center node $x_{i,j}$ is connected to four other pixels. In Figure 1.2b, the center node is connected to eight other pixels. The connections between pixels are undirected.

1.1.1 Markov Random Fields

Markov Random Fields are formally defined as follows:

If F is a family of random variables on the set S then F is said to be a Markov Random Field [Li01] on S with respect to a neighborhood system N if and only if the following two conditions are satisfied:

$$P(f) > 0, \forall \in F \qquad \text{Positivity} \tag{1.1}$$

and

$$P(f_i | f_{S \backslash i}) = P(f_i | f_{N_i}) \qquad \text{Markovianity} \tag{1.2}$$

Here f_i represents the i^{th} configuration of F and $S \backslash i$ is the set of all variables except i. N_i represents the all neighbours of i.

Typically, MRF models are posed in a probabilistic framework and have been shown to be mathematically equivalent to Gibbs distributions [Mou74, HC71]. The Hammersley and Clifford theorem allows us to specify a MRF as a Gibbs distribution as

$$p(\mathbf{x}; \theta) = \frac{1}{Z} \exp \left(- \sum_{k=1}^{N_c} V_k(\mathbf{x}_k; \theta) \right) \tag{1.3}$$

where the sum is over all N_c cliques, k, in the graph. A set of nodes form a clique when all pairs of nodes in the set are connected by a link. Each function is a clique potential function associated with clique $\mathbf{x}_k$. The potential functions are also dependent on a set of parameters, θ. The term $\frac{1}{Z}$, also known as the partition function, is a normalization constant that ensures the probability density integrates or sums to 1 and is defined as

$$Z = \sum_{\mathbf{x}} \exp\left(-\sum_{k=1}^{N_c} V_k(\mathbf{x}_k;\theta)\right) \tag{1.4}$$

Equation 1.3 is usually represented using an energy function $E(\mathbf{x}_k;\theta)$ as

$$p(\mathbf{x};\theta) = \frac{1}{Z(\theta)} \exp\left(-E(\mathbf{x};\theta)\right) \tag{1.5}$$

where

$$Z = \sum_{\mathbf{x}} \exp\left(-E(\mathbf{x};\theta)\right) \tag{1.6}$$

and

$$E(\mathbf{x};\theta) = \sum_{k=1}^{N_c} V_k(\mathbf{x}_k;\theta) \tag{1.7}$$

1.1.2 Inference

The presence of Z leads to a key problem encountered in the use of MRF models: inference and parameter estimating. Inference is the task of determining the unobserved or hidden variables given the observed variables. This can be viewed as either finding the solution with the highest probability, often referred to as the maximum-a-posteriori (MAP) solution, or the task of estimating the marginal probability distribution of every variable in the MRF.

The fundamental difficulty in working with MRF models is the fact that when the graph representation of the MRF has loops or cycles, both types of inference are, in all but a few cases, NP-complete and intractable for the size of problems that are typically encountered in low-level vision[BVZ01, Wai06]. This is due to the fact that Z has to be computed by summing over all possible configurations; a task which is intractable with most problems in low-level vision. This practical intractability has large ramifications on the problem of estimating MRF model parameters and many MRF models usually have parameters which are hand-specified due to the fact that inference is usually intractable.

Unlike maximum-likelihood learning in MRF models, which require the ability to estimate the marginal probability distributions of the node cliques in the model, other non-probabilistic methods such as [Col02] or [TLJ06a], require the ability to find the MAP solution. The method presented in this thesis is based on a non-probabilistic method.

1.2 Goals and Research Outline

This thesis[1] will introduce a new approach for discriminatively training MRF parameters. The MRF model will be trained by optimizing its parameters so that the minimum energy solution of the model is as similar as possible to the ground-truth. While previous work has relied on time-consuming iterative approximations [Tap07] or stochastic approximations [LH08], It will be demonstrated how implicit differentiation, can be used to analytically differentiate the overall training loss with respect to the MRF parameters. This leads to an efficient, flexible learning algorithm that can be applied to a number of different models.

The effectiveness of the proposed learning method will then be demonstrated by learning the parameters of two related models applied to the task of denoising images. Denoising refers to

[1]This work contains material previously published in [ST09]

the recovery and restoration of an image from one which has been corrupted by noise. The two models used for the research are the Robust Derivative [Tap07] and Field of Experts [RB05] models. Research was done using use those two models as examples to investigate how the proposed learning method can be practically applied. The necessary derivatives for each model used in the research are presented and the training process is explained. Results will be presented showing that the proposed learning algorithm produces comparable and at times better performance over other learning methods.

In addition to the introduction of a new learning algorithm, we will also introduce a new segmentation model. The novelty of this new segmentation model is how it incorporates priors on the size of the segments in a way that is straightforward to implement. Unlike related systems based on minimizing a criterion, in this model, the number of segments are not specified and the system produces more realistic segmentations. Using this new segmentation model, we will further investigate and demonstrate the effectiveness and flexibility of the proposed learning method by using it to train the proposed segmentation model.

1.3 Organization of Dissertation

The rest of dissertation will present the learning method and show its effectiveness in two different applications. A discussion of related works will be presented in chapter 2. This will help give the reader additional background information and help place this work in context. The proposed learning model is described in chapter 3. The main steps and necessary derivatives are shown using a generalized model. Next, in chapter 4, it is shown how two related models can be trained using the proposed method. The results presented will show how this proposed method compares will with other work in this area. Another application is demonstrated in chapter 5. A new segmentation model is presented and is then trained using the proposed method. This will further demonstrate the flexibility of the proposed training method. Finally, chapter 6 will summarize the research presented and provide some possible areas for future work.

CHAPTER 2
LITERATURE REVIEW AND RELATED WORK

Recent years have seen the introduction of several new approaches for learning the parameters of continuous-valued Markov Random Field models [LH08, Hin02, SP07, RB05, TLA07, WF07]. As mentioned in earlier in the introduction, learning the parameters of continuous-valued Markov Random Field models is normally done by either a maximum likelihood approach or a discriminative training approach where the training loss is minimized.

This chapter gives a brief overview of some approaches taken to learn the parameters of a Markov Random Field. Section 2.1 will review the maximum likelihood approach while Section 2.2 will describe current discriminative training approaches.

2.1 Maximum Likelihood Approaches

When taking the Maximum Likelihood approach the goal is to maximize the likelihood $M(\theta) = p(\mathbf{x}|\mathbf{y}, \theta))$, where **x** represents the hidden or unobserved quantity, **y** represents the observed quantity and θ represents the model parameters. Taking the $\log$ and using Equation 1.3 we get

$$\begin{aligned}\log M(\theta) &= \log p(\mathbf{x}|\mathbf{y},\theta)) \\ &= -\log Z - \sum_{k=1}^{N_c} V_k(\mathbf{x}_k;\theta) \qquad (2.1)\end{aligned}$$

While it is possible to differentiate the second term in Equation 2.1 in order to proceed with what is now a minimization problem, as stated earlier, it is generally intractable to do so for the first term $-\log Z$. This is especially the case in models which contain loops. Since this value is intractable to compute, one has to resort to various approximation techniques.

2.1.1 Sampling Methods

One popular approximation technique is to use sampling. The approach proposed by Roth and Black [RB05] to learn the parameters of the Field of Experts model uses a Markov chain Monte Carlo(MCMC) sampling and the idea of contrastive divergence [Hin02] to estimate the expectation over the model distribution. Using this estimate Roth and Black [RB05] perform gradient ascent on the log-likelihood to update the parameters. Regular Markov chain Monte Carlo sampling can take a long time to converge. Hinton [Hin02] introduced the method of contrastive divergence where the Markov chain is only run for a few steps. Though, when compared to regular MCMC sampling techniques, contrastive divergence simplifies the process it still, however, produces a bias in the estimates [WA02, KAH05]. Also, this method only computes approximate gradients and there is no guarantee that the contrastive divergence method converges.

2.1.2 Most Probable Explanation(MPE) Approximation

The method proposed by Scharstein and Pal [SP07] uses most-probable-explanation (MPE) estimates to compute approximate expectation gradients to learn parameters of a Conditional Random Field(CRF) model.

This approach, however, produces stability issues with models with a large number of parameters. The method proposed in this dissertation seeks to overcome that weakness and is able to effectively train a model with a large number of parameters. In their proposed method, Scharstein and Pal [SP07] use the fast alpha-expansion graphs cuts algorithm [BVZ01] to minimize their function to obtain their MPE estimate which is then used to calculate the approximate expectation. This method is similar to that used by Kumar et al.[KAH05] where the expectations are estimated using simple piecewise constant functions. Sutton et al. [SMR07] have also done similar work but used loopy belief propagation instead of graph cuts to estimate the MPE and then approximate expectations.

2.2 Optimizing the Training Loss

One approach where the training loss is minimized and is closely related to the proposed method in this thesis is the Variational Mode Learning approach proposed in by Tappen[Tap07]. As in my proposed method, the parameters are learnt by minimizing the training loss using the following optimization framework:

$$\begin{aligned} &\min_{\theta} \quad L(\mathbf{x}, \mathbf{t}) \\ &\text{s.t. } \mathbf{x} = \arg\min_{\mathbf{x}} \; -\log p(\mathbf{x}|\mathbf{y}; (\theta)) \end{aligned} \tag{2.2}$$

In this setup, the loss function $L(\mathbf{x}, \mathbf{t})$ penalizes the difference between **t**, the training image, and **x**, the current estimate. Since the negative log likelihood, $\arg\min_{\mathbf{x}} \; -\log p(\mathbf{x}|\mathbf{y}; (\theta))$ is not differentiable with respect to θ, it is replaced with an approximate solution. A series of variational optimization steps is treated as a continuous function and differentiating the result with respect to the model parameters. The key advantage of the proposed approach over Variational Mode Learning, is that the result of the variational optimization must be recomputed every time the gradient of the loss function is recomputed which often required 20-30 steps. This translates into 20 to 30 calls to the matrix solver each time a gradient must be recomputed. On the other hand, the method proposed in this work only requires one call to the matrix solver to compute a gradient.

Another approach proposed by Li and Huttenlocher [LH08] to learn the parameters of a continuous-state MRF model of optical flow uses simultaneous perturbation stochastic approximation (SPSA) [Spa92] to minimize the training loss across a set of ground truth images instead of the maximum likelihood approach. The motivation for minimizing the training loss instead of the maximum likelihood is similar to ours. Since an approximation method has to be used to determine expectation and then maximum likelihood, the results from a minimization of the maximum likelihood tends to reflect that fact and might be noisy or unpredictable. SPSA is an iterative optimization method where at each iteration all the model parameters are randomly perturbed and the function being minimized is evaluated using those perturbed values to determine the direction of descent. The model parameters are then updated approximately. When using SPSA it is difficult to determine exact convergence. Multiple runs are usually required to reduce the variance of the learnt parameters. Also, SPSA requires various coefficients which have to be ’tweaked’ to get optimal performance. While there are guidelines on how those coefficients can be chosen [Spa95], it is still a matter of trial and error in determining the right values to achieve the best performance.

In another closely related work, Do et al. [DFN07] were able to learn hyper-parameters, rather than parameters, of a Conditional Random Field (CRF) model by using a chain-structured model. Using a chain-structured model makes it possible to compute the hessian of the CRF’s density function, thus enabling the method to learn hyper-parameters. In this workHere, we consider problems where the density makes it impossible to compute this hessian, as is the case in non-Gaussian models with loops. Because we cannot compute the Hessian, we cannot learn hyper-parameters.

CHAPTER 3
GRADIENT-BASED MRF LEARNING

3.1 Introduction

In this chapter the learning method is described using a general, generic framework. Using this general framework, it will be shown how the gradient vector of the loss can be computed with respect to some parameter θ_i using the implicit differentiation method. Once the gradient of the loss is computed, the optimizing of the parameters can proceed. We will show how the optimization can be implemented using standard techniques.

3.2 Basic Learning Model

Similar to the discriminative, max-margin framework proposed by Taskar et al. [TCK05] and the Energy-Based Models [LH05], we learn the MRF parameters by defining a loss function that measures the similarity between the ground truth **t** and the minimum energy configuration of the MRF model. Throughout this paper, we will denote this minimum energy configuration as **x***.

This discriminative training criterion can be expressed formally as the following minimization:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \quad L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t}) \\ &\qquad \text{where } \mathbf{x}^*(\boldsymbol{\theta}) = \arg\min_{\mathbf{x}} E(\mathbf{x}, \mathbf{y}; w(\boldsymbol{\theta})) \end{aligned} \tag{3.1}$$

The MRF model is defined by the energy function $E(\mathbf{x}, \mathbf{y}; w(\boldsymbol{\theta}))$, where $\mathbf{x}$ denotes the current state of the MRF, the observations are denoted $\mathbf{y}$, and parameters θ. The function $w(\cdot)$ serves as a function that transforms θ before it is actually used in the energy function. It could be as simple as $w(\theta) = \theta$ or could aid in enforcing certain conditions. For instance, it is common to use an exponential function to ensure positive weighting coefficients, such as $w(\theta) = e^{\theta}$ for a scalar θ, as employed in work like [RB05]. Our primary motivation for introducing $w(\cdot)$ here is to aid in the derivation below.

Note that if $E(\mathbf{x}, \mathbf{y}; w(\theta))$ is non-convex, then one can only expect $\mathbf{x}^*$ to be a local minimum. Practically, we have found that our method is still able to perform well when learning parameters for non-convex energy functions, as we will show in Section 5.4.

3.3 Gradient Derivations

Taskar et al. have shown that for certain types of energy and loss functions, the learning task in Equation 2.2 can be accomplished with convex optimization algorithms [TCK05, TLJ06b]. However, the similarity of Equation 3.1 to solutions for learning hyper-parameters, such as [DFN07, Ben00, KSC07, CVB02], suggests that it may be possible to optimize the MRF parameters θ using simpler gradient-based algorithms. In the hyper-parameter learning work, the authors were able to use implicit differentiation to compute the gradient of a loss function with respect to hyper-parameters.

In this section, we will show how the same implicit differentiation technique can be applied to calculate the gradient of $L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})$ with respect to the parameters θ. This will enable us to use basic gradient descent techniques to learn the parameter $\boldsymbol{\theta}$. In chapter 4, it will be shown how this technique can be applied to a specific, filter-based MRF image prior.

3.3.1 Calculating the Gradient with Implicit Differentiation

We will begin by showing how the gradient vector of the loss can be computed with respect to some parameter θ_i using the implicit differentiation method used in hyper-parameter learning. We start by calculating the derivative vector of $\mathbf{x}^*(\theta)$ with respect to θ_i. Once we can differentiate $\mathbf{x}^*(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, a basic application of the chain rule will enable us to compute $\dfrac{\partial L}{\partial \theta_i}$.

Because $\mathbf{x}^*(\boldsymbol{\theta}) = \arg\min_{\mathbf{x}} E(\mathbf{x}, \mathbf{y}; w(\boldsymbol{\theta}))$, we can express the following condition using $\mathbf{x}^*$:

$$\left.\frac{\partial E(\mathbf{x}; \mathbf{y}, w(\boldsymbol{\theta}))}{\partial \mathbf{x}}\right|_{\mathbf{x}^*(\boldsymbol{\theta})} = \mathbf{0} \tag{3.2}$$

This simply states that the gradient at a minimum must be equal to the zero vector. For clarity in the remaining discussion, we will replace $\dfrac{\partial E(\mathbf{x}, \mathbf{y}; w(\theta))}{\partial \mathbf{x}}$,with the function $g(\mathbf{x}, w(\boldsymbol{\theta}))$, such that

$$g(\mathbf{x}, w(\boldsymbol{\theta})) \triangleq \frac{\partial E(\mathbf{x}, \mathbf{y}; w(\boldsymbol{\theta}))}{\partial \mathbf{x}}$$

Notice that we have retained $\boldsymbol{\theta}$ as a parameter of $g(\cdot)$, though it first passes through the function $w(\cdot)$. We retain θ because it will be eventually be treated as a parameter that can be varied, though in Equation 3.2 it is treated as a constant.

Using this notation, we can restate Equation 3.2 as

$$g(\mathbf{x}^*(\boldsymbol{\theta}), w(\boldsymbol{\theta})) = \mathbf{0} \tag{3.3}$$

Note that $g(\cdot)$ is a vector function of vector inputs. If $\mathbf{x}$ is an $N \times 1$ vector then $g(\cdot)$ is also an $N \times 1$ vector.

We can now differentiate both sides with respect to the parameter θ_i. After applying the chain rule, the derivative of the left side of Equation 3.3 is

$$\frac{\partial g}{\partial \theta_i} = \frac{\partial g}{\partial \mathbf{x}^*}\frac{\partial \mathbf{x}^*}{\partial \theta_i} + \frac{\partial g}{\partial w}\frac{\partial w}{\partial \theta_i} \tag{3.4}$$

Note that if $\mathbf{x}$ and $\mathbf{x}^*$ are $N \times 1$ vectors, then $\frac{\partial g}{\partial \mathbf{x}^*}$ is an $N \times N$ matrix.

A result of Equation 3.3 is that $\frac{\partial g}{\partial \theta_i} = \mathbf{0}$. Using this result, we can now solve for $\frac{\partial \mathbf{x}^*}{\partial \theta_i}$:

$$\begin{aligned} \mathbf{0} &= \frac{\partial g}{\partial \mathbf{x}^*}\frac{\partial \mathbf{x}^*}{\partial \theta_i} + \frac{\partial g}{\partial w}\frac{\partial w}{\partial \theta_i} \\ \frac{\partial \mathbf{x}^*}{\partial \theta_i} &= -\left(\frac{\partial g}{\partial \mathbf{x}^*}\right)^{-1}\frac{\partial g}{\partial w}\frac{\partial w}{\partial \theta_i} \end{aligned} \tag{3.5}$$

Note that because θ_i is a scalar, $\frac{\partial g}{\partial w}\frac{\partial w}{\partial \theta_i}$ is an $N \times 1$ vector.

The matrix $\frac{\partial g}{\partial \mathbf{x}^*}$ is easily computed by recognizing that $g(\mathbf{x}^*, w(\theta))$ is the gradient of $E(\cdot)$, with each term from **x** replaced with a term from $\mathbf{x}^*$. This makes the $\frac{\partial g}{\partial \mathbf{x}^*}$ term in the above equations just the Hessian matrix of $E(\mathbf{x})$ evaluated at **x***.

Using Equation 3.5 and applying the chain rule leads to the derivative of the overall loss function with respect to θ_i:

$$\begin{aligned}
\frac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \theta_i} &= \frac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \mathbf{x}^*}^T \frac{\partial \mathbf{x}^*}{\partial \theta_i} \\
&= -\frac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \mathbf{x}^*}^T \left(\frac{\partial g}{\partial \mathbf{x}^*}\right)^{-1} \frac{\partial g}{\partial w} \frac{\partial w}{\partial \theta_i} \\
&= -\frac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \mathbf{x}^*}^T \left(H_E(\mathbf{x}^*)\right)^{-1} \frac{\partial g}{\partial w} \frac{\partial w}{\partial \theta_i} \qquad (3.6)
\end{aligned}$$

In the above Equation 3.6, we have used $H_E(\mathbf{x}^*)$ to represent the Hessian of $E(\mathbf{x})$ evaluated at **x***.

Now that we have $\frac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \theta_i}$, we have all the necessary derivatives for the optimizations required by the learning algorithm.

3.4 Learning Algorithm

One of the benefits of the proposed formulation is that it provides us with a convenient framework to learn the parameter $\boldsymbol{\theta}$ using basic optimization routines. The following describes the necessary steps in the learning procedure.

1. Compute $\mathbf{x}^*(\boldsymbol{\theta})$ for the current parameter vector $\boldsymbol{\theta}$. Recall that $\mathbf{x}^*(\boldsymbol{\theta})$ is the minimum of the energy function, $E(\mathbf{x}, \mathbf{y}; w(\theta))$. This minimization can be accomplished using common optimization routines such as gradient descent or non-linear conjugate gradient optimization.

2. Compute the Hessian matrix at $\mathbf{x}^*(\theta)$, $H_E(\mathbf{x}^*)$.

3. Compute the training loss, $L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})$

4. Compute the gradient of the training loss with respect to $\mathbf{x}^*$, $\dfrac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \mathbf{x}^*}$.

5. Compute the gradient of the $L(\cdot)$ with respect to θ using Equation 3.6. As will be described in Section 3.4.1, performing the computations in the correct order can lead to significant gains in computational efficiency.

Once we have $\dfrac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \theta}$, we can use it to update the parameter, θ. The process can then be repeated until convergence.

3.4.1 Computational Considerations

Previous authors have pointed out two important points regarding Equation 3.6 [DFN07, TLA07]. First, the Hessian does not need to be inverted. Instead, only the value $\frac{\partial L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial\, \mathbf{x}^*}^T H_E(\mathbf{x}^*)^{-1}$ needs to be computed. This can be accomplished efficiently in a number of ways, including iterative methods like conjugate-gradients.

Second, by computing $\frac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \mathbf{x}^*}^T H_E(\mathbf{x}^*)^{-1}$ rather than $H_E(\mathbf{x}^*)^{-1}\frac{\partial g}{\partial w}\frac{\partial w}{\partial \theta_i}$, only one call to the solver is necessary to compute the gradient for all parameters in the vector $\boldsymbol{\theta}$.

The first observation to make here is to realize that since $\frac{\partial L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial \mathbf{x}^*}$ is a $N \times 1$ vector and $H_E(\mathbf{x}^*)$ is symmetric we have the following:

$$\frac{\partial L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial \mathbf{x}^*}^T H_E(\mathbf{x}^*)^{-1} = H_E(\mathbf{x}^*)^{-1}\frac{\partial L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial \mathbf{x}^*} \tag{3.7}$$

We then let $H_E(\mathbf{x}^*)^{-1}\frac{\partial L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial \mathbf{x}^*} = \mathbf{v}$ which can be rewritten as

$$H_E(\mathbf{x}^*)\mathbf{v} = \frac{\partial L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial \mathbf{x}^*} \tag{3.8}$$

This simply is your regular matrix equation system, $A\mathbf{v} = b$, where $A = H_E(\mathbf{x}^*)$ and $b = \frac{\partial L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial \mathbf{x}^*}$. Thus, solving for **v** in Equation 3.8 gives the product $\frac{\partial L(\mathbf{x}^*(\boldsymbol{\theta}), \mathbf{t})}{\partial \mathbf{x}^*}^T H_E(\mathbf{x}^*)^{-1}$ without explicitly calculating $H_E(\mathbf{x}^*)^{-1}$.

3.4.2 Advantages of Discriminative Learning

Typically, MRF models have been posed in a probabilistic framework. Likewise, parameter learning has been similarly posed probabilistically - often in a maximum likelihood framework. In many models, computing the partition function and performing inference are intractable, so approximate methods, such as the tree-based bounds of Wainwright et. al [WJW05] or Hinton's contrastive divergence method [Hin02], must be used.

A key advantage of the discriminative training criterion is that it is not necessary to compute the partition function or expectations over various quantities. Instead, it is only necessary to minimize the energy function. Informally, this can be seen as reducing the most difficult step in training from integrating over the space of all possible images, which would be required to compute the partition function or expectations, to only having to minimize an energy function.

A second advantage of this discriminative training criterion is that it better matches the MAP inference procedure typically used in large, non-convex MRF models. MAP estimates are popular when inference, exact or approximate, is difficult. Unfortunately, parameters that maximize the likelihood may not lead to the highest-quality MAP solution. This issue will be further explored using the Field of Experts model in Section 4.6.3.

3.5 Summary

This chapter has presented the proposed learning algorithm using a generalized framework. It was shown how we can learn the MRF parameters by defining a loss function that measures the similarity between the ground truth **t** and the minimum energy configuration of the MRF model. The derivative of the loss function with respect to the MRF parameters can be found explicitly using implicit differentiation. This enables us to use an optimization routine like gradient-descent to determine the MRF parameters such that the quality of MAP estimates are directly optimized.

CHAPTER 4
APPLICATION IN DENOISING IMAGES

4.1 Introduction

This chapter demonstrates the effectiveness of the proposed learning algorithm by applying the method to learn the parameters of two related denoising models: the Robust Derivative [Tap07] and Field of Experts [RB05] models. Denoising can be described as the process of recovering an original image from one which has been corrupted by noise. Figure 4.1 gives an example of original image and a noisy one where artificial white gaussian noise of standard deviation, $\sigma = 25$ was added. We begin by describing the models used and how the parameters are determined using the proposed method. In addition to presenting the training process of the two models, results will also be presented which demonstrate the effectiveness of the training procedure.

4.2 Background: Field of Experts Model(FoE)

Recent work has shown that image priors created from the combination of image filters and robust penalty functions are very effective for low-level vision tasks like denoising, in-painting, and de-blurring [LFD07, RB05, Tap07]. The Field of Experts model is defined by a set of linear filters, $f_1, ...f_{N_f}$, and their associated weights, $\alpha_1, ..., \alpha_{N_f}$. The Lorentzian penalty function, $\rho(x) =$

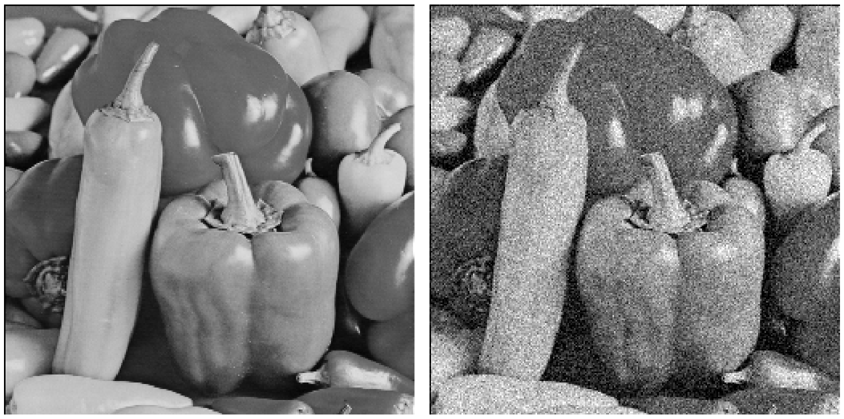


Figure 4.1: This figure shows the (a) original image compared to (b) one with artificial white Gaussian noise of standard deviation $\sigma = 25$.

$\log(1 + x^2)$ is used to define the clique potentials. This leads to a probability density function over an image, **x**, to be defined as:

$$p(\mathbf{x}) = \frac{1}{Z}\text{exp}\left(-\sum_{i=1}^{N_f}\alpha_i\sum_{p=1}^{N_p}\rho(\mathbf{x}_p * f_i)\right) \tag{4.1}$$

where N_p is the number of pixels, N_f is the number of filters and $(\mathbf{x}_p * f_i)$ denotes the result of convolving the patch at pixel p in image **x** with the filter f_i.

When applied to denoising images, the probability density in Equation 4.1 is used as a prior and combined with a Gaussian likelihood function to form a posterior probability distribution of an image given by:

$$p(\mathbf{x}|\mathbf{y}) = \frac{1}{Z}\text{exp}\left(-\sum_{i=1}^{N_f}\alpha_i\sum_{p=1}^{N_p}\rho(\mathbf{x}_p * f_i) - \frac{1}{2\sigma^2}\sum_{p=1}^{N_p}(\mathbf{x}_p - \mathbf{y}_p)^2\right) \tag{4.2}$$

where σ is the standard deviation of the Gaussian noise and **y** is a noisy observation of **x**.

4.2.1 Background: Robust Derivative Model

The Robust Derivative model is a variant of the Field of Experts model. However, unlike the FoE model, the Robust Derivative model combines a *fixed set* of high-order image derivative filters, $f_1, ...f_{N_f}$, with the Lorentzian penalty function, $\rho(x) = \log(1 + x^2)$. We arrive at the final form of the energy function, $E(\mathbf{x}, \mathbf{y}; \alpha, \beta)$, by adding parameters to adjust the weight of different filters and control the shape of the penalty function leads to the final form of the energy function :

$$E(\mathbf{x}, \mathbf{y}; \alpha, \beta) = \sum_{p=1}^{N_p} (\mathbf{x}_p - \mathbf{y}_p)^2 + \sum_{i=1}^{N_f} \exp(\alpha_i) \sum_{p=1}^{N_p} \rho((\mathbf{x}_p * f_i), \beta_i) \tag{4.3}$$

where N_p is the number of pixels, $\rho(z, \beta) = \log(1 + (\beta z)^2)$, and $(\mathbf{x}_p * f_i)$ denotes the result of convolving the patch at pixel p in image **x** with the filter f_i.

The set of first, second and third derivative filters used in the implementation of the model are shown in Figure 4.2.

The underlying idea behind the Robust Derivative Model is that the image can be modeled with a piecewise smooth prior. As described in [BR96], using the Lorentzian robust penalty is equivalent to using a quadratic penalty on the derivative values, combined with a line-process variable that allows for sharp breaks, which corresponds to edges in the image [GG84]. The

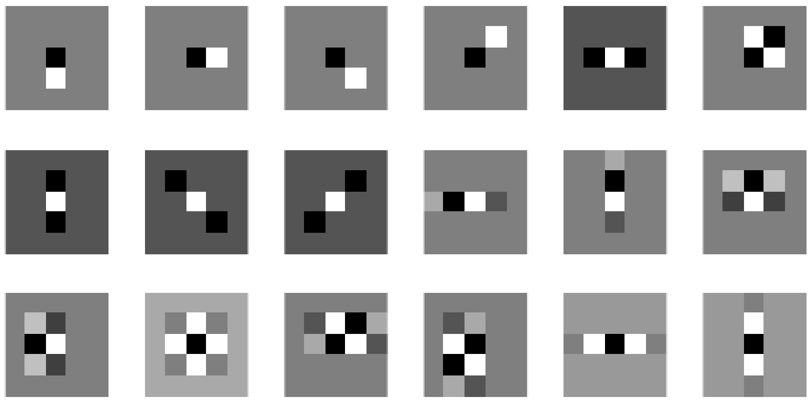


Figure 4.2: The set of first, second and third derivative filters used in the implementation of the Robust Derivative model.

advantage of this model is that it is easily interpretable and performs comparably to similar models such as the Field of Experts model or the model proposed recently by Weiss and Freeman [WF07].

4.3 Learning: Robust Derivative Model

In this section, we will describe how the parameters of the Robust Derivative model can be learnt using the gradient-based learning method. The model will be trained to denoise images with varying degrees of additive noise. We will begin with this model since it involves fewer parameters than the Field of Experts model. The results show that the training method is in fact effective. When compared with the Field of Experts model the Robust Derivative model trained using the proposed approach produces competitive results even though less parameters are needed.

4.3.1 Calculating Gradients in the Robust Derivative Model

For clarity, I will begin by describing how to compute derivatives in the Robust Derivative Model by considering a model with one filter. The derivative of Equation 4.3 with respect to **x** is given by:

$$\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} = e^{\alpha} f^{-} * \frac{2\beta^2(\mathbf{x} * f)}{1 + (\beta(\mathbf{x} * f))^2} + 2(\mathbf{x} - \mathbf{y}). \tag{4.4}$$

Rather than using the convolution notation from Equation 4.4, convolution of **x** with a filter f will be denoted as the product of a doubly-Toeplitz matrix F and **x**. F is a N_pxN_p matrix, where N_p represents the number of pixels in **x**. From this point in the derivations, **x** will be a N_px1 vector formed by stacking the rows of the image to form one column.

Using this notation, the gradient of the energy function $E(\mathbf{x})$, with respect to **x** is written as:

$$\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} = F^T \rho'(\mathbf{u}) + 2(\mathbf{x} - \mathbf{y}), \tag{4.5}$$

where $\rho'(z)$ is a function that applied element-wise to the vector $\mathbf{u} = F\mathbf{x}$. This function has the form

$$\rho'(z) = \exp(\alpha) \frac{2\beta^2 z}{1 + (\beta z)^2}. \tag{4.6}$$

Following the steps in Chapter 3, the derivative vector of **x*** with respect to the parameter α is computed from the combination of the derivative of Equation 4.5 and the Hessian of $E(\mathbf{x}^*)$.

Recall that at $\mathbf{x}^*$ we have $\left.\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}^*} = \mathbf{0}$. That is,

$$F^T \rho'(\mathbf{u}) + 2(\mathbf{x}^* - \mathbf{y}) = \mathbf{0}. \tag{4.7}$$

Differentiating both sides of Equation 4.7 with respect to the parameter α and rearranging we have,

$$\begin{aligned}
0 &= F^T \rho'(\mathbf{u}^*) + F^T W F \frac{\partial \mathbf{x*}}{\partial \alpha} + 2I \frac{\partial \mathbf{x*}}{\partial \alpha} \\
0 &= F^T \rho'(\mathbf{u}^*) + \left(F^T W F + 2I\right) \frac{\partial \mathbf{x*}}{\partial \alpha} \\
\frac{\partial \mathbf{x*}}{\partial \alpha} &= -\left(F^T W F + 2I\right)^{-1} F^T \rho'(\mathbf{u}^*). \qquad (4.8)
\end{aligned}$$

Here, W refers to a N_pxN_p diagonal matrix where the entries $[W]_{i,i}$ along the diagonal are computed by applying the function

$$\rho''(z) = \frac{2\beta^2(1+(\beta z)^2) - 2\beta^2 z(2\beta z(\beta))}{(1+(\beta z)^2)^2} \tag{4.9}$$

to the entries of the vector $\mathbf{u}^* = F\mathbf{x*}$. The expression in Equation 4.8 is easily extended to a set of different filters, $f_1 \ldots f_{N_f}$, and the corresponding matrices, $F_1 \ldots F_{N_f}$.

If the loss function, $L(\mathbf{x*}, \mathbf{t}, \theta)$, is the pixel-wise squared error

$$L(\mathbf{x*}(\theta), \mathbf{t}) = \sum_{p=1}^{N_p} (\mathbf{x}_p - \mathbf{t}_p)^2 \tag{4.10}$$

then the overall expression for finding the derivative of the loss function with respect to α becomes

$$\begin{aligned}
\frac{\partial L(\mathbf{x*}, \mathbf{t}, \theta)}{\partial \alpha} &= 2(\mathbf{x*} - \mathbf{t})^T \frac{\partial \mathbf{x*}}{\partial \alpha} \\
&= -2(\mathbf{x*} - \mathbf{t})^T \left(F^T W F + 2I\right)^{-1} F^T \rho'(\mathbf{u}^*). \qquad (4.11)
\end{aligned}$$

4.4 Evaluating Denoising Results using the Robust Derivative model

Testing was done on two datasets. The first set comprised of 68 images also from the Berkeley database[MFT01] and the second test set were images used by Portilla et. al. [PSW03] in their denoising experiments. Similar to [RB05] we assume a known noise distribution when performing our experiments.

There were 40 training images taken from the Berkeley segmentation database[MFT01]. We chose four 51x51 patches from each training image, giving us a total of 160 training patches.

In order to compare the results, the test images were denoised using the Field of Experts implementation provided by Roth and Black. Those experiments were run for 5000 iterations per image with a step size $\eta = 0.6$.

The denoising performance in this set of experiments was primarily measured using the peak signal-to-noise ratio(PSNR) values. The PSNR is a popular metric used to evaluate denoising performance and is defined as

$$PSNR = 10 \log_{10} \frac{255^2}{\sqrt{mse}} \tag{4.12}$$

The mse is the mean squared error calculated pixelwise between the original image and the test image. The square root of the mse gives the standard deviation.

Performance was compared on the Berkeley dataset with noisy images created by adding Gaussian noise at standard deviation of $\sigma = 15$ and $\sigma = 25$. There was similar performance when denoising images compared to Field of Experts as shown in Table 4.1. Overall, the model trained using the proposed method performed almost as well as Field of Experts on images with $\sigma = 15$

Table 4.1: This table compares the average RMS error and PSNR on images from the Berkeley dataset between our model and the FoE model.

Model	$\sigma = 15$ Average RMS Error	$\sigma = 15$ Average PSNR	$\sigma = 25$ Average RMS Error	$\sigma = 25$ Average PSNR
Our Model	8.02	30.33	10.79	27.92
Field of Experts	7.90	30.44	10.81	27.71

and slightly better at $\sigma = 25$. Figures 4.3 and 4.4 show that the results are also visually similar to that of Field of Experts.

Our results in terms of the peak signal-to-noise ratio on the second test dataset[PSW03] are given in Table 4.2 . Again, our results are close to those reported by Roth and Black using the Field of Experts model. We attain similar results on most of the images and in the other cases our result was either below or above their results by at most 0.3dB. A visual comparison for an example image from that dataset is shown in Figure 4.5.

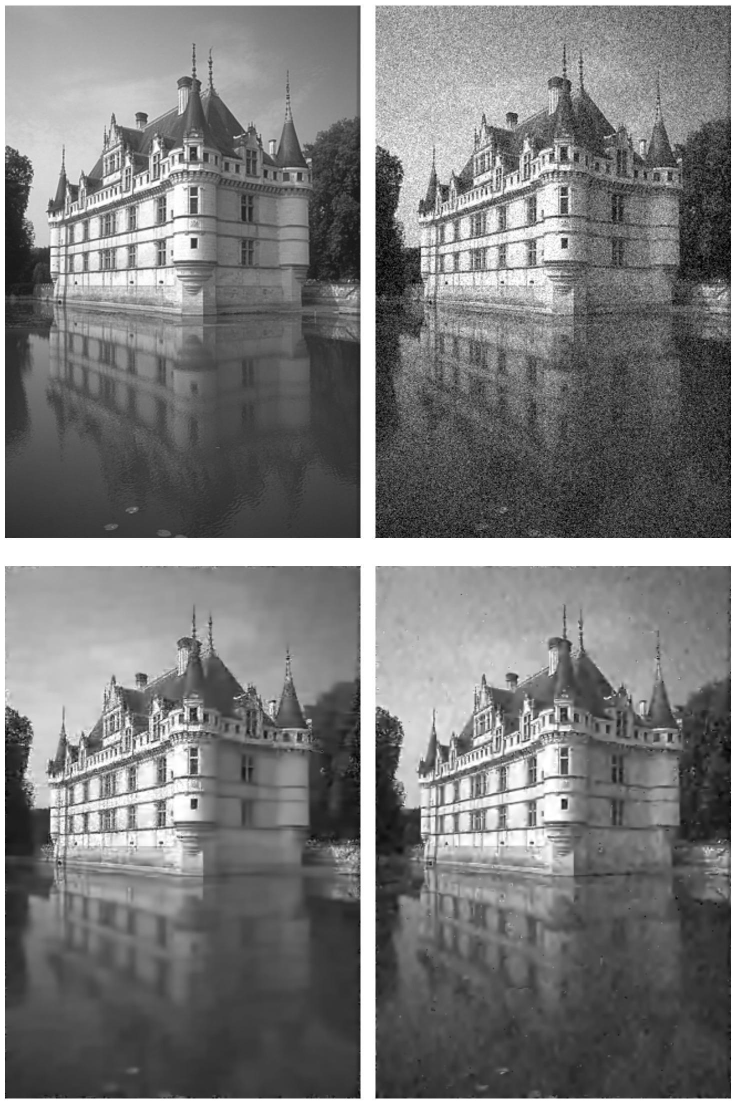
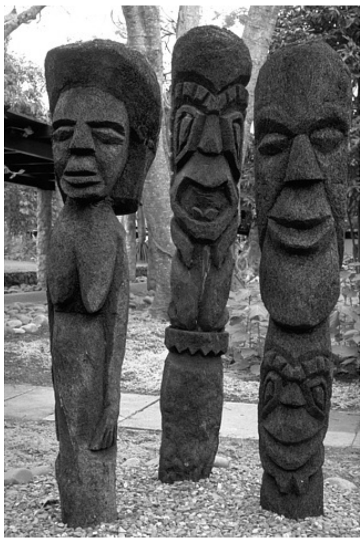
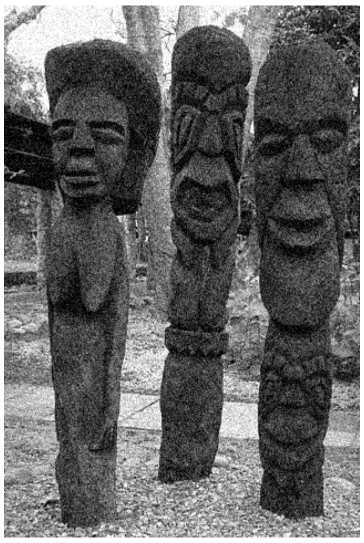


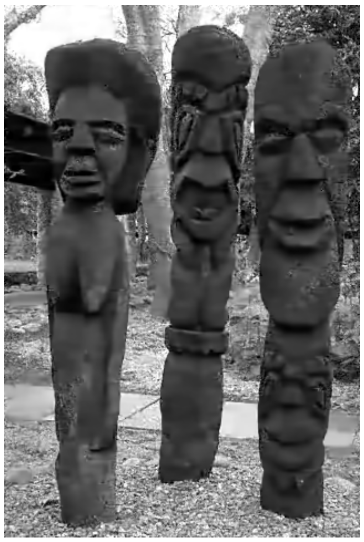
Figure 4.3: An example from the Berkeley dataset comparing the Field of Experts model to our denoising model. (a) Original image (b) Noisy image, $\sigma = 25$, PSNR = 20.16 (c) Denoised image using FoE model, PSNR = 28.40 (d) Denoised image using our model, PSNR = 28.71.



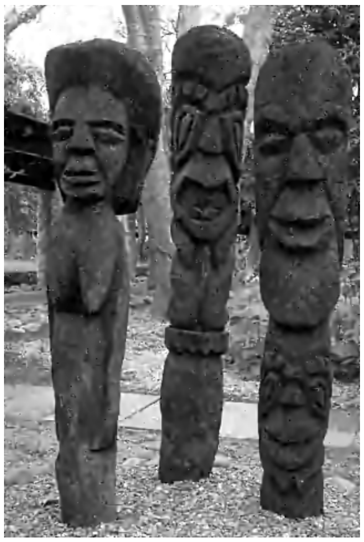
(a)



(b)



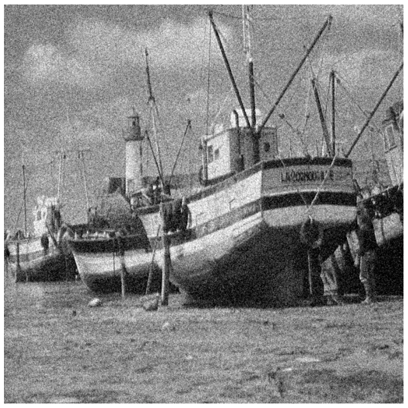
(c)



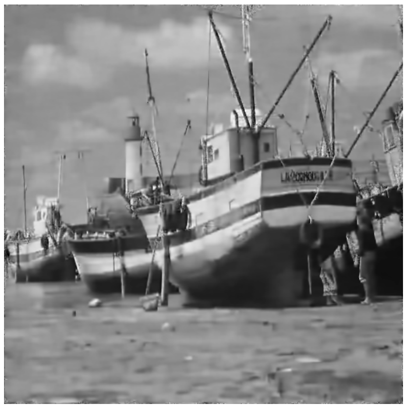
(d)

Figure 4.4: Another example from the Berkeley dataset comparing the Field of Experts model to our denoising model. (a) Original image (b) Noisy image, $\sigma = 25$, PSNR = 20.17 (c) Denoised image using FoE model, PSNR = 24.84 (d) Denoised image using our model, PSNR = 25.16.

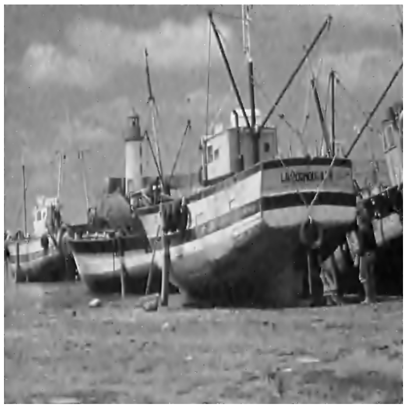
(a)



(b)



(c)



(d)

Figure 4.5: Comparing the Field of Experts model to Robust Derivative model trained using the proposed method. The example here is an image from the Portilla dataset. (a) Original image (b) Noisy image, $\sigma = 25$, PSNR = 20.26 (c) Denoised image using FoE model, PSNR = 28.71 (d) Denoised image using our model, PSNR = 28.85.

Table 4.2: PSNR in dB for denoised images from [PSW03] using the Robust Derivative model trained using our method

σ	Lena	Barbara	Boats	House	Peppers
10	34.98	32.48	33.05	34.97	34.34
15	33.23	29.98	31.23	33.44	32.25
20	31.93	28.21	29.84	32.15	30.70
25	30.85	26.66	28.75	31.05	29.51

4.5 Learning: Field of Experts Model

4.5.1 Energy and Loss Function Formulation

As stated in Section 3.2 the MRF parameters are learnt by minimizing a loss function that measures the similarity between the ground truth **t** and the minimum energy configuration of the MRF model. We use the negative log-posterior given in Equation 4.2 to form the energy function as:

$$E(\mathbf{x}, \mathbf{y}; \alpha, \beta) = \sum_{i=1}^{N_f} e^{\alpha_i} \sum_{p=1}^{N_p} \log(1 + (F_i \mathbf{x}_p)^2) + \sum_{p=1}^{N_p} (\mathbf{x}_p - \mathbf{y}_p)^2. \tag{4.13}$$

Here, multiplication with a doubly-Toeplitz matrix F_i is used to denote convolution with a filter f_i. Each filter, F_i, is formed from a linear combination of a set of basis filters $B_1, , .., B_{N_B}$. The parameters β determine the coefficients of the basis filters, that is, F_i is defined as

$$F_i = \sum_{j=1}^{N_B} \beta_{ij} B_j. \tag{4.14}$$

This formulation allows the filters, $F_i, ..F_{N_f}$, to be learnt via the parameters β as well as their respective weights via α.

In the following section, the loss function, $L(\mathbf{x}^*(\theta), \mathbf{t})$, is assumed to be the pixelwise squared error between the **x** and **t**. That is,

$$L(\mathbf{x}^*(\theta), \mathbf{t}) = \sum_{p=1}^{N_p} (\mathbf{x}_p - \mathbf{t}_p)^2, \tag{4.15}$$

where the parameters α and β are grouped into a single vector θ. It should be noted that the proposed formulation is flexible enough to allow the user to choose a loss function that matches their notion of image quality.

4.5.2 Calculating Gradients in the FoE Model

As in Section 4.3, I will describe how to compute the required derivatives in the denoising formulation by considering a model with one filter. In this case the gradient of the energy function $E(\mathbf{x}, \mathbf{y}; \alpha, \beta)$ can then be written as

$$\frac{\partial E(\mathbf{x}, \mathbf{y}; \alpha, \beta)}{\partial \mathbf{x}} = F^T \rho(\mathbf{u}) + 2(\mathbf{x} - \mathbf{y}), \tag{4.16}$$

where $\rho(\mathbf{u})$ is a function that is applied element-wise to the vector $\mathbf{u} = F\mathbf{x}$ and defined as

$$\rho(z) = \exp(\alpha) \frac{2z}{1 + z^2}. \tag{4.17}$$

Following the steps in Section 3.3.1 leads to the derivative of $\mathbf{x}^*$ with respect to the parameter β. Since when evaluated at $\mathbf{x}^*$, that is, $\left.\frac{\partial E(\mathbf{x}, \mathbf{y}; \alpha, \beta)}{\partial \mathbf{x}}\right|_{\mathbf{x}^*} = \mathbf{0}$, we have

$$\mathbf{0} \;=\; F^T \rho(\mathbf{u}^*) + 2(\mathbf{x}^* - \mathbf{y}) \tag{4.18}$$

differentiating with respect to β gives

$$\mathbf{0} \;=\; B^T \rho(\mathbf{u}^*) + F^T \gamma(\mathbf{u}^*) B\mathbf{x} + \left(F^T W F + 2I\right) \frac{\partial \mathbf{x}^*}{\partial \beta}. \tag{4.19}$$

Solving for $\frac{\partial \mathbf{x}^*}{\partial \beta_j}$ gives

$$\frac{\partial \mathbf{x}^*}{\partial \beta} = -\left(F^T W F + 2I\right)^{-1} \left(B^T \rho(\mathbf{u}^*) + F^T \gamma(\mathbf{u}^*) B\mathbf{x}\right), \tag{4.20}$$

where W is a N_pxN_p diagonal matrix and a $[W]_{i,i}$ entry is determined by applying the function

$$\gamma(z) = \exp(\alpha) \frac{2 - 2z^2}{(1 + z^2)^2} \tag{4.21}$$

element-wise to the vector $\mathbf{u}^* = F\mathbf{x}^*$.

This leads to the overall expression for computing the derivative of the loss function with respect to β to be defined as

$$\frac{L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial \beta} = -(\mathbf{x}^* - \mathbf{t})^T \left(F^T W F + 2I\right)^{-1} C_\beta, \tag{4.22}$$

where $C_\beta = (B^T \rho(\mathbf{u}^*) + F^T \gamma(\mathbf{u}^*) B\mathbf{x}^*)$.

In a similar manner, the expression for the derivative of $\mathbf{x}^*$ with respect to the parameter α is computed by first differentiating Equation 4.18 with respect to α to get

$$\mathbf{0} \;=\; F^T \rho(\mathbf{u}^*) + \left(F^T W F + 2I\right) \frac{\partial \mathbf{x}^*}{\partial \alpha}. \tag{4.23}$$

Solving for $\frac{\partial \mathbf{x}^*}{\partial \alpha}$ gives

$$\frac{\partial \mathbf{x}^*}{\partial \alpha} = -\left(F^T W F + 2I\right)^{-1} F^T \rho(\mathbf{u}^*) \tag{4.24}$$

and derivative of the loss function with respect to α can be defined as

$$\frac{L(\mathbf{x}^*(\theta), \mathbf{t})}{\partial \alpha} = -(\mathbf{x}^* - \mathbf{t})^T (F^T W F + 2I)^{-1} C_\alpha, \tag{4.25}$$

where $C_\alpha = F^T \rho(\mathbf{u}^*)$.

The above equations can be easily extended for multiple filters, $f_1, ...f_{N_f}$, and their corresponding doubly-Toeplitz matrices, $F_1, ...F_{N_f}$.

4.6 Evaluating denoising results using the Field of Experts model

Similar to the experiments using the Robust Derivative model, tests were done using the images from the Berkeley segmentation database[MFT01] identified by Roth and Black in their experiments. We used the same 40 images for training and 68 images for testing. Since the proposed approach made it possible to train on larger patches than those used by Roth and Black, the results reported in this paper are all from training the system using four randomly selected 51 x 51 patches from each training image, giving a total of 160 training patches. Figure 4.6 shows a few of the training images and Figure 4.7 shows a few examples of the patches used for training. A set of 24 filters with dimension 5 x 5 pixels were learnt using the same basis as Roth and Black. Training was also done using 2000 randomly selected 15x15 patches with slightly lower results.

Figure 4.6: A few examples of the training images used from the Berkeley segmentation database. [MFT01] A total of 40 different images were used in the training process.

4.6.1 Implementation Highlights

The training was implemented in Matlab along with MatLabMPI [Kep04] which provided parallel computing support. Since this algorithm can easily be parallelized, one can make more efficient use of available computing resources. Training took about 8-10 hours on average using 10 CPUS.

4.6.2 Denoising Results

In order to compare the results, the test images were also denoised using the Field of Experts(FoE) implementation provided by Roth and Black. The parameters in this implementation

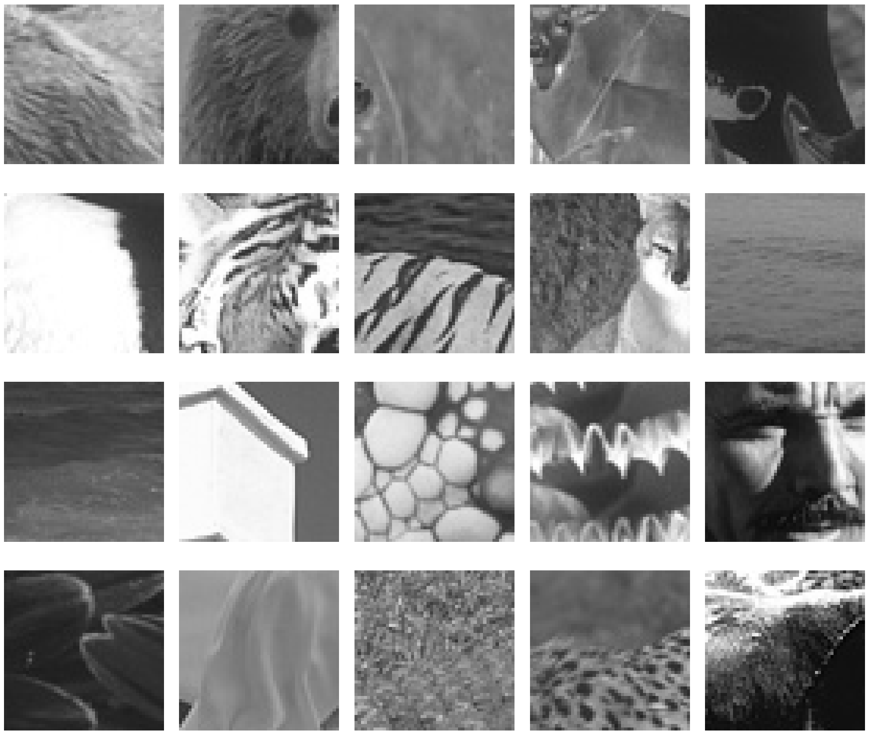


Figure 4.7: A few examples of the 51x51 pixels training patches used. These patches were randomly chosen from the 40 training images.

were learnt using the contrastive divergence method mentioned in Chapter 2 and uses gradient ascent to denoise images. For convenience, this system will be referred to as FoE-GA (for gradient ascent). The experiments were run for 2500 iterations per image as suggested by Roth and Black using a step size $\eta = 0.6$. Another denoising system was also implemented using the same parameters from the FoE-GA system but using conjugate gradient descent instead of gradient ascent. This system will be referred to as FoE-CG (for conjugate gradient) and results are reported when the system converges at a local minimum.

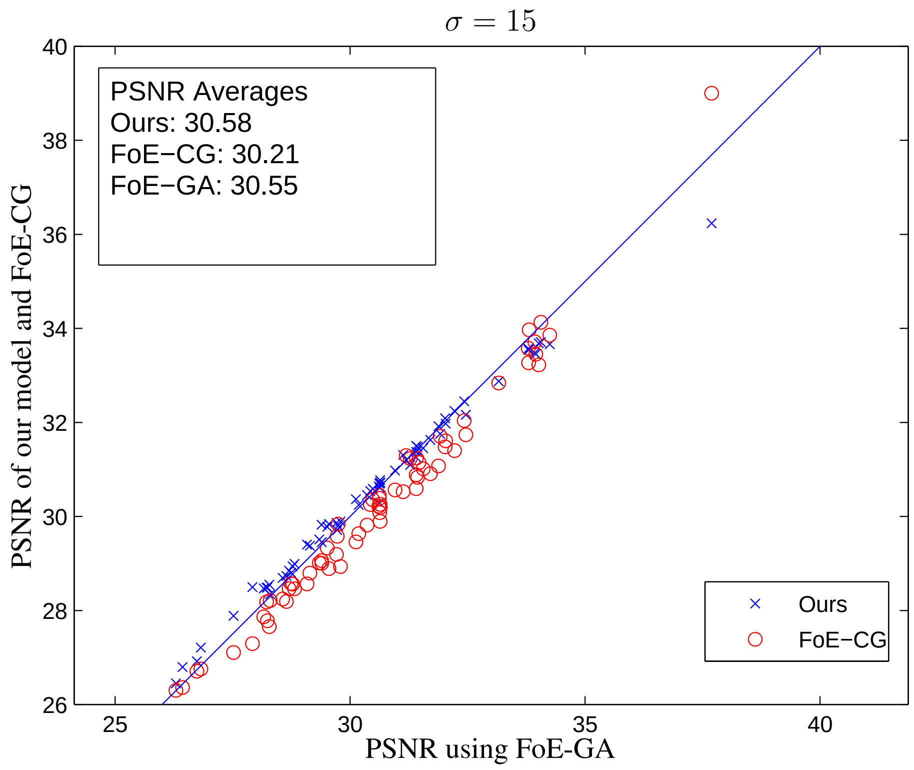


Figure 4.8: Scatterplots comparing the PSNR between our results and Field of Experts' using contrastive divergence on the test images with Gaussian noise of standard deviation, $\sigma = 15$. A point above the line means performance is better than FoE-GA. As can be seen, our performance is generally better than FoE-GA. On the other hand,FoE-CG does worse than FoE-GA

As shown in Figures 4.8 and 4.9, performance was similar and, at times, better at convergence when compared to FoE-GA across the test images when the MAP inference is allowed to terminate at a local minimum. It should be reiterated that FoE-GA terminates after a fixed number of iterations, which is chosen to maximize performance and not when the system reaches a local minimum. When compared to FoE-CG our performance is noticeably better. This shows a significant and important difference using my proposed training method: if the minimization is allowed

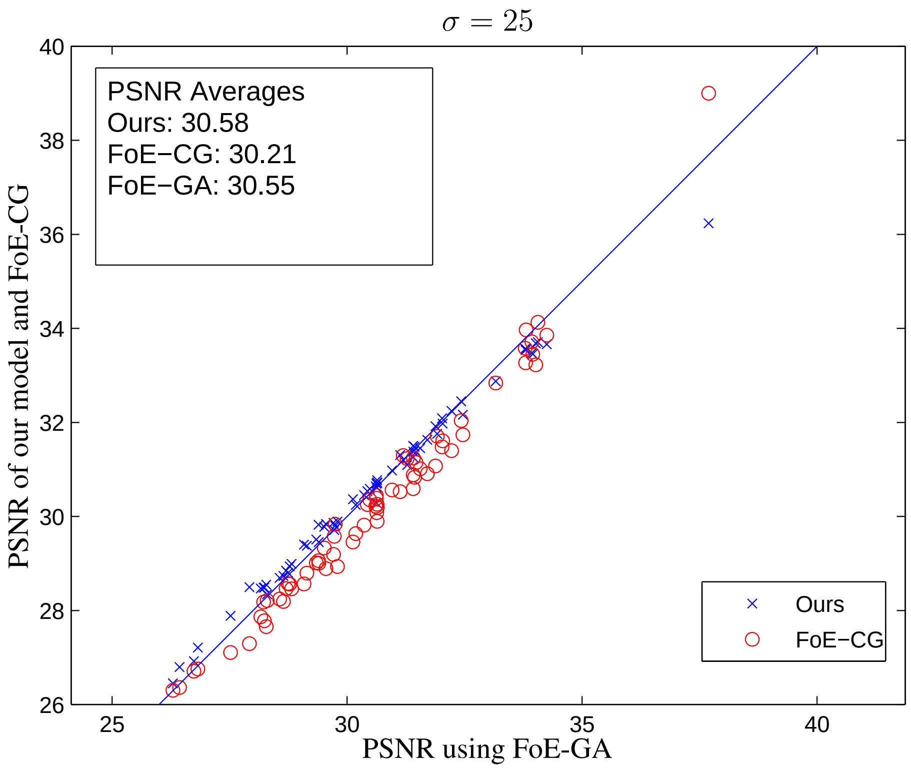


Figure 4.9: Scatterplots comparing the PSNR between our results and Field of Experts' using contrastive divergence on the test images with Gaussian noise of standard deviation, $\sigma = 25$. A point above the line means performance is better than FoE-GA. As can be seen, our performance is generally better than FoE-GA. On the other hand,FoE-CG does worse than FoE-GA

to terminate at a local minimum then the performance is much better. Convergence is also achieved much faster using the proposed training method as shown in Table 4.3. The same stopping criteria was used for testing results and FoE-CG. Figure 4.10 shows how the PSNR and the energy changes during the course of the minimization for the image shown in Figure 4.16.

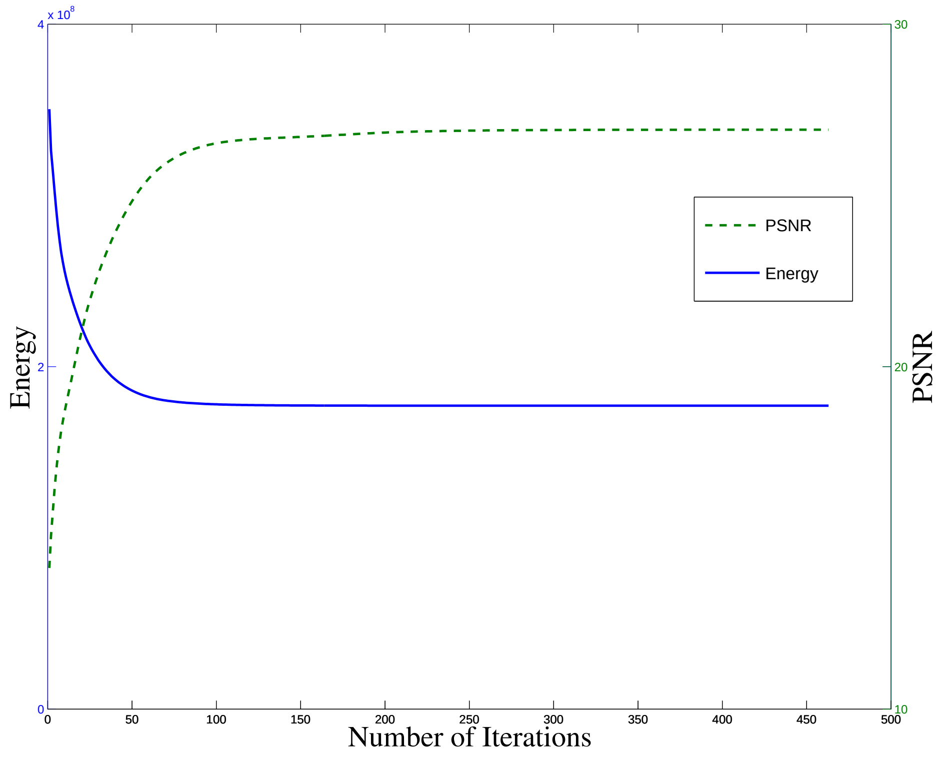


Figure 4.10: This figure shows the relationship between the energy and the PSNR during the minimization for the test image shown in figure 4.16.

In addition to calculating the PSNR, we also computed the perceptually based structural similarity index (SSIM)[WBS04] to measure the denoising results. The SSIM metric values vary between 0 and 1, where 1 is the best.

Tables 4.4 and 4.5 give the average PSNR and SSIM index computed from the denoised images for $\sigma = 25$ and $\sigma = 15$. Figures 4.11 and 4.12 show examples in which our results are better in terms of PSNR and SSIM. Visually, the texture is preserved better in our denoised images. However, in images which are composed of relatively smooth regions then FoE-GA and FoE-CG produce better results as can be seen in Figure 4.14.

Table 4.3: This table shows the average number of conjugate gradient iterations taken before convergence when performing denoising on the 68 test images. All iterations have the same complexity; so in this case fewer iterations means faster performance.

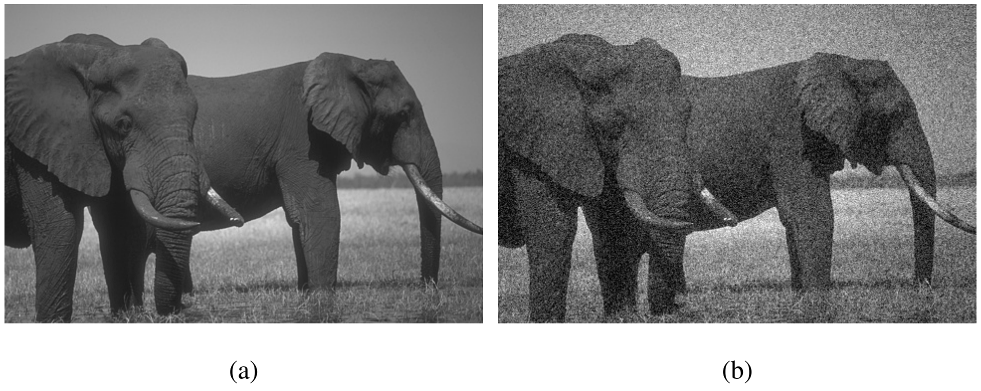
	FoE-CG	Ours
$\sigma = 25$	1874	227
$\sigma = 15$	1009	94

Table 4.4: This table compares the average PSNR and SSIM index on the test images between our results and FoE-GA/CG for images with Gaussian noise of standard deviation, $\sigma = 25$.

System	Average PSNR	Average SSIM
Ours	27.86	0.7760
FoE-GA	27.75	0.7632
FoE-CG	27.17	0.7266

Table 4.5: This table compares the average PSNR and SSIM index on the test images between our results and FoE-GA/CG for images with Gaussian noise of standard deviation, $\sigma = 15$.

System	Average PSNR	Average SSIM
Ours	30.58	0.8555
FoE-GA	30.56	0.8592
FoE-CG	30.21	0.5358



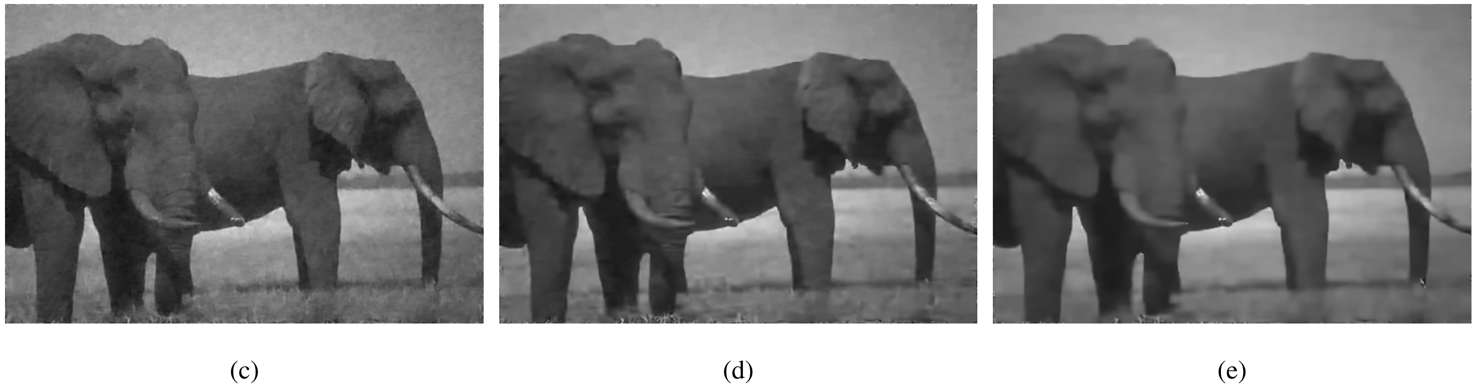


Figure 4.11: An example from the Berkeley dataset comparing denoising results between the different systems. (a)Original image (b) Noisy image, $\sigma = 25$ (c) Our denoised image, PSNR = 29.33, SSIM = 0.7341 (d) FoE-GA denoised image, PSNR = 28.98, SSIM = 0.7169 (e) FoE-CG denoised image, PSNR = 28.58, SSIM = 0.6988.

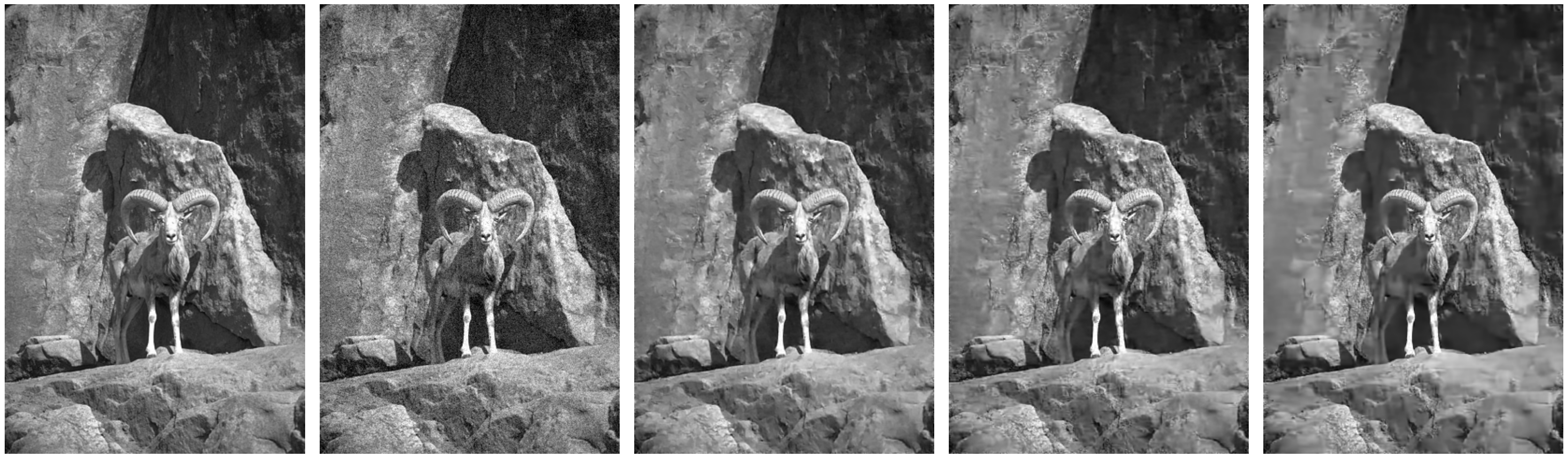


Figure 4.12: A second example from the Berkeley dataset comparing denoising results between the different systems. The over smoothing by FoE-GA/CG can be seen on the flat rock surface on the left of the image. (a) Original image (b) Noisy image, $\sigma = 15$ (c) Our denoised image, PSNR = 28.59, SSIM = 0.7990 (d) FoE-GA denoised image, PSNR = 28.28, SSIM = 0.7845 (e) FoE-CG denoised image, PSNR = 27.66, SSIM = 0.7203.

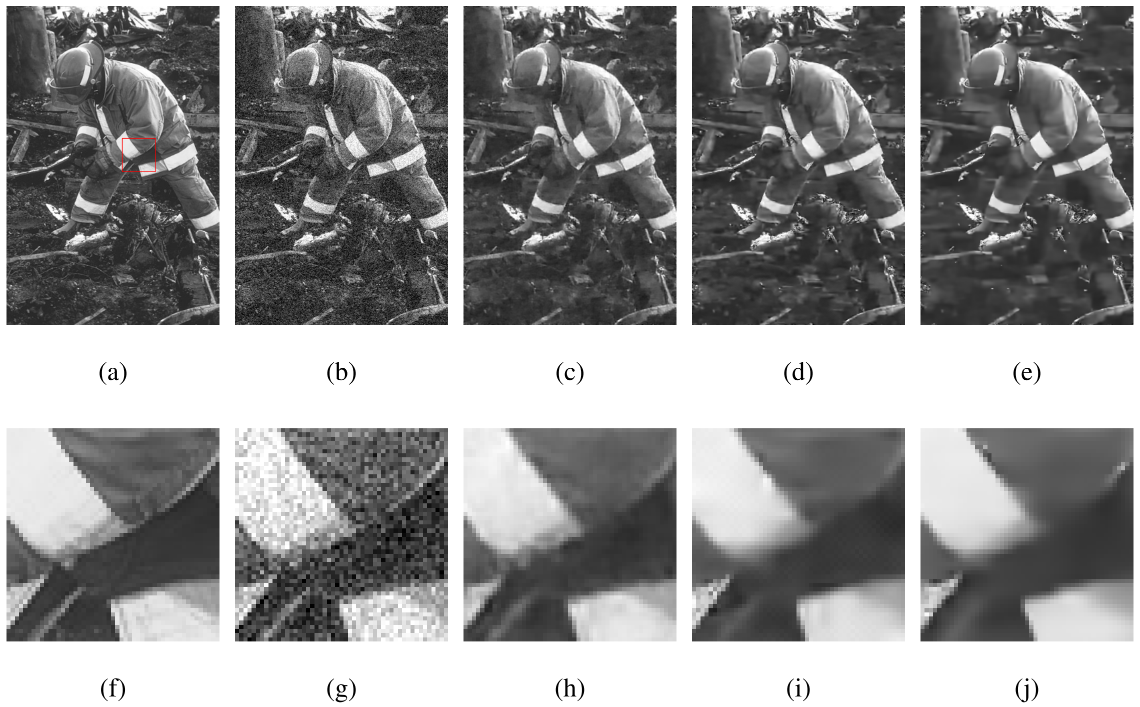


Figure 4.13: An example from the Berkeley dataset comparing denoising results between the different systems. Top row: Full size images. Bottom row: closeup view of a selected patch. From left to right: Original image, Noisy image, Our denoised image, FoE-GA denoised image, FoE-CG denoised image

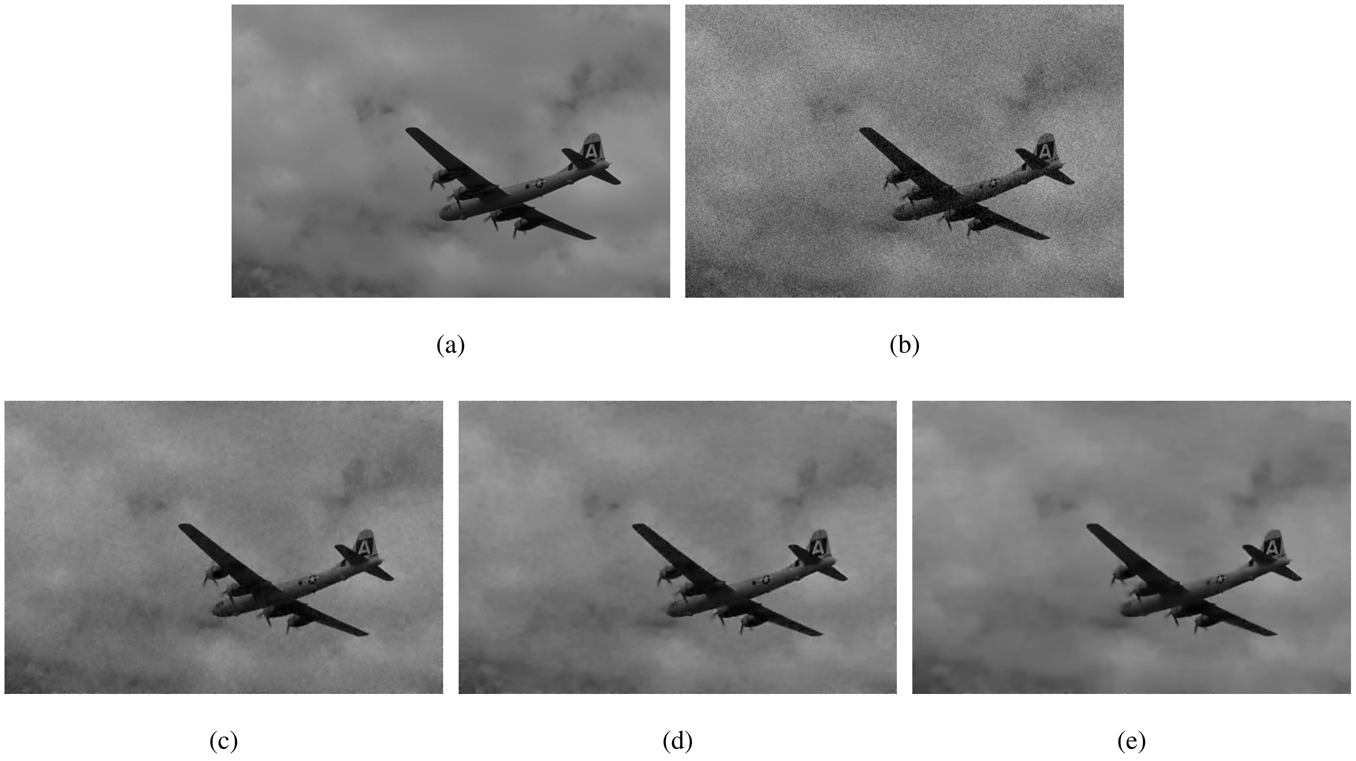


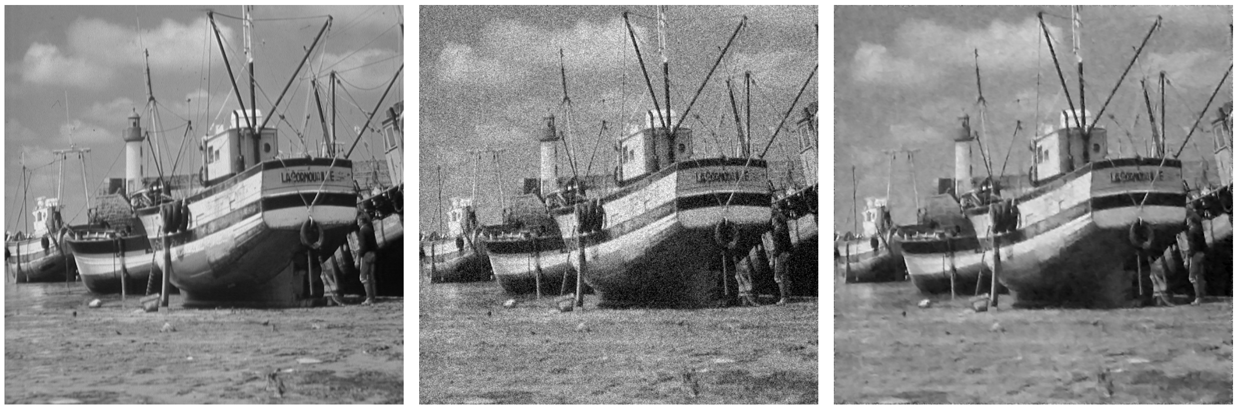
Figure 4.14: An example from the Berkeley dataset comparing denoising results between the different systems on an image with flat regions. (a) Original image (b) Noisy image, $\sigma = 15$ (c) Our denoised image, PSNR = 36.24, SSIM = 0.9087 (d) FoE-GA denoised image, PSNR = 37.69, SSIM = 0.9424 (e) FoE-CG denoised image, PSNR = 39.00, SSIM = 0.9707.

Table 4.6: Denoising results given by the PSNR in dB for the images from [PSW03]

σ	Lena	Barbara	Boats	House	Peppers
5	38.31	37.39	36.56	38.09	37.78
10	35.23	33.32	33.50	35.27	34.38
15	32.89	29.83	31.02	32.65	31.47
20	31.67	28.13	29.74	31.34	30.05
25	30.57	27.27	28.88	30.55	29.15
50	27.52	23.32	25.36	26.92	25.37

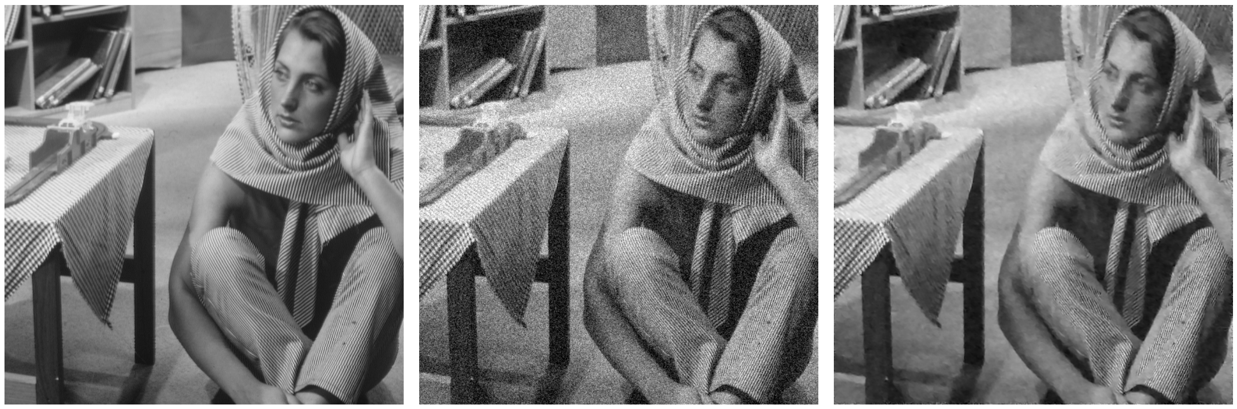
The trained model was also tested on images from a second dataset used by Portilla et al. [PSW03]. The results for additive gaussian noise at $\sigma = 5, 10, 15, 20, 25, 50$ are shown in Table 4.6. Figures 4.15,4.16, 4.17 and 4.18 show some qualitative results for images from the Portilla dataset.

For completeness, we also report the performance our implementations of other methods, which were mentioned in Chapter 2, applied to learning the parameters of the FoE model for denoising. As shown in Table 4.7 our proposed training approach produced the best testing results. The performance differences between our results and the other training methods are statistically significant at a 95% level using a signed rank test.



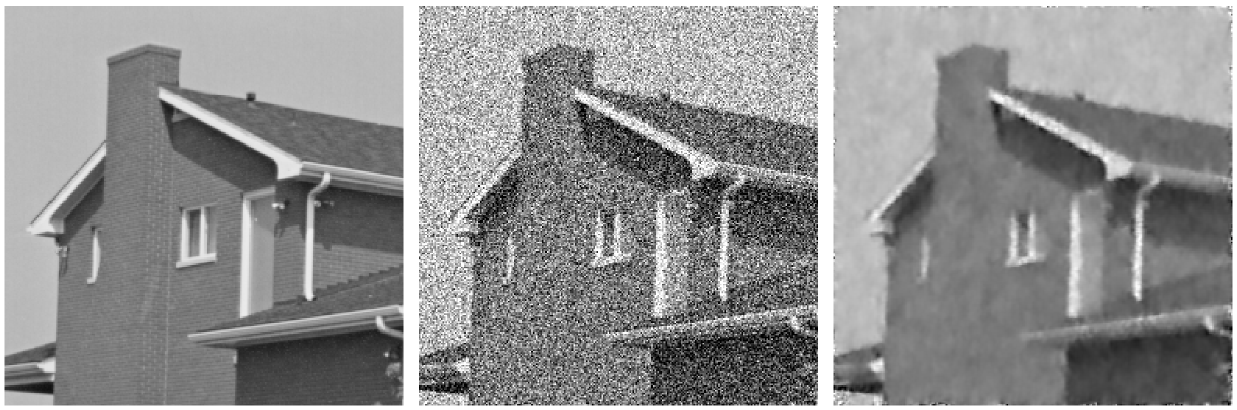
(a) (b) (c)

Figure 4.15: The example here is an image from the Portilla dataset. (a) Original image (b) Noisy image, $\sigma = 25$, PSNR = 21.53 (c) Denoised image using our model, PSNR = 28.88.



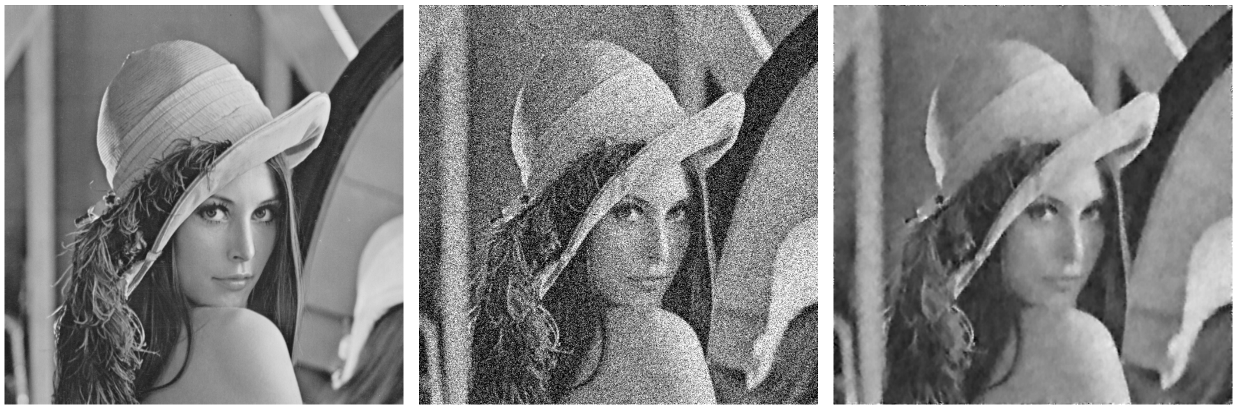
(a) (b) (c)

Figure 4.16: The example here is an image from the Portilla dataset. (a) Original image (b) Noisy image, $\sigma = 25$, PSNR = 20.26 (c) Denoised image using our model, PSNR = 27.27.



(a) (b) (c)

Figure 4.17: The example here is an image from the Portilla dataset. (a) Original image (b) Noisy image, $\sigma = 50$, PSNR = 14.12 (c) Denoised image using our model, PSNR = 26.92.



(a) (b) (c)

Figure 4.18: The example here is an image from the Portilla dataset. (a) Original image (b) Noisy image, $\sigma = 50$, PSNR = 14.16 (c) Denoised image using our model, PSNR = 27.52

Table 4.7: This table compares the results among the various methods used to learn the parameters of the FoE denoising model. The test images all had gaussian noise of standard deviation, $\sigma = 15$. Note that the FoE-GA results are obtained by terminating the optimization early, rather than finding the MAP solution in the FoE model. This ties the model to a specific optimization procedure.

Training System	Average PSNR
Scharstein and Pal [SP07]	29.56
SPSA[LH08]	29.87
Foe-CG	30.21
Variational Mode Learning[Tap07]	30.25
FoE-GA	30.55
Ours	30.58

4.6.3 Learning Advantage

In the original Field of Experts implementation , the parameters were trained in a maximum-likelihood fashion, using the Contrastive Divergence method to compute approximate gradients. Using this model, the best results, with the highest PSNR, are found by terminating the minimization of the negative log-likelihood of the model before reaching a local minimum. As can be seen from Figure 4.19, if the FoE model trained using contrastive divergence is allowed to reach a local minimum then the performance decreases significantly. This happens because maximum-

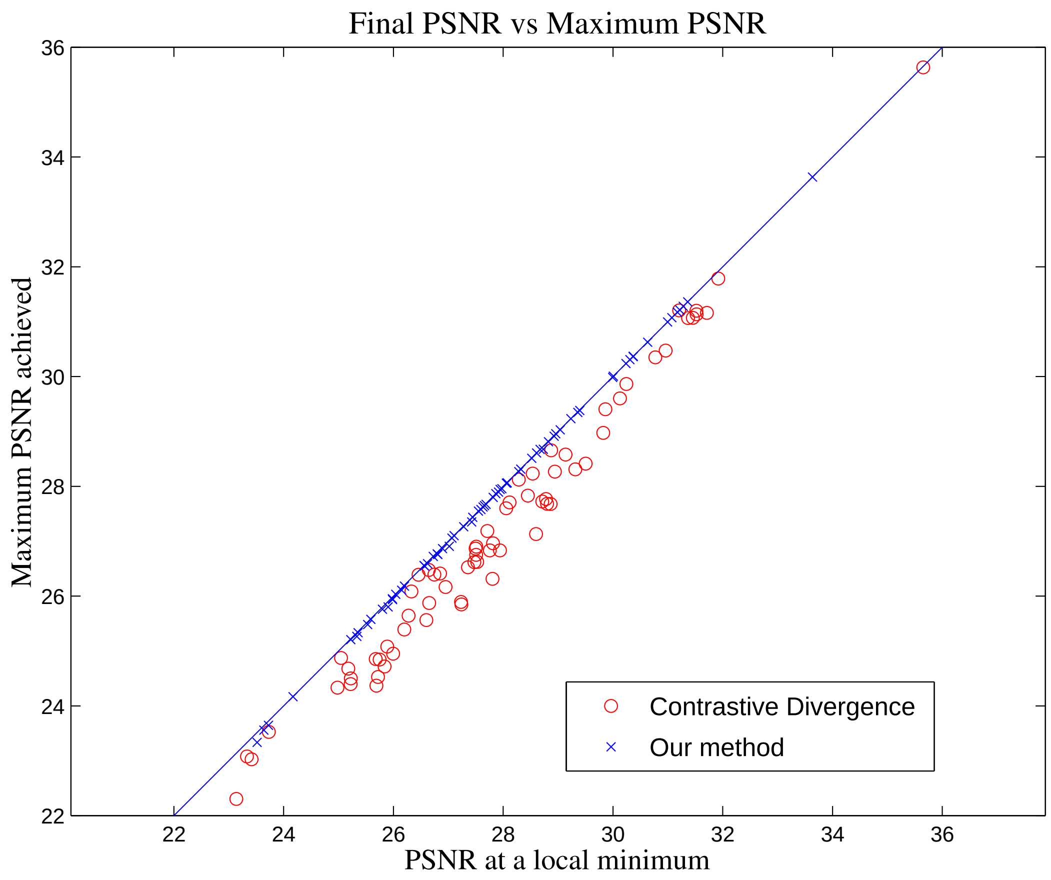


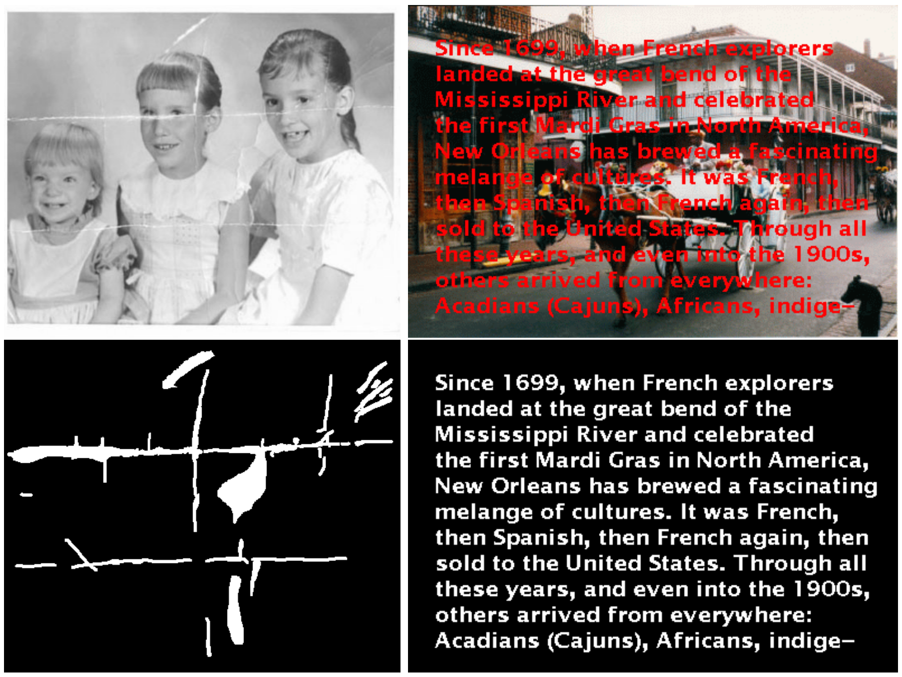
Figure 4.19: This figure shows the difference between training the FoE model using contrastive divergence and our proposed method. At termination, that is at a local minima, our training method produces results that are very close to the maximum PSNR achieved over the course of the minimization on images with additive Gaussian noise, $\sigma = 15$.

likelihood criterion that was used to train the system is related, but not directly tied to the PSNR of the MAP estimate.

The argument is that if the intent is to use MAP estimates and evaluate the estimates using some criterion, like PSNR, a better strategy is to find the parameters such that the quality of MAP estimates are directly optimized. Returning to the Field of Experts model in the previous paragraph, if the goal is to maximize PSNR, maximizing a likelihood and then hoping that it leads to MAP estimates with good PSNR values is not the best strategy. It can be argued that one should instead choose the parameters such that the MAP estimates have the highest PSNR possible. It should be noted that our method is not tied to the PSNR image quality metric - any differentiable image loss function can be used. As Figure 4.19 shows, when the FoE model is trained using our proposed approach, the PSNR achieved when the minimization terminates is very close to the maximum PSNR achieved over the course of the minimization. This is important because the quality of the model is not tied to a particular minimization scheme. Using the model trained with our method, different types of optimization methods could be used, while maintaining high-quality results.

4.6.4 Inpainting

The trained model was also used to perform inpainting in a similar manner to what is done by Roth and Black [RB05]. The goal here is to remove unwanted parts of an image such as scratches and occluding objects while keeping the reconstructed parts as realistic as possible. Only the FoE prior is used thus making the energy function to be minimized as follows:



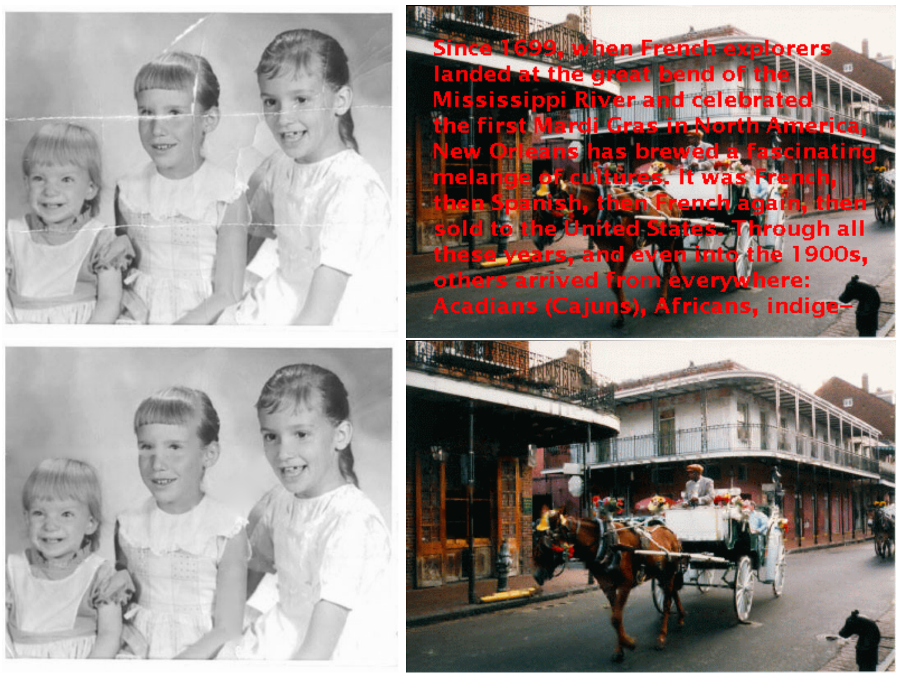
Since 1699, when French explorers landed at the great bend of the Mississippi River and celebrated the first Mardi Gras in North America, New Orleans has brewed a fascinating melange of cultures. It was French, then Spanish, then French again, then sold to the United States. Through all these years, and even into the 1900s, others arrived from everywhere: Acadians (Cajuns), Africans, indige–

Figure 4.20: This figure shows the masks used when computing the gradients. Only the gradients corresponding to the white pixels were updated during the minimization process Top: Original image. Bottom: Image Masks

$$E(\mathbf{x}, \mathbf{y}; \alpha, \beta) = \sum_{i=1}^{N_f} e^{\alpha_i} \sum_{p=1}^{N_p} \log(1 + (F_i \mathbf{x}_p)^2). \tag{4.26}$$

Rather than updating all the pixels in the image, only the gradient of pixels specified by a user supplied mask are updated. Examples of the masks are shown in Figure 4.20. The gradient of all pixels outside of the mask are set to zero during the minimization. The minimization in this case was done using gradient descent with the 'bold driver' method.

There was no additional learning done for the inpainting. The same parameters learnt for denoising were used for inpainting. Since the parameters were learnt for the denoising of gray



Since 1699, when French explorers
landed at the great bend of the
Mississippi River and celebrated
the first Mardi Gras in North America,
New Orleans has brewed a fascinating
melange of cultures. It was French,
then Spanish, then French again, then
sold to the United States. Through all
these years, and even into the 1900s,
others arrived from everywhere:
Acadians (Cajuns), Africans, indige-

Figure 4.21: This figure shows the results when our trained model was applied to inpainting. Top: Original image. Bottom: Restored image

scale images, the colour images used for the inpainting were converted to the YCbCr color space. Each channel was then minimized separately and the results recombined at the end.

Figure 4.21 shows the inpainting results which are qualitatively similar to those in [RB05]. This shows that even though the model is trained differently it is still able to perform the same tasks as well as, or even better, than the FoE model trained using contrastive divergence.

4.7 Summary

To summarize, we have introduced a new approach for discriminatively training MRF parameters. In this approach, the MRF parameters are directly optimized so that the MAP solution scores as well as possible. This new approach allows one to develop a framework that is flexible, intuitively easy to understand and relatively simple to implement.

We have demonstrated the effectiveness of the proposed approach by using it to learn the parameters for the Field of Experts model for denoising images. The performance of the model trained using the proposed method compares well with the results of the Field of Experts model trained using contrastive divergence when applied to the same tasks.

CHAPTER 5
ENERGY-BASED SEGMENTATION

5.1 Introduction

Image segmentation is one of the fundamental tasks in computer vision. Segmented images play an important role in a wide range of applications such as object classification or identification. However, this is not a trivial task due to the diversity and complexity in images and as a result, segmentations from current methods are generally not up to par with human segmentations of natural images.

Segmentation methods can be broadly grouped into two main categories: (i) object-specific segmentation [HZC04, KTZ05, LW06, VT07] and (ii) general segmentation techniques that are driven by local image features, such as edges or pixel color. Graph-based methods, such as [SM00, Wei99, WL93], are a particularly powerful and popular family of segmentation approaches because they make it possible to control the overall system behavior with local affinities that can properly handle edges and other boundaries.

We present a general segmentation system based on an energy minimization framework that is novel in how it incorporates priors on the size of the segments in a way that is straightforward to implement. For a segmentation system based on minimizing a criterion, such as the normalized cuts method, the criterion must incorporate the size of the segments. As pointed out by Shi and

Malik[SM00], if the energy function does not consider the size of the segments, the result of the minimization will tend to result in tiny segments. Thus, the normalized cuts criterion is structured to prefer segmentations where the segments are roughly the same size.

However, recent work by Sudderth et al.[SJ09] has pointed out that the size of segments in natural images tend to follow a heavy-tailed distribution. This distribution can be interpreted as stating that natural images tend to have many smaller segments with a few large segments. Interestingly, these properties can also be seen in the scale invariant properties [TO03] of natural images, histograms of contour lengths [RM03], in addition to the segment areas [MFT01] of human segmentations. Building on this, Sudderth et al. propose a probabilistic method for segmentation, built on the Pitman-Yor process [PY97]. As shown by Sudderth et al.[SJ09], this model is able to produce segmentations that match human-created segmentations better than those produced with the normalized cuts methods. The segmentations themselves are produced using a series of variational updates.

We describe the general model in more detail in Section 5.2. As will be shown, unlike Sudderth et al.[SJ09], the segmentation is achieved using gradient-based optimization of an energy function which is straightforward to implement with standard techniques. The effectiveness of our proposed model is demonstrated in Section 5.4. We will show that the method produces segmentations which overcome the uniform segment size limitations of normalized cuts and are also of comparable quality to Sudderth et al.[SJ09], while being easier to implement.

5.2 General algorithm

In this work, the segmentation problem is setup as a multi-class labeling task using an energy minimization framework. Each image is represented with a set of continuous-valued responses per segment. The goal is to determine the response image, $\mathbf{x}_k$, for each segment, k, which is then converted to probabilities using a softmax function to produce the final classification. While the number of segments is not specified, the process begins with at most ten segments per image and the system automatically refines the number of segments.

5.2.1 Softmax

Since we are using a multi-class formulation we use a multinomial logistic model or softmax function [Fig05], to transform the response images to probabilities. The probability of a pixel p belonging to class k is given by the softmax function [Bis06] defined as:

$$p(C_k|\mathbf{x}_p) = y_p^k = \frac{e^{\mathbf{x}_p^k}}{\sum_{\bar{c}} e^{\mathbf{x}_p^{\bar{c}}}} \tag{5.1}$$

where $\mathbf{x}_p^k$ represents the response for pixel p in segment k.

5.2.2 Cost function formulation

In this section we describe the underlying cost function and the contribution of the different terms of which it is comprised.

5.2.2.1 Primary cost function

At the heart of the segmentation model is the cost function, $E(\mathbf{x})$, defined as:

$$\begin{aligned} E(\mathbf{x}) &= K_1 \sum_k^{N_k} \sum_{<p,q>} w_{(p,q)} (\mathbf{x}_p^k - \mathbf{x}_q^k)^2 \\ &+ K_2 \sum_k^{N_k} -S_k \log\left(\frac{1}{N_p} \sum_p^{N_p} \frac{e^{\mathbf{x}_p^k}}{\sum_{\bar{c}} e^{\mathbf{x}_p^{\bar{c}}}}\right) \\ &+ K_3 \sum_p^{N_p} -\log \sum_k^{N_k} \left(\exp\left(\frac{e^{\mathbf{x}_p^k}}{\sum_{\bar{c}} e^{\mathbf{x}_p^{\bar{c}}}}\right)\right), \end{aligned} \tag{5.2}$$

where N_p is the number of pixels and N_k is the current number of segments being considered. The parameters $w_{(p,q)}$ and S_k are discussed in Sections 5.2.3.1 and 5.2.3.2

This cost function is comprised of three terms, described below, whose contribution to the overall function is determined by the parameters K_i.

We can use the definition of the probability y_p^k in Equation 5.1 to rewrite Equation 5.2 more concisely as

$$\begin{aligned} E(\mathbf{x}) &= K_1 \sum_k^{N_k} \sum_{<p,q>} w_k(p,q) (\mathbf{x}_p^k - \mathbf{x}_q^k)^2 \\ &+ K_2 \sum_k^{N_k} -\lambda_k S_k \log\left(\frac{1}{N_p} \sum_p^{N_p} y_p^k\right) \\ &+ K_3 \sum_p^{N_p} -\log \sum_k^{N_k} \left(\exp\left(y_p^k\right)\right) \end{aligned} \tag{5.3}$$

and its derivative as

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{x}_p^j} &= 2K_1 w_j(p,q)(\mathbf{x}_p^j - \mathbf{x}_q^j) \\
&- K_2 \sum_k^{N_k} \lambda_k S_k \left(\sum_i^{N_p} y_i^k \right)^{-1} y_p^k (I_{jk} - y_p^j) \\
&- K_3 \left(\sum_k^{N_k} \exp\left(y_p^k\right) \right)^{-1} \cdot \exp\left(y_p^k\right) \cdot y_p^k (I_{jk} - y_p^j),
\end{aligned} \tag{5.4}$$

where I_{jk} are elements of the identity matrix.

In this formulation, we can now use a standard numerical optimization routine to minimize the cost function to determine the optimal response images. We use conjugate gradient to perform the experiments in this work.

5.2.3 Decomposition of the cost function

In this section we give a more detailed explanation of each of the terms in the cost function. For clarity, we define the three terms in Equation 5.3 as: (i) Smoothness term (E_{Sm}), (ii) Segment-Size Prior term (E_{SSP}) and (iii) Separation term (E_{Sep}).

5.2.3.1 Smoothness term

$$E_{Sm} = \sum_k^{N_k} \sum_{<p,q>} w_{(p,q)} (\mathbf{x}_p^k - \mathbf{x}_q^k)^2 \tag{5.5}$$

This term implements inter-pixel relationships and has a smoothing effect on the responses. Local feature cues are incorporated in this term since it is weighted by w, which is defined using the probability of boundary(Pb) [MFM04] values of the images based on color and texture. We define w as:

$$w_{p,q} = \exp(-\gamma \| Pb(p) - Pb(q) \|) \tag{5.6}$$

The parameter γ determines how strong the difference between pixels counts. This weighting encourages discontinuities or breaks in the response images at points which correspond to higher probability of boundary values.

5.2.3.2 Segment-Size prior term

$$E_{SSP} = \sum_{k}^{N_k} -S_k \log \left(\frac{1}{N_p} \sum_{p}^{N_p} \frac{e^{\mathbf{x}_p^k}}{\sum_{\bar{c}} e^{\mathbf{x}_p^{\bar{c}}}} \right) \tag{5.7}$$

This term enforces the prior on the size of the segments. The size of the segments are specified with the parameters S_k. While S_k values do not have to be exact matches to the proportion of the segment in the image, they do,however, need to be specified in descending order of size. The contribution of this term is demonstrated in Figure 5.1. We show the resulting segmentations with S specified as well as a result with the segment-size term turned off. In order to model a heavy-tailed distribution we use $S = 1/k$, where segments, $k = 1$ to N, are specified in decreasing order of size. Since the segmentation is also highly affected by the image-specific cues in the smoothness term it is not necessary for this term to precisely match the distribution of segment sizes.

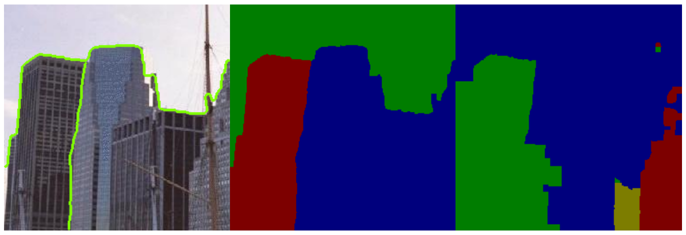


Figure 5.1: Segmentation results showing the effect of the segment-size term. (a) Image showing edge boundaries from the segmentation with the segment-size term turned on (b) Segmentation with segment-size term turned on (c) Segmentation with segment-size term turned off

This term is based on the log Dirichlet distribution, which is the conjugate prior for the multinomial distribution. Because the Dirichlet distribution is a distribution over distributions, it is used to measure whether the distribution of pixels in segments matches the sizes defined with the S_k terms.

While Sudderth et al.[SJ09] points out that the Pitman-Yor process can model distributions with heavier tails, we have found that basing this term on a Dirichlet distribution works well and makes implementation easier.

5.2.3.3 Separation term

$$E_{Sep} = \sum_{p}^{N_p} - \log \sum_{k}^{N_k} \left(\exp \left(\frac{e^{\mathbf{x}_p^k}}{\sum_{\bar{c}} e^{\mathbf{x}_p^{\bar{c}}}} \right) \right) \tag{5.8}$$

In this formulation there are two ways of satisfying the segment-size prior constraints. If, for example, we wish to get two segments, A and B, with sizes corresponding to 55% and 45% of the image respectively, then two scenarios would satisfy the segment size constraint:

1. Having 55% of pixels with probability 1 and the remaining 45% with probability 0 of belonging to segment A and vice versa for segment B
2. Having all pixels with a probability of 0.55 belonging to segment A and all pixels with a probability 0.45 belonging to segment B.

In computing the size of a segment we make the assumption that a pixel would move towards probability 1 in a chosen segment while moving to 0 in the remaining segments. This is not the case in scenario (2), so, as a result, we introduce the separation term to avoid scenario (2). This term is based on a differentiable approximation of the maximum defined as $\max(a, b, ..., c) \approx \log(e^a + e^b + ... + e^c)$. In our setup, minimizing Equation 5.8 helps to move the probability of a pixel belonging to a certain segment towards 1 while minimizing its probability across the other segments. In other words, it forces the pixel to choose one segment as in scenario (1).

5.3 Implementation details

5.3.1 Enforcing segment size constraints

In order to enforce the segment size constraints, our model expects the segments to be in decreasing order of size. To enforce this restriction, we propose two methods: (i)sorting and (ii)fusion moves.

5.3.1.1 Sorting and pruning

In sorting, after a fixed number of iterations, we compute the probabilities for each of the current response images and generate an intermediate segmentation. The segments are then sorted according to their proportional size in the image.

For additional speed, after sorting, the response images for segments that account for less than a chosen percentage of the image can be eliminated. Based on empirical evidence, we suggest a choice of 5%. The sorting and pruning procedure is outlined in Figure 5.2. The pruning of segments helps the system to dynamically adjust the number of segments under consideration thereby speeding up the final optimization stage since there are less segments to consider.

It should be noted that the number of segments remaining after these two steps is not necessarily the final number of segments in the image. As can be seen in Figure 5.2, four segments were left after the sorting and pruning stages, however, only two segments were in the final seg-

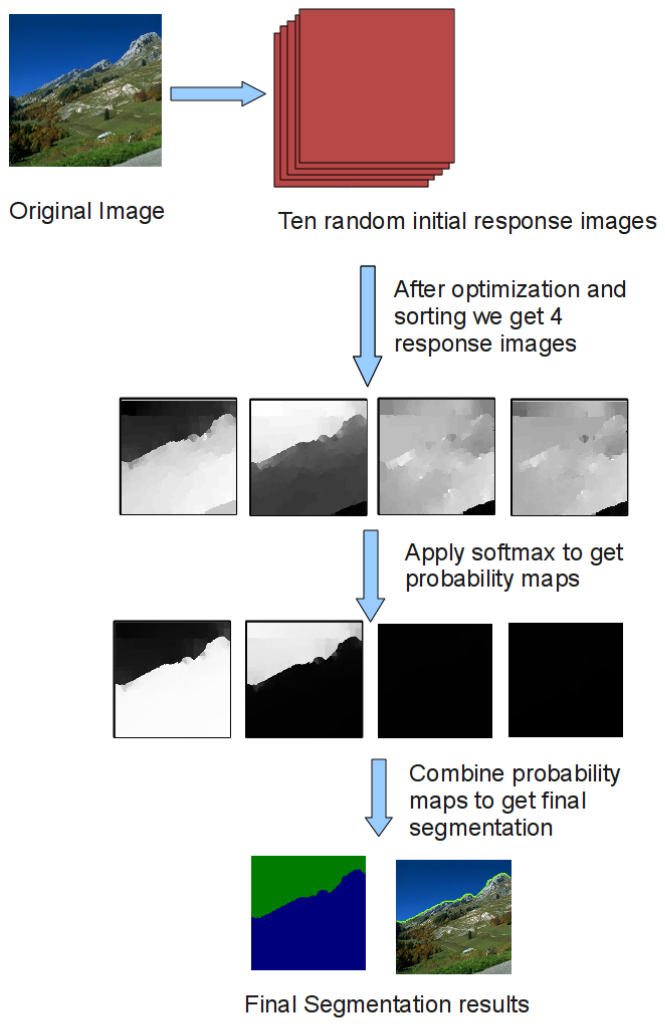

Original Image
Ten random initial response images
After optimization and sorting we get 4 response images
Apply softmax to get probability maps
Combine probability maps to get final segmentation
Final Segmentation results

Figure 5.2: This figure shows the main steps in the segmentation process when sorting and pruning are used. In this case, six of the ten original segments were eliminated during the sorting phases.

mentation. The fewer segments we have to consider the faster the final optimization stage will be.

5.3.1.2 Fusion moves

This method allows the segments to grow or shrink appropriately to match the size constraints. The cost function defined earlier in Equation 5.2 is modified to introduce fusion moves. Motivation for the fusion moves comes from graph-cut based methods like α-expansion [BVZ01] or LogCut [LRB07]. Related work using this idea of fusion moves on stereo problems has recently been proposed by [TPC, LRR09] where two proposed solutions are fused to form a more accurate solution. Inspired by this approach, we use the same principle to fuse two response images so that the overall energy function is minimized.

Given a response image from the intermediate segmentation, x^{α}, representing label α, a fusion move between another label, β, produces a new response image, $x^{n\alpha}$. This move is defined as:

$$x^{n\alpha} = (1 - \phi(z_n))x^{\alpha} + \phi(z_n)(x^{\beta}), \tag{5.9}$$

where $\phi(z_n)$ is defined as the logistic function $\phi(z_n) = \dfrac{1}{1 + \exp(-z_n)}$. In this setup, a pixel, p, in $x^{n\alpha}$ either retains its original value or takes the value of the corresponding pixel in x^{β} depending on the value of $z_n(p)$. If the value of z_n is small(less than -10) then $\phi(z_n) \approx 0$, whereas more positive values of z_n move ϕ towards 1.

Substituting $x^{n\alpha}$ appropriately in **x** leads to $E(\mathbf{x})$ now being dependent on z. As a result, in the optimization, we minimize E by differentiating with respect to z_n. The gradient is calculated using the chain rule as

$$\frac{\partial E}{\partial z_n} = \frac{\partial E}{\partial x^{n\alpha}} \frac{\partial x^{n\alpha}}{\partial \phi} \frac{\partial \phi}{\partial z_n} \tag{5.10}$$

To avoid any bias towards a particular segment, the fusion moves between the response images are performed in random order. This was done twice to allow the response image values to stabilize adequately. Also, z is uniformly initialized to -10 making $\phi(z_n) \approx 0$. This means that initially $x^{n\alpha} = x^{\alpha}$.

The effect of the fusion moves are shown in Figure 5.3. In this figure the results are shown after each stage in the segmentation process.

5.3.2 Superpixels

To speed up the segmentation process we performed the optimization on superpixels rather than all the pixels in the image. The images are over-segmented into about 500-600 superpixels using the watershed function from Matlab(R) applied to the probability of boundary(Pb) map provided by Martin et al. [MFM04]. We can introduce the superpixels into our formulation by constructing a superpixel matrix, S, similar to that used by Tappen et al. [TSD08]. Our optimization is performed using the intermediate image $\mathbf{x}_s$, and we recover the full response image, **x**, at the end using $\mathbf{x} = S\mathbf{x}_s$.

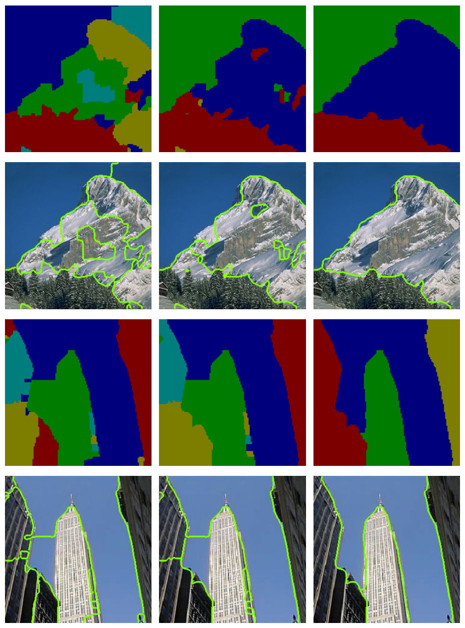


Figure 5.3: Results from two images showing examples of the intermediate results in the segmentation process when fusion is used. From left to right: initial segmentation, segmentation after fusion step, final segmentation

Since the distribution of segment labels are measured over the actual number of pixels in the image we also need to recover the probabilities per pixel using a similar relationship, $\mathbf{p} = S\mathbf{p}_s$, where $\mathbf{p}$ represents the probability per pixel and $\mathbf{p}_s$ represents the probability per superpixel. Multiplying by the superpixel matrix copies the current value of the superpixel to the member pixels in the full-size representation.

In our experience, using superpixels improves speed but does not sacrifice performance.

5.3.3 Learning

The segmentation system was trained using the method proposed in Chapter 3. The discriminative setup is formulated as the following minimization:

$$\begin{aligned} &\min_{\theta} \quad L(\mathbf{x}^*(\theta), \mathbf{t}) \\ &\qquad \text{where } \mathbf{x}^*(\theta) = \arg\min_{\mathbf{x}} E(\mathbf{x}; \theta). \end{aligned} \tag{5.11}$$

The parameters are represented by θ and in this case, training was done to learn the parameters K_i and γ in Equation 5.2. The loss function used was the negative log-likelihood defined as:

$$L(\mathbf{x}^*(\theta), \mathbf{t}) = \sum_{k}^{N_k} \sum_{p}^{N_p} \log(1 + \exp(-t_p^k y_p^k)). \tag{5.12}$$

The ground-truth probability of a pixel, p, belonging to segment k is represented by t_p^k. The probability of the pixel belonging to segment k is y_p^k calculated using Equation 5.1.

Learning was done using the cost function defined in Equation 5.2, after the initial sorting or fusing steps using the intermediate segmentation as the input. The learning was done at this stage since we would not be able to compute the necessary gradients, required for the learning procedure, between the sorting or fusion steps if it was done earlier in the process. Being able to learn the parameters not only provides the benefit of avoiding hand-tuned parameters but allows one to take advantage and use the labeled datasets currently available. There was also an increase of at least 0.05 in the average Rand index using the learnt parameters compared to the best hand-tuned results.

5.4 Experiments and results

The segmentation algorithm was tested using images from Oliva and Torralba's [TO03] scene category dataset from the LabelMe database[1]. Three scene categories were used: tall buildings, mountains and coastal images. This is the same dataset used by Sudderth et al. [SJ09] and results are compared in a similar fashion against the multi-scale normalized cuts clustering method [CBS05], with varying numbers of segments ranging from 2 - 10.

The results presented in this section were obtained using the learnt parameters with fusion moves to enforce the correct ordering of segments. One optimization step was performed to obtain a basic initial segmentation. For the sake of overall speed, the optimization in this step is not allowed to converge completely. The fusion move optimization is then done to allow segments to adjust sizes appropriately. The third and final optimization step is similar to the first except it is allowed to converge.

[1]Available at `http://labelme.csail.mit.edu/browseLabelMe/spatial_envelope_256x256_static_8outdoorcategories.html`.

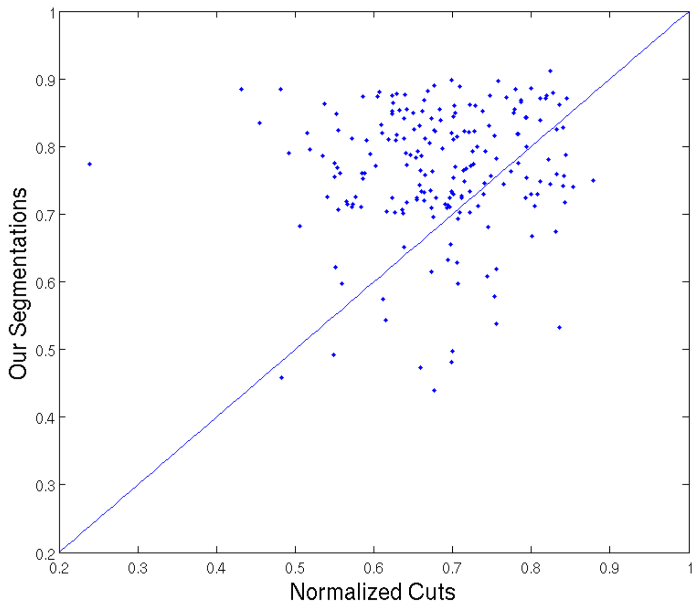


Figure 5.4: Scatter plot comparing the Rand index of our segmentations vs. NCut(6) segmentations for 200 mountain images

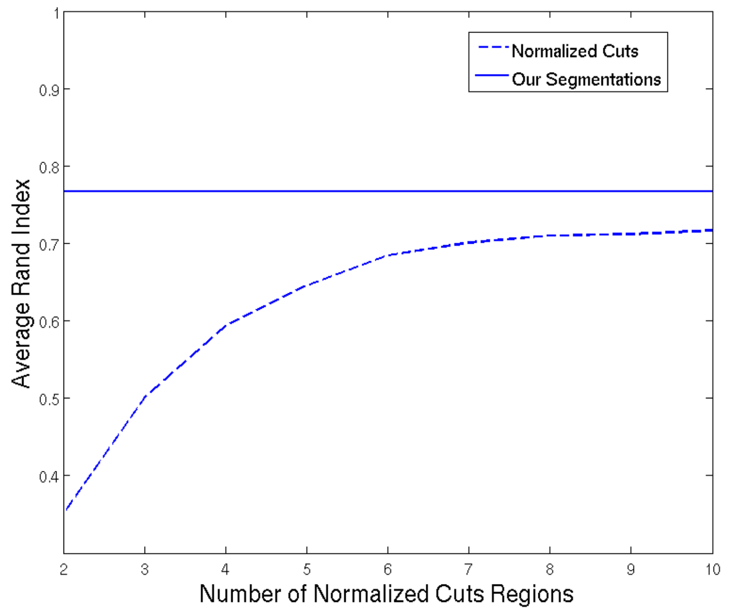


Figure 5.5: Average Rand Indexes for 200 mountain images for NCut(2-10)

The segmentation results and the NCuts segmentations were compared against the LabelMe human segmentation using the standard Rand Index[Ran71] defined as:

$$RI(S,S') = 1 - \frac{1}{\binom{N}{2}}\left(\frac{1}{2}\left(\sum_i\left(\sum_j p_{ij}\right)^2 + \sum_j\left(\sum_i p_{ij}\right)^2\right) - \sum_{i,j} p_{ij}^2\right), \tag{5.13}$$

where p_{ij} is the number of pixels with label i in segment S and label j in segment S'. N represents the total number of pixels in the image.

Figures 5.5 and 5.7 show the average Rand Index of the segmentations produced using our method and the normalized cuts segmentations computed on 200 randomly chosen mountain and tall building images.

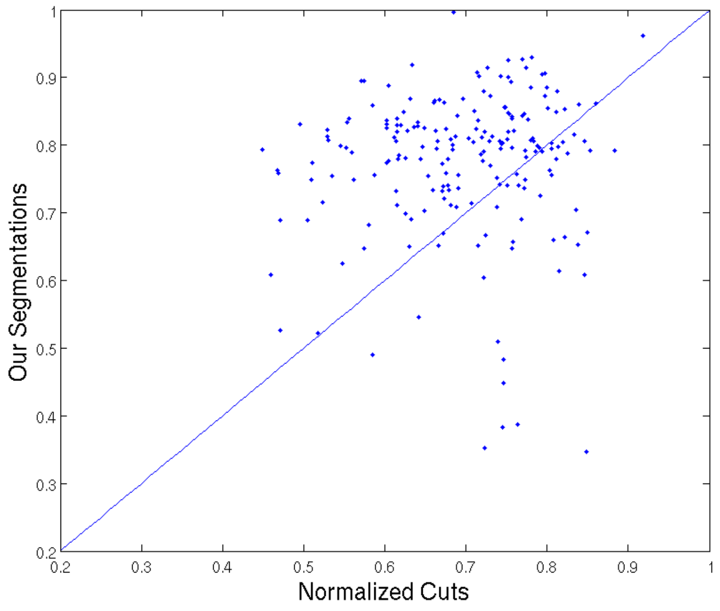


Figure 5.6: Scatter plot comparing the Rand index of our segmentations vs. Ncut(6) segmentations for 200 tall building

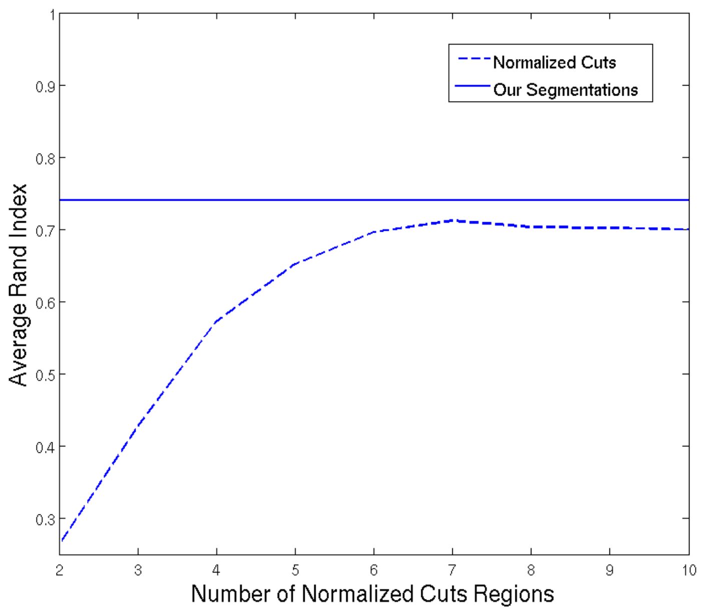


Figure 5.7: Average Rand Indexes for 200 tall building images for NCut(2-10)

While a complete direct comparison to [SJ09] cannot be shown due to the unavailability of the data, the segmentations produced using this new approach are in most cases just as good or better to those produced using the system from Sudderth et al. [SJ09]. A comparison between a few images extracted directly from Sudderth et al. [SJ09] is shown in Figure 5.8.

In addition to using the LabelMe dataset, experiments were also conducted using the Berkeley Segmentation Dataset(BSDS) [MFT01]. The dataset contains 200 training images and 100 test images along with ground-truth segmentation from multiple human subjects. In this set of experiments, the segmentations were evaluated using the probabilistic Rand Index and Variation of Information[Mei05]. The probabilistic Rand Index is computed by averaging the standard Rand Index[Ran71] among the different ground-truth segmentations for each image. The Variation of

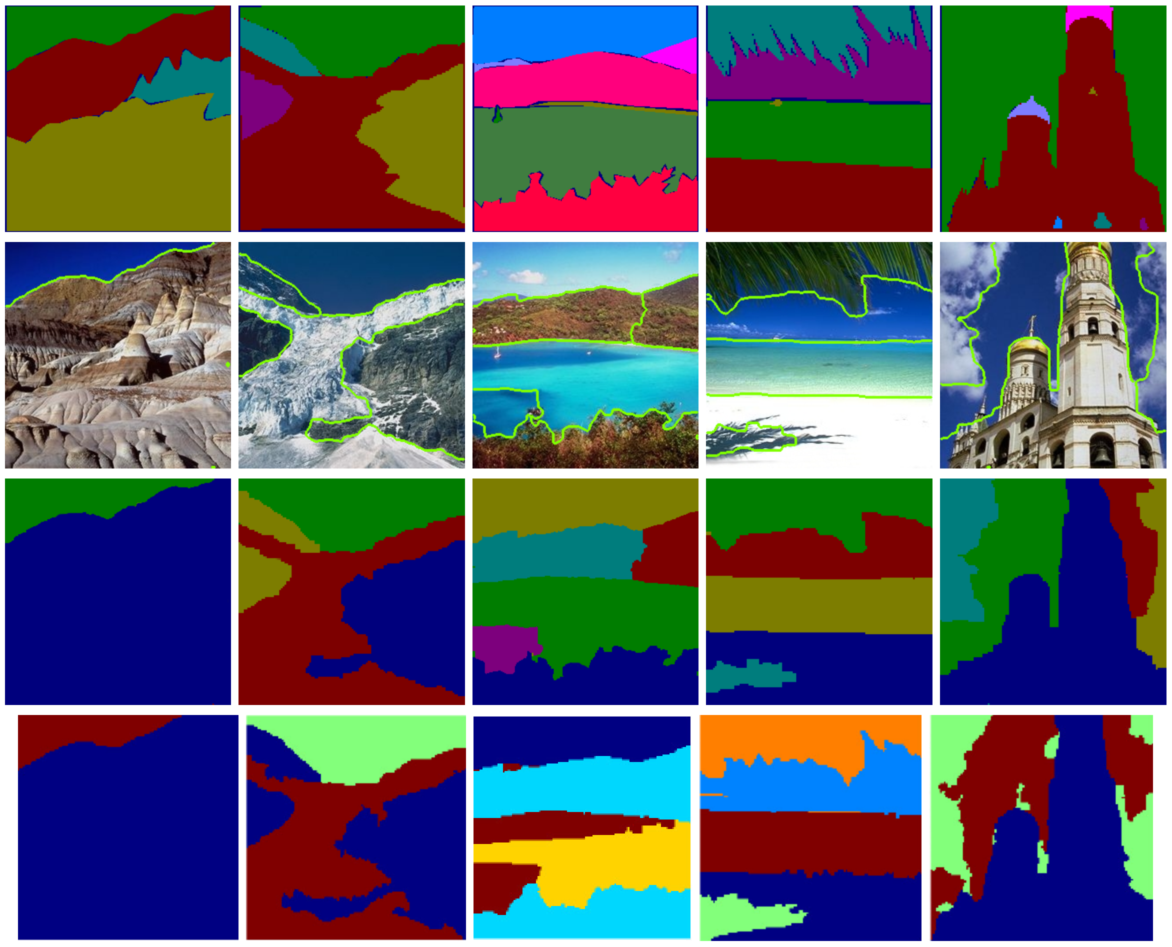


Figure 5.8: Sample results comparing our results to images extracted from [SJ09]. From Top to Bottom: LabelMe human segments, original image with boundaries inferred from our segmentation, our segmentation, PY- Edge segmentation [Best viewed in color]

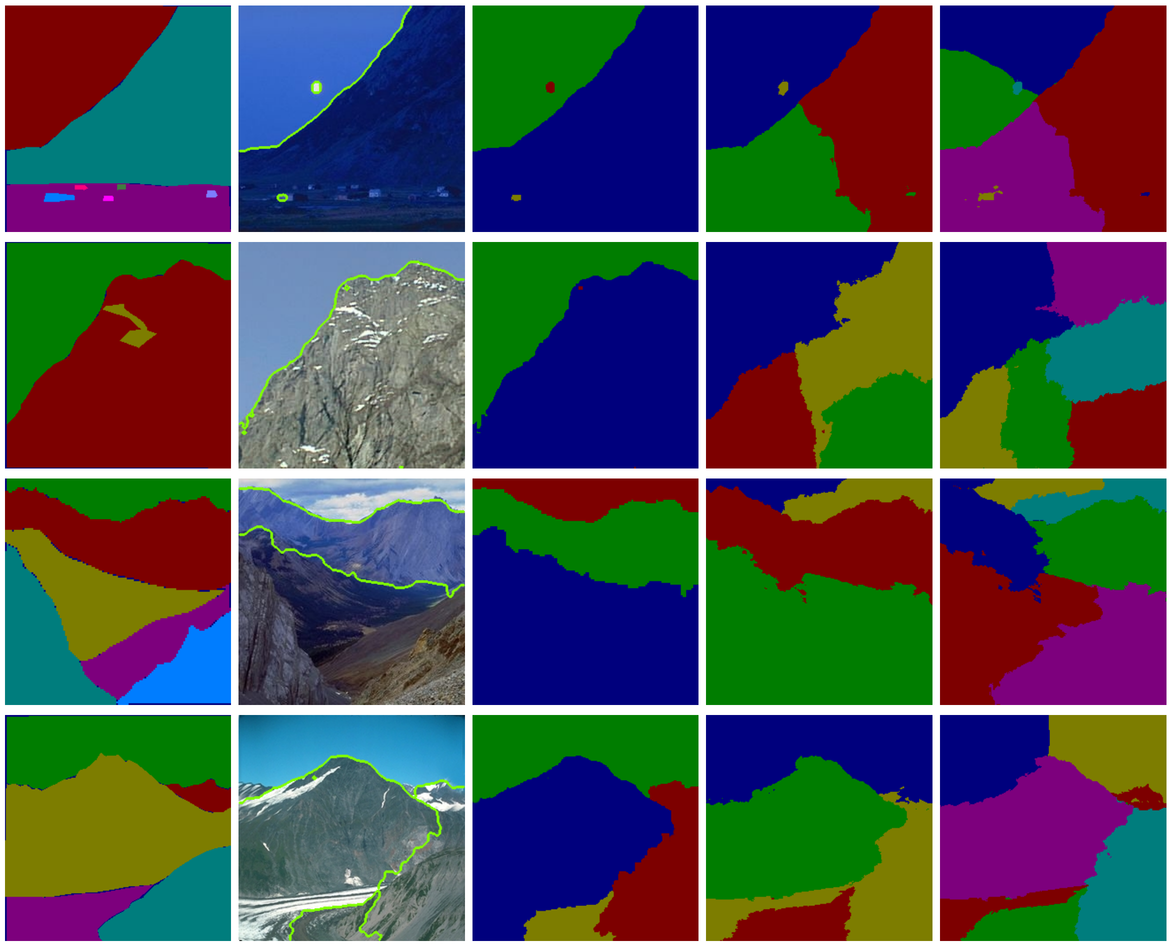


Figure 5.9: Sample results from the mountain images. From left to right: LabelMe human segments, original image with boundaries inferred from our segmentation, our segmentation, NCut(4), NCut(6) [Best viewed in color]

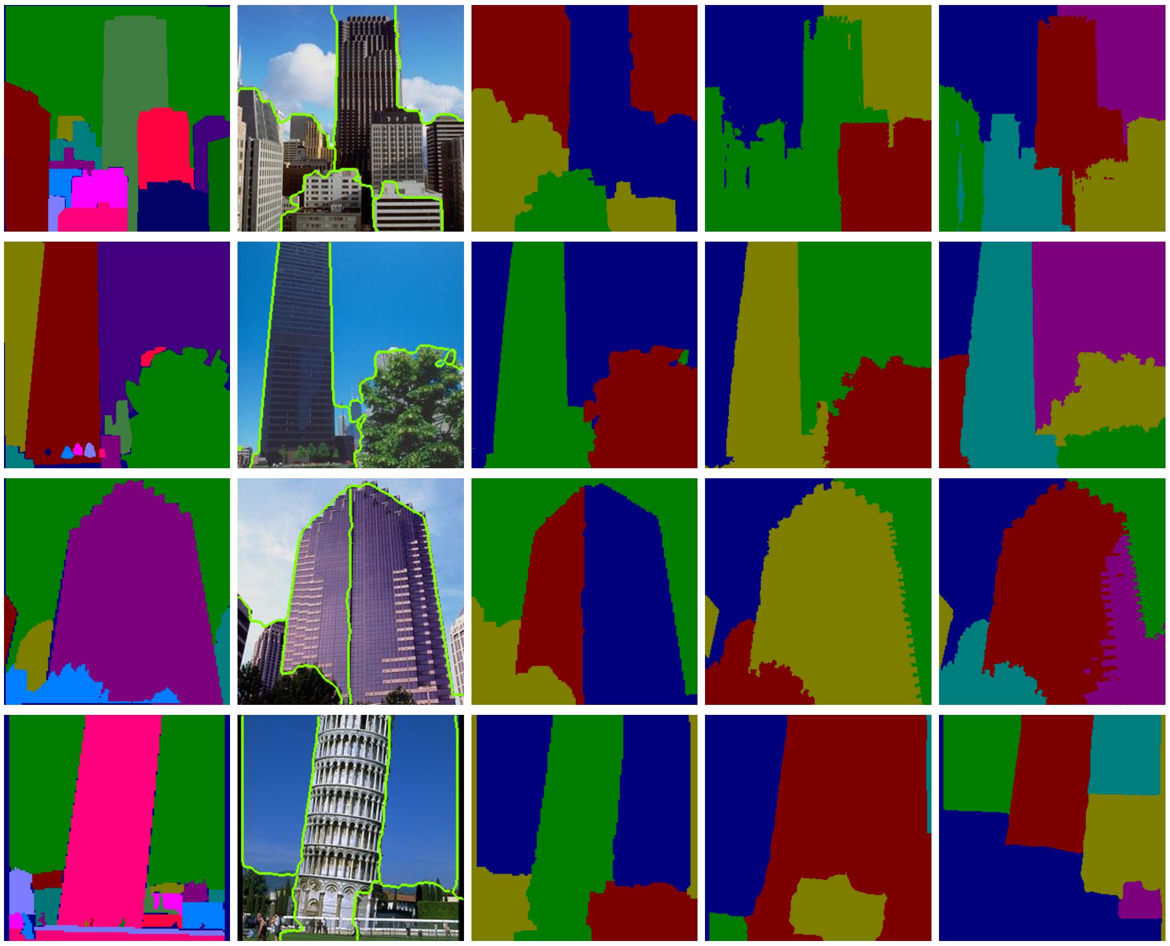


Figure 5.10: Sample results from the tall building images. From left to right: LabelMe human segments, original image with boundaries inferred from our segmentation, our segmentation, NCut(4), NCut(6) [Best viewed in color]

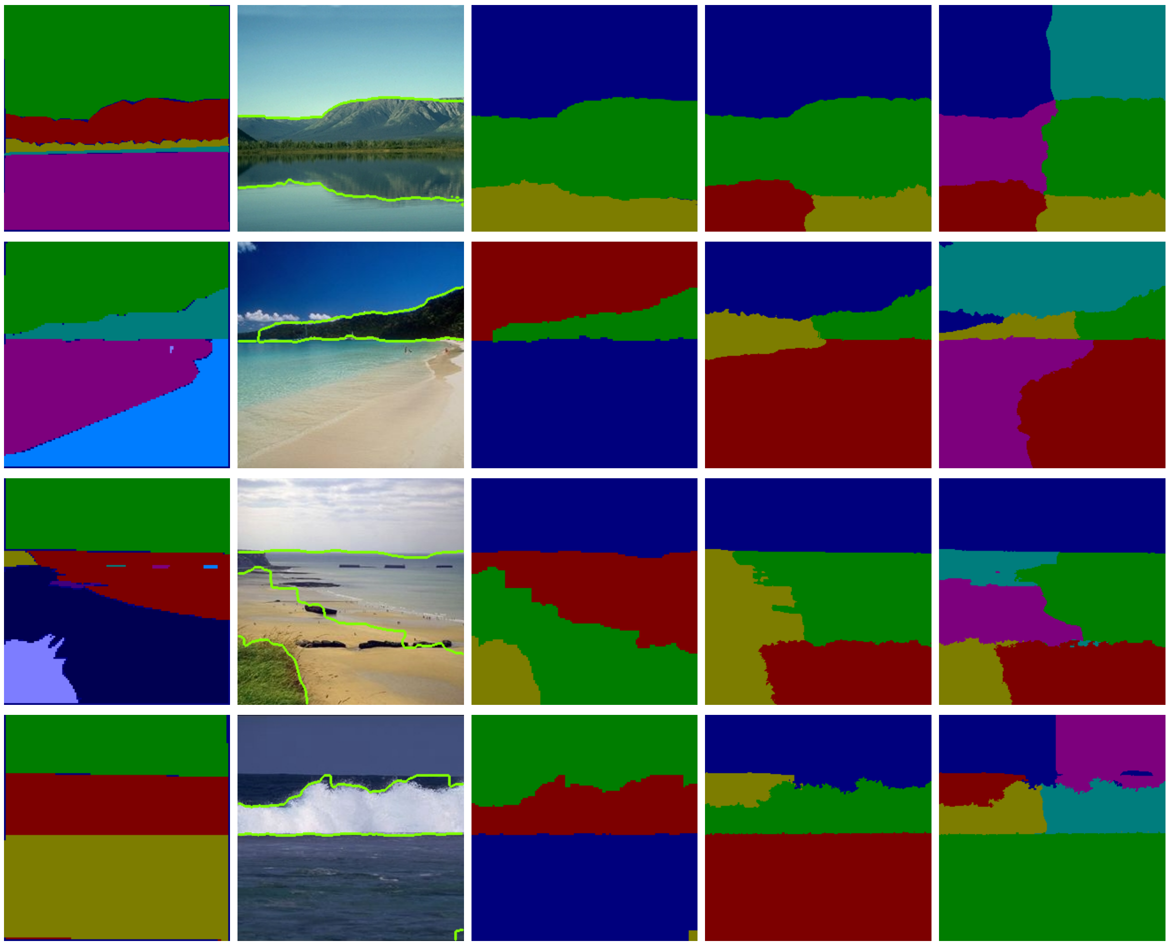


Figure 5.11: Sample results from the coastal images. From left to right: LabelMe human segments, original image with boundaries inferred from our segmentation, our segmentation, NCut(4), NCut(6) [Best viewed in color]

Table 5.1: Here results are shown comparing segmentation methods against their respective ground-truths using the probabilistic Rand Index(PRI) and Variation of Information(VI) benchmarks.

Method	PRI	VI
Ours	0.80	1.73
Multiscale NCuts	0.75	2.16
Mean Shift	0.78	1.78
gPb-owt-ucm[2]	0.81	1.68

Information (VI) benchmark is based on the entropy between two segmentations and is defined as

$$VI(C, C') = H(C) + H(C') - 2I(C, C'), \tag{5.14}$$

where H represents the entropies and I the mutual information between two segmentations C and C'. The smaller the VI value, the closer the segmentations being compared are to each other.

As shown in Table 5.1, the algorithm performs the best when compared to the multi-scale normalized cuts [CBS05] and mean shift [CMM02] algorithms. Results from Arbelaez et al.[AMF09] have also been included. Arbelaez et al. [AMF09] proposed a two step segmentation method that produces segmentations from contours using an oriented watershed transform and an ultrametric contour map. While our proposed segmentation method also uses contours as an input, we are able to generate comparable segmentations using a relatively straightforward energy minimization framework.

[2]Averages are taken from [AMF09] and are based on all 300 BSDS images

5.5 Summary

A new approach for segmenting images using low-level features was presented in this chapter. The segmentation problem is setup in a energy minimization framework that is easy to implement and maintain. This approach is intrinsically multi-class and provides better segmentations unlike previous data-driven methods which tend to produce similar sized segments. It is also able to automatically discover the segments present without any user input.

CHAPTER 6
CONCLUSION

This thesis introduces a new approach for discriminatively training MRF parameters. The MRF model is trained by optimizing its parameters so that the minimum energy solution of the model is as similar as possible to the ground-truth. We demonstrate how implicit differentiation, can be used to analytically differentiate the overall training loss with respect to the MRF parameters. This leads to an efficient, flexible learning algorithm that can be applied to a number of different models.

The effectiveness of the proposed learning method is demonstrated by learning the parameters of two related models applied to the task of denoising images. The two models used are the Robust Derivative [Tap07] and Field of Experts [RB05] models. Those two models are used as examples to show how the proposed learning method can be practically applied. It is also shown that the proposed learning algorithm produces comparable and, at times, better performance over other learning methods.

In addition to the introduction of a new learning algorithm, a new segmentation model is also introduced. The novelty of this new segmentation model is how it incorporates priors on the size of the segments in a way that is straightforward to implement. Unlike related systems based on minimizing a criterion, the number of segments are not specified and the system produces more realistic segmentations. The effectiveness and flexibility of the proposed learning method is shown by using it to train the proposed segmentation model.

LIST OF REFERENCES

[AMF09] Pablo Arbelaez, Michael Maire, Charless C. Fowlkes, and Jitendra Malik. "From contours to regions: An empirical evaluation." In *IEEE Computer Vision and Pattern Recognition*. IEEE Computer Society, 2009.

[Ben00] Yoshua Bengio. "Gradient-Based Optimization of Hyperparameters." *Neural Comput.*, **12**(8):1889–1900, 2000.

[Bis06] C.M. Bishop. *Pattern Recognition and machinve Learning*. Springer-Verlag, 2006.

[BR96] Michael J Black and A Rangarajan. "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision." *International Journal of Computer Vision*, **19**(1):57–92, July 1996.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. "Fast Approximate Energy Minimization via Graph Cuts." *IEEE Transactions of Pattern Analysis and Machine Intelligence*, **23**(11):1222–1239, 2001.

[CBS05] Timothee Cour, Florence Benezit, and Jianbo Shi. "Spectral Segmentation with Multiscale Graph Decomposition." In *IEEE Computer Vision and Pattern Recognition*, Washington, DC, USA, 2005. IEEE Computer Society.

[CMM02] Dorin Comaniciu, Peter Meer, and Senior Member. "Mean shift: A robust approach toward feature space analysis." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**:603–619, 2002.

[Col02] Michael Collins. "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms." In *EMNLP 2002*, 2002.

[CVB02] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. "Choosing Multiple Parameters for Support Vector Machines." *Mach. Learn.*, **46**(1-3):131–159, 2002.

[DFN07] Chuong Do, Chuan-Sheng Foo, and Andrew Ng. "Efficient multiple hyperparameter learning for log-linear models." In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pp. 377–384. MIT Press, Cambridge, MA, 2007.

[Fig05] M A T Figueiredo. “Bayesian Image Segmentation Using Gaussian Field Priors.” In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 74–89, 2005.

[GG84] S Geman and D Geman. “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.” *IEEE Transactions of Pattern Analysis and Machine Intelligence*, (6):721–741, 1984.

[HC71] JM Hammersley and P Clifford. “Markov fields on finite graphs and lattices.” 1971.

[Hin02] Geoffrey Hinton. “Training products of experts by minimizing contrastive divergence.” *Neural Computation*, **14**(7):1771–1800, 2002.

[HZC04] X.M. He, R.S. Zemel, and M. Á. Carreira-Perpiñián. “Multiscale conditional random fields for image labelling.” In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[KAH05] S. Kumar, J August, and M Hebert. “Exploiting Inference for Approximate Parameter Learning in Discriminative Fields: An Empirical Study.” In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2005.

[Kep04] Jeremy Kepner. “MatlabMPI.” *J. Parallel Distrib. Comput.*, **64**(8):997–1005, 2004.

[KSC07] S. Sathiya Keerthi, Vikas Sindhwani, and Olivier Chapelle. “An Efficient Method for Gradient-Based Adaptation of Hyperparameters in SVM Models.” In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pp. 673–680. MIT Press, Cambridge, MA, 2007.

[KTZ05] M. P. Kumar, P. H. S. Torr, and A. Zisserman. “OBJ CUT.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005.

[LFD07] Anat Levin, Rob Fergus, Frédo Durand, and William T. Freeman. “Image and depth from a conventional camera with a coded aperture.” In *ACM SIGGRAPH 2007*, p. 70, New York, NY, USA, 2007. ACM.

[LH05] Yann LeCun and Fu Jie Huang. “Loss Functions for Discriminative Training of Energy-Based Models.” In *Proc. of the 10-th International Workshop on Artificial Intelligence and Statistics (AIStats’05)*, 2005.

[LH08] Yunpeng Li and Daniel P. Huttenlocher. “Learning for Optical Flow Using Stochastic Optimization.” In *ECCV (2)*, pp. 379–391, 2008.

[Li01] Stan Z. Li. *Markov random field modeling in image analysis*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[LRB07] Victor S. Lempitsky, Carsten Rother, and Andrew Blake. “LogCut - Efficient Graph Cut Optimization for Markov Random Fields.” In *International Conference on Computer Vision, ICCV*, 2007.

[LRR09] Victor Lempitsky, Carsten Rother, Stefan Roth, and Andrew Blake. “Fusion Moves for Markov Random Field Optimization.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **99**, 2009.

[LW06] A Levin and Y Weiss. “Learning to Combine Bottom-Up and Top-Down Segmentation.” In *European Conference on Computer Vision (ECCV)*, Graz, Austria, May 2006.

[Mar82] David Marr. *Vision*. W. H. Freeman, 1982.

[Mei05] Marina Meilă. “Comparing clusterings: an axiomatic view.” In *ICML ’05: Proceedings of the 22nd international conference on Machine learning*, New York, NY, USA, 2005. ACM.

[MFM04] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. “Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(5):530–549, 2004.

[MFT01] D. Martin, C. Fowlkes, D. Tal, and J. Malik. “A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics.” In *Proc. 8th Int’l Conf. Computer Vision*, volume 2, pp. 416–423, July 2001.

[Mou74] John Moussouris. “Gibbs and Markov random systems with constraints.” **10**:11–33, 1974.

[PSW03] J. Portilla, V. Strela, M. Wainwright, and E.P. Simoncelli. “Image Denoising Using Scale Mixtures of Gaussians in the Wavelet Domain.” *IEEE Transactions on Image Processing*, **12**(11):1338–1351, November 2003.

[PY97] J. Pitman and M. Yor. *Annals of Probability*, **25**(2):855–900, 1997.

[Ran71] W.M. Rand. “Objective Criteria for the Evaluation of Clustering Methods.” *Journal of the American Statistical Association*, **66**(336):846–850, 1971.

[RB05] Stefan Roth and Michael Black. “Field of Experts: A Framework for Learning Image Priors.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pp. 860–867, 2005.

[RM03] Xiaofeng Ren and Jitendra Malik. “Learning a classification model for segmentation.” In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pp. 10–17, 2003.

[SJ09] Erik Sudderth and Michael Jordan. "Shared Segmentation of Natural Scenes Using Dependent Pitman-Yor Processes." In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pp. 1585–1592. 2009.

[SM00] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

[SMR07] Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. "Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data." *J. Mach. Learn. Res.*, **8**:693–723, 2007.

[SP07] D. Scharstein and C. Pal. "Learning Conditional Random Fields for Stereo." *Computer Vision and Pattern Recognition, 2007.*, pp. 1–8, June 2007.

[Spa92] J.C. Spall. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation." *Automatic Control, IEEE Transactions on*, **37**(3):332 –341, mar 1992.

[Spa95] J. C. Spall. "Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization." *American Statistician*, 1995. Submitted.

[ST09] Kegan G. G. Samuel and Marshall F. Tappen. "Learning Optimized MAP Estimates in Continuously-Valued MRF Models." In *IEEE Computer Vision and Pattern Recognition*, 2009.

[Tap07] M. F. Tappen. "Utilizing Variational Optimization to Learn Markov Random Fields." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR07)*, 2007.

[TCK05] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. "Learning Structured Prediction Models: A Large Margin Approach." In *The Twenty Second International Conference on Machine Learning (ICML-2005)*, 2005.

[TLA07] Marshall F. Tappen, Ce Liu, Edward H. Adelson, and Willian T. Freeman. "Learning Gaussian Conditional Random Fields for Low-Level Vision." In *IEEE Conference on Computer Vision & Pattern Recognition*, 2007.

[TLJ06a] Ben Taskar, Simon Lacoste-Julien, and Michael Jordan. "Structured Prediction via the Extragradient Method." In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

[TLJ06b] Ben Taskar, Simon Lacoste-Julien, and Michael Jordan. "Structured Prediction via the Extragradient Method." In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pp. 1345–1352. MIT Press, Cambridge, MA, 2006.

[TO03] Antonio Torralba and Aude Oliva. “Statistics of natural image categories.” In *Network: Computation in Neural Systems*, pp. 391–412, 2003.

[TPC] W. Trobin, T. Pock, D. Cremers, and H. Bischof. “Continuous Energy Minimization Via Repeated Binary Fusion.”.

[TSD08] Marshall F. Tappen, Kegan G. G. Samuel, Craig V. Dean, and David M. Lyle. “The Logistic Random Field - A convenient graphical model for learning parameters for MRF-based labeling.” In *IEEE Computer Vision and Pattern Recognition*, 2008.

[VT07] Jakob J. Verbeek and Bill Triggs. “Region Classification with Markov Field Aspect Models.” In *IEEE Computer Vision and Pattern Recognition*, 2007.

[WA02] C. K. I. Williams and F. V. Agakov. “An Analysis of Contrastive Divergence Learning in Gaussian Boltzmann Machines.” Informatics Research Report, May 2002.

[Wai06] Martin J Wainwright. “Estimating the “wrong” graphical model: Benefits in the computation-limited setting.” *Journal of Machine Learning Research*, **7**:1829–1859, September 2006.

[WBS04] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, Student Member, Eero P. Simoncelli, and Senior Member. “Image Quality Assessment: From Error Visibility to Structural Similarity.” *IEEE Transactions on Image Processing*, **13**:600–612, 2004.

[Wei99] Y. Weiss. “Segmentation using eigenvectors: a unifying view.” In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pp. 975–982 vol.2, 1999.

[WF07] Yair Weiss and William T Freeman. “What Makes a Good Model of Natural Images.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[WJW05] M J Wainwright, T S Jaakkola, and A S Willsky. “MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches.” *IEEE Transactions on Information Theory*, **51**(11):3697–3717, November 2005.

[WL93] Z. Wu and R. Leahy. “An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(11):1101–1113, 1993.