Bayesian Optimization for Image Segmentation, Texture Flow Estimation and Image Deblurring

Yu-Wing, Tai

A THESIS SUBMITTED

FOR THE DEGREE OF PHILOSOPHY

DEPARTMENT OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2008

Abstract

This thesis addresses three important problems within computer vision: image segmentation, texture flow estimation, and image/video deblurring. While these three topics differ significantly in the underlying parametric models used to formulate the problems, the uniting theme throughout this thesis is the use of a Bayesian optimization framework to solve each specific problem. In particular, we show how each of these problems can be formulated into one of a maximum a posterior (MAP) estimation, where the likelihood and prior probabilities are uniquely defined for each problem. To solve these non-convex optimizations, an alternating optimization algorithm that iteratively solves for model parameters is used. Our experimental results show that this Bayesian approach provides excellent performance that is either on par or superior to the current state-of-the-art for each topics' respective area.

This thesis is organized to begin with an overview on Bayesian formulation of parameter estimation, followed by self-contained chapters for the problems of image segmentation, texture flow estimation, and image/video deblurring. A summary chapter is included to categorically summarize our contributions and discuss future work.

Acknowledgments

I would like to thank my supervisor Dr. Michael S. Brown for his guidance and support during the years, for his insightful discussions on several topics and projects, for his helpfulness, for his encouragement, and for his instructions on both the technical and non-technical aspects of my Ph.D. training. I would like to thank my previous supervisor Dr. Chi-Keung Tang in the Hong Kong University of Science and Technology for his training during my undergraduate and master study in HKUST. I would also like to thank my mentor Dr. Steve Lin in the Microsoft Research Asia for his brilliant insights and suggestions on the research project we have cooperated on.

Thanks also to the members of research group in NUS, MSRA, NTU and HKUST where I have worked. I benefited greatly through the interactions my colleagues who are all very good people to work with. I am sure that the friendships I have formed there will continue on throughout my career.

Finally, I would also like to express my deepest gratitude to my family, my parents and my two younger sister for their continuous supporting and unfailing love. Special thank to my friends in Hong Kong who helped me in various stages of my live.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

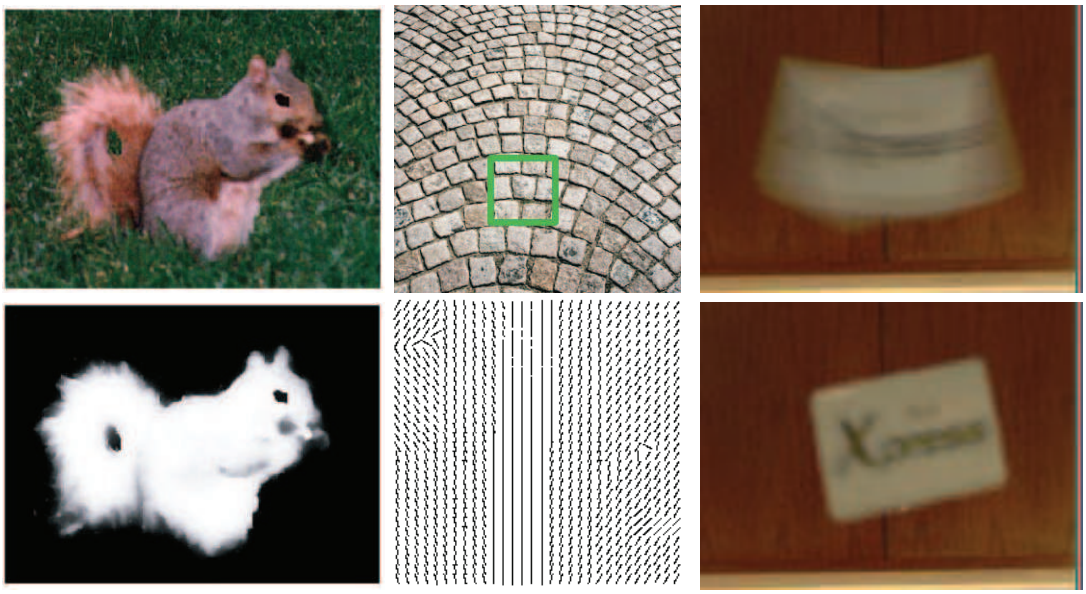


Figure 1.1: Examples of the three problems we are going to present in this thesis. The first row shows our inputs and the second row shows our outputs.

This thesis is organized as three self-contained chapters addressing three distinct problems of soft color image segmentation (chapter 2), texture flow estimation (chapter 3), and image and video deblurring (chapter 4). Figure 1.1 shows examples of the input and output of each

of these problems. The central theme uniting these three problems is of the use of a Bayesian optimization framework to formulate a solution. In this introduction, a review of the Bayesian method is provided in section 1.2, along with a rationale for its use in computer vision problems. Common techniques including linear regression, alternating optimization (expectation maximization) and belief propagation for solving such optimization problems are presented in section 1.3. This chapter also gives an introduction to the three problems addressed in this thesis in section 1.4 and the overall structure of this thesis in 1.5. Chapter 5 concludes this thesis with an discussion on the work presented and a summary of contributions.

1.2 Bayesian Method

The Bayesian method has been widely used by various research disciplines and is not limited to computer vision and image processing problems. In this section, we give an overview of how the Bayesian approach can be used to estimate model parameters. This is followed by a discussion on why it is often a popular choice for addressing computer vision problems.

1.2.1 Bayes Rule and the Bayesian Model

Bayes rule was developed by the Reverend Thomas Bayes in 18th century and is stated as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad \text{(Eq. 1.1)}$$

where $P(A)$ and $P(B)$ are the prior probability of A and B respectively, $P(A|B)$ and $P(B|A)$ are the conditional probability of A given B and probability of B given A respectively. In Bayes' theorem, $P(A|B)$ is called the posterior probability, $P(B|A)$ is called the likelihood probability, $P(A)$ is called the prior probability and $P(B)$ is known as the normalization

constant. Typically, A is denoted as the *Hypothesis Model*, and B is denoted as the *Observation*. For general cases that have multiple variables, such that $A = \{A_1, \ldots, A_n\}$ and $B = \{B_1, \ldots, B_m\}$, equation (Eq. 1.1) can be generalized as follows:

$$P(A_1, \ldots, A_n | B_1, \ldots, B_m) = \frac{P(B_1, \ldots, B_m | A_1, \ldots, A_n) P(A_1, \ldots, A_n)}{P(B_1, \ldots, B_m)} \qquad \text{(Eq. 1.2)}$$

To provide a better understanding on how the probabilities are defined, consider the following typical example given in [79]:

Suppose there is a test for cancer. Let A be the event that the person tested has cancer and let B be the event that the test result is positive, i.e. the test says the person has cancer. We are interested in knowing whether the person has cancer or not given the positive testing results. That is we want to find the posterior probability, $P(A = \text{has cancer} | B = \text{positive})$. For simplicity of representation, let us denote $A =$ has cancer, $\bar{A} =$ does not have cancer; $B =$positive, $\bar{B} =$ negative. Suppose the test is 95% accurate, i.e. if the person has cancer, the probability that the test will give a positive result is $P(B|A) = 0.95$. Similar definition is given to $P(\bar{B}|\bar{A}) = 0.95$. If we further know that the probability of a person having cancer is $P(A) = 0.005$. Then, according to Bayes rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\bar{A})P(\bar{A})} = 0.087$$

This means given a positive test result, there is only an 0.087 probability the person has cancer.

In this example, we can see how the Hypothesis Model A, Observation B and the probabilities are defined. In many real world situations, $P(A|B)$ is difficult to define or cannot be defined directly, while $P(B|A)$ and $P(A)$ are easier to define. In these situations, Bayes rule can be used to define the probability model of $P(A|B)$. These kind of models are called Bayesian models. In later sections, we will discuss how the likelihood probability $P(B|A)$ and the prior probability $P(A)$ are defined in typical computer vision problems. First, we describe some important properties of the Bayesian model.

There are several advantages of using the Bayesian model [50]. First, the Bayesian model allows us to learn and/or model causal relationships between hypothesis model parameters and observations. This is useful when we are trying to gain understanding about a problem domain. In addition, the Bayesian model encodes the strength of causal relationships with probabilities. To better understand the causal relationship in Bayesian model, let us consider the previous example. If we want to find the probability that a person having cancer would have a positive test results, i.e. $P(A, B) = P(B|A)P(A)$. This causal relationship can be expressed by the following figure:

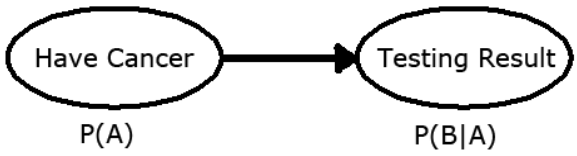


Figure 1.2: Bayesian network of causal relationship is a directed acyclic graph encoding conditional dependencies among variable.

The causal relationship derived directly from Bayes rule is called a Bayesian network which describes conditional dependence among variables. In most situations, prior probability variables are defined before likelihood probability variables unless the variables are independent/conditionally independent. By exploiting the causal relationships among variables, we can simplify the representation of probability. For example, if the hypothesis model variables $A = \{A_1, \ldots, A_n\}$ are all independent and the observation variables $B = \{B_1, \ldots, B_m\}$ are all conditionally independent. We can simplify equation (Eq. 1.3) into:

$$\frac{P(B_1, \ldots, B_m|A_1, \ldots, A_n)P(A_1, \ldots, A_n)}{P(B_1, \ldots, B_m)} = \frac{\prod_{i=1}^{m} P(B_i|A_1, \ldots, A_n) \prod_{j=1}^{n} P(A_j)}{P(B_1, \ldots, B_m)} \quad \text{(Eq. 1.3)}$$

Second, the Bayesian model is easy and flexible in combining any prior knowledge about the problem domain and data into the prior probability $P(A)$. Let us again using the pre-

vious example. Assume now that we know that the person smokes and this habit would increase probability of having cancer. Let A_2 be the prior knowledge that the person smokes, the probability that the person has cancer with positive testing result would be $P(A, A_2|B) = \frac{P(B|A,A_2)P(A|A_2)P(A_2)}{P(B)} = \frac{P(B|A)P(A|A_2)}{P(B)}$ which asserts that A_2 is conditionally independent of B and $P(A_2) = 1$ by exploring causal relationship between parameters. Notice that the definition of likelihood $P(B|A)$ in this example is still the same. Given the current Bayesian model, if we have new priors or new observations about the problem, we can incorporate these new priors and new observations into the current Bayesian model. This allows minimum changes of probability defined in current model.

Third, the Bayesian model can handle incomplete data observations or noisy data observations through incorporating proper priors into the model. For example, consider a classification or regression problem where two of the input variables are strongly anti-correlated. This correlation is not a problem for standard supervised learning techniques, provided all inputs are measured in every case. When one of the inputs is not observed, however, most models will produce an inaccurate prediction, because they do not encode the correlation between the input variables. The Bayesian model offers a natural way to encode such dependencies into the prior probabilities $P(A)$. In later section, we will see how priors are effective in dealing with noisy observations especially in image domains where priors can be modeled as a pairwise energy term in a Markov network.

Fourth, the Bayesian model offers an efficient and principled approach for avoiding the over-fitting of data. From a statistical point of view, the Bayesian method is a probabilistic method that find the most likely decision boundaries or model parameters. It has been shown that the solution from the Bayesian method is a global optimal if the Bayesian probabilistic model is a convex function. For more details on the optimality of Bayesian method, see [50].

In many situations, we are not interested in calculating the posterior or joint probability of probabilistic model. Instead, we are interested in finding a hypothesis model (or hypothesis

model parameters) that can maximize the posterior or joint probability given the current observations. The problem of finding a hypothesis model that maximizes posterior probability is called the Maximum A Posteriori (MAP) problem:

$$\begin{aligned} \arg\max_A P(A|B) &= \arg\max_A \frac{P(B|A)P(A)}{P(B)} \\ &= \arg\max_A P(B|A)P(A) \end{aligned} \quad \text{(Eq. 1.4)}$$

Note that the $P(B)$ is omitted since it has no effect on the estimation of A. It is also easy to see that maximizing posterior probability is equal to maximizing joint probability. In situations where we do not have any prior knowledge, i.e. $P(A)$ is uniformly distributed over the whole observation domain, the solution of MAP problem is equal to the solution of the Maximum Likelihood (ML) problem since $P(A)$ is also omitted.

1.2.2 Likelihood Probability

In this section, we describe a common method to define likelihood probability. The likelihood probability defines how the observations are generated from the hypothesis model. Typically, we assume the process of generating observations from the hypothesis model are identical and independent which means that the generation of B_i does not depend on the result of any of other B_j. In order words, given multiple observations, $B = \{B_1, \ldots, B_m\}$, we assume the observations are all conditionally independent. This assumption is valid in many real world situation.

For a better understand on how likelihood probability is defined, we use image denoising as an example. Suppose we have a noisy image, I_N, and we want to estimate a clear image I assuming that each pixel is potentially corrupted by noise that is identical and independent. We first need to choose a distribution to model how noise is generated. A common approach is to use Gaussian distribution with mean μ and standard derivation, σ. Now, we can define the likelihood probability of the image denoising problem as:

$$
\begin{aligned}
P(I_N|I,\mu,\sigma) &= \prod_{(x,y)} P(I_N(x,y)|I(x,y),\mu,\sigma) \\
&= \prod_{(x,y)} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{||(I_N(x,y)-I(x,y))-\mu||^2}{\sigma^2}) \qquad \text{(Eq. 1.5)}
\end{aligned}
$$

where (x, y) denotes image coordinate and $I_N(x, y) - I(x, y)$ denotes the noise magnitude at an (x, y) position. Typically, μ is chosen to be zero mean and it is omitted in many technical papers.

In some problems, the Gaussian distribution is not the best probabilistic model. Other common distribution used in computer vision included Laplacian distribution, Poisson distribution, Chi-square distribution, and so on. To make the definition more general, mixture of distributions can also be used. The model can be represented in either parametric form or non-parametric form. Though the models are often complicated, the definition of likelihood probability which measures the distance between observations to the selected model is somewhat similar. The drawback of using a complicated model is that it would make the solution space significantly more complicated and the estimation results are more likely to result in a local maxima/minima or to be over-fitted. While using a simple model is computationally efficient and can achieve a global maxima/minima, a simple model may not be the best fit to the observation distribution. This is a tradeoff between using complicated models versus simple models. Choosing the appropriate model for a problem is the responsibility of the algorithm designer.

1.2.3 Prior Probability

In the previous section, we described how the likelihood probability is defined. Now, let us look at the image denoising example again and discuss why we need the prior probability. In the image denoising problem, our goal is to obtain a clear image I, and not the calculation

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

Figure 1.3: Image denoising example from [38]. (a) Input noisy image. Denoised image using (b) 5×5 Gaussian filter, (c) 5×5 median filter, (d) MRF with neighborhood smoothness prior, and (e) MRF with neighborhood smoothness prior and discontinuity prior. (f) Ground truth image.

the posterior probability. As we have discussed in the section 1.2, this problem is a MAP problem. Without prior probability, the MAP problem is equal to ML problem and a trivial solution to the image denoising problem is $I = I_N$. However, this is not the solution we want to obtain. A common prior for image denoising problem is the neighborhood smoothness prior which says that for a pixel at certain point (x, y) of image, its value $I(x, y)$ should not be very different from pixel values $I(x', y')$ within a local neighbor region, $(x', y') \in \mathcal{N}(x, y)$. A simple definition of neighbor region is to use first order neighborhood, i.e. the 4 or 8 adjacent pixels. Now, we can derived the prior probability of the image denoising problem:

$$\begin{aligned} P(I) &= \prod_{(x,y)} \prod_{(x',y') \in \mathcal{N}(x,y)} P_S(I(x,y), I(x',y')) \\ &= \prod_{(x,y)} \prod_{(x',y') \in \mathcal{N}(x,y)} \frac{1}{\sigma' \sqrt{2\pi}} \exp\left(-\frac{||(I(x,y) - I(x',y')) - \mu'||^2}{\sigma'^2}\right) \quad \text{(Eq. 1.6)} \end{aligned}$$

where μ' and σ' are mean and standard derviation of Gaussian distribution. In this example, we use the Gaussian distribution again to model prior probability distribution. Similar to the definition of likelihood probability, we can use any other probability distribution to model prior probability distribution. Notice that although the definition of likelihood probability and prior

probability in this example are very similar (they both use Gaussian distribution), their physical meaning and effects are not the same. A simple way to distinguish likelihood probability and prior probability in an energy function is to identify whether or not the energy term has an observation measurement included. If there is no observation measurement in an energy term, this energy term is prior probability, otherwise, it is likelihood probability.

With the definition of prior probability in the image denoising problem, $I = I_N$ is no longer the optimal solution. The likelihood probability and prior probability defined above formed a Markov network in which the likelihood probability corresponds to the data term and the prior probability corresponds to the pairwise energy term of a Markov network. Since image intensity is discrete, the problem of image denoising can then be transformed into a discrete labeling problem of a Markov Random Field (MRF) for which a local optimal solution can be achieved by using techniques such as graph-cut [58] or belief propagation [106].

The neighborhood smoothness prior used in the image denoising example produce results that are over smoothed which some edge features of an image are also smoothed out. This is shown in figure 1.3(d). To preserve sharp edge features, we can include another prior, edge preserving discontinuity prior, which says that the smoothness prior should not be applied across edges:

$$P_D(I(x,y), I(x',y')) = \begin{cases} 1 & , |I(x,y) - I(x',y')| > t \\ 0 & , |I(x,y) - I(x',y')| \leq t \end{cases} \qquad \text{(Eq. 1.7)}$$

where t is a threshold to define discontinuities. Figure 1.3(e) shows the results of adding in this discontinuity prior. Although a more complicated model can be used for this edge preserving discontinuity prior, we choose the simplest model for better illustration. The edge preserving discontinuity prior defined here is independent of the likelihood probability and the neighborhood smoothness prior defined above given the values of I. To combine $P_D(I(x,y), I(x',y'))$ into the current Bayesian model, we can simply multiply it to the current model. This example

demonstrates the flexibility of the Bayesian model to include new prior into current model. The neighborhood smoothness prior and the edge preserving discontinuity prior are also commonly used in other computer vision problems such as the stereo matching problem and optical flow estimation.

Having examined how the prior probabilities are defined for image denoising, one may notice that the prior probabilities are problem specific. Indeed, the performance of the Bayesian model strongly depends on how the prior probabilities are defined. The prior probabilities encode our prior knowledge or our desirable behaviors of the estimated solution. In classification problems, we may want the decision boundary to be smoothed to avoid over fitting of the data, this desired behavior can be encoded in prior probability as a regularization term during the training process. In image segmentation problems, we may want regions of the same segment to exhibit some global similarity which can also be encoded in prior probability.

1.3 Common techniques for solving Bayesian model

To solve the parameters in a Bayesian model, there are several standard techniques. For example, linear regression is often used when model parameters can be written in a linear form. Alternating optimization (AO) [1] is used when the model parameters are interdependent. Graph-cut or belief propagation can be used when the Bayesian model can be transformed into a discrete labeling problem of Markov Random Field. Some other common techniques included Markov Chain Monte Carlo (MCMC), Expectation Propagation, Kernel methods, and so on. In this section, we describe three techniques that are most commonly used in computer vision area: the Linear Regression, the Alternating Optimization (Expectation Maximization Algorithm), and the Belief Propagation.

[1] Expectation Maximization is a special case of AO which convergence of EM has been proved while AO does not guarantee to converge.

1.3.1 Linear Regression

Linear regression is often used when the model parameters that we want to estimate can be written in a linear equation which is a straight line in high dimensional feature space. The results of data fitting are subjected to statistical analysis. Suppose that there is a set of observations $B = \{B_1, \ldots, B_m\}$ which each observation has N features denoted as $B_i(j)$, $1 \leq i \leq m$ is observations index, $1 \leq j \leq N$ is feature index. We assume these observation are generated by a linear model with $N + 1$ parameters:

$$A_0 + A_1 B(1) + \ldots + A_N B(N) = 0 \qquad \text{(Eq. 1.8)}$$

we further assume the observations follow the identical and independent distribution (iid) and the estimation errors following Gaussian distribution with zero mean and standard derivation σ. Also, we assume that the model parameters $A = \{A_0, \ldots, A_N\}$ are all independent. Then, we can define our Bayesian-MAP objective function as follow:

$$\begin{aligned}
\arg\max_A P(A|B) &= \arg\max_A P(B|A)P(A) \\
&= \arg\max_A \prod_{i=0}^{m} P(B_i|A) \\
&= \arg\min_A -\log(\prod_{i=0}^{m} P(B_i|A)) \\
&= \arg\min_A -\sum_{i=0}^{m} \log(P(B_i|A)) \\
&= \arg\min_A -\sum_{i=0}^{m} \log(\frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{||A_0 + A_1 B_i(1) + \ldots + A_N B_i(N)||^2}{2\sigma^2})) \\
&= \arg\min_A \sum_{i=0}^{m} ||A_0 + A_1 B_i(1) + \ldots + A_N B_i(N)||^2 \qquad \text{(Eq. 1.9)}
\end{aligned}$$

In this example, we make no assumptions on $P(A)$ and therefore it is removed and the MAP problem is reduced to a ML problem. We have also removed the normalization constant $\frac{\log(\sigma\sqrt{2\pi})}{2\sigma^2}$ since it is independent of the model parameters A or observations B[2]. A mono-

[2]If each observation B_i has its own σ_i, $\frac{\log(\sigma_i\sqrt{2\pi})}{2\sigma_i^2}$ would become the weight of observations and it cannot be omitted.

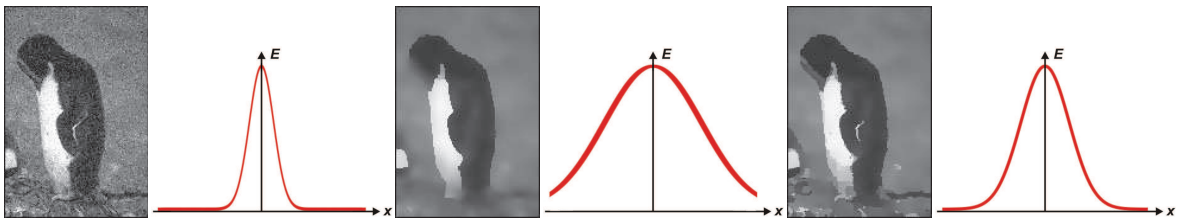
tonic function $\log(\cdot)$ is used to convert multiplications into summations to increase numerical stability and a negative sign is added to convert the $\arg\max$ problem into an $\arg\min$ problem. After these mathematical manipulations, our objective function becomes a standard linear least-squares regression problem. Re-writing equation (Eq. 1.9) into matrix form, we get:

$$\begin{aligned} & \arg\min_{A} \begin{bmatrix} A_0 & \cdots & A_N \end{bmatrix} \begin{bmatrix} 1 & \cdots & B_1(N) \\ \vdots & \ddots & \vdots \\ 1 & \cdots & B_m(N) \end{bmatrix}^T \begin{bmatrix} 1 & \cdots & B_1(N) \\ \vdots & \ddots & \vdots \\ 1 & \cdots & B_m(N) \end{bmatrix} \begin{bmatrix} A_0 \\ \vdots \\ A_N \end{bmatrix} \\ = & \arg\min_{A} \mathcal{A}^T \mathcal{B}^T \mathcal{B} \mathcal{A} \end{aligned} \quad \text{(Eq. 1.10)}$$

which is a global optimal solution of $\mathcal{A}$ corresponding to the minimum eigenvector of $\mathcal{B}^T\mathcal{B}$. This can be reliably solved by using standard numerical routines such as LU decomposition or singular value decomposition (SVD). The solution found by linear regression is guaranteed to be a global maximum/minimum since the linear equation is a convex function. In practise, we can normalize the values of observation $B_i(j)$ between 0 and 1 to further increase numerical stability.

Linear regression is a very reliable approach but it can fail to produce a good result for several reasons:

- The underlying model is not a linear model which means that our basic assumptions about the problem is incorrect and we need to re-formulate the problem.
- The estimation errors do not follow Gaussian distribution. For example, if there are outliers presented in the observations, the estimated model parameters will be biased by these outliers. One simple way to alleviate this problem is to assign different weight to the observations and then performed weighted least-square fitting. Another method is to perform outlier removal before linear regression.
- The Euclidean distance in input space is not the best distance measurements for the observations. One method to solve this problem is to transform the input space into some



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

Figure 1.4: Effect of parameters in MRF. In this example, we analysis the effect of σ in equation (Eq. 1.5). Denoised image with (a) small σ, (c) large σ and (e) optimal σ. (b)(d) and (f) shows the respective likelihood distribution of different σ. With small σ, the likelihood distribution is concentrated on the original noisy value, therefore its denoised result I is closer to I_N. On the other hand, a large σ allows larger difference between I and I_N; However, the denoised result is over smoothed. In this example, there exists an optimal σ for which the energy function is minimum. The optimal σ can be found by using alternating optimization.

other high dimensional space such that the parameters become linear in the transformed space. This method is especially useful for classification problems using the support vector machine (SVM).

1.3.2 Alternating Optimization (Expectation Maximization Algorithm)

Alternating Optimization is often used when model parameters are interdependent. It is typically used in many computer vision applications (e.g. [23, 18, 27, 39, 108, 120, 134]). Unlike linear regression, Alternating Optimization or Expectation Maximization only guaranteed to converge to local minima/maxima, while solutions found by linear regression are always global minima/maxima.

In alternating optimization, the model parameters will be divided into disjoint subsets of parameters, in which each subset of parameters represents different physical meaning about the model. Take the image denoising problem as an example. The effect of the parameter σ in this problem is demonstrated in figure 1.4. As we can see, different values of σ can produce different results and there exists an optimal σ that allows us to produce the best result. In this situation, we can consider $\{\mu, \sigma\}$ of the Gaussian distribution used to model noise distribution

of an image to be the parameters we want to estimate. Now, the total number of parameters we want to estimate are $\{I, \mu, \sigma\}$. We can divide the parameters $\{I, \mu, \sigma\}$ into two disjoint subset: $\{I\}$ and $\{\mu, \sigma\}$, where each subset represents a different physical meaning about the image denoising problem and they are interdependent. Alternating Optimization is an iterative gradient descent optimization process[3]. For each subset of parameters, the parameters inside the subset would have its own iterative update rules with the parameters in the other subset fixed during the processing of one update rule. The update rules will be operated alternatively and iteratively, thus, the name of alternating optimization. In the image denoising example, using the same objective function defined in (Eq. 1.5),(Eq. 1.6) and (Eq. 1.7), the update rule for I is given by solving the MRF equation defined above with fixed $\{\mu, \sigma\}$ and the update rule for $\{\mu, \sigma\}$ is defined by the likelihood terms with fixed I:

$$\begin{aligned} \mu &= \frac{1}{M}\sum_{(x,y)}(I(x,y) - I_N(x,y)) \\ \sigma &= \sqrt{\frac{1}{M}\sum_{(x,y)}||(I(x,y) - I_N(x,y)) - \mu||^2} \end{aligned} \qquad \text{(Eq. 1.11)}$$

where M is total number of pixels in image. We ignore the prior terms here, since they has no influence on the estimation of $\{\mu, \sigma\}$. An example of using alternating optimization for finding the model parameters in a MRF stereo problem is presented in [134, 135]. Alternating Optimization is guaranteed to converge if each of the update rule is monotonic decreasing/increasing about the global objective function. For more details about the convergence of alternating optimization, see [11].

One well-known example of Alternating Optimization is the Expectation-Maximization (EM) algorithm. The EM algorithm is a maximum-likelihood parameter estimation algorithm which is useful in handling incomplete data or data that has missing values. Unlike alternating optimization, the convergence of the EM algorithm to a local maxima/minima is guaranteed.

[3]Note that for AO their can be more than two disjoint subsets. In chapter 2, we will formulate our problem with three subsets of parameters.

Let us assume that the data $\mathcal{X}$ is observed and is generated by an unknown distribution. We call $\mathcal{X}$ the incomplete data observations. We assume that a complete data set exists $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$, a complete-data likelihood probability is defined as:

$$P(\mathcal{Z}|\Theta) = P(\mathcal{X}, \mathcal{Y}|\Theta) = P(\mathcal{Y}|\mathcal{X}, \Theta)p(\mathcal{X}|\Theta) = \prod_{i=1}^{N} P(y_i|x_i, \Theta)P(x_i|\Theta) \qquad \text{(Eq. 1.12)}$$

where Θ is the set of model parameters, N is total number of observed data. The likelihood $p(\mathcal{X}|\Theta)$ is referred to as the incomplete-data likelihood function. The parameters we want to estimate are $\{P(\mathcal{Y}|\mathcal{X}, \Theta), \Theta\}$, where we can think of $P(\mathcal{Y}|\mathcal{X}, \Theta)$ as a probability density function with $\mathcal{X}$ and Θ are constants and $\mathcal{Y}$ is a random variable. Again, the parameters can be divided into two disjoint subsets $\{P(\mathcal{Y}|\mathcal{X}, \Theta)\}$ and $\{\Theta\}$. Hence, the EM algorithm is a two step alternating optimization algorithm. The two steps in the EM algorithm are usually referred as the Expectation step (E-step) and the Maximization step (M-step).

In the E-step, the EM algorithm finds the expected value of the complete-data log-likelihood $\log P(\mathcal{X}, \mathcal{Y}|\Theta)$ with respect to the unknown data Y given the observed data X and the current parameter estimated Θ^{t-1}:

$$Q(\Theta, \Theta^{t-1}) = E[\log P(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta^{t-1}] = \int_y P(y|\mathcal{X}, \Theta^{t-1}) \log P(\mathcal{X}, y|\Theta)dy \quad \text{(Eq. 1.13)}$$

where Θ is the new set of parameters that we optimize to increase Q, $P(y|\mathcal{X}, \Theta^{t-1})$ is the marginal distribution of the unobserved data and is dependent on both the observed data $\mathcal{X}$ and the current parameters Θ^{t-1}. One important thing to notice is that $\mathcal{X}$ and Θ^{t-1} are constants. Using Bayes rule, we get:

$$P(y|x, \Theta^{t-1}) = \frac{P(y, x|\Theta^{t-1})}{P(x|\Theta^{t-1})} = \frac{P(x|y, \Theta^{t-1})P(y|\Theta^{t-1})}{\int_x P(x|y, \Theta^{t-1})P(y|\Theta^{t-1})dx} \qquad \text{(Eq. 1.14)}$$

In the M-step, the EM algorithm finds the parameters Θ that maximize the expectation we computed in the first step. That is, we find:

$$\Theta^t = \arg\max_{\Theta} Q(\Theta, \Theta^{t-1}) \qquad \text{(Eq. 1.15)}$$

The E-step and the M-step are iterated alternatively. Each iteration is guaranteed to increase the log likelihood and hence the algorithm is guaranteed to converge to a local maximum of the likelihood function.

To let us gain a better understanding of the EM algorithm, we use the Gaussian mixture model (GMM) parameter estimation problem as an example. We assume the data x_i are generated from one of the Gaussian distributions in the GMM. The incomplete-data likelihood for x_i is then defined as follow:

$$P(x_i|\Theta) = \sum_{j=1}^{M} \alpha_j P(x_i|\mu_j, \Sigma_j) = \sum_{j=0}^{M} \frac{\alpha_j}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp(-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j)) \tag{Eq. 1.16}$$

where M is number of Gaussian distribution in the mixture, α_j are mixing weight of Gaussian such that $\sum_{j=1}^{M} \alpha_j = 1$, $\{\mu_j, \Sigma_j\}$ are the mean and covariance matrix of the j-th Gaussian in the GMM and d is observed data dimension.

In this example, we consider $\mathcal{Y} = \{y_i = j, 1 \leq i \leq N, 1 \leq j \leq M\}$, which is a random variable indicating which Gaussian distribution "generated" the observed data x_i. Given $\{\mu_j, \Sigma_j\}$, we can compute $P(x_i|y_i = j, \mu_j, \Sigma_j)$ for each i and j. In addition, the mixing weight, α_j can be considered as prior probabilities of each mixture component, that is $P(y_i = j|\mu_j, \Sigma_j) = \alpha_j$. According to the definition in equation (Eq. 1.14), the E-step update rule is:

$$\begin{aligned} P(y_i = j|x_i, \mu_j, \Sigma_j) &= \frac{\alpha_j P(x_i|y_i = j, \mu_j, \Sigma_j)}{\sum_{k=1}^{M} P(x_i|y_i = k, \mu_k, \Sigma_k)} \\ &= \frac{\frac{\alpha_j}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp(-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j))}{\sum_{k=1}^{M} \frac{\alpha_k}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp(-\frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1}(x_i - \mu_k))} \end{aligned} \tag{Eq. 1.17}$$

To compute the update rule used in the M-step, we can perform partial derivatives on the objective function Q with respect to α_j, μ_j and Σ_j and then set the partial derivative to zero.

After some mathematical arrangements, we get the following update rules:

$$\begin{aligned}
\alpha_j^t &= \frac{1}{N}\sum_{i=1}^{N} P(y_i = j|x_i, \mu_j^{t-1}, \Sigma_j^{t-1}) \\
\mu_j^t &= \frac{\sum_{i=1}^{N} x_i P(y_i = j|x_i, \mu_j^{t-1}, \Sigma_j^{t-1})}{\sum_{i=1}^{N} P(y_i = j|x_i, \mu_j^{t-1}, \Sigma_j^{t-1})} \\
\Sigma_j^t &= \frac{\sum_{i=1}^{N} P(y_i = j|x_i, \mu_j^{t-1}, \Sigma_j^{t-1})(x_i - \mu_j^t)(x_i - \mu_j^t)^T}{\sum_{i=1}^{N} P(y_i = j|x_i, \mu_j^{t-1}, \Sigma_j^{t-1})}
\end{aligned} \quad \text{(Eq. 1.18)}$$

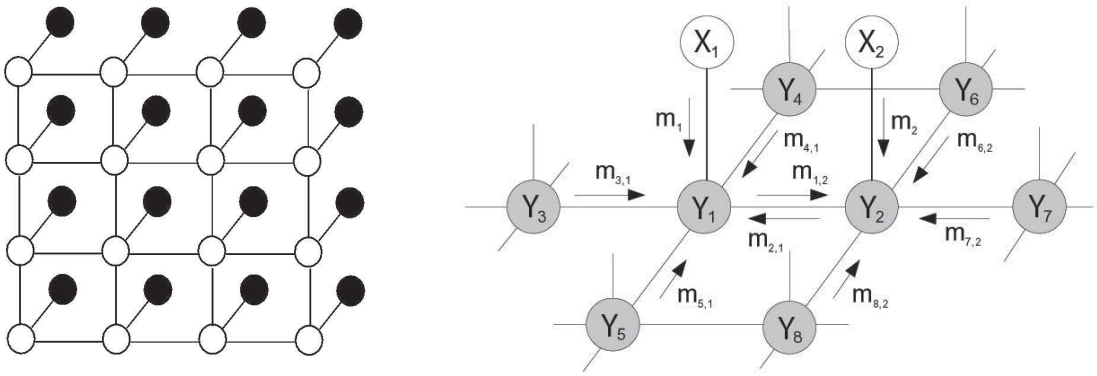
The mathematic details about the EM algorithm and the GMM parameters estimation example can be found in [15].

1.3.3 Belief Propagation

Belief Propagation (BP) is used for the discrete labeling problem in pairwise Markov Random Fields (MRF). Since image intensity is discrete, BP is commonly used in image denoising. BP is also commonly used in the stereo problem, where the depth is quantized into discrete labels; and in segmentation problem, where each segment corresponds to one and only one discrete label. Another common technique for solving discrete labeling problem in pairwise MRF is the Graph cut algorithm [58]. A comparative study between belief propagation and graph cut is presented in [112] which concludes that their results are comparable.

Denote by $\mathcal{X}$ the observations, we assume there are hidden variables $\mathcal{Y}$ which encode the relationships between observations. These hidden relationships between observations allow us to form a pairwise Markov network. Our goal is then to find a configuration of $\mathcal{Y}$ such that it maximizes the following joint probability energy function:

$$\begin{aligned}
\arg\max_{\mathcal{Y}} P(\mathcal{X}, \mathcal{Y}) &= \arg\max_{\mathcal{Y}} P(\mathcal{X}|\mathcal{Y})P(\mathcal{Y}) \\
&= \arg\max_{\mathcal{Y}} \prod_i P(x_i|y_i) \prod_i \prod_{y_j \in N(y_i)} P(y_i, y_j) \\
&= \arg\min_{\mathcal{Y}} \sum_i L(x_i|y_i) + \sum_i \sum_{y_j \in N(y_i)} L(y_i, y_j)
\end{aligned} \quad \text{(Eq. 1.19)}$$



(a) (b)

Figure 1.5: (a) The Pairwise Markov Network, it is an undirected graph encoding the pairwise relationships between hidden variables. (b) Local message passing in a Markov Network. Images for this figure are from [106].

where $L(\cdot) = -log(P(\cdot))$, $N(y)$ is the first order neighborhood of y. An illustration of the energy function defined in equation (Eq. 1.19) is shown in figure 1.5(a). The energy function forms a pairwise Markov network. In a Bayesian formulation, $P(\mathcal{X}|\mathcal{Y})$ is the likelihood probability, and P(Y) is the prior probability.

Belief propagation (BP) is an iterative inference algorithm that propagates messages in the network. There are two different algorithms for implementing belief propagation: the sum-product algorithm and the max-product algorithm. The sum-product algorithm computes the marginal distributions of each node, while the max-product algorithm computes the MAP estimate of the whole MRF. In this thesis, we discuss the max-product algorithm.

Let $m(y_j, y_i)$ be the message that hidden node y_j sends to y_i, $m(x_i, y_i)$ be the message that observed node x_i sends to hidden node y_j, and $b(y_i)$ be the belief at node y_i. Note that the message sent from y_j is different from the message sent from x_i. The message sent from x_i is defined by the likelihood probability, and the message sent from y_i is defined by the prior probability. The belief $b(y_i)$ is a vector encodes the current states of y_i with different confidence. The standard max-product algorithm is given below:

(i) Initialize all messages $m(y_j, y_i)$ as uniform distributions and messages $m(x_i, y_i) = P(x_i|y_i)$.

(ii) Update messages $m(y_j, y_i)$ iteratively for $i = 1 : T$:

$$m(y_j, y_i)^{t+1} = \kappa \max_{y_j} P(y_j, y_i) m(x_i, y_i) \prod_{y_k \in N(y_j) \setminus y_i} m(y_k, y_j)^t$$

(iii) Compute beliefs

$$b(y_i) = \kappa m(x_i, y_i) \prod y_j \in N(y_i) m(y_j, y_i)$$

$$b(y_i)^{MAP} = \arg\max_{y_k} b(y_k)$$

where κ is a normalization constant. Figure 1.5(b) illustrates the message passing procedure in belief propagation. The computational complexity of a standard max-product belief propagation algorithm is $O(TNL^2)$, where N is the number of nodes in the Markov network, T is the number of iterations and L is number of discrete labels. An example of using belief propagation for the stereo problem is given in [106] and an example for photometric stereo is given in [125]. More information on the belief propagation algorithms can be found in [128, 129].

1.4 Using Bayesian Optimization: Our Contribution

This section gives a brief introduction to the problems we have studied: image segmentation, texture flow estimation and image/video deblurring. Following the definitions of the Bayesian model, we formulate the three problems into Bayesian ML/MAP optimization problems. The central idea is on how to identify useful information available in each problem and define the likelihood probability and the prior probability properly.

[Soft Color Segmentation] In image segmentation, we present an algorithm to perform soft color segmentation given a color image. Unlike traditional image segmentation approaches, our segmentation approach is designed to address a large class of image-based problems which

require soft segments with an appropriate amount of overlap and transparency. We formulated this problem in a Bayesian optimization framework. Our global objective function consists of both global and local parameters. The global parameters define similarity of same segmented regions, and dissimilarity across different segmented regions. We argue that a Gaussian Mixture Model (GMM) is sufficient to represent the global color statistics of an image and each segment corresponds to a Gaussian distribution of GMM. To handle spatial and color coherence among soft segments while preserving discontinuities, we introduce a set of local parameters, and we assign to each pixel a set of soft labels corresponding to their respective color distributions. The global and local parameters are interdependent. Our Bayesian optimization framework simultaneously exploits the reliability given by global color statistics and flexibility of local image compositing. We use Alternating optimization to solve our problem in which global and local parameters are refined iteratively and alternatively. We performed extensive experiments to compare our segmentation result to many current image segmentation algorithms that included k-means clustering [32], Mean Shift [26], Expectation-Maximization (EM) [28, 6], Watershed [116], Jseg [29], DDMCMC [114], Information-Bottleneck [44], Multiscale graph-based techniques [101, 42], the statistical region merging [80], and user-assisted image matting [10, 98, 21]. This work has been published in CVPR'05 [110] and PAMI'07 [108].

[Texture Flow] For texture flow estimation, we propose a novel texture feature representation that is suitable for estimating the orientation and scale of texture pixels, while making no assumptions about the underlying texture properties. Our texture flow estimation begins with a small example patch of the texture that is specified by the user. From this example patch, a set of principal features are extracted that are used to compute the likelihood probability about the orientation and scale of each pixel lying in a distorted texture region. Combined with neighborhood smoothness and discontinuity priors, we formulate the final texture flow estimation problem using the Bayesian method as a discrete labeling problem of a Markov Random Field (MRF) and solve it using a variant of belief propagation. We demonstrate the effectiveness of

this approach on a variety of inputs, including natural and synthesized images and show the usefulness of this extracted flow field for texture remapping. This work has been published in CVPR'07 [107].

[Image Deblurring] In image deblurring problem, we propose a novel approach which is based on the hybrid camera framework proposed by Ben-Ezra and Nayar [7, 8] to reduce spatially varying motion blur. The work in [7, 8] focused on correcting motion blur due to ego motion in a still-camera, and therefore was limited to addressing global translational motion. Their method also processed only a single still image. In this work, we addresses the broader problem of deblurring with spatially varying motion blur, and we target the problem of correcting a temporal sequence (i.e. video footage). The central idea in our Bayesian formulation is to combine the benefits of both deconvolution and super-resolution. Deconvolution of motion blurred, high-resolution images yields high frequency details, but with ringing artifacts due to lack of low-frequency components. In contrast, super-resolution-based reconstruction from low-resolution images recovers artifact-free low-frequency results that lack high-frequency detail. We show that the deblurring information from deconvolution and super-resolution are complementary to each other, and can be used together to elevate deblurring performance. We demonstrate that this approach produces excellent results in deblurring spatially varying motion blur compared to state-of-the-art techniques. In addition, the availability of the low-resolution imagery, and subsequently derived motion vectors, further allows us to perform super-resolution in the temporal domain. This work has been published in CVPR'08 [109] and is currently in its second revision for PAMI.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 disucsses soft color segmentation; Chapter 3 present the work on texture flow estimation; Chapter 4 details our work on image and video deblurring; and Chapter 5 presents a summary. Chapters 2, 3 and 4

are self-contained. Each chapter describes the problem definition, related work, the Bayesian formulation of the problem and associated optimization procedure used. Each chapter also includes the results, discussion, and summary pertaining to its associated problem. Chapter 5 concludes this thesis with summary of each problem, a discussion on how to forumalate problems using a Bayesian framework as well as some future research directions.

Chapter 2

Soft Color Segmentation and its applications

2.1 Overview

This describes an automatic approach to soft color segmentation, which produces soft color segments with appropriate amount of overlapping and transparency essential to synthesizing natural images for a wide range of image-based applications. While many state-of-the-art and complex techniques are excellent at partitioning an input image to facilitate deriving a semantic description of the scene; to achieve seamless image synthesis, we advocate a segmentation approach designed to maintain spatial and color coherence among soft segments while preserving discontinuities, by assigning to each pixel a set of soft labels corresponding to their respective color distributions. We formulated this problem into a Bayesian optimization framework. Our global objective function consists of both global and local parameters. We optimize the global objective function which simultaneously exploits the reliability given by global color statistics and the flexibility of local image compositing, leading to an image model where the global color statistics of an image is represented by a Gaussian Mixture Model (GMM): while the color of a pixel is explained by a local color mixture model where the weights are defined by the soft labels to the elements of the converged GMM. Transparency is naturally introduced in our probabilistic framework which infers an optimal mixture of colors at an image pixel.

To adequately consider global and local information in the same framework, an alternating optimization scheme is proposed to iteratively solve for the global and local model parameters. Our method is fully automatic, and is shown to converge to a local optimal solution. We perform extensive evaluation and comparison, and demonstrate that our method achieves good image synthesis results for image-based applications: such as image matting, color transfer, image deblurring, and image colorization.

2.2 Background and motivation

Given a color image, our algorithm performs soft color segmentation, producing overlapping and transparent segments suitable for a wide range of important image-based applications: such as image matting [10, 98, 21] (figure 2.20–2.21), color transfer between images [95, 110] (figure 2.22–2.24), image deblurring [55] (figure 2.25), image denoising [36, 86] (figure 2.26). Unlike traditional approaches, our segmentation approach is designed to address a large class of image-based problems which require soft segments (with appropriate amount of overlapping and transparency). This approach is translated into an alternating optimization (AO) algorithm which is more straightforward to implement than many state-of-the-art and complex segmentation techniques, which are geared to produce a semantic segmentation of the input image for tasks such as recognition and interpretation.

We present a probabilistic framework to address soft color segmentation, where a global objective function is modeled by global and local parameters. These parameters are alternately optimized until convergence. Since our goal is to maintain natural color and texture transition across soft segments rather than assigning semantics to each segmented region, it is sufficient to model *global* color statistics of an image by Gaussian Mixture Model (GMM). Each pixel's color can be explained by a *local* mixture of colors derived from the optimized GMM weighted by the inferred soft labels.

Our segmentation goal is different but related to that of traditional segmentation approaches. In this chapter, we evaluate and compare our automatic method with k-means clustering [32], Mean Shift [26], Expectation-Maximization (EM) [28, 6], Watershed [116], Jseg [29], Data Driven Markov Chain Monte Carlo (DDMCMC) [114], Information-Bottleneck [44], Multi-scale graph-based techniques [101, 42], the statistical region merging [80], and user-assisted image matting [10, 98, 21] to show that better or comparable results are obtained, in terms of region transparency, color coherence and spatial coherence. Our method produces results comparable to the Bayesian matting [21] in terms of extracting a foreground matte from an image. In [21], a user-supplied trimap is required while our method is fully automatic. Our proposed algorithm is also applied to various image applications such as transferring color between images, image deblurring, image denoising, and colorizing grayscale images.

The chapter is organized as follows: Section 2.3 reviews the related work on color and image segmentation. Section 2.4 describes in detail our alternating optimization (AO) algorithm which estimates optimal global and local model parameters. We perform experiments to show the good optimality and convergence of our AO algorithm while the theoretical aspects of these issues are addressed in [11]. In section 2.5, we evaluate and analyze our AO algorithm using synthetic and real images. Results and comparisons are presented in section 2.6. In section 2.7, we apply our soft color segmentation to various image synthesis applications and show that significantly better results can be obtained by employing soft segments produced by our algorithm. We conclude this chapter in section 2.8. Proposals of future research direction as discussed in chapter 5.

2.3 Related work

We review in this section previous work most relevant to ours in image segmentation.

2.3.1 Hard segmentation

The Watershed algorithm [116] is a region-based technique where "watershed" lines are used to mark the boundaries of regions. The morphological operations of closing (or opening) are then introduced to smooth ridges (or fill in valleys) of the topographical map produced. This method is sensitive to intensity changes, so a large number of small regions is usually produced. The Watershed algorithm is often used as a preprocessing step to obtain an over-segmented image to preserve as much detail as possible for further processing.

The Expectation-Maximization (EM) algorithm, which is one form of alternating optimization, was employed in [6] to address the problem of color and texture segmentation. The joint distribution of color and texture is modeled using a mixture of Gaussians in a six-dimensional space (three dimensions for color and three for texture). Because the grouping is performed in a 6D space and no spatial coordinates are considered, small and fragmented regions are produced. A separate spatial grouping step is then applied to obtain pixel connected components.

The JSeg [29] is an unsupervised algorithm for color and texture segmentation. The first color quantization step creates a class-map of color labels. The second spatial segmentation step uses the class-map to create a J-image to identify color or texture regions. The two steps are sequential, where the second step is dependent upon the results produced by the first one.

The Mean Shift segmentation [26] is a clustering algorithm that can perform color and texture segmentation. The algorithm takes as input a feature bandwidth, a spatial bandwidth, and a minimum region area (in pixels). Salient clusters are successively extracted by applying a kernel in the feature space, which shifts toward significant cluster center. Because the feature space is a high dimensional one, in order to reduce the number of shifts for achieving fast convergence, a set of random locations in the feature space is usually considered for selecting the initial location with the highest density of feature vectors.

Graph-based approaches for image segmentation and grouping have gained much attention. The Normalized Cuts [102] is one such algorithm which uses a global criterion on the total

dissimilarity among (and similarity within) different pixel groups, where discrete region labels output after graph optimization.

The statistical region merging was proposed in [80], which consists of a semi-supervised statistical region refinement algorithm for color image segmentation. Based on certain principles on perceptual grouping and an image generation model, a simple merging method was proposed to produce visually coherent color segments.

2.3.2 Soft segmentation

The concept of soft segmentation is not a new one. For example, the traditional k-means clustering [32] can be considered as one form of soft color segmentation. In essence, each point in the feature space is associated with a label and its confidence value calculated using some function related to the distance to each converged cluster. If spatial and color coordinates are considered simultaneously for preserving the coherence, the resulting feature space becomes sparse and high-dimensional, making the method vulnerable to local optima.

The split-and-link algorithm [87] computes overlapping segments in a pyramidal framework where the levels are overlapped so that each pixel is a descendant of four others in the pyramid. The linking is done based on similarity to ameliorate some problems in initial splitting. In [44], unsupervised image clustering was proposed to cluster images, subject to minimizing the loss of mutual information between the clusters and image features. The proposed clustering can be regarded as soft label classification, where GMMs are used to model the feature space. A graph-based approach was proposed in [101] which combines multiscale measurements of intensity contrast, texture differences and boundary integrity. The method optimizes a global measurement over a multiscale pyramidal structure of the image, and maintains fuzzy relationship between nodes in successive levels. A follow-up of the work [42] made use of multiscale aggregation of filter responses to handle complex textures.

In [78], a clustering-based algorithm was proposed to segment color textures, where multiscale smoothing and initial clustering are first performed to determine a set of core clusters to

which a subset of pixels should belong. Soft labels are then assigned and updated iteratively at all other pixels at multiple scales.

A unifying framework known as DDMCMC was proposed [114] which exploits Markov Chain dynamics to explore the complex solution space and achieves a nearly global optimal solution regardless of initial segmentations. Since features occur at multiple scales, the method incorporates intrinsic ambiguities in image segmentation, and utilizes data-driven techniques (such as clustering where soft assignment is made in the feature space) for computing importance proposal probabilities.

Fuzzy connectedness [115] groups image elements (pixels) by assigning a strength of connectedness to every possible path between every possible pair of image elements. The connectedness strength is related to the region that the image element belongs to. An image element can be associated with more than one region with different connectedness strength. The method has been extensively experimented in segmenting delicate tissues from medical images.

In computer graphics, the class of image matting algorithms can be considered as a special case of soft color segmentation. Smith and Blinn [103] were the first to present the blue screen matting systematically. Knockout [10] is one method that gathers color samples by estimating the foreground and background with weighted averages of the pixel colors within a neighborhood. Ruzon *et al.* [98] sampled colors by a mixture of Gaussians, and proposed to use the color with the maximum probability. In Bayesian matting [21], the authors formulated the matting problem using Bayesian optimization, where the maximum *a posteriori* (MAP) estimation is performed to estimate the optimal alpha matte for foreground extraction. This method performs pixelwise optimization without exploiting any spatial coherence information. Grabcut [97] and Poisson matting [105] consider matte continuity among pixels. Note that all the above matting techniques are not automatic, requiring some form of user interaction, usually in the form of a user-supplied trimap which specifies “definite foreground,” “definite background” and “uncertain” regions, in order to produce satisfactory matting results.

2.3.3 Comparison with our work

The approaches described in the previous section have made significant contributions in image segmentation. However, they are not suitable to be used in an image-based application which requires soft color segments with an appropriate amount of overlapping and transparency due to one or more of the following reasons:

- While the previous methods produce excellent image segmentation results for natural images, they are designed to solve the general segmentation problem, which may not be ideal for image-based applications. For instance, to obtain a good image interpretation, general image segmentation aims to cluster similar patterns or textures. However, as shown in the result section, the details inside each pattern should be preserved so that distinct colors will not get mixed up in the synthesized image. Furthermore, the resulting segments reported in the above literature are mostly hard segments which do not preserve smooth color transition among segments.

- To maintain spatial and color coherence, many algorithms concatenate spatial and feature vectors resulting in a sparse and high dimensional feature space. A careful initialization is therefore needed to ensure fast convergence to a reasonable solution.

- On the other hand, spatial grouping and color clustering are considered, by certain approaches, as independent rather than interdependent processes, so errors produced in one step are propagated to the following steps.

- All matting methods are interactive requiring somewhat careful initialization (e.g. a user-supplied trimap).

In this chapter, we propose an automatic color segmentation approach to address the above issues. To maintain spatial and color coherence, instead of using a high-dimensional feature

space, an alternating optimization framework is adopted: our method optimizes a global objective function that combines the advantages given by global color statistics and local image compositing. Using a global objective function, global and local information is properly integrated by using a Markov network that optimizes for the soft labels at each image pixel, subject to spatial and color coherence while preserving underlying discontinuities. The global color statistics of an image is specified by the inferred three dimensional Gaussian Mixture Model (GMM). A local mixture model is introduced to account for the observed color at each pixel, where soft labels are introduced to naturally encode transparent and overlapping regions in our probabilistic framework. We propose an alternating optimization (AO) algorithm to estimate an optimal set of model parameters. Readers may refer to [11] for AO's convergence. Our method also proves the convergence empirically by extensive experiments on a variety of natural and complex images. We demonstrate the efficacy of our approach in a wide variety of image-based applications, and show that less human interaction or better results can be obtained using our soft color segmentation method.

2.4 Soft Color Segmentation

Our approach in soft color segmentation takes into consideration both global and local color information within the same framework. *Global color statistics* models the overall colors of the input image. *Local color compositing* models the mixture of colors contributing to the observed color at a pixel, where the colors are derived from the optimized global statistics.

In our framework, the global and local models cooperate with each other subject to the spatial and color coherency constraints in each pixel's neighborhood, where the similarity within the same region and dissimilarity across regions are preserved. An *alternating optimization* scheme [27, 11] is adopted to iteratively optimize the global and local parameters. The notations are summarized in Table 2.1.

Notation	Meaning
I	Image space
$I(x, y)$	RGB color at pixel (x, y)
$\mathcal{G}$	The 3D GMM $\mathcal{G} = \{G(i; \mu_i, \Sigma_i)\}, i = 1, \cdots, N$
μ_i	Mean of Gaussian component $i, i = 1, \cdots N$
Σ_i	Covariance matrix of Gaussian component $i, i = 1, \cdots N$
$\mathcal{L}(x, y)$	Soft labels to $\mathcal{G}$ at pixel (x, y): $\mathcal{L}(x, y) = \{\ell_i(x, y) \vert i = 1, \cdots N\}$
$\mathcal{C}(x, y)$	Compositing colors derived from $\mathcal{G}$ at pixel (x, y): $\mathcal{C}(x, y) = \{c_i(x, y) \vert i = 1, \cdots N\}$

Table 2.1: Notation used in this chapter. For local parameters, the coordinates (x, y) may be skipped when we refer to the entire set of parameters across the whole image. Note that μ_i, $I(x, y)$, $c_i(x, y)$ are RGB tuples. N is the number of Gaussians.

2.4.1 Problem modeling and formulation

Global color statistics By observing a large number of *natural* images (figure 2.1), we find that the global color statistics can be represented by a set of overlapping regions and modeled by a mixture of Gaussians. In other words, the color of a pixel can be predicted by a Gaussian Mixture Model (GMM), where each Gaussian is three dimensional for encoding the R, G, and B channels, and is parameterized by the corresponding mean and covariance matrix. Let us denote the GMM by $\mathcal{G} = \{G(i; \mu_i, \Sigma_i)\}$, where $1 \leq i \leq N$ and N is the number of Gaussians, μ_i is the mean, and Σ_i is the covariance matrix.

Note that global color statistics alone, such as GMM, are not sufficient for semantic segmentation or image understanding. This explains why many approaches that use GMM (e.g. [6, 44]) introduce additional constraints to address the semantic segmentation problem. On the other hand, given that the global color statistics of a natural image can be well modeled by a GMM, and that our goal is to infer overlapping color segments (which may not separate well in the color space, without any a priori information), it is a natural choice to use GMM to represent global statistics in the color space. On the other hand, our method does not purely rely on GMM. As we shall describe shortly, a local mixture model is used where MRFs are incorporated so that spatial and color coherence are optimized at each pixel's neighborhood

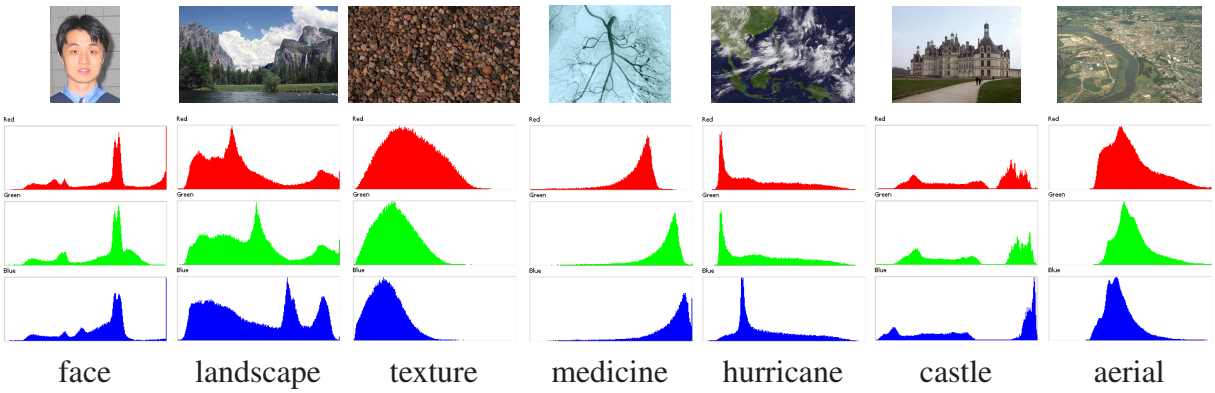


Figure 2.1: The global color statistics of a natural image can be modeled by a mixture of Gaussians. Top: typical images. Bottom: For illustration, the three rows shown are the histograms of the R, G, and B color channels, whereas three dimensional Gaussians are used in our computation.

during the alternating optimization process.

Local color compositing By generalizing the image compositing equation in [89], we propose the following model for local color compositing at each pixel (x, y) which uses a mixture of colors to encode overlapping and transparency:

$$I(x,y) = \sum_{i=1}^{N} \ell_i(x,y) c_i(x,y) \qquad \text{(Eq. 2.1)}$$

where $\mathcal{L}(x,y) = \{\ell_i(x,y) | i = 1, 2, \cdots N\}$ is the set of soft labels corresponding to the N color segments (or Gaussians) in $\mathcal{G}$, and $\mathcal{C}(x,y) = \{c_i(x,y) | i = 1, 2, \cdots N\}$ is the set of compositing colors.

Problem formulation In summary, our color model consists of: a global term representing the color model for the whole image described by $\mathcal{G}$, and a local term for each pixel described by $\mathcal{L}$ and $\mathcal{C}$. The set of unknowns consists of

$$\{c_i(x,y), \ell_i(x,y), \mu_i, \Sigma_i | i = 1, 2, ..., N \text{ and } (x,y) \in I\}.$$

We formulate the problem of soft color segmentation as follows: Given an image I and the number of Gaussians (segments) N, we seek to optimize for the global and local model

parameters $\mathcal{G}, \mathcal{L}, \mathcal{C}$ subject to the necessary spatial and color coherency and discontinuity inherent in the image. Using the Minimum Description Length principle [46, 45] (also used in the Expectation-Maximization algorithms in [6, 44]) which avoids model overfitting, the number of Gaussians N can be inferred. In practice, we fix the value of N where $N < 10$ in all cases we tested. The weights of insignificant Gaussians will be converged to zero. We describe in the following our alternating optimization (AO) algorithm to solve for the optimal $\mathcal{L}, \mathcal{C}$ and $\mathcal{G}$ iteratively.

2.4.2 The global optimization function

Given a color image I, we maximize the *a posteriori* probability to infer all unknowns. We formulate the problem in a Bayesian framework as follows:

$$\begin{aligned} \arg\max_{\mathcal{G},\mathcal{L},\mathcal{C}} \mathrm{P}(\mathcal{G},\mathcal{L},\mathcal{C}|I) &\propto \arg\max_{\mathcal{G},\mathcal{L},\mathcal{C}} \mathrm{P}(I|\mathcal{G},\mathcal{L},\mathcal{C})\mathrm{P}(\mathcal{C}|\mathcal{G},\mathcal{L})\mathrm{P}(\mathcal{L},\mathcal{G}) \\ &\propto \arg\max_{\mathcal{G},\mathcal{L},\mathcal{C}} \mathrm{P}(I|\mathcal{L},\mathcal{C})\mathrm{P}(\mathcal{C}|\mathcal{G},\mathcal{L})\mathrm{P}(\mathcal{L}) \qquad \text{(Eq. 2.2)} \end{aligned}$$

where $\mathrm{P}(I|\mathcal{L},\mathcal{C}) = \mathrm{P}(I|\mathcal{G},\mathcal{L},\mathcal{C})$ since $I(x,y)$ is not directly related to $\mathcal{G}$ by (Eq. 2.1), $\mathrm{P}(\mathcal{L},\mathcal{G}) = \mathrm{P}(\mathcal{L})\mathrm{P}(\mathcal{G})$ since in our model $\mathrm{P}(\mathcal{L})$ and $\mathrm{P}(\mathcal{G})$ are independent. Finally, $\mathrm{P}(\mathcal{G})$, the global color statistics, is assumed to be a uniform distribution without any prior knowledge. So $\mathrm{P}(\mathcal{G})$ is omitted in (Eq. 2.2).

2.4.2.1 Matching likelihood $\mathrm{P}(I|\mathcal{L},\mathcal{C})$

We assume the observation noise follows an independent identical distribution (i.i.d.), so we define the likelihood $\mathrm{P}(I|\mathcal{L},\mathcal{C})$ as a product of likelihoods at each pixel:

$$\begin{aligned} \mathrm{P}(I|\mathcal{L},\mathcal{C}) &\propto \prod_{(x,y)} \mathrm{P}(I(x,y)|\mathcal{L}(x,y),\mathcal{C}(x,y)) \\ &= \prod_{(x,y)} \frac{1}{\sqrt{2\pi}\sigma_c} \exp\left(-\frac{(I(x,y) - \sum_{i=1}^{N} \ell_i(x,y)c_i(x,y))^2}{2\sigma_c^2}\right) \qquad \text{(Eq. 2.3)} \end{aligned}$$

which models the fidelity how $I(x,y)$ conforms to the local model $\sum_{i=1}^{N} \ell_i(x,y)c_i(x,y)$ with standard deviation σ_c. We set $\sigma_c = 0.1$ in all our experiments.

2.4.2.2 Matching likelihood $\mathrm{P}(\mathcal{C}|\mathcal{G},\mathcal{L})$

We define the matching likelihood $\mathrm{P}(\mathcal{C}|\mathcal{G},\mathcal{L})$ in the same way, using a product of likelihoods at each pixel:

$$\mathrm{P}(\mathcal{C}|\mathcal{G},\mathcal{L}) \quad \propto \quad \prod_{(x,y)} \mathrm{P}(\mathcal{C}(x,y)|\mathcal{G},\mathcal{L}(x,y)) \tag{Eq. 2.4}$$

Based on the information theory [45], the relative entropy (Kullback-Leibler divergence) of $\mathcal{L}(x,y)$ and $\mathrm{P}(\mathcal{C}(x,y)|\mathcal{G})$ is defined as:

$$\sum_{i=1}^{N} \ell_i(x,y) \log \frac{\ell_i(x,y)}{\mathrm{P}(c_i(x,y)|G_i)} \tag{Eq. 2.5}$$

where $\ell_i(x,y)$ measures the confidence that the value c_i is generated by Gaussian element i. The relative entropy can be minimized as we maximize its negation:

$$\begin{aligned} & \arg\min_{\mathcal{C}} \sum_{i=1}^{N} \ell_i(x,y) \log \frac{\ell_i(x,y)}{\mathrm{P}(c_i(x,y)|G_i)} \\ \propto \quad & \arg\max_{\mathcal{C}} \sum_{i=1}^{N} (\log \mathrm{P}(c_i(x,y)|G_i)^{\ell_i(x,y)} - \log \ell_i(x,y)^{\ell_i(x,y)}) \\ \propto \quad & \arg\max_{\mathcal{C}} \sum_{i=1}^{N} \log \mathrm{P}(c_i(x,y)|G_i)^{\ell_i(x,y)} - \sum_{i=1}^{N} \log \ell_i(x,y)^{\ell_i(x,y)} \end{aligned} \tag{Eq. 2.6}$$

In the representation of the above likelihood, $\mathcal{L}$ is the observation. Omitting $\sum_{i=1}^{N} \log \ell_i(x,y)^{\ell_i(x,y)}$ will not influence the estimation result. Thus (Eq. 2.6) can be simplified as follows when we take the exponent:

$$\arg\max_{\mathcal{C}} \sum_{i=1}^{N} \log \mathrm{P}(c_i(x,y)|G_i)^{\ell_i(x,y)} \propto \arg\max_{\mathcal{C}} \prod_{i=1}^{N} \mathrm{P}(c_i(x,y)|G_i)^{\ell_i(x,y)} \tag{Eq. 2.7}$$

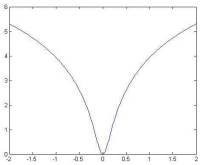
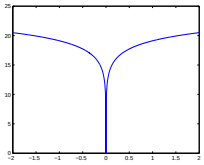
(a)  (b) 

Figure 2.2: The robust function for encoding the discontinuity-preserving function: plotting the Lorentzian estimator $\log(1 + \frac{1}{2}(\frac{x}{\sigma})^2)$ vs. x with (a) $\sigma = 0.1$, (b) $\sigma = 0.0005$, In all cases, the curves are bounded when $x \rightarrow \pm\infty$, which is more robust than the usual norm-squared function (i.e. the unbounded x^2) in terms of encoding the error term.

Given $G_i(i; \mu_i, \Sigma_i)$, the likelihood $\mathrm{P}(c_i(x,y)|G_i)$ is modeled by the deviation of $c_i(x,y)$ from the Gaussian G_i:

$$\mathrm{P}(c_i(x,y)|G_i) \propto \frac{1}{(2\pi)^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{(c_i(x,y)-\mu_i)^T\Sigma_i^{-1}(c_i(x,y)-\mu_i)}{2}\right) \qquad \text{(Eq. 2.8)}$$

Substitute (Eq. 2.8) into (Eq. 2.7), we obtain the function to be maximized:

$$\prod_{i=1}^{N} \frac{1}{\left((2\pi)^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}\right)^{\ell_i(x,y)}} \exp\left(-\ell_i(x,y)\frac{(c_i(x,y)-\mu_i)^T\Sigma_i^{-1}(c_i(x,y)-\mu_i)}{2}\right) \qquad \text{(Eq. 2.9)}$$

2.4.2.3 Prior $\mathrm{P}(\mathcal{L})$

Effective modeling of the prior is very important in producing good results in a Bayesian formulation. To maintain spatial and color coherence while introducing transparency among overlapping segments, we apply a Markov Random Field (MRF) model, which asserts that the conditional probability of a site in the field depends only on the information of its neighboring sites.

MRFs are effective in avoiding noise or highly fragmented segments while maintaining color smoothness across and within the segmented regions. In our soft color segmentation, although the resulting segments may sometimes not correspond to those produced by manual or semantic segmentation, as will be shown in the experimental section, they are adequate for image-based applications which require soft segments for synthesizing seamless images.

The prior $\mathrm{P}(\mathcal{L})$ encodes the probability that one pixel falls in different segments. We apply the following pairwise constraint:

$$\mathrm{P}(\mathcal{L}) \propto \prod_{(x,y)} \prod_{(x',y') \in \mathcal{N}(x,y)} \exp(-\psi(\mathcal{L}(x,y), \mathcal{L}(x',y'))) \qquad \text{(Eq. 2.10)}$$

where $\psi(\mathcal{L}(x,y), \mathcal{L}(x',y'))$ is the joint clique potential function of sites (x,y) and its first-order neighborhood sites (x',y').

We adopt the Lorentzian estimator (figure 2.2), a robust function [16] that preserves discontinuity implicitly by penalizing any occurrence of color discontinuity between sites (x,y) and (x',y'). The joint potential function is defined as follows:

$$\psi(\mathcal{L}(x,y), \mathcal{L}(x',y')) \quad = \quad \log\left(1 + \frac{1}{2}\left(\frac{||\mathcal{L}(x,y) - \mathcal{L}(x',y')||}{\sigma_p}\right)^2\right) \qquad \text{(Eq. 2.11)}$$

where σ_p is set to 0.1 in all our experiments. The use of a robust discontinuity-preserving function is typical, for instance, in MRF stereo [58, 106] and MRF photometric stereo [111, 125].

2.4.3 The alternating optimization

Combining the likelihoods of (Eq. 2.3) and (Eq. 2.9) and the prior in (Eq. 2.11), we solve the following global optimization problem by maximizing the posterior function:

$$\begin{aligned}\arg\max_{\mathcal{G},\mathcal{L},\mathcal{C}} \quad & \prod_{(x,y)} \frac{1}{\sqrt{2\pi}\sigma_c} \exp\left(-\frac{(I(x,y) - \sum_{i=1}^{N} \ell_i(x,y) c_i(x,y))^2}{2\sigma_c^2}\right) \prod_{(x,y)} \prod_{(x',y') \in \mathcal{N}(x,y)} \left(\frac{1}{1 + \frac{1}{2}\left(\frac{||\mathcal{L}(x,y) - \mathcal{L}(x',y')||}{\sigma_p}\right)^2}\right) \\ & \prod_{(x,y)} \prod_{i=1}^{N} \frac{1}{\left((2\pi)^{\frac{3}{2}} |\Sigma|^{\frac{1}{2}}\right)^{\ell_i(x,y)}} \exp\left(-\ell_i(x,y) \frac{(c_i(x,y) - \mu_i)^T \Sigma_i^{-1} (c_i(x,y) - \mu_i)}{2}\right) \end{aligned} \qquad \text{(Eq. 2.12)}$$

Obviously, this optimization problem cannot be directly solved due to the large number of unknowns. To solve (Eq. 2.12), we propose an alternating optimization (AO) algorithm [27]: by fixing the values of a subset of parameters in each iteration of the optimization process, the global objective function is maximized by optimizing each subproblem alternately.

2.4.3.1 Fix $\mathcal{G}, \mathcal{C}$ and optimize $\mathcal{L}$

In the first subproblem, we use the optimized values of $\mathcal{G}$ and $\mathcal{C}$ from the previous iterations to optimize $\mathcal{L}$. In the literature of probabilistic graph models, a Markov network is an undirected graph where the nodes $\{\mathrm{x}_s\}$ are used to encode hidden variables and the nodes $\{\mathrm{y}_s\}$ are used to encode observed variables. Taking $X = \{\mathrm{x}_s\}$ and $Y = \{\mathrm{y}_s\}$, the posterior $\mathrm{P}(X|Y)$ can be factorized as:

$$\mathrm{P}(X|Y) = \prod_s \psi_s(\mathrm{x}_s, \mathrm{y}_s) \prod_s \prod_{t \in \mathcal{N}(s)} \psi_{st}(\mathrm{x}_s, \mathrm{x}_t) \qquad \text{(Eq. 2.13)}$$

where $\psi_{st}(\mathrm{x}_s, \mathrm{x}_t)$ is the compatibility matrix between nodes x_s and x_t, encoding the cost between two neighboring pixels, and $\psi_s(\mathrm{x}_s, \mathrm{y}_s)$ is the local evidence for node x_s, representing the observation probability from the hidden variables $p(\mathrm{y}_s|\mathrm{x}_s)$.

In this subproblem, it can be observed that our posterior (Eq. 2.2) and (Eq. 2.13) are the same, if we define

$$\psi_{st}(\mathrm{x}_s, \mathrm{x}_t) = \exp(-\psi(\mathcal{L}(x, y), \mathcal{L}(x', y'))) \qquad \text{(Eq. 2.14)}$$

$$\psi_s(\mathrm{x}_s, \mathrm{y}_s) = \mathrm{P}(I(x, y)|\mathcal{L}(x, y), \mathcal{C}(x, y))\mathrm{P}(\mathcal{C}(x, y)|\mathcal{G}, \mathcal{L}(x, y)) \qquad \text{(Eq. 2.15)}$$

Thus finding the MAP solution of (Eq. 2.12) is equal to solving the Markov network where each hidden node encodes $\mathcal{L}(x, y)$. In belief propagation terms, N is the number of labels, and $\mathcal{L}$ is passed as 1D messages among the hidden nodes. Thus, the MAP estimation can be solved (approximately) by loopy belief propagation via a message passing procedure [84]. Because $\ell_i(x, y) \in [0, 1]$ is fractional and $\sum_{i=1}^{N} \ell_i(x, y) = 1$, transparency is naturally encoded at (x, y) by the soft labels. The memory and computational complexity of our belief propagation algorithm is $O(ZN)$ and $O(TZN)$ respectively, where Z is total number of pixels, N is number of Gaussians and T is number of iterations.

2.4.3.2 Fix $\mathcal{G}, \mathcal{L}$ and optimize $\mathcal{C}$

In the second subproblem, we use the optimized values of $\mathcal{G}$ and $\mathcal{L}$ obtained from the previous iterations to optimize $\mathcal{C}$. Since $\mathcal{L}$ is fixed, $\mathrm{P}(\mathcal{L})$ is a constant. Equation (Eq. 2.12) in this subproblem is simplified to:

$$\arg\max_{\mathcal{C}} \quad \prod_{(x,y)} \frac{1}{\sqrt{2\pi\sigma_c}} \exp\left(-\frac{(I(x,y)-\sum_{i=1}^{N}\ell_i(x,y)c_i(x,y))^2}{2\sigma_c^2}\right) \prod_{(x,y)}\prod_{i=1}^{N} \frac{1}{\left((2\pi)^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}\right)^{\ell_i(x,y)}} \exp\left(-\ell_i(x,y)\frac{(c_i(x,y)-\mu_i)^T\Sigma_i^{-1}(c_i(x,y)-\mu_i)}{2}\right) \quad \text{(Eq. 2.16)}$$

By taking the logarithm of (Eq. 2.16), we obtain a polynomial in several variables with constant coefficients. We perform pixelwise optimization by taking the partial derivative at each pixel and setting the equations equal to zero to compute the stationary points. In the following, the constant terms are ignored if they disappear after taking the partial derivative:

$$\frac{\partial}{\partial c_i}\left(\frac{(I(x,y)-\sum_{i=1}^{N}\ell_i(x,y)c_i(x,y))^2}{2\sigma_c^2} + \sum_{i=1}^{N}\ell_i(x,y)\frac{(c_i(x,y)-\mu_i)^T\Sigma_i^{-1}(c_i(x,y)-\mu_i)}{2}\right) = 0$$

Hence, we obtained the following $3N \times 3N$ linear equation system:

$$\begin{bmatrix} \ell_1(x,y)\Sigma_1^{-1}+\mathbf{I}\ell_1(x,y)^2/\sigma_c^2 & \dots & \mathbf{I}\ell_1(x,y)\ell_N(x,y)/\sigma_c^2 \\ \vdots & \ddots & \vdots \\ \mathbf{I}\ell_1(x,y)\ell_N(x,y)/\sigma_c^2 & \dots & \ell_N(x,y)\Sigma_N^{-1}+\mathbf{I}\ell_N(x,y)^2/\sigma_c^2 \end{bmatrix} \begin{bmatrix} c_1(x,y) \\ \vdots \\ c_N(x,y) \end{bmatrix} = \begin{bmatrix} \ell_1(x,y)\Sigma_1^{-1}\mu_1 + I(x,y)\ell_1(x,y)/\sigma_c^2 \\ \vdots \\ \ell_N(x,y)\Sigma_N^{-1}\mu_N + I(x,y)\ell_N(x,y)/\sigma_c^2 \end{bmatrix} \quad \text{(Eq. 2.17)}$$

where $\mathbf{I}$ is the identical matrix and $\sigma_c = 0.1$. Note that when $\ell_i(x,y)$ equals to 0, we do not need to estimate $c_i(x,y)$ since the equation is already balanced, and $c_i(x,y)$ has no effect to the objective function. Therefore, we can reduce the dimension of the linear system by removing the entries with $\ell_i(x,y) = 0$. In our implementation, we only estimate the color of pixels with

$|\ell_i(x,y)^t - \ell_i(x,y)^{t-1}| > \varepsilon$, where t is the iteration number, and ε is a small threshold set to be 0.01 in all our experiments. The above linear system is solved by using singular value decomposition (SVD).

2.4.3.3 Fix $\mathcal{L}, \mathcal{C}$ and optimize $\mathcal{G}$

In the last subproblem, we use the optimized values of $\mathcal{C}$ and $\mathcal{L}$ obtained from the previous iterations to optimize the unknown $\mathcal{G}$. Since $\mathcal{L}$ and $\mathcal{C}$ are fixed, $\mathrm{P}(I|\mathcal{L},\mathcal{C})$ and $\mathrm{P}(\mathcal{L})$ are constant in this subproblem. Taking the negative logarithm of our global objective function (Eq. 2.12) we obtain:

$$\begin{aligned} & \arg\max_{\mathcal{G}} \prod_{(x,y)} \prod_{i=1}^{N} \frac{1}{\left((2\pi)^{\frac{3}{2}}|\Sigma_i|^{\frac{1}{2}}\right)^{\ell_i(x,y)}} \exp\left(-\ell_i(x,y)\frac{(c_i(x,y)-\mu_i)^T\Sigma_i^{-1}(c_i(x,y)-\mu_i)}{2}\right) \\ = \; & \arg\min_{\mathcal{G}} \sum_{(x,y)} \sum_{i=1}^{N} \ell_i(x,y)\left(\frac{3\log(2\pi)+\log(|\Sigma_i|)+(c_i(x,y)-\mu_i)^T\Sigma_i^{-1}(c_i(x,y)-\mu_i)}{2}\right) \end{aligned} \tag{Eq. 2.18}$$

Taking the derivative with respect to μ_i and setting it to zero, we get:

$$\sum_{(x,y)} \Sigma_i^{-1}(c_i(x,y)-\mu_i)\ell_i(x,y) = 0 \tag{Eq. 2.19}$$

which is reduced to

$$\mu_i = \frac{\sum_{(x,y)} \ell_i(x,y)c_i(x,y)}{\sum_{(x,y)} \ell_i(x,y)} \tag{Eq. 2.20}$$

To find Σ_i, we can write (Eq. 2.18) as (constant terms are ignored here since they disappear after taking derivative):

$$\frac{1}{2}\sum_{(x,y)} \sum_{i=1}^{N} \ell_i(x,y)\left(\text{trace}(\Sigma_i^{-1}(c_i(x,y)-\mu_i)(c_i(x,y)-\mu_i)^T) - \log(|\Sigma_i^{-1}|)\right) \tag{Eq. 2.21}$$

Taking derivative with respect to Σ_i^{-1}, we get:

$$\frac{1}{2}\sum_{(x,y)} \sum_{i=1}^{N} \ell_i(x,y)\left(2\mathbf{M} - \text{diag}(\mathbf{M})\right) = 2S - \text{diag}(S)$$

where $\mathbf{M} = (c_i(x,y) - \mu_i)(c_i(x,y) - \mu_i)^T - \Sigma_i$, and $S = \frac{1}{2}\sum_{(x,y)}\sum_{i=1}^{N} \ell_i(x,y)\mathbf{M}$. Setting the derivative equal to 0, i.e. $2S - \text{diag}(S) = 0$, implies that $S = 0$, which gives:

$$\frac{1}{2}\sum_{(x,y)}\sum_{i=1}^{N} \ell_i(x,y)\left((c_i(x,y) - \mu_i)(c_i(x,y) - \mu_i)^T) - \Sigma_i\right) = 0 \qquad \text{(Eq. 2.22)}$$

Rearranging the equation we obtain:

$$\Sigma_i = \frac{\sum_{(x,y)} \ell_i(x,y)(c_i(x,y) - \mu_i)(c_i(x,y) - \mu_i)^T}{\sum_{(x,y)} \ell_i(x,y)} \qquad \text{(Eq. 2.23)}$$

Thus, (Eq. 2.20) and (Eq. 2.23) together give the optimal $\mathcal{G}$.

2.4.4 Summary

In summary, our alternating optimization algorithm is as follows: Initialize the unknowns $(\mathcal{G}, \mathcal{L}, \mathcal{C})$ and iterate the following steps until convergence or reaching a fixed number of iterations.

- Compute $\mathcal{L}$ by loopy belief propagation with (Eq. 2.13), (Eq. 2.14), and (Eq. 2.15).
- Compute $\mathcal{C}$ with (Eq. 2.17).
- Compute $\mathcal{G}$ with (Eq. 2.20) and (Eq. 2.23).

To initialize the optimization, we use the results produced by k-means clustering, where the mean and covariance for each Gaussian component of GMM $\mathcal{G}$ is initialized as the mean and covariance of the corresponding cluster. At each pixel, $\mathcal{L}$ is initialized as the soft labels obtained via k-means clustering. $\mathcal{C}(x,y)$ is set to $I(x,y)$ for all pixels (x,y).

2.4.5 Convergence

While the alternating optimization guarantees convergence to one type of global optimal solution [11], in this section, we experimentally test the convergence of our alternating optimization algorithm for soft color segmentation. In each respective step of estimating $\mathcal{L}$, $\mathcal{G}$ or $\mathcal{C}$, we use

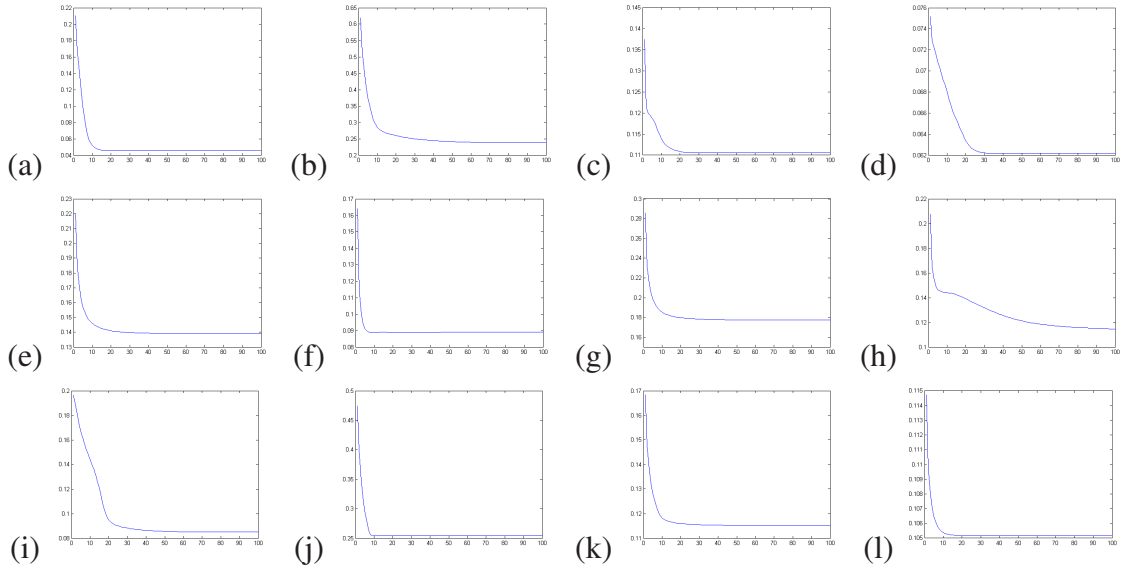


Figure 2.3: We plot the negative logarithm of the global objective function (Eq. 2.2) against the number of iterations. (a) *Synthetic image (figure 2.5)*,28sec. (b) *Graffiti (figure 2.6)*,114sec. (c) *Lighthouse (figure 2.8)*,75sec. (d) *Camellia (figure 2.9)*,39sec. (e) *River 1 (figure 2.12)*94sec. (f) *Blobworld (figure 2.11)*16sec. (g) *Motion blur*,106sec. (h) *Hurricane (figure 2.18)*,183sec. (i) *Nebula (figure 2.19)*,178sec. (j) *River 2 (figure 2.23)*,86sec. (k) *Castle (figure 2.24)*46sec. (l) *Gray level image* ,34sec. Our goal is to maximize (Eq. 2.2) which is equal to minimizing the negative logarithm of (Eq. 2.2). The values shown on the vertical axis are normalized by the number of pixels and the number of clusters in the converged $\mathcal{G}$. Empirically, our approach converges to a good minima within 30 to 40 iterations. Below each graph shows the actual running time (in second) to achieve convergence. Our AO approach typically runs in 180 secs for images with resolution 256×256 on a notebook computer with a 1.40GHz Intel Pentium M processor and 1.00GB RAM.

the values obtained for the two unknowns in the previous step to compute the maximum of the third unknown, which makes the estimated value of $\mathrm{P}(\mathcal{G}, \mathcal{L}, \mathcal{C}|I)$ increase monotonically. Hence, the convergence of our method is guaranteed and the maximum of $\mathrm{P}(\mathcal{G}, \mathcal{L}, \mathcal{C}|I)$ can be reached upon convergence. In our implementation, we terminate our iterations in either one of the following situations:

- $\sum_{0 \leq i < N} (\mu_i^t - \mu_i^{t-1})^2 < \eta$, where η is a predefined threshold and t is current iteration, or
- $t = 40$.

To demonstrate the optimization efficiency, we run our alternating optimization algorithm for a total of 100 iterations for each case. Figure 2.3 shows the graphs plotting the negative logarithm of the global objective function against the number of iterations. Note that maximizing the objective function (Eq. 2.2) is equivalent to minimizing its negative logarithm. Typically, our method converges within 30 to 40 iterations. Figure 2.4 shows some intermediate results during the AO iterations. We use different colors to represent each Gaussian components in $\mathcal{G}$, and label $I(x, y)$ as color i if $\ell_i(x, y)$ is largest in $\{\ell_1(x, y), \ell_2(x, y), \cdots, \}$. The output shown in the figure is hard color segmentation to facilitate visual evaluation. As the number of iterations increases, the spatial connectivity and color homogeneity are progressively refined until final convergence. In the result section, we shall show that our converged soft segmentation allows smooth and natural color transition for a wide range of image synthesis applications and produces satisfactory results.

2.5 Evaluation and Analysis

This section presents examples to evaluate our AO algorithm to perform soft color segmentation. Evaluations on synthetic and real data are first presented; followed by the study of the effects of $\mathcal{C}$ and $\mathcal{G}$ on the soft labels $\mathcal{L}$ estimation.

2.5.1 Synthetic image

Figure 2.5(a) shows a synthetic image used in our evaluation. This example presents a challenge because a single pixel's color can be explained by as many as six colors. The result shows that our automatic method is capable of segmenting the image into six coherent regions. Figure 2.5(b) shows a re-synthesized image $\sum_{i=1}^{6} \ell_i(x, y)c_i(x, y)$ generated by compositing the estimated $\mathcal{L}$ and $\mathcal{C}$ at each pixel. The image difference between the input image and the synthesized image $|I(x, y) - \sum_{i=1}^{6} \ell_i(x, y)c_i(x, y)|$ is shown in figure 2.5(c). We achieve an

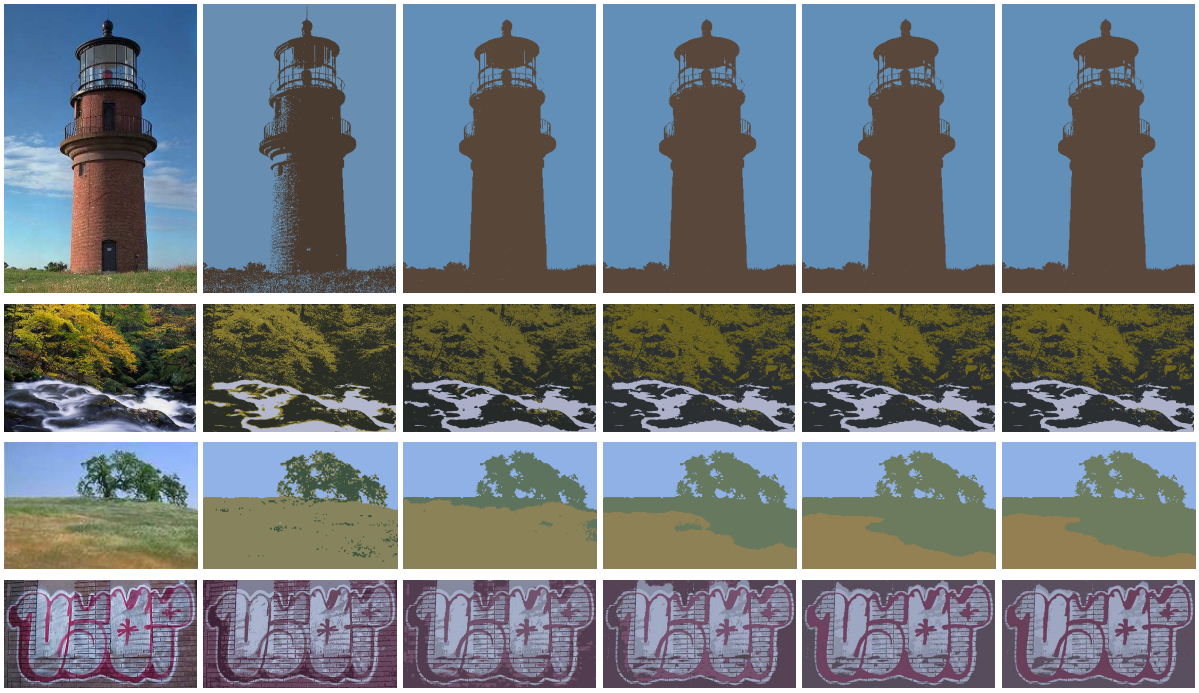


Figure 2.4: We show the intermediate results of the AO algorithm. Since each pixel may belong to more than one Gaussians, we show at each pixel the Gaussian component i with the maximum ℓ_i. Initially, no spatial connectivity is considered by k-means clustering, which performs clustering in the RGB space. Spatial and color coherence are progressively preserved as the number of iterations increases.

average pixel error of 0.0147, given by $\frac{\sum_{(x,y)} |I(x,y) - \sum_{i=1}^{N} \ell_i(x,y) c_i(x,y)|}{Z}$ where Z is the total number of pixels. We plot the soft labels $\mathcal{L}$ at sample pixels in figure 2.5(d) to show the transparent boundaries of the resulting soft regions. Figure 2.5(e) shows the segmentation results of each converged color region, displayed as $I_i(x, y) = \ell_i(x, y)c_i(x, y), 1 \leq i \leq 6$.

2.5.2 Real image

In this section we use real images to evaluate our method in the presence of rich textures and colors. We first use Poisson matting [105] to extract the graffiti c^* and the alpha matte ℓ^* from the original image. They are composited onto a set of new background images rich in textures and colors (shown in figure 2.6(a)). Finally, we segment the graffiti from the com-

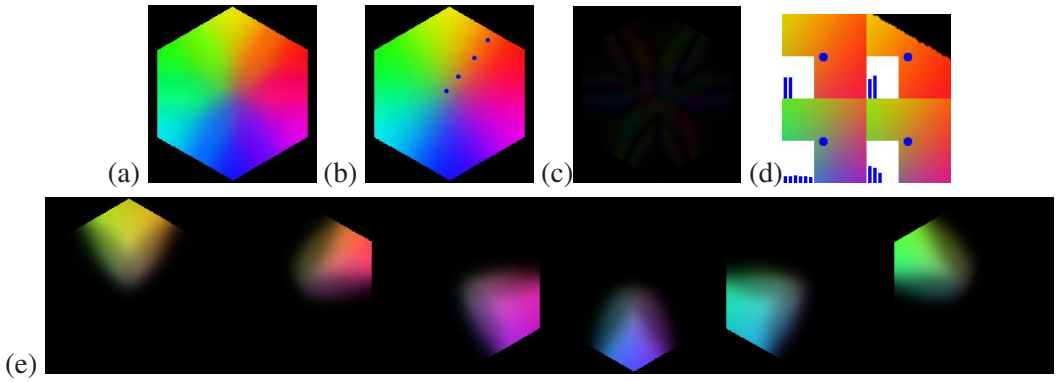


Figure 2.5: Evaluation using a synthetic image. (a) Here the observed color of a pixel may be explained by a mixture of as many as six colors. (b) The re-synthesized image generated by compositing $\mathcal{L}$ and $\mathcal{C}$ obtained upon the convergence of our AO algorithm. (c) Image difference between (a) and (b). (d) The soft labels $\mathcal{L}$ of sample pixels. (e) The soft segments, displayed as $I_i(x, y) = \ell_i(x, y)c_i(x, y)$, $1 \leq i \leq 6$, depict transparent and overlapping boundaries.

posited images using our soft color segmentation method. The segmented graffiti, shown in figure 2.6(b) as $\ell(x, y)c(x, y)$, is compared with $\ell^*(x, y)c^*(x, y)$ at each pixel (x, y) and the image differences are shown in figure 2.6(c). Note the small residual achieved by our method. Figure 2.6(d) shows the ℓ image for the graffiti segment, and figure 2.6(e) shows the difference $|\ell(x, y) - \ell^*(x, y)|$. The segmentation accuracy, defined as $1 - \frac{\sum_{(x,y)} |\ell(x,y) - \ell^*(x,y)|}{\sum_{(x,y)} \ell^*(x,y)}$, is over 70% in all the cases, while the segmented soft regions are visually indistinguishable from the ground truth.

2.5.3 Effect of color re-estimation

Next, we show the necessity of re-estimating $\mathcal{C}$ in our AO algorithm. The color estimation step corresponds to the second subproblem in our AO approach, which improves the results at the overlapping regions. Figure 2.7(b) shows one example where we do not perform color re-estimation, and we simply set all components of $\mathcal{C}$ to I. Comparing to figure 2.7(a) obtained using our proposed method, the segmented Gaussian components in (b) are less separable. A real example is shown in figure 2.8(a)–(b).

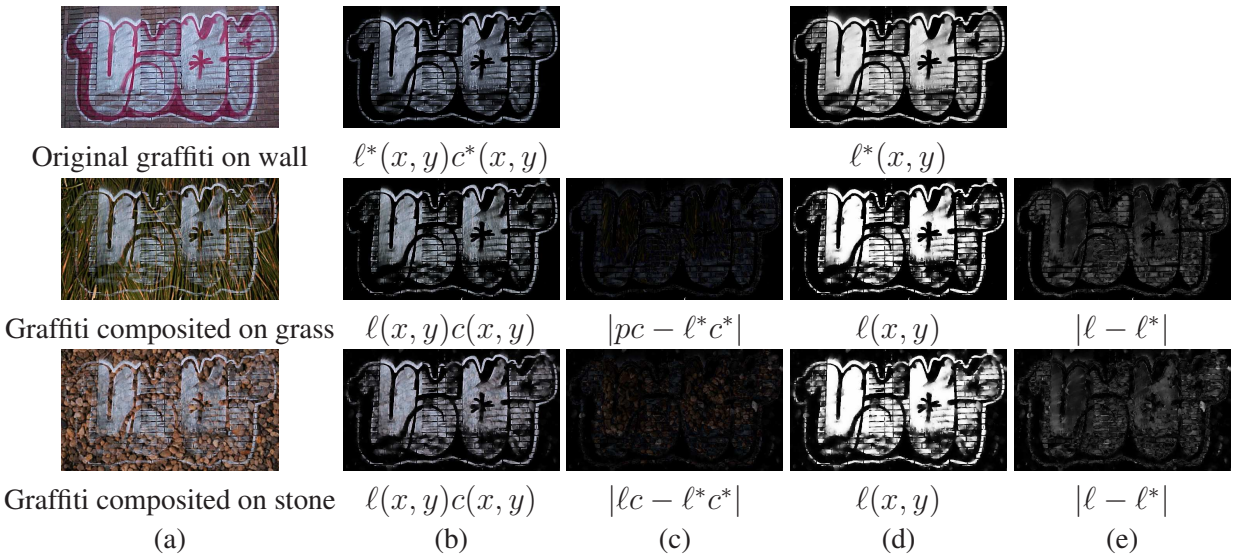


Figure 2.6: Evaluation using real image. (a) Original image and image composites. (b) The segmented graffiti. (c) Image difference between the ground truth segment $\ell^*(x,y)c^*(x,y)$ and the graffiti segmented from the new image composite. (d) The $\ell(x,y)$ image for the segmented graffiti. (e) Image difference between the ground truth ℓ^* and the converged ℓ.

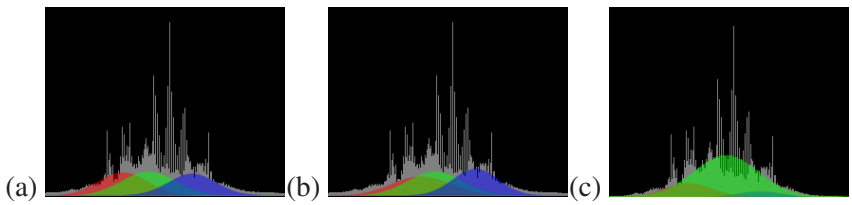


Figure 2.7: The three estimated Gaussians overlaid onto the histogram of the graffiti image. Only the R channel is shown here. (a) With color estimation. (b) Without color estimation. (c) shows the Gaussians estimated using the original EM algorithm. The estimated Gaussian components are better separated using our AO algorithm.

(a) (b) (c)

Figure 2.8: The images of soft label ℓ for a lighthouse image. The subimage on the lower right side shows the compositing result using the estimated ℓ and c on a green background. (a) Segmentation result by our AO approach. (b) Segmentation result without $\mathcal{C}$ re-estimation. Comparing the result in (a), better transparent regions are obtained with color re-estimation. (c) Segmentation result without $\mathcal{G}$ re-estimation. The input image is shown in figure 2.20.

2.5.4 Effect of GMM re-estimation

Finally, we show the necessity of re-estimating $\mathcal{G}$ in our AO algorithm. $\mathcal{G}$ encodes the global color statistics for soft color segmentation. It is initialized by k-means clustering. Without $\mathcal{G}$ re-estimation, our approach is reduced to one that runs belief propagation on the result produced by k-means clustering, followed by the color estimation step.

Figure 2.8(c) shows the result without $\mathcal{G}$ re-estimation, which is susceptible to errors given by the initial clustering, therefore resulting in the suboptimal estimation of $\mathcal{L}$ and $\mathcal{C}$, as depicted in the figure.

2.6 Results and comparison

We have evaluated our method on real and synthetic images and studied the effects of $\mathcal{C}$ and $\mathcal{G}$ on $\mathcal{L}$ in the AO algorithm. In this section, we present the results and comparisons with state-of-the-art segmentation techniques, focusing on: shading and shadows, and highly textured scenes.

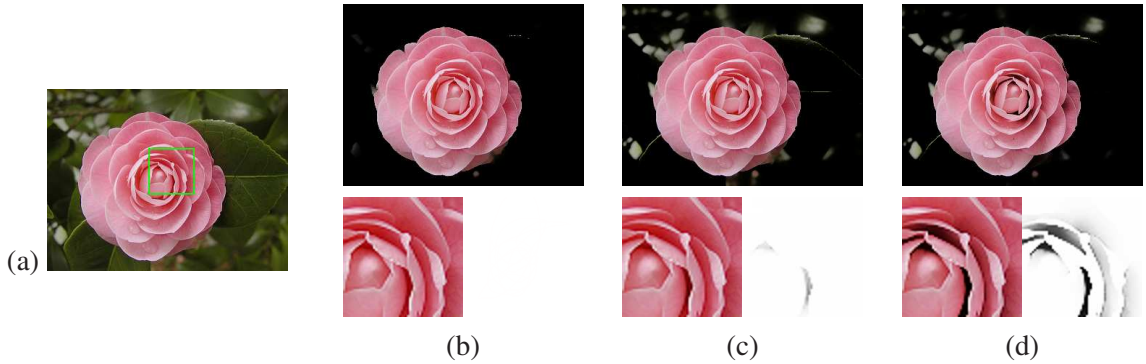


Figure 2.9: (a) Input image. Soft shading and shadows are present between the petals. The segmentation result using (b) covariance matrix (oriented Gaussians), (c) standard derivation (unoriented Gaussians), (d) k-means clustering with belief propagation and color estimation. Shading and soft shadows are better captured by an oriented Gaussian.

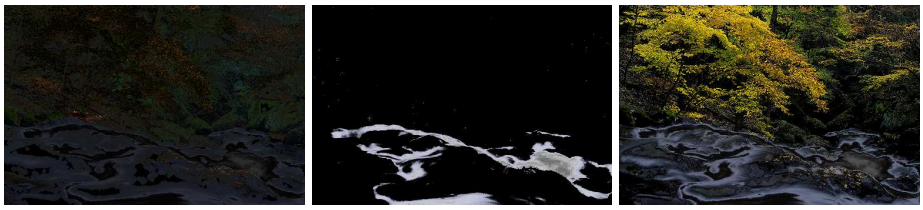


Figure 2.10: Segmentation result by the original EM algorithm. Without spatial consideration, spatially dissimilar patterns are mixed up. See the online version for color visualization.

2.6.1 Shading and soft shadows

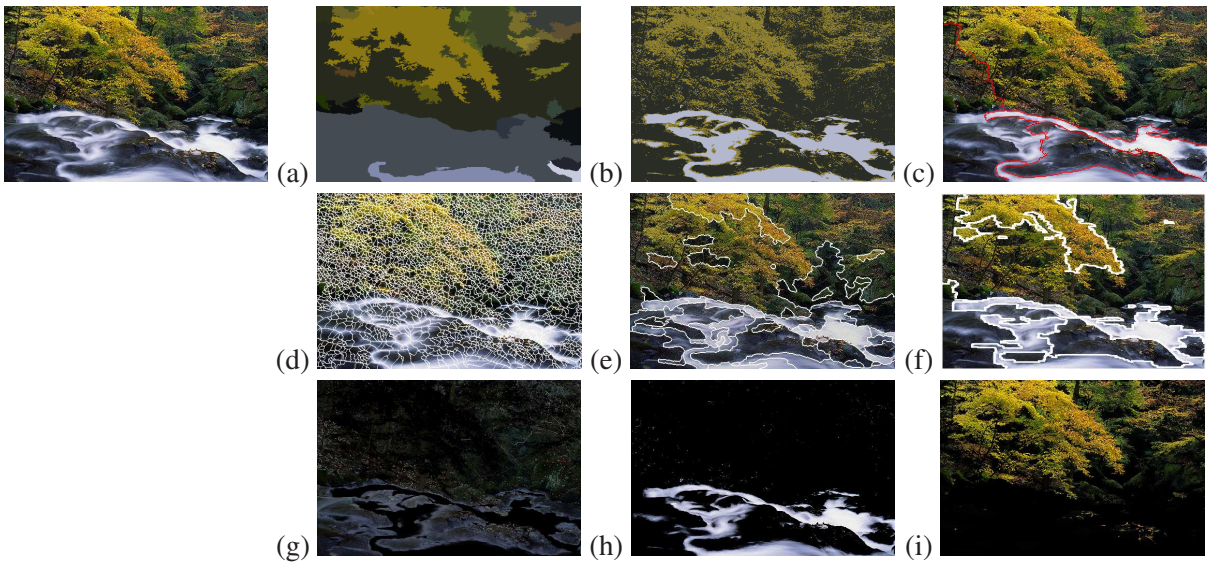
Because our approach is designed to produce soft color segments, soft shading and shadows can be handled uniformly. We compare our method by using the following alternatives:

- standard deviation in modeling $\mathcal{G}$,
- k-means clustering with belief propagation and color estimation.

Figure 2.9 shows that our AO method produces significantly better results. This can be explained by the use of Σ which encodes an oriented Gaussian and captures non-uniform color distribution due to shadow and shading.

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

Figure 2.11: (a) Input image [6]. (b) Segmentation result from [6]. (c)–(e) Our soft color segmentation results $\ell_i c_i$.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

$\ell_1 c_1$ image $\ell_2 c_2$ image $\ell_3 c_3$ image

Figure 2.12: Input image shown on the left. Results produced by (a) Mean shift segmentation, (b) k-means clustering with $k = 3$, (c) Normalized cuts, (d) Watershed algorithm, (e) JSeg method, (f) statistical region merging, (g)–(i): our AO algorithm. The three segmented regions correspond to the three basic color components underlying the image. See the online version for color visualization.

2.6.2 Highly textured scenes

We first compare our AO algorithm with Expectation-Maximization (EM) algorithm, which is one specific form of alternating optimization. Then comparisons are made with other representative segmentation methods.

The **original EM** algorithm does not have the prior probability $\mathrm{P}(\mathcal{L})$ nor color label $\mathcal{C}$ in its objective function, it instead segments an image by considering the color domain only.

Fractional boundaries can be obtained if we do not use the maximum-vote filter such as the one proposed in [6]. Figure 2.10(a)–(c) show the results. Since no spatial information is considered, spatially different patterns, such as the water and the leaves, cannot be well separated.

The **revised EM** algorithm described in [6] performs GMM estimation in the 6D feature space, in which color/texture segmentation is performed. Figure 2.11 shows our results compared with [6]. Because no spatial information is considered during their EM iteration, the maximum-vote filter and the connected component algorithm are used to enforce spatial connectivity. As shown in figure 2.11, although their approach can group relevant region centers, the undecided region boundaries are output as unsegmented regions. Our automatic approach produces fractional boundaries to faithfully maintain the smooth color transition among segments.

For the complex scene shown in figure 2.12, **Mean Shift** [26] cannot segment the river well. The complex region boundaries cannot be preserved, as shown in figure 2.12(a). Because Mean Shift performs segmentation by first concatenating color and spatial coordinates, the resulting sparse and high dimensional feature space makes the segmentation more challenging.

K-means segmentation [32] segments the image by clustering in the RGB space ($k = 3$ in our example). The result is good from the global statistical point of view, but since spatial information is not considered, isolated point clusters result (figure 2.12(b)).

The image segmentation problem is formulated into **Normalized cuts** in [102], where a graph partitioning problem is solved. For this example, the complex region boundaries are not well preserved (figure 2.12(c)).

The **Watershed** algorithm [116] segments the image into many small partitions (figure 2.12(d)), which are suitable for applications requiring an over-segmentation of the scene.

JSeg [29] is an image-based segmentation technique which consists of regions splitting, growing, and merging. The segmentation results are subject to a user-defined color quantization threshold and a region merging threshold, whereas our automatic alternating maximization

algorithm does not have critical threshold to set. The segmentation results produced by JSeg is not satisfactory for this complex scene as shown in figure 2.12(e).

The implicit assumption used in the **statistical region merging** [80] is that the observed color variations inside the same region should be smaller than those across different regions. The use of local statistics and a single scale are not adequate in handling a complex image with multi-scale features. Over-merging and under-merging are therefore possible, as shown in figure 2.12(f).

In our **alternating optimization**, the global color statistics as well as the local spatial coherence are considered. In figure 2.12(g)–(i), the three basic colors of the image are separated. Specifically, figure 2.12(g) shows the dark gray transparent segment which provides a smooth color transition between the leaves and the river. The interweaving green and yellow leaves are reasonably segmented from the complex scene.

Although our soft color segmentation does not necessarily generate a semantic segmentation of the scene, it does produce good synthesis results which are often used to evaluate the segmentation quality. For instance, in the **DDMCMC** method [114], the input is re-synthesized using $p(I|W)$ where W is the segmentation result. In our case, we re-synthesize the input using $p(I|\mathcal{L}, \mathcal{C})$. Some results are shown in figure 2.13. Notice that, however, the goal of DDMCMC method is to produce coherent segmentation for image understanding, while our method optimizes for overlapping and transparent segments to preserve natural and smooth color transition.

2.6.3 Multiscale Processing

Our soft color segmentation can be used to process images at multiple scales. Since we use a GMM to represent colors and a Gaussian kernel to perform prefiltering before subsampling, the scale-space theory [124] asserts that no new edge features will be produced while processing the subsampled data after Gaussian prefiltering.

We propose to use a Gaussian pyramid to reduce dissimilarity of ℓ_i among pixels in a single textured region. The result obtained in a higher level is incorporated as a soft constraint while

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

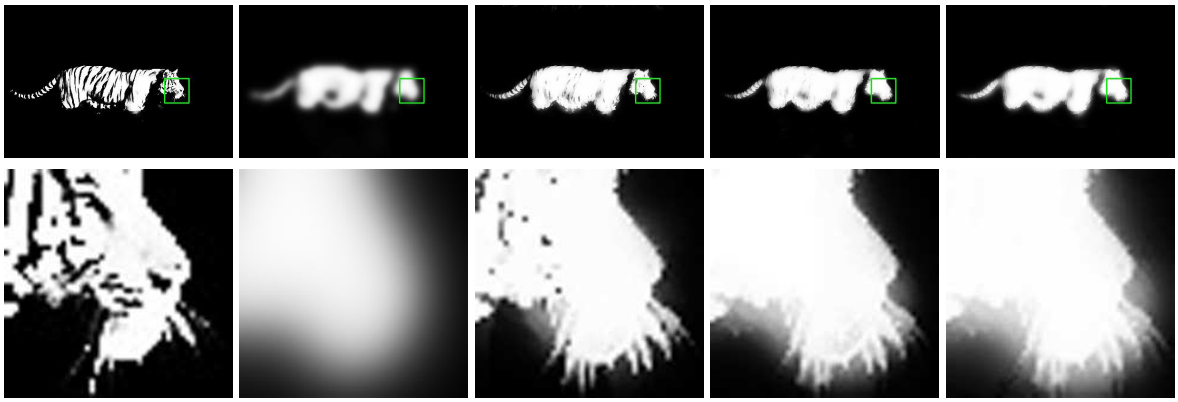
Figure 2.13: Evaluation by re-synthesis. (a) Input images. (b)-(d) Our segmentation results displayed as $\ell_i(x,y)c_i(x,y)$, $1 \leq i \leq 3$. (e) Re-synthesized results of our approach using $p(I|\mathcal{L},\mathcal{C})$. See the online version for color visualization.

optimizing for the result in the immediately lower level. By enforcing the same N (obtained in the lowest level) to all levels, we introduce the following prior term to replace (Eq. 2.10) as follows:

$$\mathrm{P}(\mathcal{L}) \propto \prod_{(x,y)} \exp(-(\frac{\mathcal{L}(x,y) - \mathcal{L}_{ms}(x,y)}{\sigma_{ms}})^2) \prod_{(x,y)} \prod_{(x',y')\in\mathcal{N}(x,y)} \exp(-\psi(\mathcal{L}(x,y), \mathcal{L}(x',y'))) \quad \text{(Eq. 2.24)}$$

where $\mathcal{L}_{ms}(x,y)$ is the result obtained in the immediate higher level in the pyramid, and σ_{ms} controls the similarity between $\mathcal{L}(x,y)$ and $\mathcal{L}_{ms}(x,y)$. Equation (Eq. 2.24) indicates that a small value of σ_{ms} gives more penalty which in turn favors that $\mathcal{L}(x,y)$ and $\mathcal{L}_{ms}(x,y)$ be similar in distribution. Figure 2.14 shows the effect of σ_{ms}. Note in image-based applications, partitioning one texture pattern into exactly one segmented region may not be always desired, so the choice of (Eq. 2.10) or (Eq. 2.24) in modeling the prior $\mathrm{P}(\mathcal{L})$ depends on the applications.

When equation (Eq. 2.24) is used, we set $\sigma_{ms} = 0.5$. Figure 2.15 shows the comparison. The top row compares the result on one example presented by Galun *et al.* [42]. Our result



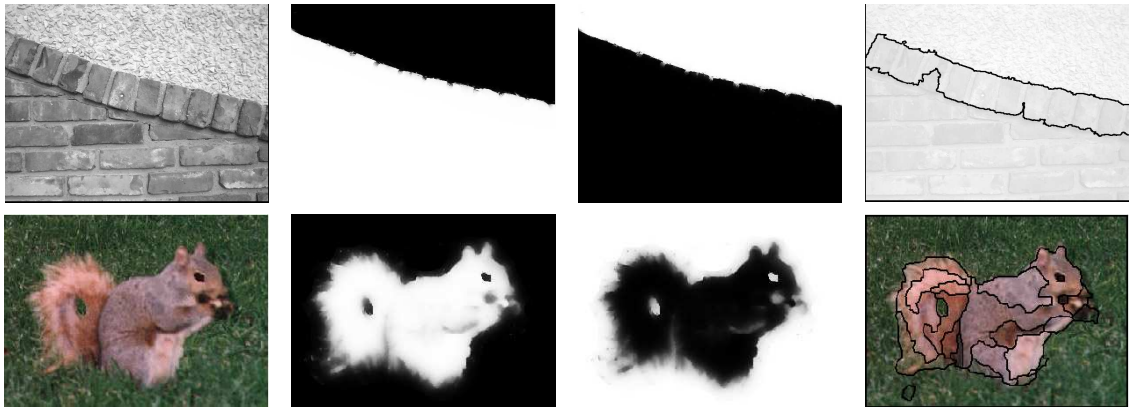
(a) (b) (c) (d) (e)

Figure 2.14: (a) Image of $\ell_1(x, y)$ estimated using (Eq. 2.10) without Gaussian pyramid construction. (b)-(e) The results using a 5-level Gaussian pyramid. (b) $\ell_1(x, y)$ estimated in the highest level. Taking the segmentation result obtained in the immediately higher level as a soft constraint and applying (Eq. 2.24), $\ell_1(x, y)$ is estimated in the original input scale with (c) $\sigma_{ms} = 1$, (d) $\sigma_{ms} = 0.5$ and (e) $\sigma_{ms} = 0.01$. By using smaller σ_{ms}, the output soft labels are more uniform in the textured region.

can successfully segment the two patterns on the wall. In the bottom row of figure 2.15, we compare our result with DDMCMC [114] in texture segmentation.

Using color transfer [95], we can re-color the segments at different scales. In figure 2.16(b), we show $\ell_1(x, y)$ images produced by using (Eq. 2.24). The top row of figure 2.16 compares the result on one example presented in Galun *et al.* [42] where the camouflage patterns are progressively segmented. In the bottom row, we compare our result using an input image from Sharon *et al.* [101]. Our method produces good results where the water is separated from the tiger while the stripped textures are maintained. We re-color the resulting segments using a different color (using small scale of analysis) or re-color them using the same color (using large scale of analysis).

In figure 2.17, we show one instance that our multiple scale segments produced are consistent among each other, allowing the user to specify and focus on regions of interest. For example, using the multiscale soft segments, we can re-color all the dining facilities along the waterfront using a single color (figure 2.17(b)), or re-color the individual tables and chairs (figure 2.17(c)), while the faraway mountain and windmills are re-colored using a small scale in



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

Figure 2.15: We compare our multiscale results with the those in Galun *et al.* [42] (top) and DDM-CMC [114] (bottom). Using the multiscale prior (Eq. 2.24), our method automatically converges to two soft segments, that is, ℓ_1 and ℓ_2. (a) Input images, (b) ℓ_1 images, (c) ℓ_2 images. Our multiscale method deals with textures and produces soft segments with appropriate boundary transparency and spatial coherence. The results from [42, 114] are shown in (d). Note that our method works in RGB and does not work better in intensity images with one grayscale channel.

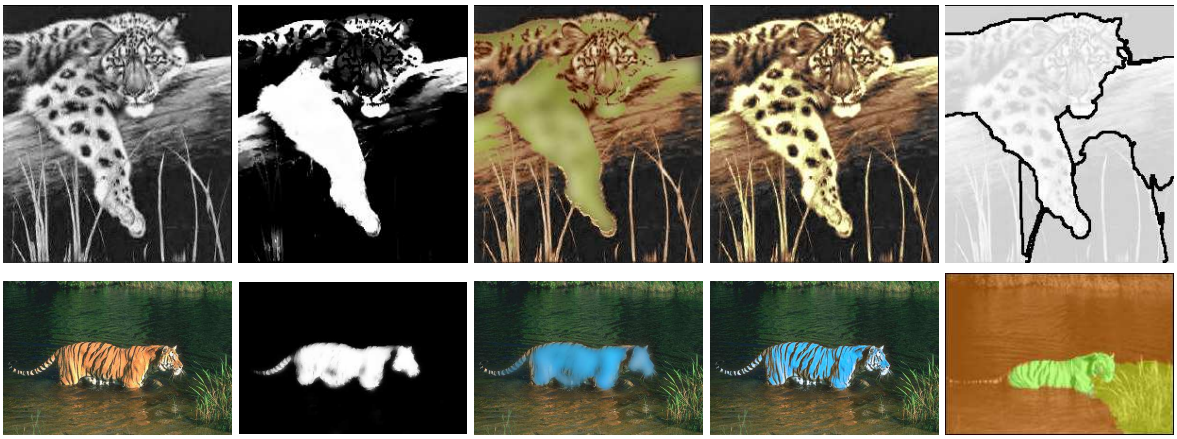
both cases.

2.7 Applications

In this section, we present the results and applications of soft color segmentation. Our approach provides a general framework for the following applications which are traditionally addressed by separate algorithms. Without any human interaction, the results produced by our method are more reasonable or comparable to previous methods, where user assistance may be required.

2.7.1 Soft color segmentation

Hurricane images Soft color segmentation can be applied to process satellite images of hurricane, where the specification of a trimap for image matting is difficult in complex images such as the input shown in figure 2.18(a). Hurricane segmentation from satellite images is helpful in identifying, analyzing and predicting the formation of hurricanes. Hurricanes are non-solid and partially transparent. Figure 2.18(b)–(d) show the results produced by our soft



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

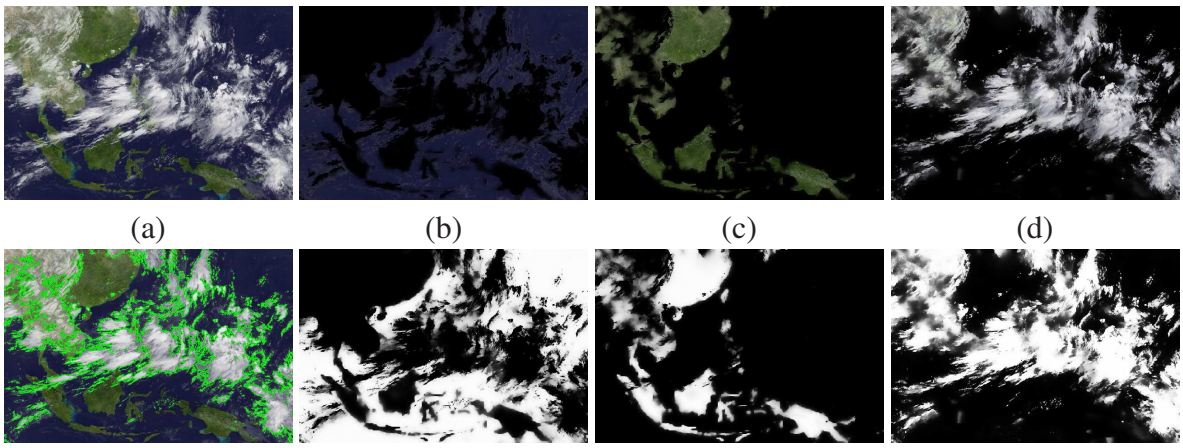
Figure 2.16: To show that our approach is capable of producing scene segmentation at multiple scales, we process (by re-coloring) the selected segment at different scale. These two examples are respectively presented in Galun *et al.* [42] (top) and Sharon *et al.* [101] (bottom). The selected segments are the leg of the leopard and the torso of the tiger, respectively. (a) Input image. (b) $\ell_1(x, y)$ estimated with scale prior (Eq. 2.24). Re-colored results using (c) large scale and (d) small scale. The segmentation results from [42] and [101] are shown in (e).

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

Figure 2.17: Consistency of multiple scale segments. (a) Input grayscale image. To show that our approach is capable of producing soft segmentation at multiple scales, we re-color the image using (b) large scale and (c) small scale at the chosen regions or scales of interest (the dining facilities along the waterfront). The result from [114] is shown in (d).

color segmentation. Note that the inferred ℓ values for the hurricane are proportional to the cloud density. Figure 2.18(e) shows an unsatisfactory segmentation result produced by hard segmentation.

Nebulas In nebula analysis, different colors of a nebula represent different components and temperatures of the nebula. Figure 2.19(a) shows a nebula image, Messier Object M20, captured by the Hubble Space Telescope. The red emission nebula with its young stars clustered near its center is surrounded by a blue reflection nebula, where different components and tem-



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

(e) (f) ocean (g) land (h) hurricane

Figure 2.18: Segmentation of a satellite image of a hurricane. (a) Input image. (b)–(d) Our segmentation results displayed as $\ell_i(x,y)c_i(x,y)$, $1 \leq i \leq 3$. (e) Hard segmentation result. (f)–(h) The corresponding ℓ_i images for (b)–(d). Our approach segments the land from the hurricane. The ℓ image in (h) indicates the cloud density of the hurricane. See the online version for color visualization.

peratures exist across the nebula. Similar to hurricanes, the nebula is not a solid object and the transition boundary between the red and blue nebulas should be soft and smooth. Results obtained by hard segmentation cannot faithfully reflect this phenomenon. Figure 2.19(b)–(d) show our soft color segmentation results in which smooth and natural transition across the nebulas are achieved.

2.7.2 Image matting

Image matting, in particular Bayesian matting [21], can be regarded as a user-assisted form of soft color segmentation for the specific case $N = 2$, if $\mathcal{L} = \{\alpha, 1-\alpha\}$ where $0 \leq \alpha \leq 1$, $\mathcal{C} = \{F, B\}$: where F and B are the respective optimal foreground and background colors, when $\mathcal{G}$ is restricted in a local neighborhood governed by a user-supplied trimap.

Figure 2.20 shows a result on natural image matting. Our result is comparable to the results obtained by Bayesian matting [21], while our automatic approach does not use any trimap. As depicted in our result, the fences of the lighthouse are not smoothed out. Figure 2.21 shows

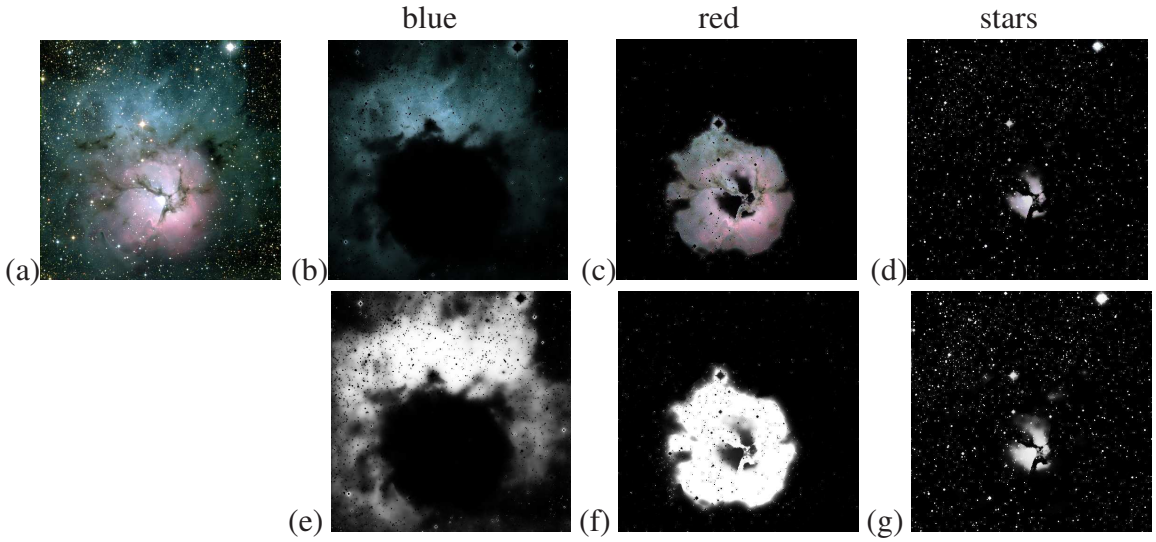


Figure 2.19: Segmentation of a nebula image. (a) Input image, Messier Object M20. (e)–(g) The soft labels $\ell_i(x, y)$ corresponding to (b)–(d). Our algorithm segments the red and blue nebulas with fractional boundaries. Note that the bright stars are not smoothed out due to the discontinuity-preserving property of our MRF formulation. See the online version for color visualization.

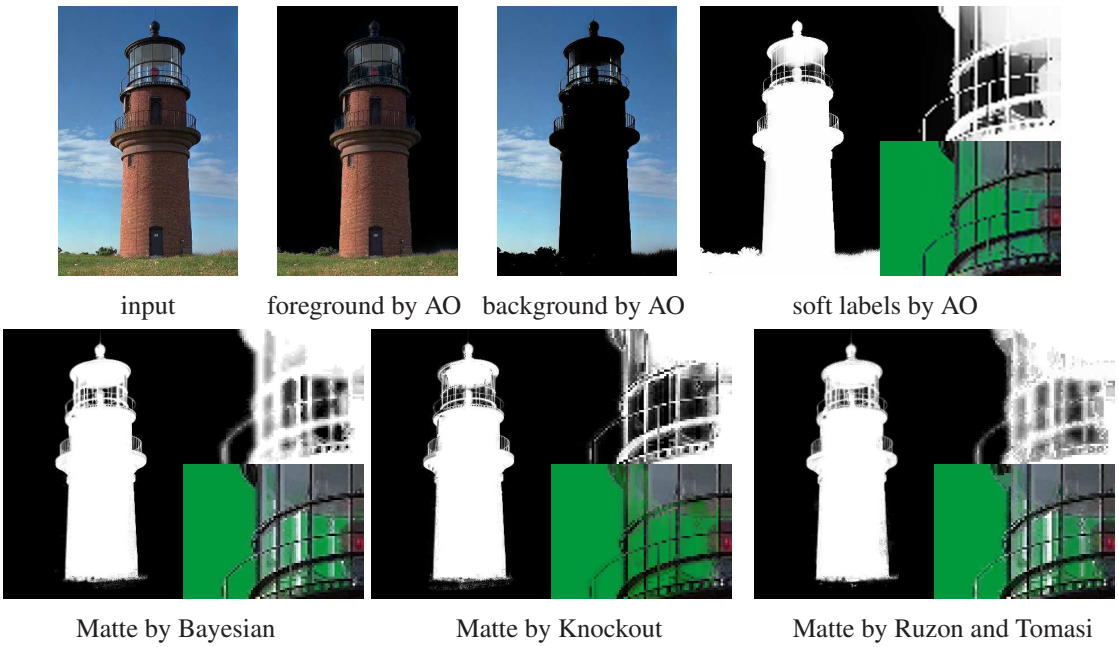


Figure 2.20: Comparison with image matting. While approaches in natural image matting use a user-specified trimap or other user-supplied hints, our method is fully automatic. See the online version for color visualization.

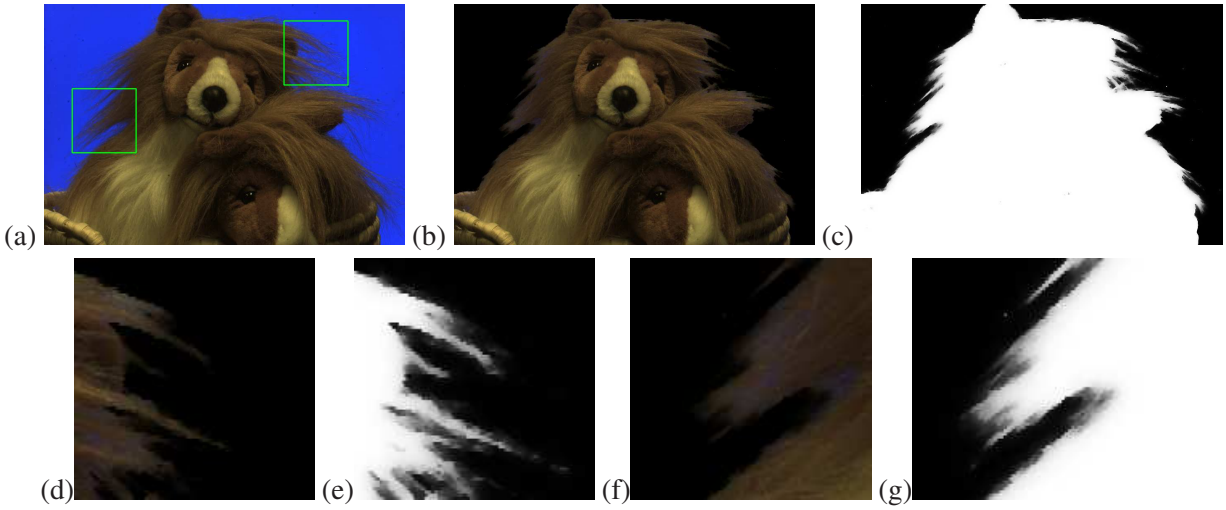


Figure 2.21: Boundary smoothness and transparency for an object with long hairs. (a) Input image. (b) One automatically segmented region. (c) ℓ image of the region (or alpha matte). (d)–(g) zoom-in views of the results obtained using our method. Because no trimap is used, long and thin hairs are missed while short hairs are still preserved. See the online version for color visualization.

an example in which long hairs are present. Our segmentation result is satisfactory for the example except for the long hairs because no trimap is used.

2.7.3 Color transfer between images

A single Gaussian is used in [95] to model the global color statistics of the source and target images. If diversified colors are present in any of inputs, image patches must be manually specified to divide the colors into separate clusters. However, for complex images, it is difficult for user to specify the right patches, and a small number of patches is inadequate to discern different color statistics. Here we propose to perform soft color segmentation on both the source and target images before applying color transfer so that the transfer process is fully automatic. Recall that our soft color segmentation models the global color distribution by a GMM, that is, each color region (not necessarily connected) corresponds to a component of

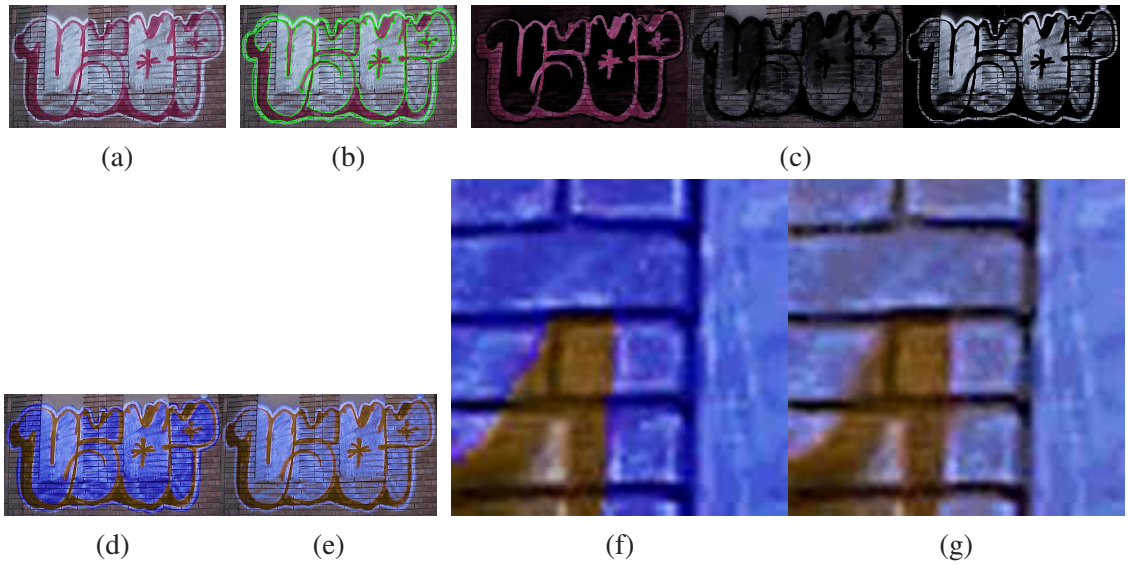


Figure 2.22: (a) The graffiti image. (b) Hard segmentation manually produced, where the region boundaries are indicated by the solid green curves. (c) Soft segmentation produced by our AO algorithm. The three converged color regions are shown. (d) Color transfer result using the hard segmentation shown in (b). (e) Color transfer result using the soft segmentation shown in (c). Zoom-in views of the result for (f) hard segmentation and (g) soft segmentation. The color transfer result in (g) is smoother and more natural. The color transfer equation in [95] is used to generate (d)–(g). See the online version for color visualization.

the converged GMM. We therefore define the final transferred color for a pixel $g(I_T(x,y))$ by:

$$g(I_T(x,y)) = \sum_j \ell_{T_j}(x,y)(\frac{\sigma_{S_i}}{\sigma_{T_j}}(I_T(x,y) - \mu_{T_j}) + \mu_{S_i}) \qquad \text{(Eq. 2.25)}$$

where $\ell_{T_j}(x,y)$ is the soft label corresponding to the T_j-th Gaussian component of the GMM of the target image, obtained by our AO algorithm for soft color segmentation. The following examples show that our GMM model is more suitable than a single Gaussian, used in [95], in guiding the color transfer process. Our natural color transfer with soft color segmentation achieves smoother and more natural color transition, especially for regions rich in colors and textures.

Figure 2.22 first demonstrates color transfer on a complex scene based on soft color segmentation, which is more preferable in comparison to the one based on hard/binary segmentation. The transferred result using soft color regions (figure 2.22(g)) looks smoother and more

natural, as opposed to that given by hard segmentation where unnatural and abrupt changes in color are observed among adjacent color patches (figure 2.22(f)).

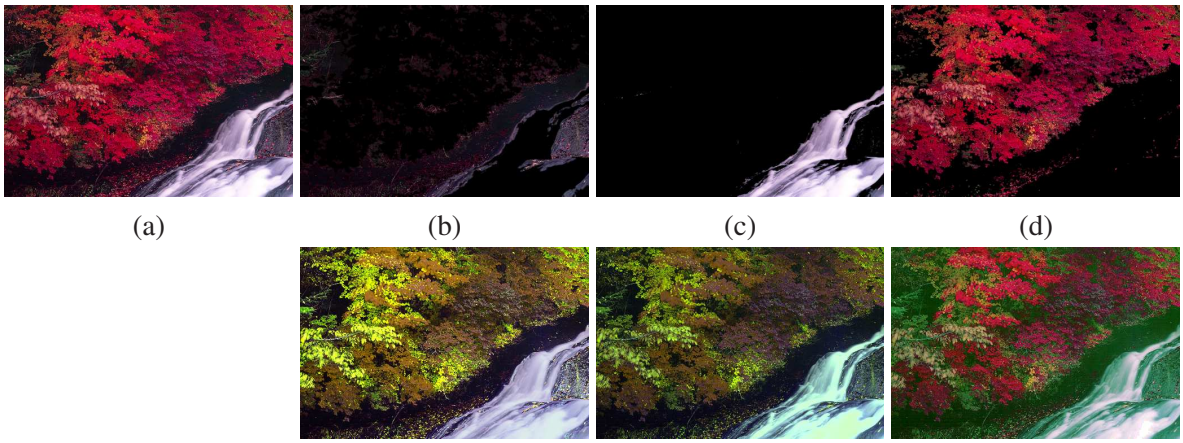
Figure 2.23 and figure 2.24 compare the color transfer results using a GMM model obtained by our AO algorithm as opposed to using a single Gaussian model [95] for guiding the color transfer process. For figure 2.23, the source image and its soft segments were already shown in figure 2.12(g)–(i). The soft segments for the target segments are shown in figure 2.23(b)–(d). Our transfer result is shown in figure 2.23(e). Using the GMM model and soft color segments to guide the color transfer process, the color of the rivulet (figure 2.12) is faithfully transferred to the river (figure 2.23). In comparison with [95], using a single Gaussian, which does not adequately model the global color distribution of the image, as depicted in figure 2.23(f) and (g). The transfer results are unsatisfactory as undesirable mixture of colors of the leaves and the river is easily observed.

Figure 2.24(a) and (b) show another source/image pair. Figure 2.24(c) is the transfer result generated using our approach. Comparing with the respective results generated by [95] in figure 2.24(d) and histogram equalization in figure 2.24(e), our result is more natural and suffers less saturation.

2.7.4 Image correction using image pairs

We have extended the idea of color transfer using image pairs to the task of image intensities corrections:

Image deblurring using normal/low exposure pair Given two images of the same scene taken almost simultaneously and without a tripod: 1) one is acquired under normal exposure and so motion and shape blur may be present, 2) the other is taken with a short shuttle speed and so the image is crisp but under-exposed, we want to transfer the color from image 1 to image 2 so as to generate a bright and crisp image. Since the source and the target images have strong spatial similarity, after segmenting the respective images using our AO approach, we



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

(e) (f) (g)

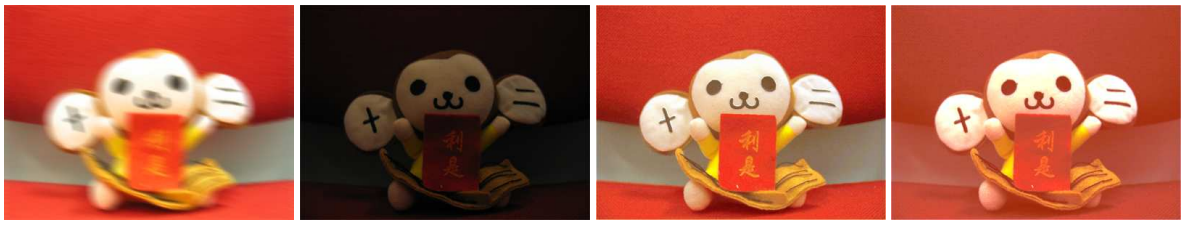
Figure 2.23: Comparison of color transfer using our approach and [95]. (a) The target image (the source image is shown in figure 2.12(a)). (b)–(d) Soft color segmentation results. (e) Color transfer result using our approach in which soft color segmentation is performed before transferring the colors. (f) Color transfer result without soft color segmentation [95]. (g) Color transfer result by histogram equalization. The results in (f) and (g) show undesirable mixture of colors of the leaves and the river. See the online version for color visualization.

can use, as the criterion, the largest overlapping area of the segmented regions for matching the Gaussians, so as to perform the associated color transfer from the source to target to deblur the image. Figure 2.25 shows our result of image deblurring.

Image denoising using flash/no flash pair Similarly, we can perform image denoising using two images: one is taken with camera flash (flashed image) and the other is captured using a high ISO configuration without flashing (non-flashed image) to preserve the original scene color and ambiance. Images captured with a high ISO setting contain a considerable amount of noise. In [86], the flashed image is used to denoise the non-flashed image. In our method, we first use a median filter to reduce the amount of noise. Then, both the source and target images are segmented using our AO algorithm. Finally, we map the colors from the non-flashed image to the flashed image to construct our denoised and sharp image which faithfully preserves the original scene ambiance. Figure 2.26 shows and compares the denoised result. Our current result does not transfer shadows though, which on the other hand can be performed as described in [36].

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

Figure 2.24: Comparison of color transfer on a natural scene using our approach and [95]. (a) An old photograph of a downtown scene captured on an overcast day. (b) The target image captured on a sunny day. (c) Our transfer result. (d) Transfer result generated by [95] where unacceptable mixture of colors are present. (e) Transfer result generated by histogram equalization in which undesirable saturation is observed. See the online version for color visualization.



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

Figure 2.25: Image deblurring using color transfer with/without soft color segmentation. (a) Source image. (b) Target image. (c) Result using our approach. (d) Result using a single Gaussian model. Note that the colors in result (d) are not preserved. See the online version for color visualization.

2.7.5 Colorization

For applications in image colorization, we only have the luminance channel in an input grayscale image. To constrain the color transfer, we assume that two pixels in the same region have similar colors if they have similar luminance value. Similar to the above transfer applications, we perform soft color segmentation in both the source and target images. Here, the only modification of our method is that we perform alternating optimization only on the luminance ℓ channel and assign the same distribution to the absent ab channels. Figure 2.27 shows one result. Note in our result the smooth and natural transition between blue sky and green trees. The whole process is fully automatic.

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

Figure 2.26: Comparison on image denoising. (a) Non-flashed source and flashed target images. Results and zoom-in views obtained by (b) our approach, (c) [86], and (d) [36]. Our method makes use of the strong spatial relationship between the source and target images given by the soft color segmentation. Therefore, the red shade of the sofa, the bottles and stones are not mixed up. Such undesirable mixture is observed in the result in (d). The result using joint bidirectional filter [86] in (c) is still very noisy. See the online version for color visualization.

2.8 Summary

We have described an algorithm based on a Bayesian optimization framework to address the problem of soft color segmentation. An alternating optimization (AO) procedure is used to

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

Figure 2.27: Color transfer to a gray scale image. (a) The source image. (b) The target image. (c) Result by our approach. (d) Result by [122]. See the online version for color visualization.

simultaneously optimize both global and local parameters in the global objective function. We have demonstrated that our method produces reasonable segmentation in the form of overlapping and transparent color regions, despite the presence of rich colors, textures, shading, soft shadows and image noises. Our soft color segmentation can be applied at multiple scales. This general approach has found useful applications such as image matting, color transfer and image correction. Our method combines the advantages of global color statistics and local image compositing, while preserving the necessary spatial and color coherence.

Chapter 3

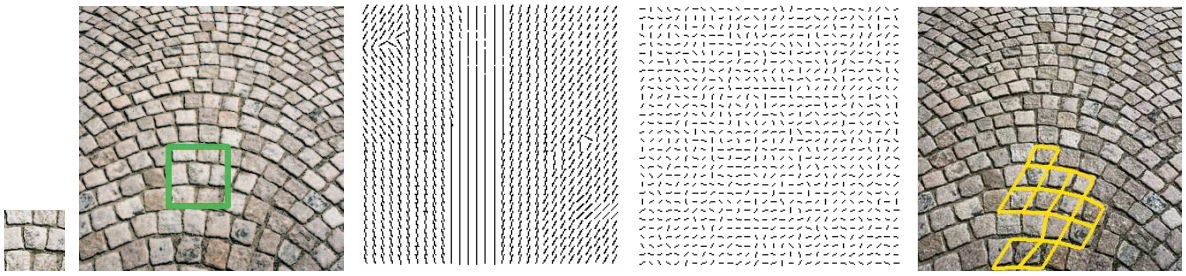
Texture Flow Estimation

3.1 Overview

Texture flow estimation is a valuable step in a variety of vision related tasks, including texture analysis, image segmentation, shape-from-texture and texture remapping. This chapter describes a novel and effective technique, based on a Bayesian optimization framework, to estimate texture flow in an image given a small example patch. The key idea consists of extracting a dense set of features from the example patch, where discrete orientations are encapsulated into the feature vector, such that rotation can be simulated as a linear shift of the vector. This dense feature space is then compressed by PCA and clustered using EM to produce a set of small set of principal features. Obtaining these principal features at varying image scales, we can compute the per-pixel scale and orientation likelihoods for the distorted texture. Combined with neighborhood smoothness prior, the final texture flow estimation is formulated as the MAP solution of a labeling Markov network, which is solved using belief propagation. Experimental results on both synthetic and real images demonstrate good results even for highly distorted examples.

3.2 Background and Motivation

Texture flow estimation serves as the starting point of many common vision tasks including segmentation, recognition, shape-from-texture, and texture remapping. Texture flow estimation



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

Figure 3.1: An input image and its texture flow estimation. (a) The user specified example patch. (b) The input image. (c) The estimated texture flow field. Results of flow-extraction from recent proposed techniques :(d) Paris *et al.* [82], and (e) Hays *et al.* [48].

is often targeted towards images of textured 3D surfaces in a natural scene [70, 74], where texture flow arises due to the geometric variation of non-planar surfaces or imaging under perspective. It is assumed that the underlying texture has some repeating structure [117] that is varying in the image (i.e. distorted) in terms of scale and orientation. We called this distortion *texture flow*. In this chapter, we address texture flow estimation for a variety of texture types that exhibit reasonably strong distortion in both scale and orientation. These include textures found in natural images as well as synthetically generated images. Figure 3.1 shows an example.

Our texture flow estimation begins with a small example patch of the texture that is specified by the user. From this example patch, a set of principal features are extracted that are used to compute the orientation and scale likelihoods of each pixel lying in a distorted texture region. Using these per-pixel likelihoods, we formulate the final texture flow estimation as a Markov Random Field (MRF) and solve it using a variant of belief propagation. We demonstrate the effectiveness of this approach on a variety of inputs, including natural and synthesized images and show the usefulness of this extracted flow field for texture remapping.

There are two main contributions of this work. First, while inspired by previous approaches, a novel texture feature, along with a procedure for extracting principal features from the sample patch, is introduced. These principal features are suitable for estimating the orientation and scale of texture pixels while making no assumptions about the underlying texture properties.

Second, the texture flow estimation problem has been formulated in Markov network, which can resolve orientation and scale to recover reasonably complex flow fields. The texture feature extraction and MRF formulation are both contributions in themselves, as the feature extraction approach can be beneficial to existing texture analysis and segmentation algorithms, as well as our MRF formulation, which can be used with other feature extraction methods to label texture flows.

3.3 Related Work

There is a great deal of work targeting texture analysis and flow estimation. We discuss examples most relevant to our approach and refer the reader to the following recent articles by Lazebnik *et al.* [62], Ben-Shahar and Zucker [9] and Lefebvre and Hoppe [63] for more thorough overviews.

In the context of flow orientation, typically for segmentation, early work by Rao *et al.* [92, 91] proposed orientation fields estimation of gradient-like texture using first derivative Gaussian filters and performed local averaging to compute coherent orientation flows. Following work on scale-space and nonlinear diffusion, Perona [85] proposed orientation diffusion which considered flow directions in the flow smoothing step to improve estimation results. In [113], Tschumperle *et al.* proposed using vector set regularization with anisotropic diffusion. Shahar *et al.* [9] proposed to incorporate curvature information and enhance global continuation of estimated texture flows. Paris *et al.* [82] use band-pass filtering to enhance flow data before estimation.

In a shape-from-texture context, the shape recovery process is generally broken into two independent steps: 1) texture flow estimation and 2) shape recovery using the estimated flow. In [74], Malik and Rosenholtz modeled texture distortion by a dense set of 2D affine transformations and estimated these transformations by spectrograms matched in the frequency domain, while Clerc and Mallat [24] showed how wavelets can be employed for this purpose.

In [41], Forsyth modeled surface texture via a marked point process and estimated the affine transformations on these points. An EM-like procedure is introduced to reconstruct global surface from the sparse texture distortion map. This idea is extended by Lobay and Forsyth in [70] to handle more complex textons. Lazebnik *et al.* [62] introduces RIFT to models texture distortion on surface by sparse local affine regions. Hays *et al.* [48] introduced a technique to detect texture regularities in natural scenes using higher-order correspondence.

In the context of texture classification, research has been focused on developing rotation and scale invariant classifiers. In [57], Kashyap and Khotanzad developed a circular simultaneous autoregressive model for the extraction of rotation invariant texture features. Cohen *et al.* [25] and Chen *et al.* [19] obtain rotation and scale invariance by training the HMM on the texture samples from a wide variety of angles. Wu *et al.* [126] proposed spiral resampling and sub-band decomposition to learn the rotation classifier. In Gabor filtering (or other filtering techniques), rotation invariance is realized by computing rotation invariant features from the filtered images or by converting rotation variant features to rotation invariant features. Representative works using filtering techniques include [14, 33, 121, 88, 90, 81, 51].

In the context of texture synthesis, several techniques incorporate texture flow to guide the synthesis output (e.g. see [35, 133, 60, 63]). Specifying "this guidance flow field" however is often done manually: such as demonstrated by Ashikhmin [2] and Liu *et al.* [69] that allow texture flow to be specified using a paint-like interface or by manipulating a mesh grid.

Distinguishing our work from these previous approaches, we note that the flow estimation for segmentation focuses mainly on gradient like textures and it utilizes high frequency details for estimation. These approaches do not consider overall structure of the underlying texture and typically cannot handle large variations in scale. For the shape-from-texture approaches, textures are assumed to be lying on smooth 3D surfaces and the estimated distortion map is generally not very complex. Also, the underlying texture for shape-from-texture is usually assumed to be an isotropic texture. In texture synthesis, the texture flow is often specified by

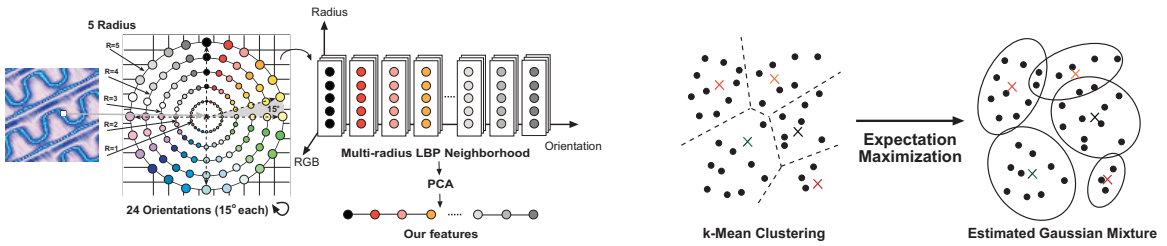


Figure 3.2: This figure overviews the feature extraction process from the example patch. Initial features are extracted about each pixel. The entire feature space (of all pixel features) is then decomposed using PCA and each feature is projected onto the first principal component of the PCA decomposition. A small set of *principal features* are extracted by first running k-means clustering on the per-pixel features and using the cluster means as the initializations of principal features. The feature space is then modeled using a gaussian mixture model (GMM) and the principle features are taken to be the means and covariance matrices of k gaussians refined via EM.

the user. While the texture flow has been proven to be effective for generating better quality texture synthesis results, there is not much attention on estimating the texture flow from natural scene or from a synthesized texture. In our work, we desire to extract flow from a variety of texture types, and are make no assumptions about the texture's structure. Moreover, while we work from an example patch, we do not assume that the target distortion is a matter of piecewise affine transforms of this given patch.

3.4 Texture Features

3.4.1 Feature Representation

Given an example patch, either extracted directly from the input image, or from an example known to be sufficiently similar the target image, a set of features (the term descriptors can also be used) are derived that will be employed to estimate orientation and scale. Our feature extraction is inspired by the local binary pattern (LBP) operator proposed by Ojala *et al.* [81] for segmentation. In this approach, a feature is constructed by collecting thresholded pixels at varying orientations at a fixed radius about a pixel which are organized into a linear array. The

benefit of this feature construction is that rotation can be simulated by shifting the linear array, thus allowing this single feature to estimate multiple orientations.

This basic idea is extended in the following manner as shown in figure 3.2. RGB pixel data is collected at 24 orientations at 5 discrete radii about each pixel in the sample patch. This results in a $24 \times 5 \times 3$ dimensioned feature *per pixel*. Assuming a reasonably sized sample patch, e.g. 36×36, this results in approximately 500000 intensity values to describe the input patch. Such dense features are too large for practical use and the following steps are taken to reduce the feature size.

First, the $24 \times 5 \times 3$ per pixel feature space sampled over the entire patch is decomposed via PCA. Each feature is then projected onto the first principal component, resulting in a single 24 dimensioned feature per pixel. This PCA projection captures the salient information from the multiple radii and multiple color channels while maintaining the linear-shift property.

3.4.2 Principal Features Extraction

The per-pixel features can further be reduced to a set of so called *principal features*. This is achieved by k-means clustering followed by expectation maximization (EM) as shown in figure 3.2. Each pixel feature is considered to be a point in high dimensional space and the distribution of these points is assumed to follow a Gaussian Mixture Model (GMM) consisting of k gaussians. The extracted *principal features* are taken to be the estimated gaussian means of the GMM. Similar procedures have been used in other work (e.g. [70, 73]) to group redundant training features.

To estimate the k means of the GMM, we first run k-means clustering on the PCA pixel features. The resulting cluster centers are used as initializers for the Gaussian means. We also determine the number of Gaussians during k-means clustering. After EM converges, principal features having very small weights assigned to their corresponding Gaussian (implying that these means model only a few instances in the sample space) are removed. Typically, after the

EM procedure, we are left with only 30 to 40 principal features that represent the entire sample texture patch. Thus, using PCA projection and EM, we have reduced our initial feature space from more than 500000 to less than 1000.

To handle scale, the principal feature extraction procedure is performed at multiple scales of the input patch. For our implementation, we use eight scales, from 0.25 to 2. The feature sampling and reduction procedure mentioned above is applied for the sample patch at the various scales.

3.5 MRF Formulation

The texture flow field estimation is formulated as an energy minimization problem with a well-defined objective function for a Markov network. The global objective function is described first, followed by the description of the likelihood and prior terms for the MRF.

3.5.1 Global Objective Function

Given the principal features extracted from the example texture patch, the goal is to estimate the texture flow in an input image with the same texture. This problem can be formulated as a discrete label assignment problem where we wish to assign an orientation label, $\mathcal{O}$, and a scale label, $\mathcal{S}$, to every pixel in the distorted texture image. We assume that our label assignment process follows the MRF property, which has been demonstrated to be effective at overcoming noise and correcting errors in several vision-related problems (see recent examples [38, 58, 106]). For our problem, a Markov network is constructed where each node $\{n_i\}_{i=1}^{N}$ (N is the number of nodes) corresponds to a pixel in the distorted texture, and for which a four-neighborhood system is defined to have edges, $\mathcal{E}$, connecting each node. Under the MRF

configuration, our global objective function is defined in the standard form:

$$E(\mathcal{L}) = \max_{\mathcal{L}} \prod_{i=1}^{N} \exp(-V_i(l_i)) \prod_{(i,j)\in\mathcal{E}} \exp(-V_{ij}(l_i, l_j))$$
$$= \min_{\mathcal{L}} \left(\sum_{i=1}^{N} V_i(l_i) + \sum_{(i,j)\in\mathcal{E}} V_{ij}(l_i, l_j) \right), \qquad \text{(Eq. 3.1)}$$

where $l \in \mathcal{O} \times \mathcal{S}$ is the set of all possible labels, $\mathcal{L} = \{l_i\}_{i=1}^{N}$ is a configuration with label l_i assigning to node n_i, $V_i(l_i)$ is the data cost function for assigning label l_i to the node n_i, $V_{ij}(l_i, l_j)$ is the pairwise potential function for assigning labels l_i and l_j to the pair of neighbor nodes n_i and n_j in the Markov network. The goal is to find a configuration $\mathcal{L}$ such that the global objective function is optimized. In a Bayesian MRF formulation, the configuration $\mathcal{L}$ corresponds to the maximum a posteriori (MAP) solution, where $\prod_{i=1}^{N} \exp(-V_i(l_i))$ corresponds to the likelihood and $\prod_{(i,j)\in\mathcal{E}} \exp(-V_{ij}(l_i, l_j))$ corresponds to the prior. While the definition of the energy function is standard for a Markov network, the definitions of the likelihood and the prior costs are unique for each different problem. Our likelihood estimation and prior term are detailed in the following subsections, however, optimizing the MRF is described first.

To minimize the MRF cost function, recent state-of-the-art MRF-based approaches that optimized their objective functions using either belief propagation (BP) ([38, 106]) or via graph-cut [58] were considered. In our implementation, orientation labels are defined at each 15° for eight different scale labels linearly from 0.25 to 2.0, resulting in in a total of $|\mathcal{O}| \times |\mathcal{S}| = 192$ labels. Our estimation results can be further improved by using even finer quantization in orientation and scale, at a trade-off of higher computational and memory costs. Using our current quantization, the number of labels for our problem is higher than those problems addressed in [38, 58, 106] and as a result, we elected to use Priority-BP [59], a variant of belief propagation, for our label assignment process.

The significant difference between BP and Priority-BP is the use of a dynamic label pruning procedure and a priority message passing scheme to help resolve the huge memory requirements in storing all possible label assignments at each node. As described in [59], Priority-BP

can tolerate label assignments with thousands of labels. While our problem does not have thousands of labels, the number is sufficiently high to warrant the use of Priority-BP. Using Priority-BP we find that we only need to store 20 to 30 labels per node for each updating iteration, while still obtaining good results.

3.5.2 Likelihood

Given the set of principal features generated from the sample texture computed at a resolution of 24 orientations and 8 different scales, denoted as labels T, where $\mathcal{T} = \{\mathcal{T}_l\}_{l=1}^{|\mathcal{O}|\times|\mathcal{S}|}$, we measure the likelihood of texture feature t_i extracted from a pixel position at n_i in input image is to be generated from each label, l, where $l \in |\mathcal{O}| \times |\mathcal{S}|$, is represented by the features in $\mathcal{T}_l$. Since each $\mathcal{T}_l$ is assumed to follow a GMM distribution, and with the assumption that observation noise follows an independent identical distribution, we define our likelihood as follows:

$$\begin{aligned} P(\{t_i\}_{i=1}^N \in \mathcal{T}|\mathcal{L}) &= \prod_{i=1}^{N} P(t_i \in \mathcal{T}_{l_i}|\mathcal{L}) \\ &= \prod_{i=1}^{N} \exp(-(t_i - \mu_{\mathcal{T}_{l_i}^k})^T \Sigma_{\mathcal{T}_{l_i}^k}^{-1}(t_i - \mu_{\mathcal{T}_{l_i}^k})) \\ &= \prod_{i=1}^{N} \exp(-V_i(l_i)) \end{aligned} \tag{Eq. 3.2}$$

where N is the number of nodes in the MRF, and $(\mu_{\mathcal{T}_{l_i}^k}, \Sigma_{\mathcal{T}_{l_i}^k})$ are the normalized mean and the covariance matrix of the k-th Gaussian in $\mathcal{T}_{l_i}$. This k-th Gaussian is selected as the principal feature from $\mathcal{T}_l$ that is most similar to t_i, i.e. $k = \arg\min_k ||t_i - T_l^k||_2$, and is denoted as $\mathcal{T}_{l_i}^k$. Although we can use the weighted sum of errors of all the Gaussian in $\mathcal{T}_{l_i}$, we find that using the most similar principal feature for each label gives the best results.

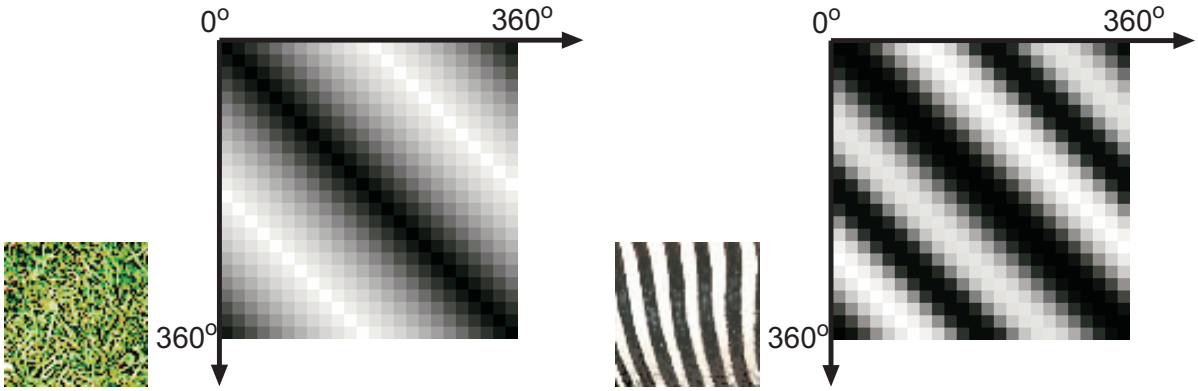


Figure 3.3: Compatibility matrix($W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) f(|l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|)$) for different texture. The darker regions in the plots represent where the texture is more similar for pairs of orientations. (a) a grass texture, (b) compatibility matrix of grass texture, (c) zebra texture, (d) compatibility matrix for zebra texture. In the grass texture, there is no rotation symmetry, while in the zebra texture, rotation symmetric exist at $180°$.

3.5.3 Prior

The Markov property asserts that the conditional probability of a site in the Markov network depends only on its neighboring sites, which means that our prior is defined over each pair of neighbor nodes in the Markov network. We adopt a smoothness assumption for our prior, which assumes the changes of both orientation and scale are small across neighboring sites. To avoid smoothing discontinuities present in the flow field a truncated linear cost model in our prior term is used as discussed in the following.

Recall that our label $l \in \mathcal{O} \times \mathcal{S}$ consists of orientation $\mathcal{O}$ and scale $\mathcal{S}$ components. Since there exists no statistical relationship between the scale and orientation, we assume the orientation prior $P(\mathcal{O})$ and the scale prior $P(\mathcal{S})$ are independent. Hence, our prior $P(\mathcal{L})$ can be expanded into the joint probability of $P(\mathcal{O})$ and $P(\mathcal{S})$:

$$
\begin{aligned}
P(\mathcal{L}) &= P(\mathcal{O})P(\mathcal{S}) \\
&= \prod_{(i,j)\in\mathcal{E}} \exp(-V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})) \prod_{(i,j)\in\mathcal{E}} \exp(-V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}})) \\
&= \prod_{(i,j)\in\mathcal{E}} \exp(-(V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}}) + V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}}))) \\
&= \prod_{(i,j)\in\mathcal{E}} \exp(-V_{ij}(l_i, l_j)) \qquad \text{(Eq. 3.3)}
\end{aligned}
$$

where $l_i^{\mathcal{O}} \in \mathcal{O}$ and $l_i^{\mathcal{S}} \in \mathcal{S}$ denote respectively the orientation and scale component of l_i, $V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})$ and $V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}})$ are the pairwise potential functions of $P(\mathcal{O})$ and $P(\mathcal{S})$ respectively.

We first define $V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}})$. Since our scale labels $\mathcal{S}$ are defined linearly, we use a simple truncated linear model [38] for our cost function as follows:

$$
V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}}) = \min(c_{\mathcal{S}}|l_i^{\mathcal{S}} - l_j^{\mathcal{S}}|, d_{\mathcal{S}}), \qquad \text{(Eq. 3.4)}
$$

where $c_{\mathcal{S}}$ is the rate of the cost, and $d_{\mathcal{S}}$ controls when to terminate increasing costs. This model is desirable because it allows discontinuities in the labeled MRF by ceasing to increase costs after the scale label difference, $c_{\mathcal{S}}|l_i^{\mathcal{S}} - l_j^{\mathcal{S}}|$, is greater than $d_{\mathcal{S}}$. A similar cost function was used in a BP approach for stereo [106], although rather than truncating the linear cost, they use a robust function that changes smoothly from zero to a constant as the cost increases. In our implementation, we set $c_{\mathcal{S}} = 10/(|\mathcal{S}| - 1)$ and $d_{\mathcal{S}} = 5$ such that the cost stops increasing when the label difference is more than half the total number of labels $|\mathcal{S}|$.

To model the orientation prior, $V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})$, we also use a truncated linear model. However, the effects of rotation symmetry found in the example patch needs to be incorporated into the cost function, $V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})$ instead of directly using the label difference. Our modification to the linear model is as follows:

$$V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}}) = \min(c_{\mathcal{O}} W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) f(|l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|), d_{\mathcal{O}}) \quad \text{(Eq. 3.5)}$$

where

$$W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) = \frac{1}{|\mathcal{T}_{l_i^{\mathcal{O}}}||\mathcal{T}_{l_j^{\mathcal{O}}}|} \sum_{m=1}^{|\mathcal{T}_{l_i^{\mathcal{O}}}|} \sum_{n=1}^{|\mathcal{T}_{l_j^{\mathcal{O}}}|} ||\mu_{\mathcal{T}_{l_i^{\mathcal{O}}}^m} - \mu_{\mathcal{T}_{l_j^{\mathcal{O}}}^n}||$$

is the average Euclidean difference between the Gaussian means of principal features in $\mathcal{T}$ at orientation defined by $l_i^{\mathcal{O}}$ and $l_j^{\mathcal{O}}$, and $W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) \in [0, 1]$ after normalization,

$$f(|l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|) = \begin{cases} |l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|, & |l_i^{\mathcal{O}} - l_j^{\mathcal{O}}| \leq |\mathcal{O}|/2 \\ |\mathcal{O}| - |l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|, & |l_i^{\mathcal{O}} - l_j^{\mathcal{O}}| > |\mathcal{O}|/2 \end{cases}$$

defines the label difference so that the angle difference between the two label $l_i^{\mathcal{O}}$, $l_j^{\mathcal{O}}$ within $[0, 180°)$. Figure 3.3 displays the compatibility matrix $W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}}) f(|l_i^{\mathcal{O}} - l_j^{\mathcal{O}}|)$ for different texture samples, we can see that our definition of $W(\mathcal{T}_{l_i^{\mathcal{O}}}, \mathcal{T}_{l_j^{\mathcal{O}}})$ models the rotation symmetric properly. We set $c_{\mathcal{O}} = 20/(|\mathcal{O}| - 1)$ and $d_{\mathcal{O}} = 5$ which truncate the costs when the angle difference between two labels is larger than $90°$. With both definitions of $V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}})$ and $V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})$, it is easy to observe that they take the same weights in joint function $V_{ij}(l_i, l_j)$.

Combining the definition of our likelihood (Eq. 3.2), prior (Eq. 3.3) and the global objective function (Eq. 3.1), our Bayesian MRF is formulated as:

$$E(\mathcal{L}) = \min_{\mathcal{L}} \left(\sum_{i=1}^{N} V_i(l_i) + \sum_{(i,j)\in\mathcal{E}} (V_{\mathcal{S}_{ij}}(l_i^{\mathcal{S}}, l_j^{\mathcal{S}}) + V_{\mathcal{O}_{ij}}(l_i^{\mathcal{O}}, l_j^{\mathcal{O}})) \right).$$

3.6 Experiments

Our algorithm is evaluated using both natural and synthetic image examples. For the natural image examples, we compare our results with those obtained by two recent flow estimation approaches proposed by Paris *et al.* [82] and Hays *et al.* [48]. We admit that this is not entirely

a fair comparison as these approaches target specific types texture types; gradient-like textures in [82] and regular textures in [48]. However, we use these recent approaches to demonstrate the advantage of our method to target diverse texture types.

For examples on synthetic examples, we use the recent texture synthesis technique introduced by Lefebvre and Hoppe [63] that synthesizes textures for over a specified flow field. This gives us the ability to compare our results using the specified flow field as a ground truth comparison.

3.6.1 Real World Examples

For the real world examples, while we make no assumption about the texture type, we do assume the user specified example patches are not distorted and contain sufficient textons to represent the texture. We also assume reasonably constant shading in the distorted texture. Significant shading or shadows should be reduced before applying our algorithm. Various techniques are available to address this task and for our purposes we will assume such pre-processing has already been applied.

Figure 3.1 shows the paved stone road example used in figure 3.1. Our estimated texture flow field (figure 3.1(b) is consistent with human perception in terms of scale and orientation of the stone pattern. Figure 3.1(d) shows a result from orientation extraction using Paris *et al.* [82]. Since their approach assumes a gradient-like texture its breaks down for this example. Figure 3.1(e) shows a result from Hays *et al.* [48]. While the approach in [48] offers the benefit of being fully automated, it targets textures with lattice structures. As a result, the approach works well for a small portion of the image, but as the texture becomes increasingly distorted under perspective projection the approach is unable to follow the texture flow.

Figure 3.4(a) shows a zebra example. We select a sample texture patch from the zebra body as indicated by the green box. The shading of the zebra texture is first normalized as shown in figure 3.4(b) which can be achieved with various techniques, such as homomorphic filtering.

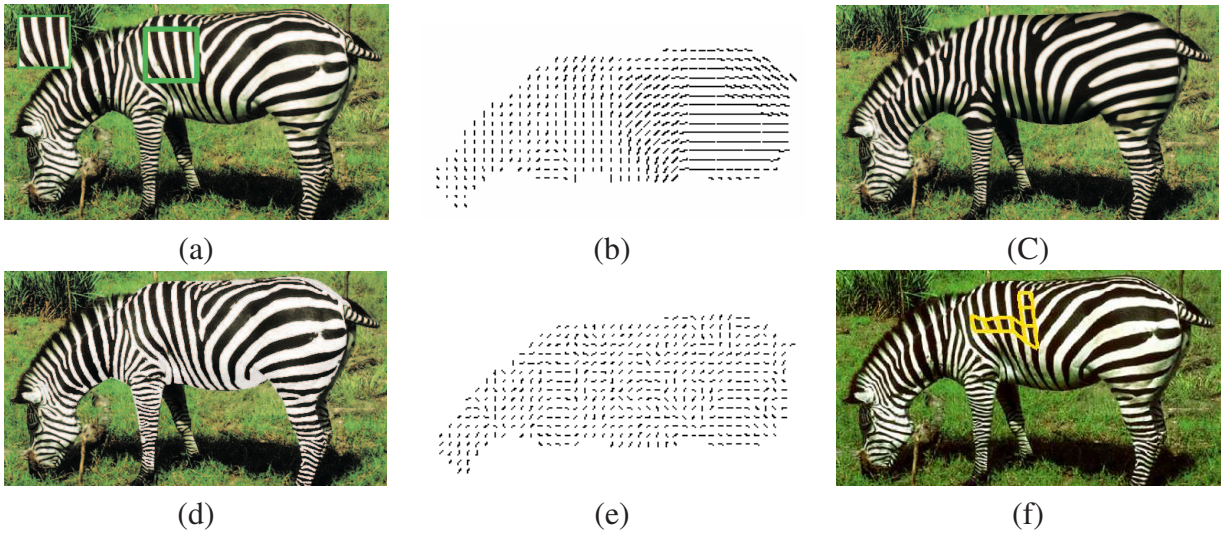
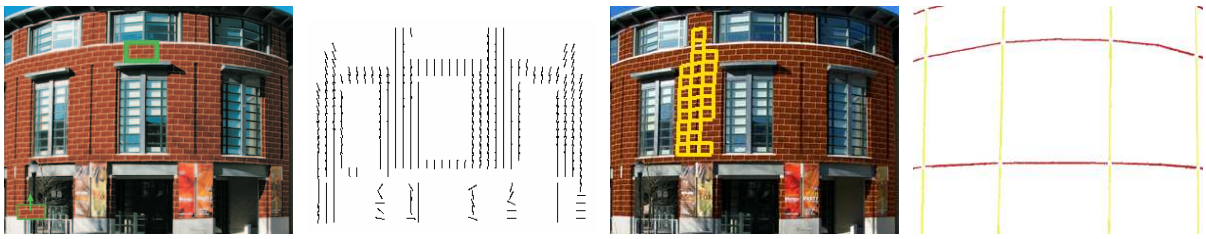


Figure 3.4: (a) Zebra image with the example patch indicated in the green box. (b) The estimated texture flow field of zebra body. (c) Texture re-synthesized using the example patch. (d) Shading normalized over the zebra's body. (e) The estimated orientation field from [82]. (f) Detected regular texture lattice from [48]
.

We show our estimated texture flow field in figure 3.4(c). Using the extracted texture flow field, we perform texture replacement using the flow-guided texture synthesis [63] as shown in figure 3.4(d). The example texture is used to replace the original zebra texture. Note how similar the re-synthesized version looks to the original. Figure 3.4(e) shows a result from [82] which scale changes are not modeled. Their approach also does not handle discontinuity and singularity well. Figure 3.4(f) shows a result from [48]. The texture violates their regular texture assumption and can only detect a small region of the texture.

Figure 3.5 compares our extract on an example demonstrated in the "near regular" texture synthesis proposed by Liu *et al.* [69] where a lattice structure is specified by user to guide the synthesis. Figure 3.5(a) show an example from in [69]. Our estimated texture flow field is shown in figure 3.5(b). Figure 3.5(c) shows the result from [48]. The user specified control lines from Liu *et al.* [69] are shown in figure 3.5(d). Our result is better then result from [48] and is comparable to the orientation of control lines manually specified in [69].



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

Figure 3.5: Texture flow estimation of real image. (a) Input image from [69]. An example texture patch is selected from the input image as shown in the green box. (b) Our estimated texture flow field. (c) Result from [48]. (d) The user specified control line from [69].

3.6.2 Synthetic Examples

Experiments for estimating the texture flow using synthetic examples are shown in figure 3.6. The benefit of synthetically generated textures is that we can control the difficult of the testcase as well as provide a mechanism to test against ground truth. Seven different sample textures are shown at the top of the figure. The example patch sizes are shown below each sample in the figure. The input images are all 256×256. The texture examples vary significantly, with different levels of symmetry, texton scale, and texton distribution.

For the first example, a texture flow field with orientation varying from 0° to 180° and a fixed the scale is used. Even though the scale is fixed, we apply our algorithm to test over all orientation and scale sizes. For each texture example, the root mean squared error (*rms*) of our estimated flow field against the known flow field is shown in parentheses for orientation and scale respectively. For orientation, the average *rms* is 9.73° for all examples. This is less than the quantization angle of 15°. In addition, only texture four (labeled at T_4 in the figure) has minor problems with scale estimation.

The second test case tests the accuracy of scale estimation. The scale of the texture flow field are changed while the orientation is fixed. In this test case, the orientation is estimated correctly (with *rms* less than 1°). The average *rms* of scale is 0.09. Again, the *rms* of each test example is smaller than the quantized scale (0.25).

The third test case tests our algorithm under perspective projection and the ground truth texture flow field is defined by a plane under projection and both orientation and scale are distorted. Our estimated average *rms* for orientation and scale are 11.26° and 0.19 respectively. The *rms* of the orientation for texture example five slightly exceeds the quantization angle. This is due to the fact that this particular texture becomes homogeneous under down sampling.

The final test case, the ground truth texture flow field is a spiral structure with both orientation and scale changing at every pixel. This complex texture flow field is found less frequently in natural structure but is common in manmade objects. Under this difficult test case, our algorithm is still reliable at estimating the texture flow field with average *rms* of orientation equal to 13.53° and average *rms* of scale equal to 0.12.

For that vast majority of the test cases, we find that the *rms* for both orientation and scale are smaller than the quantization error. This demonstrates that our extracted principal features are effective in their estimation of orientation and scale likelihoods for local texture neighborhoods, and that our MRF label assignment process is accurate and robust.

3.7 Summary

We have presented a robust technique for estimating orientation and scale flow fields in a distorted texture given a sample patch. We formulated this problem into a discrete labeling Bayesian Markov Random Field with can be solved effectively by using priority belief propagation. Our results show that our technique can be used with a variety of texture, and is able to produce good results on both natural and synthetic images for considerably distorted textures. While the use of the RGB space in our feature sampling generates good results, a perceptual color space or gradient space may help to ameliorate the effect of illumination. Future work includes exploiting our approach to help normalize the underlying texture for general use in texture-synthesis.

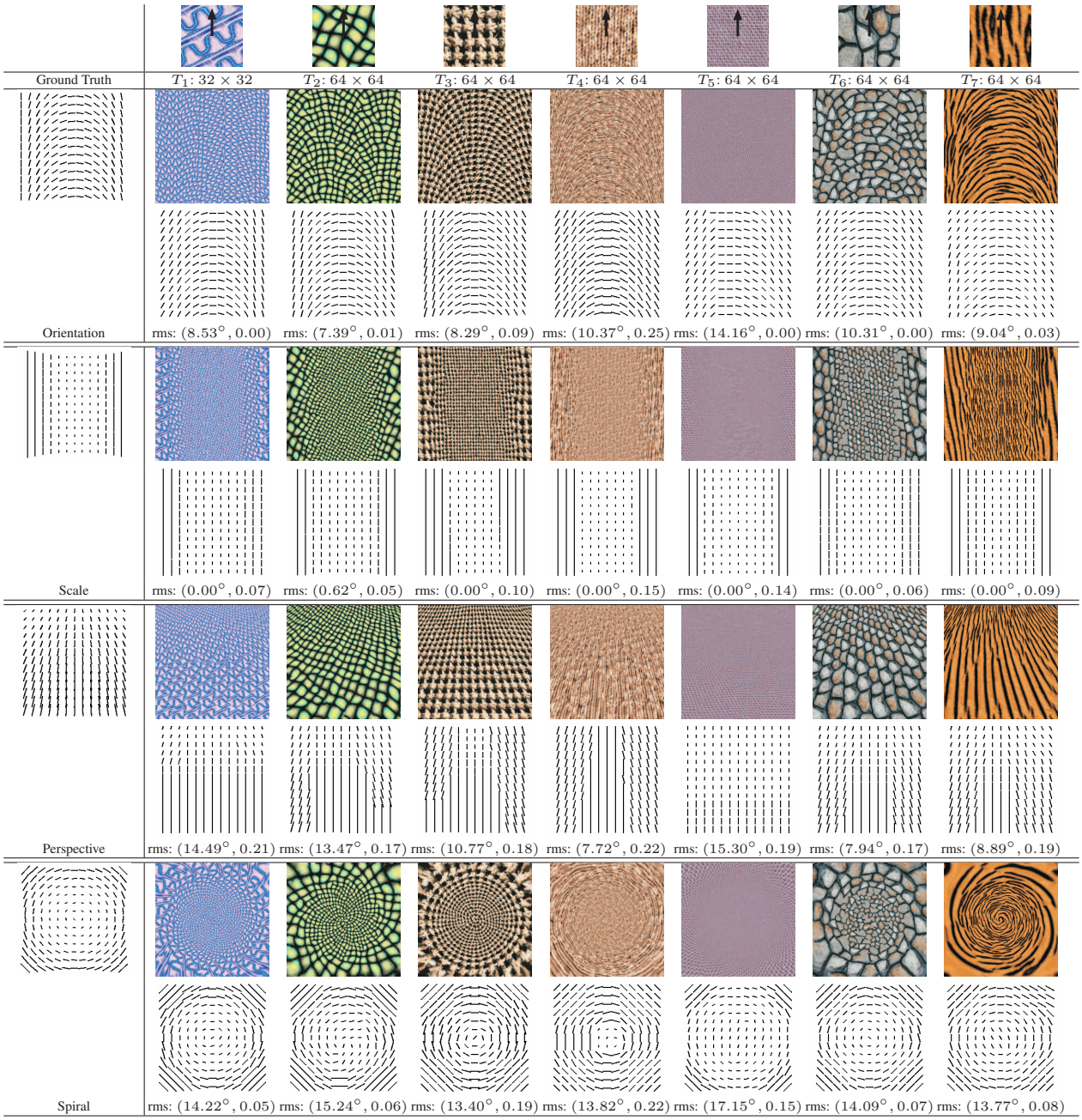


Figure 3.6: Our algorithm is tested on several synthetic examples. Seven sample textures are shown on the top row. Four test cases are demonstrated: rotation, scale, perspective projection and spiral. Ground truth for the texture flow field are shown for comparison in the left most column. For each test case, the upper row shows the distorted texture generated using anisotropic texture synthesis from [63]. The lower row shows the estimated texture flow field with root mean square (*rms*) errors for orientation angle and scale respectively. Since orientation is quantize by 15° and scale by 0.25, we find that for most of our test case, the *rms* of both orientation and scale are below our quantization size. This means that estimation errors are likely attributed to the angle and scale quantization than errors in the actual label assignment.

Chapter 4

Image/Video Deblurring using a Hybrid Camera

4.1 Overview

We describe a novel approach to reduce spatially-varying motion blur in video and images using a hybrid camera system. A hybrid camera is a standard video camera that is coupled with an auxiliary low-resolution camera sharing the same optical path but capturing at a significantly higher frame rate. The auxiliary video is temporally sharper but at a lower resolution, while the lower-frame-rate video has higher spatial resolution but is susceptible to motion blur.

Our deblurring approach uses the data from these two video streams to reduce spatially-varying motion blur in the high-resolution camera with a technique that combines both deconvolution and super-resolution. Our algorithm also incorporates a refinement of the spatially-varying blur kernels to further improve results. A Bayesian optimization framework is used to effectively combine these likelihood measurements into a single global objective function. We solved the global objective function by using alternating optimization (AO) procedure. Our approach can reduce motion blur from the high-resolution video as well as estimate new high-resolution frames at a higher framerate. Experimental results on a variety of inputs demonstrate notable improvement over current state-of-the-art methods in image/video deblurring.

(a) (b)

Figure 4.1: Tradeoff between resolution and frame rates. (a) Image from a high-resolution, low-frame-rate camera. (b) Image from a low-resolution, high-frame-rate camera.

4.2 Introduction

This chapter introduces a novel approach to reduce spatially-varying motion blur in video footage. Our approach uses a hybrid camera framework first proposed by Ben-Ezra and Nayar [7, 8]. A hybrid camera system simultaneously captures a high-resolution video together with a low-resolution video that has denser temporal sampling. The hybrid camera system is designed such that the two videos are synchronized and share the same optical path. Using the information in these two videos, our method has two aims: 1) to deblur the frames in the high-resolution video, and 2) to estimate new high-resolution video frames at a higher temporal sampling. While a high-resolution, high-frame-rate camera could be used to obtain this data, such cameras are expensive and require high-speed data streaming and substantial memory storage. As a result, such high-end cameras are generally restricted to use in a studio or laboratory settings. A hybrid camera system, on the other hand, is relatively cheap to construct.

The previous work in [7, 8] using a hybrid camera system focused on correcting motion blur in a single image due to translational ego-motion in a static scene and was limited to globally invariant motion blur. In this chapter, we address the broader problem of correcting spatially-varying motion blur and aim to deblur temporal sequences. In addition, our work achieves

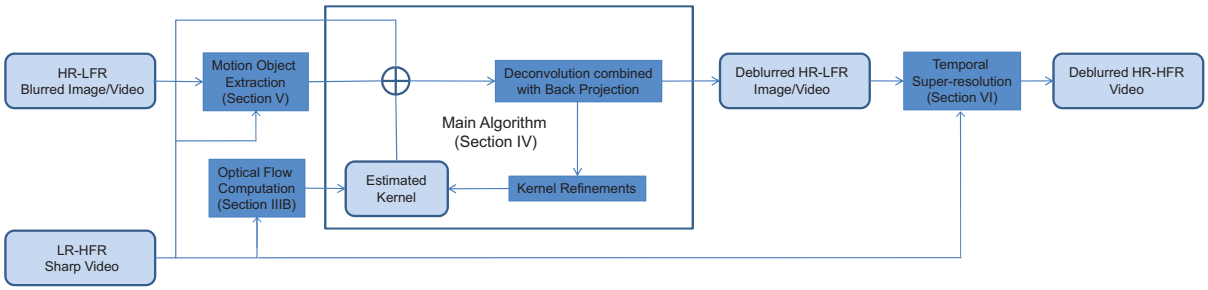


Figure 4.2: The processing pipeline of our system. Optical flows are first calculated from the Low-Resolution High-Frame-Rate (LR-HFR) video. From the optical flows, spatially-varying motion blur kernels are estimated (Section 4.4.2). Then the main algorithm performs an iterative optimization procedure which simultaneously deblurs the High-Resolution, Low-Frame-Rate (HR-LFR) image/video and refines the estimated kernels (Section 4.5). The output is a deblurred HR-LFR image/video. For the case of deblurring a moving object, the object is separated from the background prior to processing (Section 4.6). In the deblurring of video, we can additionally enhance the frame rate of the deblurred video to produce a High-Resolution, High-Frame-Rate (HR-HFR) video result (Section 4.7).

improved deblurring performance by more comprehensively exploiting the available information acquired in the hybrid camera system, including optical flow, back-projection constraints between low-resolution and high-resolution images, and temporal coherence along image sequences. In addition, our approach can be used to increase the frame rate of the high-resolution camera by estimating the missing frames.

The central idea in our formulation is to combine the benefits of both deconvolution and super-resolution. Deconvolution of motion blurred, high-resolution images yields high-frequency details, but with ringing artifacts due to lack of low-frequency components. In contrast, super-resolution-based reconstruction from low-resolution images recovers artifact-free low-frequency results that lack high-frequency detail. We show that the deblurring information from deconvolution and super-resolution are complementary to each other and can be used together to improve deblurring performance. In video deblurring applications, our method furthermore capitalizes on additional deconvolution constraints that can be derived from consecutive video frames. We demonstrate that this approach produces excellent results in reducing spatially-

varying motion blur. In addition, the availability of the low-resolution imagery, and subsequently derived motion vectors, further allows us to perform estimate new temporal frames in the high-resolution video, which we also demonstrate.

The processing pipeline of our approach is shown in Figure 4.2, which also relates process components to their corresponding section in the chapter. The remainder of the chapter is organized as follows: Section 4.3 discusses related work; Section 4.4 describes the hybrid camera setup and the constraints on deblurring available in this system; Section 4.5 describes our overall deconvolution formulation expressed in a maximum a posteriori (MAP) framework; Section 4.6 discusses how to extend our framework to handle moving objects; Section 4.7 describes how to perform temporal super-resolution with our framework; Section 4.8 provides results and comparisons with other current works, followed by a discussion and summary in Section 4.9.

4.3 Related Work

Motion deblurring can be cast as the deconvolution of an image that has been convolved with either a global motion point spread function (PSF) or a spatially-varying PSF. The problem is inherently ill-posed as there are a number of unblurred images that can produce the same blurred image after convolution. Nonetheless, this problem is well studied given its utility in photography and video capture. The following describes several related works.

The majority of related work involves traditional blind deconvolution which simultaneously estimates a global motion PSF and the deblurred image. These methods include well-known algorithms such as Richardson-Lucy [96, 72] and Wiener deconvolution [123]. For a survey on blind deconvolution readers are referred to [47, 43]. These traditional approaches often produce less than desirable results that include artifacts such as ringing.

A recent trend in motion deblurring is to either constrain the solution of the deblurred image or to use auxiliary information to aid in either the PSF estimation or the deconvolution itself (or

both). Examples include work by Fergus *et al.* [39], which used natural image statistics to constrain the solution to the deconvolved image. Raskar *et al.* [93] altered the shuttering sequence of a traditional camera to be make the PSF more suitable for deconvolution. Jia [54] extracted an alpha mask of the blurred region to aid in PSF estimation. Dey *et al.* [30] modified the Richardson-Lucy algorithm by incorporating total variation regularization to suppress ringing artifacts. Levin *et al.* [65] introduced gradient sparsity constraints to reduce ringing artifacts. Yuan *et al.* [131] proposed a multiscale non-blind deconvolution approach to progressively recover motion blurred details. Shan *et al.* [99] studied the relationship between estimation errors and ringing artifacts, and proposed the use of a spatial distribution model of image noise together with a local prior that suppresses ringing to jointly improve global motion deblurring.

Other recent approaches use more than one image to aid in the deconvolution process. Bascle *et al.* [5] processed a blurry image sequence to generate a single unblurred image. Yuan *et al.* [130] used a pair of images, one noisy and one blurred. Rav-Acha and Peleg [94] consider images that have been blurred in orthogonal directions to help estimate the PSF and constrain the resulting image. Chen and Tang [18] extend the work of [94] to remove the assumption of orthogonal blur directions. Bhat *et al.* [12] proposed a method that uses high-resolution photographs to enhance low-quality video, but this approach is limited to static scenes. Most closely related to ours is the work of Ben-Ezra and Nayar [7, 8], which used an additional imaging sensor to capture high-frame-rate imagery for the purpose of computing optical flow and estimating a global PSF. Li *et al.* [68] extend the work of Ben-Ezra and Nayar [7, 8] by using parallel cameras with different frame rates and resolutions, but their work targets depth map estimation and not deblurring.

The aforementioned approaches assume the blur to arise from a global PSF. Recent works addressing spatially-varying motion blur include that of Levin [64], which used image statistics to correct a single motion blur on a stable background. Bardsley *et al.* [4] segmented an image into regions exhibiting similar blur, while Cho *et al.* [20] used two blurred images to simultaneously estimate local PSFs as well as deconvolve the two images. Their approaches [64, 4, 20],

however, assume the motion blur to be globally invariant within each separated layer. Work by Shan *et al.* [100] allows the PSF to be spatially-varying; however, the blur is constrained to that from rotational motion. Levin *et al.* [67] proposed a parabolic camera designed for deblurring images with 1D object motion. During exposure, the camera moves in a manner that allows the resulting image to be deblurred using a single deconvolution kernel.

The problem of super-resolution can be considered as a special case of motion deblurring in which the blur kernel is a low-pass filter that is uniform in all motion directions. High-frequency details of a sharp image are therefore completely lost in the observed low-resolution image. There are two main approaches to super-resolution: image hallucination based on training data and image super-resolution computed from multiple low-resolution images. Our work is closely related to the latter approach, which is reviewed here. The most common technique for multiple image super-resolution is the back-propagation algorithm proposed by Irani and Peleg [52, 53]. The back-propagation algorithm is an iterative refinement procedure that minimizes the reconstruction errors of an estimated high-resolution image through a process of convolution, downsampling and upsampling. A brief review that includes other early works on multiple image super-resolution is given in [17]. More recently, Patti *et al.* [83] proposed a method to align low-resolution video frames with arbitrary sampling lattices to reconstruct a high-resolution video. Their approach also uses optical flow for alignment and PSF estimation. These estimates, however, are global and do not consider local object motion. This work was extended by Elad and Feuer [37] to use adaptive filtering techniques. Zhao and Sawhney [136] studied the performance of multiple image super-resolution against the accuracy of optical flow alignment and concluded that the optical flows need to be reasonably accurate in order to avoid ghosting effects in super-resolution results. Shechtman *et al.* [34] proposed space-time super-resolution in which they align video clips with different resolutions and frame rates using homographies to produce high-resolution, high-frame-rate output. Their approach, however, does not consider local motion, such that the blur of moving objects will remain.

Sroubek *et al.* [104] proposed a regularization framework for solving the multiple image super-resolution problem. This approach also does not consider local motion blur effects. Recently, Agrawal [1] proposed a method to increase the resolution of images that have been deblurred using a fluttered shutter system. Their approach can also be considered as a combination of motion deblurring and super-resolution, but is limited to translational motion.

While various previous works are related in part, our work is unique in its focus on spatially-varying blur with no assumption on global or local motion paths. Moreover, our approach takes full advantage of the rich information available from the hybrid camera system, using techniques from both deblurring and super-resolution together in a single MAP framework. We demonstrate our results on different forms of spatially-varying motion blur, and show that our approach can achieve sharper and cleaner results in comparison to state-of-the-art techniques.

4.4 Hybrid Camera System

Figure 4.1 illustrates the tradeoff between a high resolution image captured at a low frame rate, and a low resolution image captured at a high frame rate. For comparable levels of scene exposure per pixel, the high resolution image requires a longer exposure time and thus suffers from motion blur. On the other hand, a short exposure suffices for the low resolution image captured with larger sensor units, and it is therefore sharp but lacking in detail.

The advantages of a hybrid camera system are derived from the additional data acquired by the low-resolution high-frame-rate (LR-HFR) camera. While the spatial resolution of this camera is too low for most practical applications, the high-speed imagery is reasonably blur free and is thus suitable for optical flow computation. Since the cameras are assumed to be synchronized temporally and observing the same scene, the optical flow corresponds to the motion of the scene observed by the high-resolution low-frame-rate (HR-LFR) camera, whose images are blurred due to its slower temporal sampling. An obvious connection is to use the optical flow to compute the overall blur kernel of the high-resolution image for deconvolution.

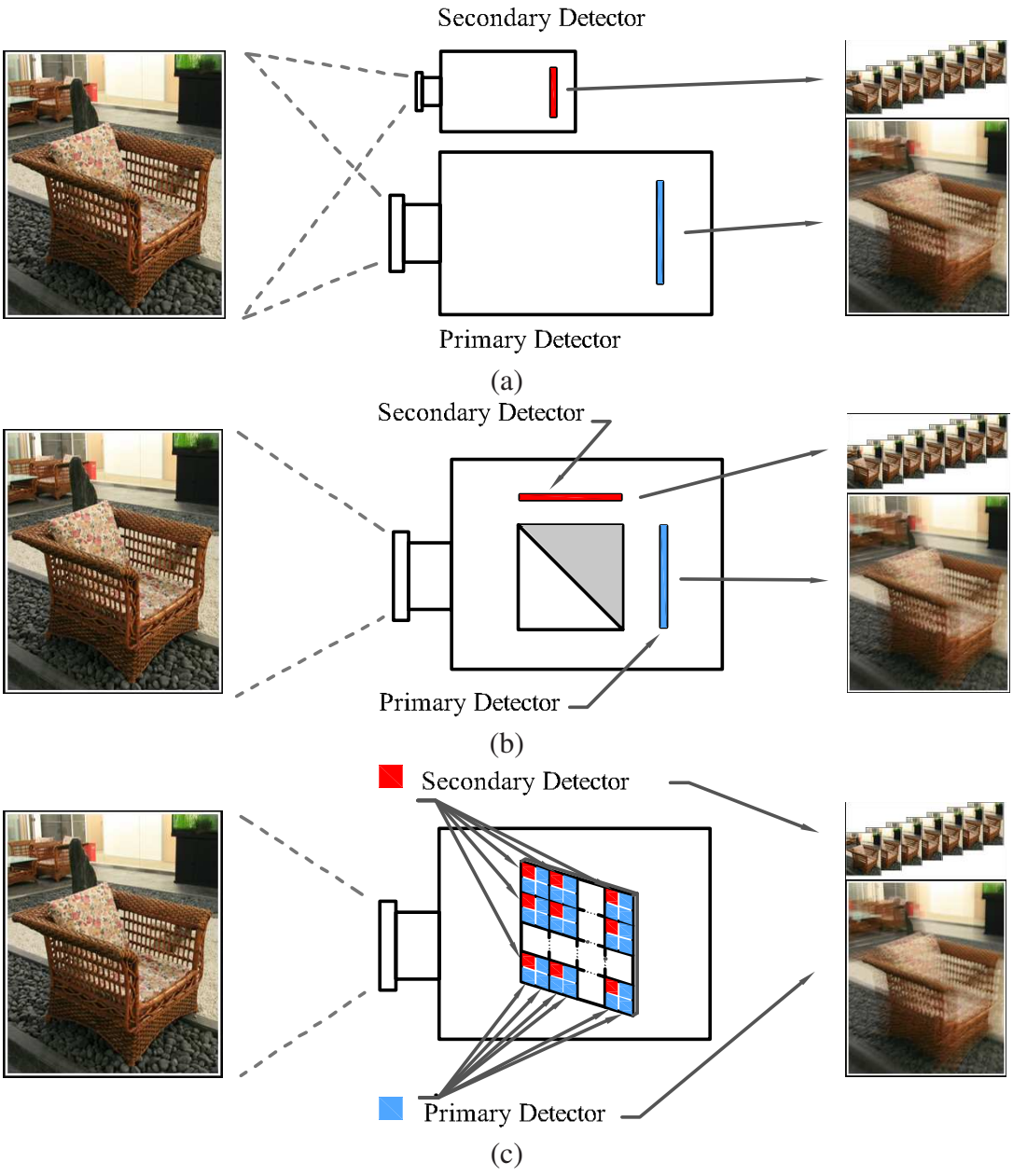


Figure 4.3: The three conceptual design of hybrid camera purposed by [7, 8]. (a) Two cameras are placed together as close as possible to simulate one single view. (b) Light rays are splitted for two detectors after passing through the same aperture. (c) A single detector contains two different sensor such that their stream rate can be adjust differently.

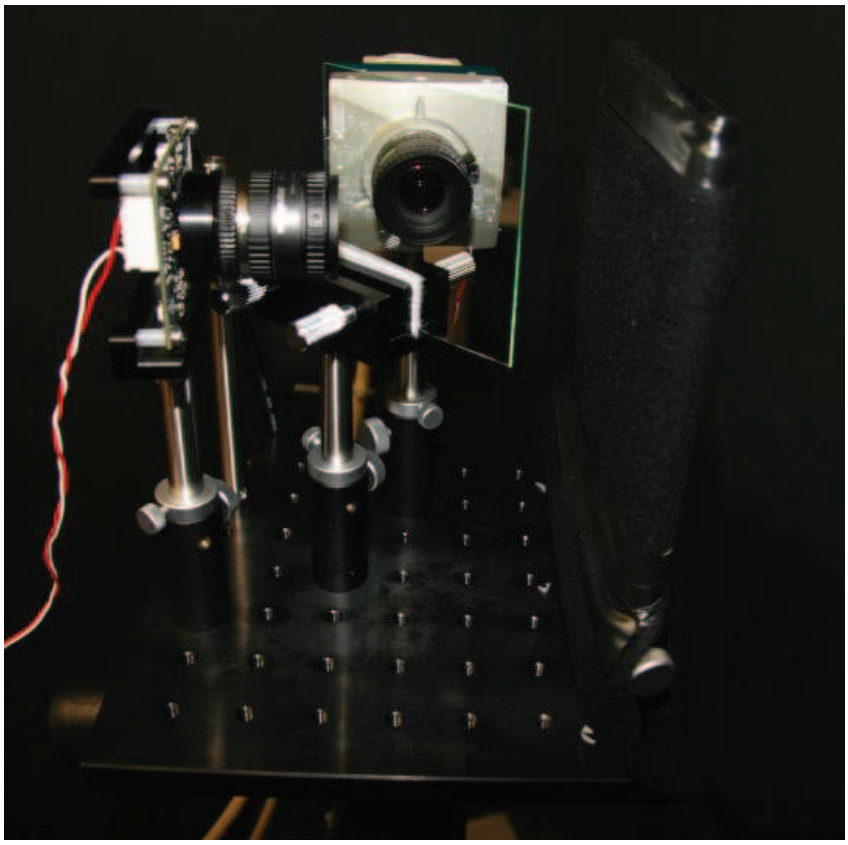


Figure 4.4: Our hybrid camera combines a Point Grey Dragonfly II camera, which captures images of 1024×768 resolution at 25 fps (6.25 fps for image deblurring examples), and a Mikrotron MC1311 camera that captures images of 128×96 resolution at 100 fps. A beam-splitter is employed to align their optical axes and respective images.

In the following, we discuss the construction of a hybrid camera, the optical flow and motion blur estimation, and the use of the low-resolution images as reconstruction constraints on the high-resolution images.

4.4.1 Camera Construction

Three conceptual designs of the hybrid camera system were discussed by Ben-Ezra and Nayar [7] (Figure 4.3). In their work, they implemented a simple design in which the two cameras are placed side-by-side, such that their viewpoints can be considered the same when viewing a distant scene. A second design avoids the distant scene requirement by using a beam splitter

(a) (b)

Figure 4.5: Spatially varying blur kernel estimation using optical flows. (a) Motion blur image. (b) Estimated blur kernels of (a) using optical flow.

to share between the two sensing devices the light rays that pass through a single aperture, as demonstrated by McGuire *et al.* [77] for the studio matting problem. A promising third design is to capture both the HR-LFR and LR-HFR video on a single sensor chip. According to [13], this can readily be achieved using a programmable CMOS sensing device.

In our work, we built a hand-held hybrid camera system based on the second design as shown in Figure 4.4. The two cameras were positioned such that their optical axes and pixel arrays are well aligned, and video synchronization is achieved using a 8051 microcontroller. To match the color responses of the two devices, we employ histogram equalization. Since the exposure levels of the two devices are set to be equal, the signal-to-noise ratios in the HR-LFR and LR-HFR images are approximately the same.

4.4.2 Blur Kernel Approximation Using Optical Flow

In the absence of occlusion, disocclusion, and out-of-plane rotation, a blur kernel can be assumed to represent the motion of a camera relative to objects in the scene. In [7], this relative motion is assumed to be constant throughout an image, and the globally invariant blur kernel is obtained through the integration of global motion vectors over a spline curve.

However, since optical flow is in fact a local estimation of motions, we can calculate spatially varying blur kernels from optical flow. We use the pyramidal Lucas-Kanade algorithm [71] to calculate the optical flow at each pixel location:

$$\arg\min_{(u,v)} \sum \left(u\frac{\delta I(x,y,t)}{\delta x} + v\frac{\delta I(x,y,t)}{\delta y} + \frac{\delta I(x,y,t)}{\delta t} \right) \quad \text{(Eq. 4.1)}$$

where (u, v) is the optical flow vector at image position (x, y, t), $\frac{\delta I(x,y,t)}{\delta x}$, $\frac{\delta I(x,y,t)}{\delta y}$ and $\frac{\delta I(x,y,t)}{\delta t}$ are the partial derivative of image I at (x, y, t) position with respect to x, y and t respectively. We follow the brightness constancy assumption of optical flow estimation to assume our motion blurred image are captured under constant illumination such that the change of pixel color of moving scene/object over the exposure peroid can be neglected. The per-pixel motion vectors are then integrated to form spatially varying blur kernels, $K(x, y)$, one per pixel [7]:

$$K(x,y) = \int S(x,y)\delta t \quad \text{(Eq. 4.2)}$$

where $S(x, y)$ is a spline curve that has been fitted to the path of optical flow at (x, y) position, δt is the time interval between each successive frame in the high frame rate video. We use a C1 continuity spline curve fit to the path of optical flow at position (x, y). The number of frame used to fit the spline curve is 16 for image examples and 4 for video examples (Figure 4.4). Figure 4.5 shows an example of spatially varying blur kernels estimated from optical flows.

The optical flows estimated with the multiscale Lucas-Kanade algorithm [71] may contain noise that degrades blur kernel estimation. We found such noisy estimates to occur mainly in smooth or homogeneous regions which lack features for correspondence, while regions with sharp features tend to have accurate optical flows. Since deblurring artifacts are evident primarily around such features, the Lucas-Kanade optical flows are effective for our purposes. On the other hand, the optical flow noise in relatively featureless regions has little effect on deblurring results, since these areas are relatively unaffected by errors in the deblurring kernel. As a measure to heighten the accuracy and consistency of estimated optical flows, we use local smoothing [127] as an enhancement of the multiscale Lucas-Kanade algorithm [71].

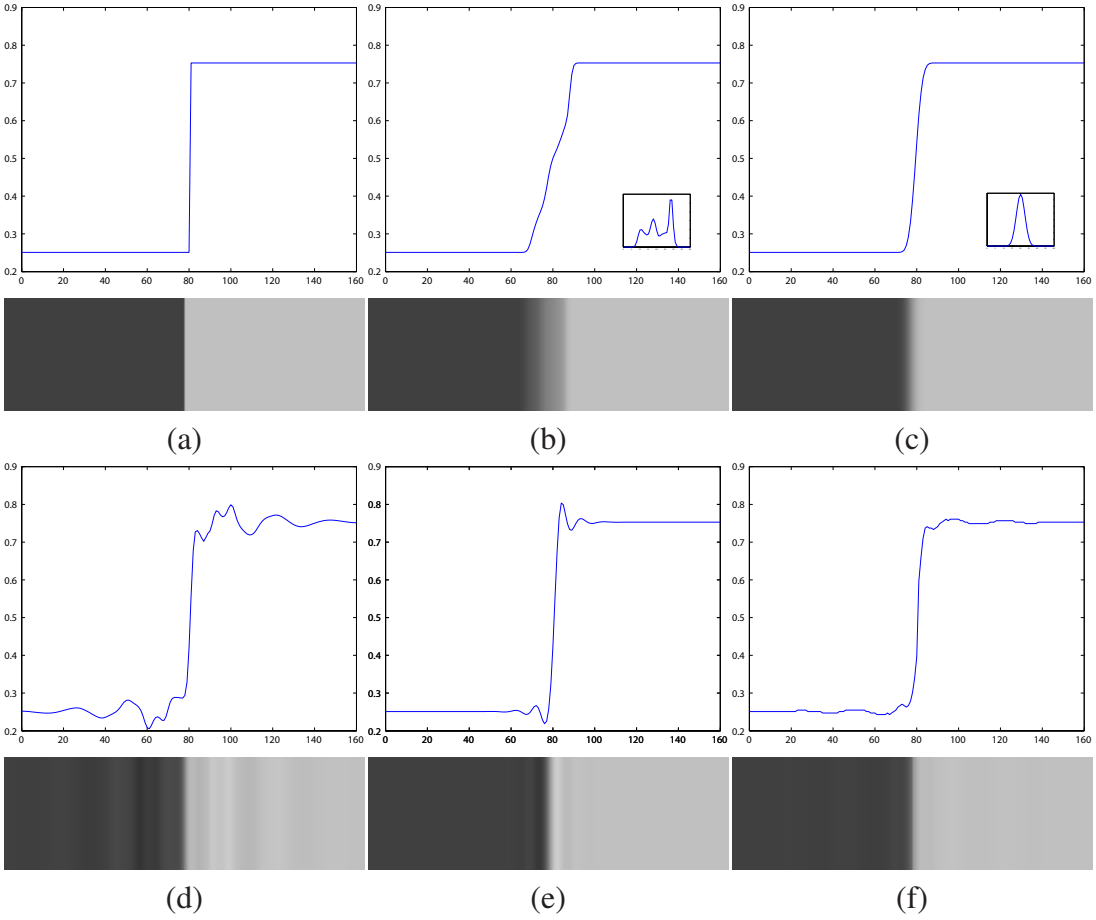


Figure 4.6: In this figure, we show the benefits of using both deconvolution and super-resolution for deblurring through a 1D illustrative example. (a) Ground truth signal. (b) Motion blurred signal of (a) with discontinuities in motion, the motion blur kernel is also shown. (c) Observed low resolution signal of (a), the gaussian low-pass kernel is also shown. (d) Deblurred signal using the given motion blur kernel. (e) Super-resolution of (c). (f) Deblurred signal using both constraints from the motion blur kernel and super-resolution. For better visualization, an image corresponds to each of the 1D signal are also shown.

These estimated blur kernels contain quantization errors due to the low resolution of the optical flows. Additionally, motion vector integration may provide an imprecise temporal interpolation of the flow observations. In our Bayesian optimization framework, we consider the blur kernels to also be parameters to be estimated, and thus are subject to refinement in the optimization procedure. The details will be discussed fully in Section 4.5.

4.4.3 Back-Projection Constraints

The capture of low-resolution frames in addition to the high-resolution images not only facilitates optical flow computation, but also provides super-resolution-based reconstruction constraints [52, 53, 83, 17, 37, 3, 34] on the high-resolution deblurring solution. The back-projection algorithm [52, 53] is one of the most common iterative techniques to minimize reconstruction error, and can be formulated as follows:

$$I^{t+1} = I^t + \sum_{j=1}^{M} (u(W(I_{l_j}) - d(I^t \otimes h))) \otimes p \qquad \text{(Eq. 4.3)}$$

where M is the number of corresponding low-resolution observations, t is an iteration index, I_{l_j} is the j-th low-resolution image, $W(\cdot)$ is the warping function that align I_{l_j} with respect to a basis reference image, $\otimes$ is the convolution operation, h is the convolution filter before downsampling, p is a filter representing the back-projection process, and $d(\cdot)$ and $u(\cdot)$ are the downsampling and upsampling processes respectively. Equation (Eq. 4.3) assumes that each observation carries the same weight. In the absence of a prior, h is chosen to be a gaussian filter with a size proportionate to the downsampling factor, and p is set equal to h.

In the hybrid camera system, a number of low-resolution frames are captured in conjunction with each high-resolution image. To exploit this available data, we align these frames according to the computed optical flows, and use them as back-projection constraints in Equation (Eq. 4.3). The number of low-resolution image constraints M is determined by the relative frame rates of the cameras. In our implementation, we choose the first low-resolution frame as the basic reference frame and the estimated blur kernel and other low resolution frames are aligned with respect to this reference frame. Of course, we can choose other low-resolution frame as the basic reference frame and this will leads to another deblurred result. As we will see in the later section 4.8, this property of producing multiple solution is an important property that allows us to enhance temporal sampling of deblurred video which is a unique property of our motion deblurring algorithm.

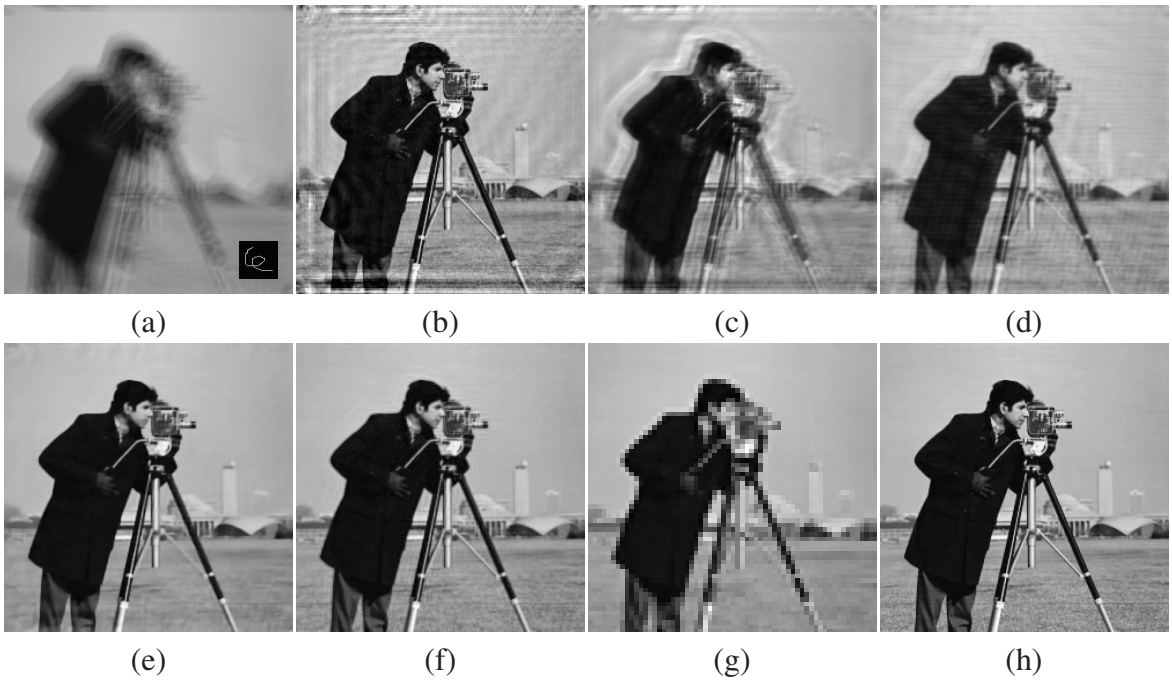


Figure 4.7: Performance comparisons for different deconvolution algorithms on a synthetic example. The ground truth motion blur kernel is used to facilitate comparison. The signal-to-noise ratio (SNR) of each result is reported. (a) A motion blurred image [SNR(dB)=25.62] with the corresponding motion blur kernel shown in the inset. Deconvolution results using (b) Wiener filter [SNR(dB)=37.0], (c) Richardson-Lucy [SNR(dB)=33.89], (d) Total Variation Regularization [SNR(dB)=36.13], (e) Gradient Sparsity Prior [SNR(dB)=46.37], and (f) our approach [SNR(dB)=50.26dB], which combines constraints from both deconvolution and super-resolution. The low resolution image in (g) is 8x downsampled from the original image, shown in (h).

The benefits of using the back-projection constraint, and multiple such constraints, is illustrated in Figure 4.6. Each of the low-resolution frames presents a physical constraint on the high-resolution solution, in a manner resembling a set of offset images in the super-resolution application.

4.5 Bayesian Optimization Framework

A brief review on the Richardson-Lucy deconvolution algorithm is given as our approach is fashioned in a similar manner. For sake of clarity, our algorithm is first discussed for use with

correcting global motion blur, followed by its extension to spatially varying blur kernels.

4.5.1 Richardson-Lucy Image Deconvolution

The Richardson-Lucy algorithm [96, 72] is an iterative deconvolution algorithm derived from Bayes Theorem that minimizes the following estimation error:

$$\arg\min_{I} n(||I_b - I \otimes K||^2) \tag{Eq. 4.4}$$

where I is the deblurred image, K is the blur kernel, I_b is the observed blur image, and $n(\cdot)$ is the image noise distribution. A solution can be obtained using the iterative update algorithm defined as follows:

$$I^{t+1} = I^t \times K * \frac{I_b}{I^t \otimes K} \tag{Eq. 4.5}$$

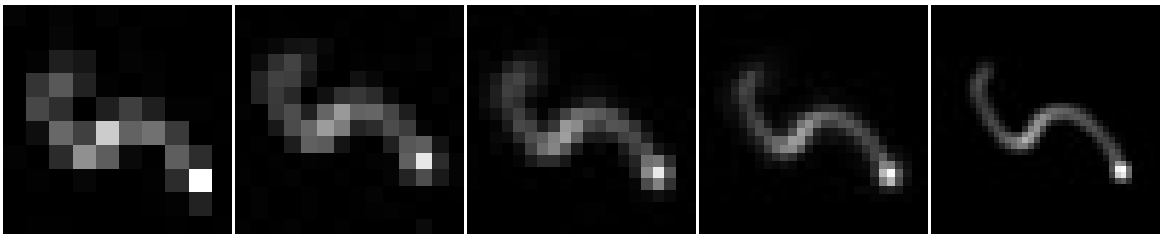
where $*$ is the correlation operation. A blind deconvolution method using the Richardson-Lucy algorithm was proposed by Fish *et al.* [40], which iteratively optimizes I and K in alternation. The same equation (Eq. 4.5) was used with positions of I and K switched during optimization iterations for K. The Richardson-Lucy algorithm assumes image noise $n(\cdot)$ to follow a Poisson distribution. If we assume the image noise $n(\cdot)$ follows a Gaussian distribution, then a least squares method can be employed [52]:

$$I^{t+1} = I^t + K * (I_b - I^t \otimes K) \tag{Eq. 4.6}$$

which shares the same iterative back-projection update rule as Equation (Eq. 4.3).

From video input with computed optical flows, multiple blurred images I_b and blur kernels K may be acquired by reversing the optical flows of neighboring high-resolution frames. These multiple observation constraints can be jointly applied in Equation (Eq. 4.6) [94] as

$$I^{t+1} = I^t + \sum_{i=1}^{N} w_i K_i * (I_{b_i} - I^t \otimes K_i) \tag{Eq. 4.7}$$



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

Figure 4.8: Multiscale refinement of blur kernel. From (a) to (e) are the refined kernel at different scales. Our kernel refinement started from coarsest level. Results of coarser level are then upsampled and are used as initial guess for the next level refinement. The motion blurred images are shown in figure 4.12.

where N is the number of aligned observations. That image restoration can be improved with additional observations under different motion blurs is a important property that we demonstrate in Section 4.8 and exploit in this work. The use of neighboring frames in this manner may also serve to enhance the temporal consistency of the deblurred video frames.

4.5.2 Optimization for Global Kernels

In solving for the deblurred images, our method jointly employs the multiple deconvolution and back-projection constraints derived from the hybrid camera input. For simplicity, we assume in this subsection that the blur kernels are spatially invariant. Our approach can be formulated into a Bayesian framework as follows:

$$\begin{aligned} & \arg\max_{I,K} P(I,K|I_b,K_o,I_l) \\ = \; & \arg\max_{I,K} P(I_b|I,K)P(K_o|I,K)P(I_l|I)P(I)P(K) \\ = \; & \arg\min_{I,K} L(I_b|I,K)+L(K_o|I,K)+L(I_l|I)+L(I)+L(K) \end{aligned} \quad \text{(Eq. 4.8)}$$

where I and K are the sharp images and the blur kernels we want to estimate, I_b, K_o and I_l are the observed blur images, estimated blur kernels from optical flows, and the high frame rate low resolution images respectively, and $L(\cdot) = -log(P(\cdot))$. In our formulation, we make no assumption on the priors $P(I)$ and $P(K)$. Assuming that $P(K_o|I,K)$ is conditionally independent of I, the estimation errors of likelihood probabilities $P(I_b|I,K)$, $P(K_o|I,K)$ and $P(I_l|I)$ follow Gaussian distributions and that each observation of I_b, K_o and I_l are independent and

identically distributed, we can then rewrite Equation (Eq. 4.8) as

$$\arg\min_{I,K} \quad \sum_{i}^{N} ||I_{b_i} - I \otimes K_i||^2 + \lambda_B \sum_{j}^{M} ||I_{l_j} - d(I \otimes h)||^2 + \lambda_K \sum_{i}^{N} ||K_i - K_{o_i}||^2 \qquad \text{(Eq. 4.9)}$$

where λ_K and λ_B are the relative weights of the error terms. To optimize the above equation for I and K, we employ alternating minimization. Combining Equations (Eq. 4.3) and (Eq. 4.7) yields our iterative update rules:

(i) Update $I^{t+1} = I^t + \sum_{i=1}^{N} K_i^t * (I_{b_i} - I^t \otimes K_i^t) + \lambda_B \sum_{j=1}^{M} h \otimes (u(W(I_{l_j}) - d(I^t \otimes h)))$

(ii) Update $K_i^{t+1} = K_i^t + \widetilde{I}^{t+1} * (I_{b_i} - I^{t+1} \otimes K_i^t) + \lambda_K (K_{o_i} - K_i^t)$

where $\widetilde{I} = I / \sum_{(x,y)} I(x,y)$, $I(x,y) \geq 0$, $K_i(u,v) \geq 0$, and $\sum_{(u,v)} K_i(u,v) = 1$. The two steps are updated in alternation until the change in I falls below a specified level. In our implementation, we set $N = 3$ in correspondence to the current, previous and next frames, and M is set according to the relative camera settings (4/16 for video/image deblurring). We also initialize $I^0 = I_b$ (the currently observed blurred image), $K_i^0 = K_{o_i}$ (the estimated blur kernel from optical flows), and set $\lambda_B = \lambda_K$.

To achieve more stable refinement and flexibility on the kernel refinement, we follows previous work [39, 130] to refine the kernel in multiscale fashion. Figure 4.8 shows an illustration of our kernel refinement step. We refine the kernels from the coarse level by down-sampled the observed images. Results of coarser level are then upsampled and refined again. The multiscale pyramid are constructed using a downsampling factor of $1/\sqrt{2}$ with five levels. Initial guess of kernel is given from the estimated kernels using optical flow and the likelihood $P(K_o|K)$ is also applied at each level of pyramid with weight decreasing so as to allow more flexibilities of refinement at finer level. We note that starting at a level coarser than the low-resolution images allows our method to recover from some error in PSF estimation from optical flows.

(a) (b) (c)

Figure 4.9: Convolution with kernel decomposition. (a) Convolution result without kernel decomposition, where full-sized kernels are generated on-the-fly per-pixel. (b) Convolution using 30 PCA-decomposed kernels. (d) Convolution using delta function decomposition of kernels, with at most 30 delta functions per pixel.

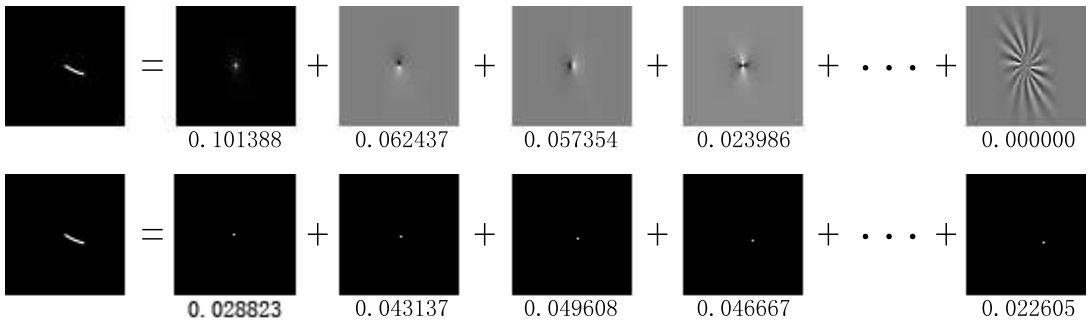


Figure 4.10: Kernel decomposition using PCA verse delta function representation. Top row is the decomposed kernel representation using PCA; The bottom row is the decomposed kernel using delta function representation. The basis kernels are derived from the spatial varying kernels of figure 4.9. We show one of the spatial varying kernels and its decomposition, the weight of each basis kernels are displayed below each of the basis kernels. The delta function representation not only guarantee positive values of basis kernel, but also provide more flexibility on kernel refinement.

4.5.3 Spatially Varying Kernels

A spatially varying blur kernel can be expressed as $K(x, y, u, v)$, where (x, y) is the image coordinate and (u, v) is the kernel coordinate. For large sized kernels, e.g. 65×65, this representation is impractical due to its enormous storage requirements. Recent work has suggested ways to reduce the storage size by constraining the motion path [100]; however, our approach places no constraints on possible motion. Instead, we decompose the spatially varying kernels

into a set of P basis kernels k_l whose mixture weights a_l are a function of image location:

$$K(x, y, u, v) = \sum_{l=1}^{P} a_l(x, y) k_l(u, v). \quad \text{(Eq. 4.10)}$$

The convolution equation then becomes

$$I(x, y) \otimes K(x, y, u, v) = \sum_{l=1}^{P} a_l(x, y)(I(x, y) \otimes k_l(u, v)). \quad \text{(Eq. 4.11)}$$

In [61], principal components analysis (PCA) is used to find the basis kernels. PCA, however, does not guarantee positive kernel values, and we have found in our experiments that PCA-decomposed kernels lead to unacceptable ringing artifacts, exemplified in Figure 4.9. The ringing artifacts in the convolution result resembles the patterns of basis kernels. We propose instead to use a delta function representation, where each delta function represents a position (u, v) within a kernel. Since a motion blur kernel is typically sparse, we store only $30 \sim 40$ delta functions for each image pixel, where the delta function positions are determined by the initial optical flows. This normally results in about $500 \sim 600$ distinct delta functions in total for an entire image, and provides a sufficient approximation of the spatially varying blur kernels in the convolution process. An example of basis kernels decomposition using PCA and delta function representation is shown in figure 4.10. The delta function representation also has more flexibility on kernel refinements, while refinements of a kernel using a PCA representation are limited to the PCA subspace.

Combining Equations (Eq. 4.11) and (Eq. 4.9), our optimization function becomes

$$\begin{aligned} \arg\min_{I,K} \quad & \sum_{i}^{N} ||I_{b_i} - \sum_{l}^{P} a_{il}(I \otimes k_{il})||^2 + \lambda_B \sum_{j}^{M} ||I_{l_j} - d(I \otimes h)||^2 \\ & + \lambda_K \sum_{i}^{N} \sum_{l}^{P} ||a_{il} k_{il} - a_{o_{il}} k_{il}||^2. \end{aligned} \quad \text{(Eq. 4.12)}$$

The corresponding iterative update rules are then

(i) Update $I^{t+1} = I^t + \sum_{i=1}^{N} \sum_{l}^{P} a_{il}^t k_{il} * (I_{b_i} - \sum_{l}^{P} a_{il}^t (I^t \otimes k_{il})) + \lambda_B \sum_{j=1}^{M} h \otimes (u(W(I_{l_j}) - d(I^t \otimes h)))$

(ii) Update $a_{il}^{t+1} = a_{il}^{t} + (\widetilde{I'}^{t+1} * (I'_{b_i} - \sum_l^P a_{il}^t (I'^{t+1} \otimes k_{il}))) \cdot k_{il} + \lambda_K (a_{o_{il}} - a_{il}^t)$

where I' and I'_b are local patches of the estimated result and the blur image. The kernel refinement step can, again, be implemented in multi-scale to allow greater flexibilities and stabilities. The number of delta functions k_{il} stored at each pixel position may be reduced when an updated value of a_{il} becomes insignificant. For greater stability, we process each update rule five times before switching to the other.

4.5.4 Discussion

Utilizing both deconvolution of high-resolution images and back-projection from low-resolution images offers distinct advantages, because the deblurring information from these two sources tend to complement each other. This can be intuitively seen by considering a low-resolution image to be a sharp high-resolution image that has undergone motion blurring with a Gaussian PSF and bandlimiting. Back-projection may then viewed as a deconvolution with a Gaussian blur kernel, and would promote recovery of lower-frequency image features without artifacts. On the other hand, deconvolution of high-resolution images with the high-frequency PSFs typically associated with camera and object motion generally supports reconstruction of higher-frequency details, especially those orthogonal to the motion direction. While some low-frequency content can also be restored from motion blur deconvolution, there is often significant loss due to the large support regions for motion blur kernels, and this results in ringing artifacts. As discussed in [94], the joint use of images having such different blur functions and deconvolution information favors a better deblurring solution.

Multiple motion blur deconvolutions and multiple back-projections can further help to generate high quality results. Differences in motion blur kernels among neighboring frames provide different frequency information; and multiple back-projection constraints help to reduce quantization and the effects of noise in low-resolution images. In some circumstances

there exists redundancy in information from a given source, such as when high-resolution images contain identical motion blur, or when low-resolution images are offset by integer pixel amounts. This makes it particularly important to utilize as much deblurring information as can be obtained.

Our current approach does not utilize priors on the deblurred image or the kernels. With constraints from the low-resolution images, we have found these priors to be unneeded. Figure 4.7 compares our approach with other deconvolution algorithms. Specifically, we compare our approach with Total Variation regularization [30] and Sparsity Priors [65], which have recently been shown to produce better results than traditional Wiener filtering [123] and the Richardson-Lucy [96, 72] algorithm. Both Total Variation regularization and Sparsity Priors produce results with less ringing artifacts. There are almost no ringing artifacts with Sparsity Priors, but many fine details are lost. In our approach, most medium to large scale ringing artifacts are removed using the back-projection constraints, while fine details are recovered through deconvolution.

Although our approach can acquire and utilize a greater amount of data, high-frequency details that have been lost by both motion blur and downsampling cannot be recovered. This is a fundamental limitation of any deconvolution algorithm that does not hallucinate detail. We also note that reliability in optical flow cannot be assumed beyond a small time interval. This places a restriction on the number of motion blur deconvolution constraints that can be employed to deblur a given frame.

Lastly, we note that iterative back-projection technique incorporated into our framework is known to have convergence problems. Empirically we have found that stopping after no more than 50 iterations of our algorithm produces acceptable results.

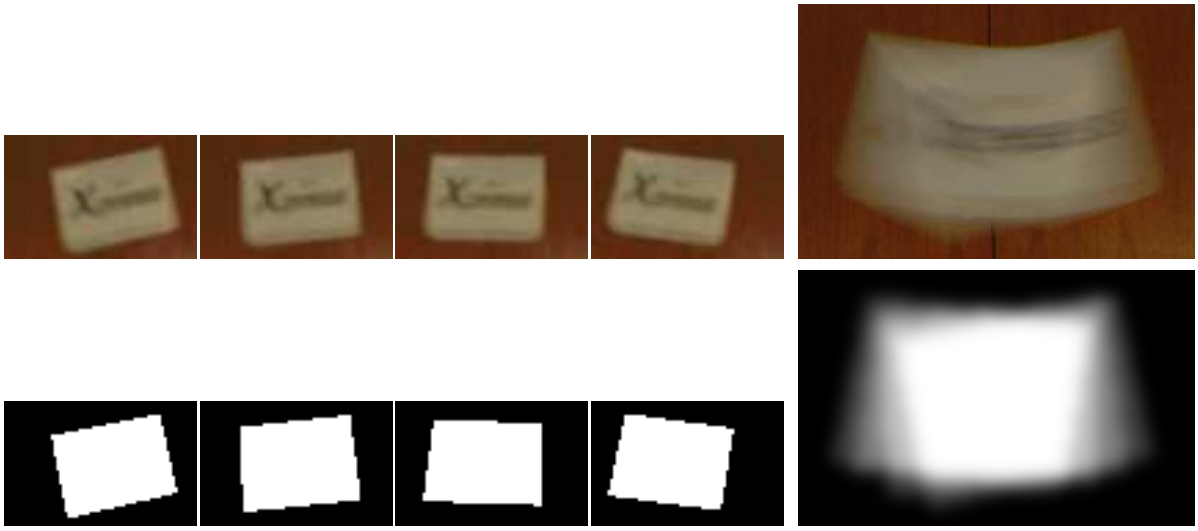


Figure 4.11: Layer separation using a hybrid camera: (a)-(d) Low resolution frames and its corresponding binary segmentation masks. (e) High resolution frame and the matte estimated by compositing the low resolution segmentation masks with smoothing.

4.6 Extension to Deblurring of Moving Objects

In the presence of moving objects (and thus occlusion and disocclusion), the high-resolution image needs to be segmented into different layers, because pixels on the blended boundaries of moving objects contain both foreground and background components, each with different relative motion to the camera. This layer separation is inherently a matting problem:

$$I = \alpha F + (1 - \alpha)B \qquad \text{(Eq. 4.13)}$$

where I is the observed image intensity, F, B and α are the foreground color, background color and alpha to be estimated respectively. The matting problem is an ill-posed problem since the number of variables are more that the number of observation. Single image approaches require user assistance to provide a trimap [23, 22, 105] or scribbles [119, 66, 118] for collecting color samples of foreground and background colors. Fully automatic approaches, however, have required either a blue background [103], multiple cameras with different focus [76], polarized

illumination [77] or a camera array [56]. In this section, we propose a simple solution to the layer separation that takes advantage of the hybrid camera system. As matting is not the primary focus of our work, we offer a brief description of our basic approach to moving object layer separation, and leave a more detailed investigation of this problem for future study.

In a hybrid-camera setup, moving objects should still remain sharp in the high frame rate video. To extract the alpha matte of a moving object, we can perform binary segmentation of the moving object in the low resolution images, and then compose the binary segmentation masks with smoothing to approximate the alpha matte in the high-resolution image. In Figure 4.11, an example of this matte extraction is demonstrated together with the moving object separation method of Zhang *et al.* [132]. We also need to estimate the foreground color F for deblurring. This can be done by assuming local color smoothness prior on F and B and solve it using approach in Bayesian matting [23]:

$$\begin{bmatrix} \Sigma_F^{-1} + \mathbf{I}\alpha^2/\sigma_I^2 & \mathbf{I}\alpha(1-\alpha)/\sigma_I^2 \\ \mathbf{I}\alpha(1-\alpha)/\sigma_I^2 & \Sigma_B^{-1} + \mathbf{I}(1-\alpha)^2/\sigma_I^2 \end{bmatrix} \begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \Sigma_F^{-1}\mu_F + I\alpha/\sigma_I^2 \\ \Sigma_B^{-1}\mu_B + I(1-\alpha)/\sigma_I^2 \end{bmatrix} \quad \text{(Eq. 4.14)}$$

where (μ_F, Σ_F) and (μ_B, Σ_B) are the local color mean and covariance matrix (Gaussian distribution) of foreground and background colors, $\mathbf{I}$ is 3×3 identical matrix and σ_I is standard derivation of I modeling the estimation errors of equation (Eq. 4.13). Given the solution of F and B, the α solution can be further refined by solving equation (Eq. 4.13) in closed form. The refinements of F, B, and α can be done alternatively to produce a better result.

Once the moving objects are separated, we can then deblur each layer separately using our framework. The alpha mattes are also deblurred for compositing, and the occluded background areas revealed after alpha mask deblurring can then be filled in either by back-projection from the low-resolution images or by the motion inpainting method of [75].

4.7 Temporal Super-resolution

Unlike deblurring of images, videos require deblurring of multiple consecutive frames in a manner that preserves temporal consistency. As described in Section 4.5.2, we can jointly use the current, previous and subsequent frames to deblur the current frame in a temporally consistent way. However, after sharpening each individual frame, temporal discontinuities in the deblurred low-frame-rate high-resolution video may become evident, through some "jumpiness" in the sequence. In this section, we describe how our method can alleviate this problem by increasing the temporal sampling rate to produce a deblurred high-frame-rate high-resolution video.

Our solution to temporal super-resolution derives directly from our deblurring algorithm. The deblurring problem is a well-known under-constrained problem since there exist many solutions that can correspond to a given motion blurred image. In our scenario, we have M high-frame-rate low-resolution frames corresponding to each low-frame-rate high-resolution motion blurred image. With our algorithm, we therefore have the opportunity to estimate M solutions using each one of the M low-resolution frames as the basic reference frame. While the ability to produce multiple deblurred frames is not a complete solution to temporal super-resolution, here the use of these M different reference frames leads to a set of deblurred frames that is consistent with the temporal sequence. This unique feature of our approach is gained through the use of the hybrid camera to capture low-resolution high-frame-rate video in addition to the standard high-resolution low-frame-rate video. The low-resolution high-frame-rate video not only aids in estimating the motion blur kernels and provides back-projection constraints, but can also help to increase the deblurred video frame rate. The result is a high-frame-rate high-resolution deblurred video.

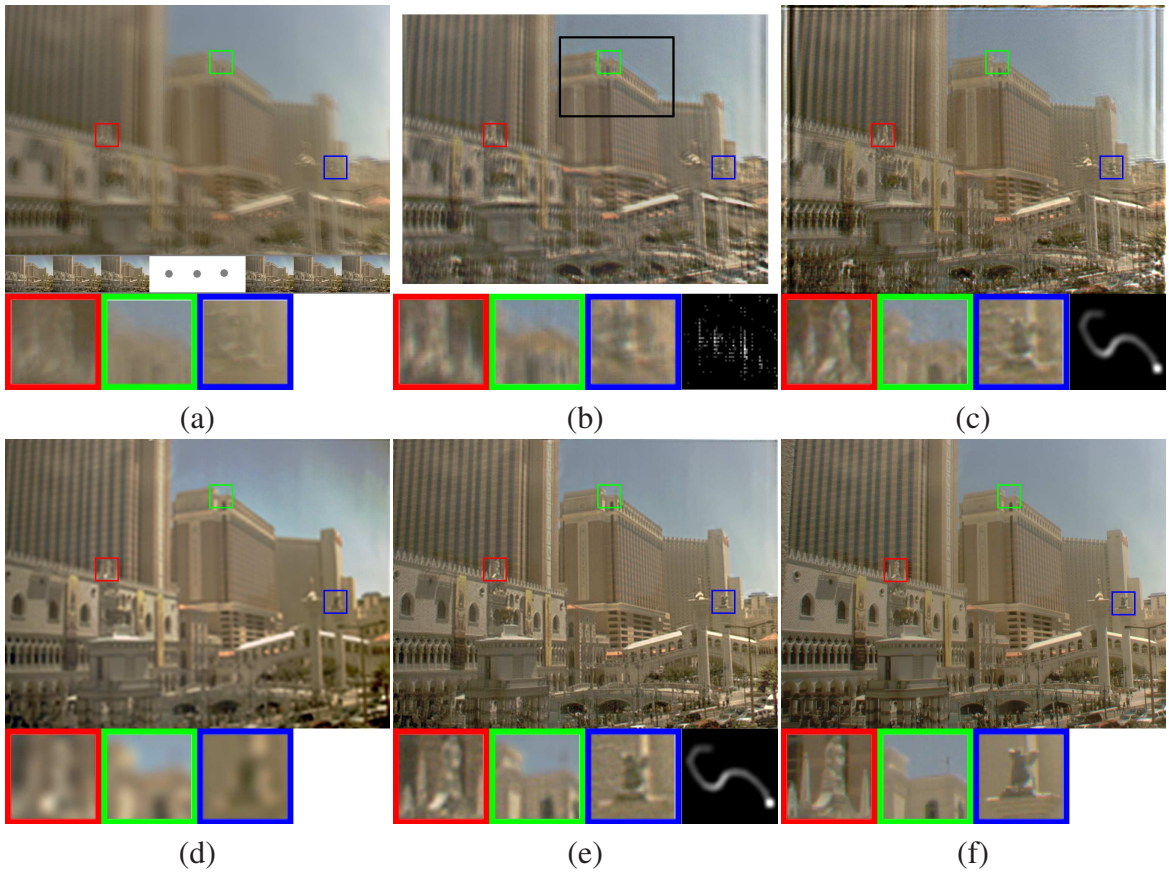


Figure 4.12: Image deblurring using globally invariant kernels. (a) Input. (b) Result generated by [39], where the user-selected regions are indicated by black boxes. (c) Result generated by [7]. (d) Result generated by back projection [52]. (e) Our results. (f) The ground truth sharp image. Close-up views and the estimated global blur kernels are also shown.

4.8 Results and Comparisons

We evaluate our deblurring framework using real images and videos. In these experiments, a ground-truth blur-free image is acquired by mounting the camera on a tripod and capturing a static scene. Motion blurred images are then obtained by moving the camera and/or introducing a dynamic scene object. We show examples of several different cases: **globally invariant motion blur** caused by hand shake, **in-plane rotational motion** of a scene object, **translational motion** of a scene object, **out-of-plane rotational motion** of an object, **zoom-in motion** caused by changing the focal length (i.e. camera's zoom setting), a combination of

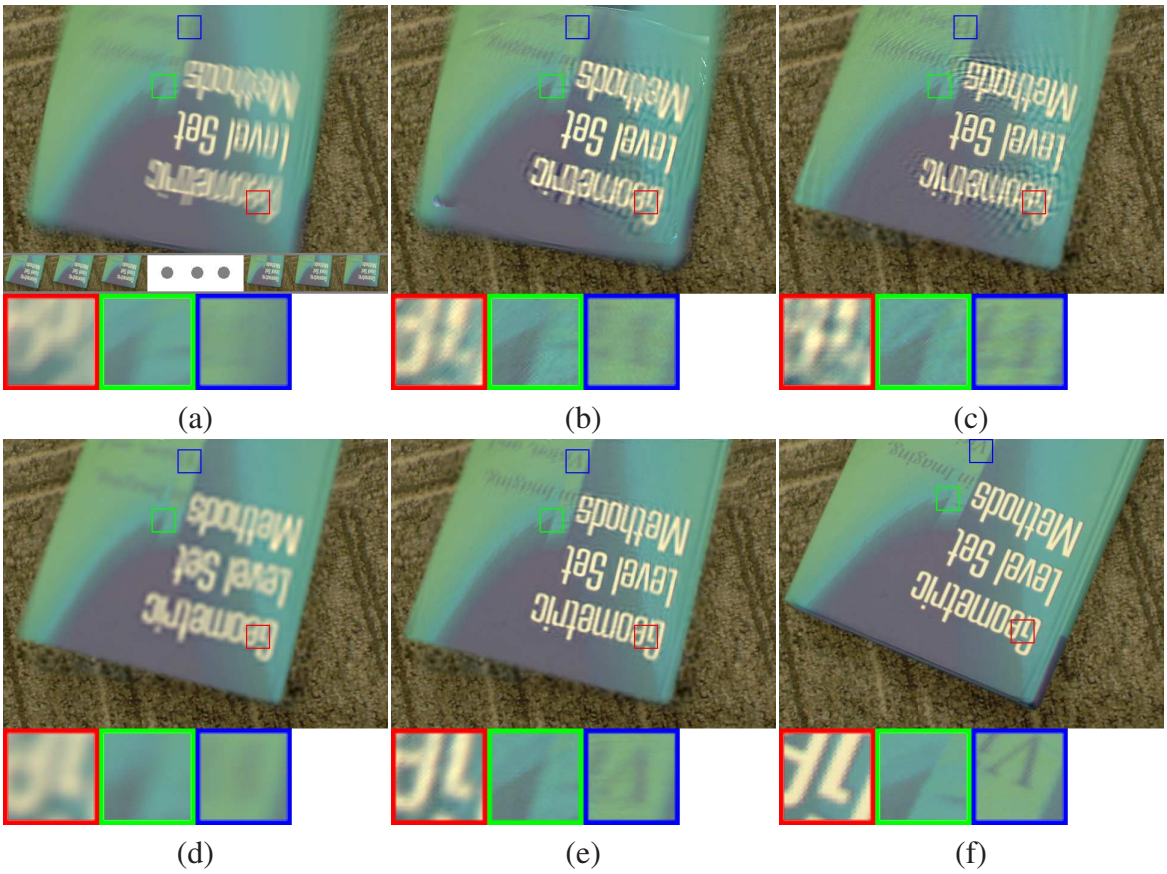


Figure 4.13: Image deblurring with spatial varying kernels from rotational motion. (a) Input. (b) Result generated by [100] (Result is obtained courtesy of the authors of [100]). (c) Result generated by [7] with spatially varying blur kernels estimated from optical flow. (d) Result generated by back projection [52]. (e) Our results. (f) The ground truth sharp image. Close-ups are also shown.

translational motion and rotational motion with **multiple frames** used as input for deblurring one frame, **video deblurring with out-of-plane rotational motion**, **video deblurring with complex in-plane motion**, and **video deblurring with a combination of translational and zoom-in motion**.

[Globally invariant motion blur] In Figure 4.12, we present an image deblurring example with globally invariant motion, where the input is one high-resolution image and several low-resolution images. Our results are compared with those generated by Fergus *et al.* [39], Ben-Ezra and Nayar [7] and back projection [52]. Fergus *et al.*'s approach is a state-of-the-art

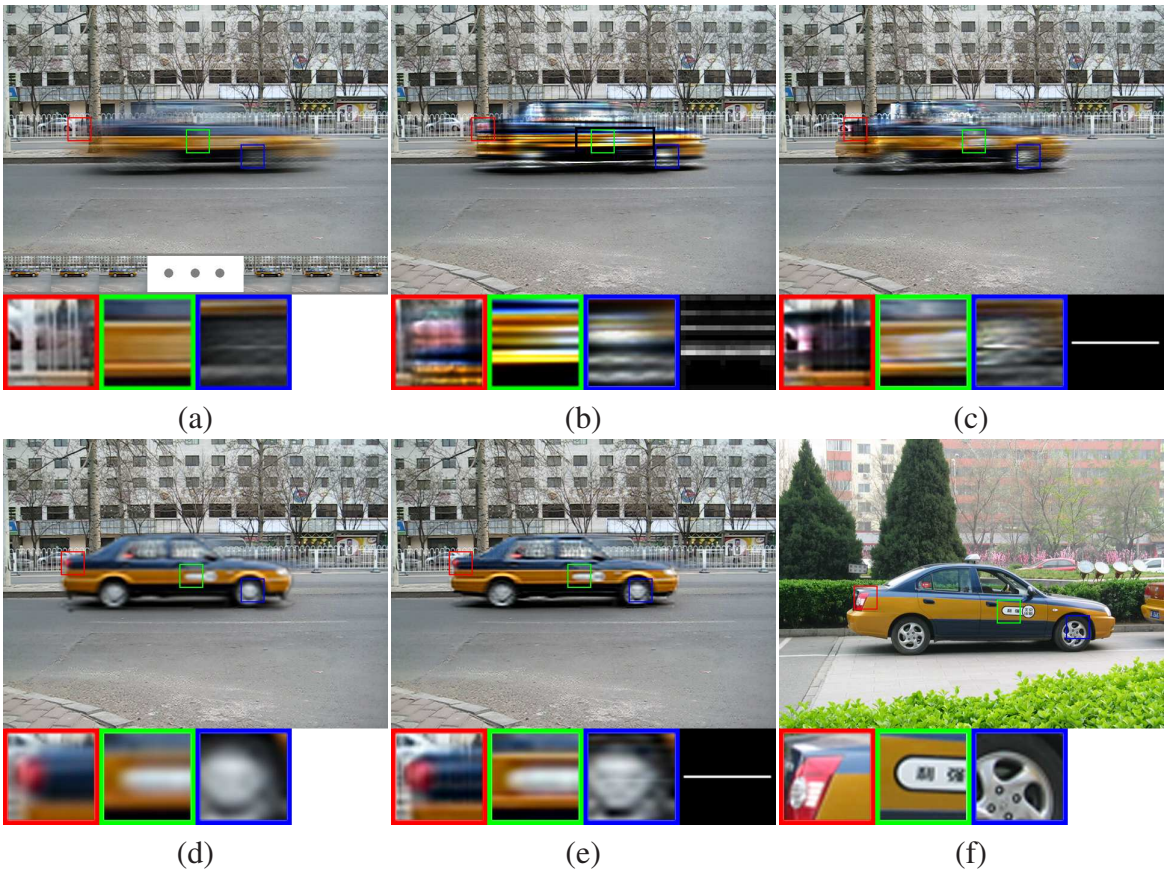


Figure 4.14: Image deblurring with translational motion. In this example, the moving object is a car moving horizontally. We assume the motion blur within the car is globally invariant. (a) Input. (b) Result generated by [39], where the user-selected regions are indicated by black boxes. (c) Result generated by [7]. (d) Result generated by back projection [52]. (e) Our results. (f) The ground truth sharp image captured from another car of same model. Close-up views and the estimated global blur kernels within the motion blur layer are also shown.

blind deconvolution technique that employs a natural image statistics constraint. However, when the blur kernel is not correctly estimated, an unsatisfactory result such as shown in (b) will be produced. Ben-Ezra and Nayar use the estimated optical flow as the blur kernel and then perform deconvolution. Their result in (c) is better than that in (b) as the estimated blur kernel is more accurate, but ringing artifacts are still unavoidable. Back-projection produces a super-resolution result from a sequence of low resolution images as shown in (d). Noting that motion blur removal is not the intended application of back-projection, we can see that

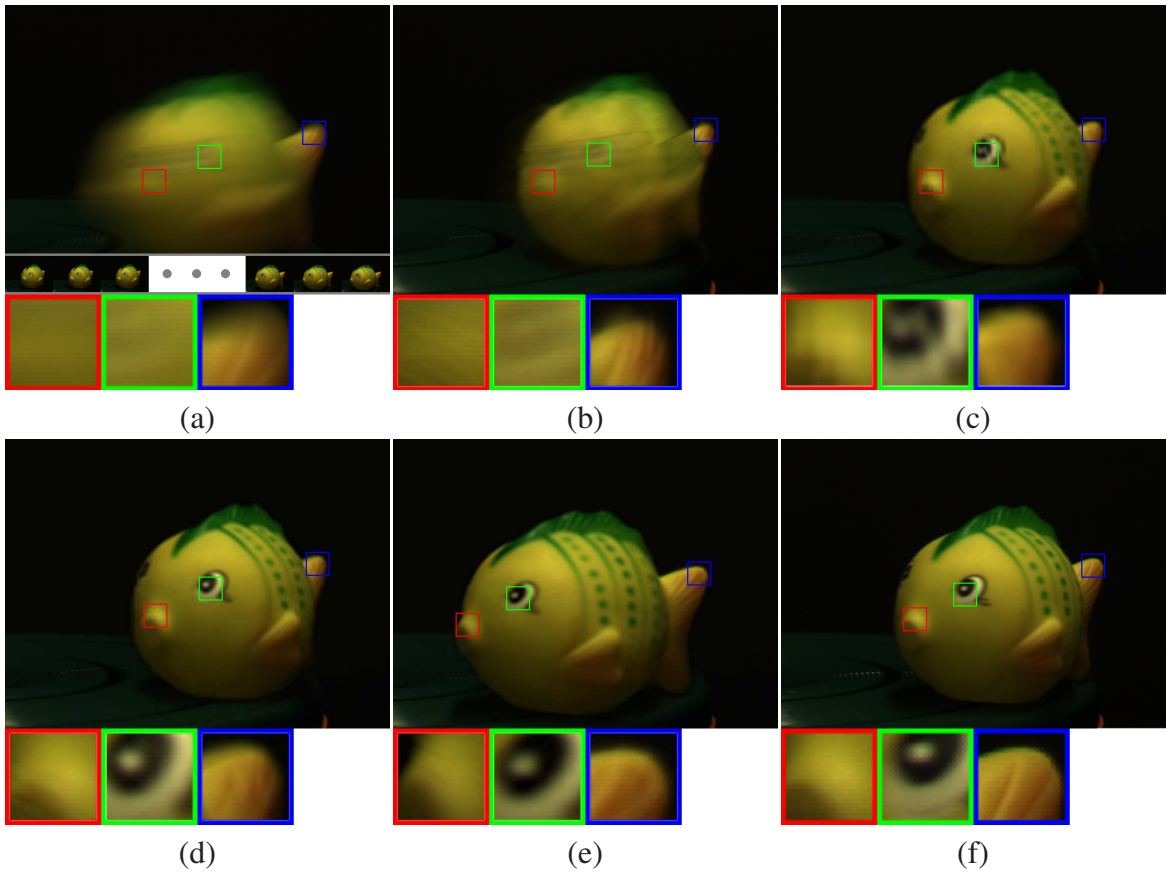


Figure 4.15: Image deblurring with spatial varying kernels. In this example, the moving object contains out-of-plane rotation with both occlusion and disocclusion exists at the object boundary. (a) Input. (b) Result generated by [7]. (c) Result generated by back projection [52]. (d) Our results using the first low resolution frame as the basis frame. (e) Our results using the last low resolution frame as the basis frame. (f) The ground truth sharp image. Close-ups are also shown.

its results are blurry since the high-frequency details are not sufficiently captured in the low resolution images. The result of our method and the refined kernel estimate are displayed in (e). The ground truth is given in (f) for comparison.

[In-plane rotational motion] Figure 4.13 shows an example with in-plane rotational motion. We compared our result with those by Shan *et al.* [100], Ben-Ezra and Nayar [7], and back projection [52]. Shan *et al.* [100] is the most recent techniques targeting to deblur in-plane rotational motion. Our approach is seen to produce less ringing artifacts compared to [100] and

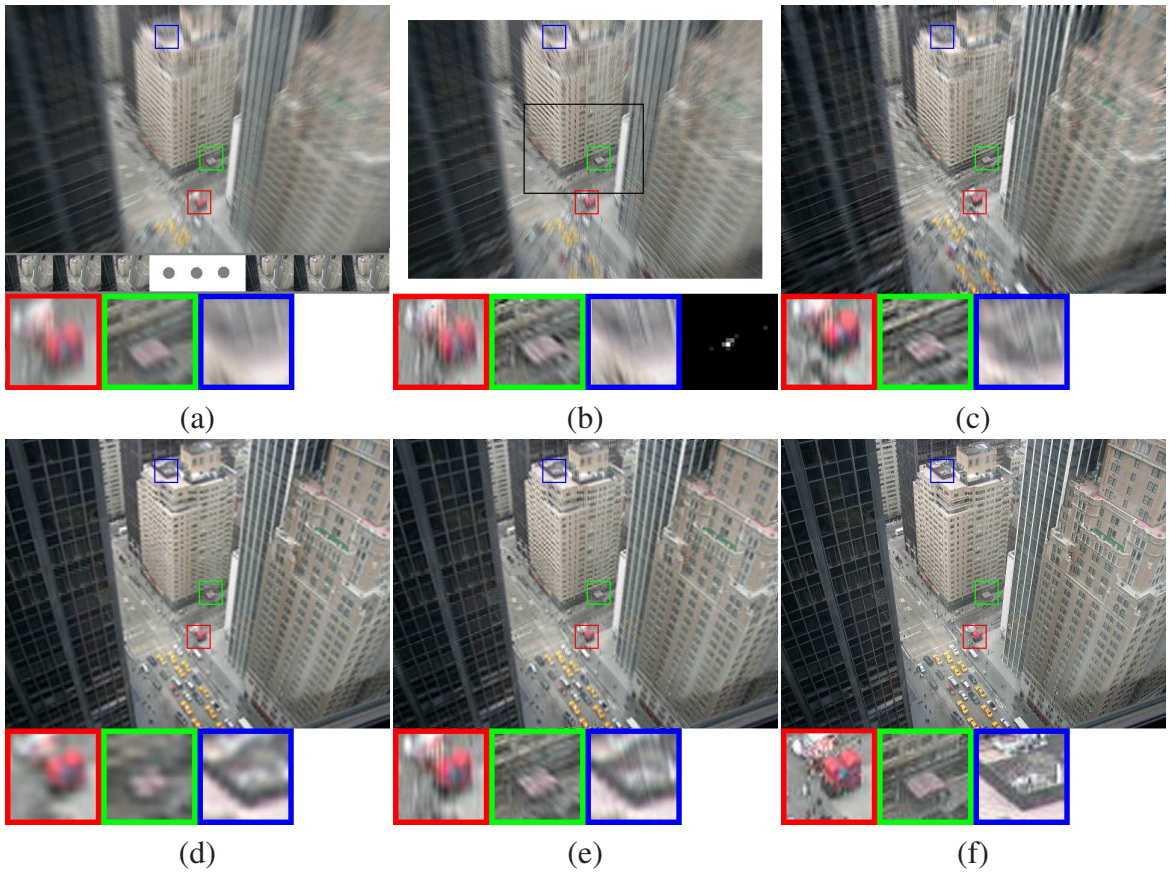


Figure 4.16: Image deblurring with zoom-in motion blur. (a) Input. (b) Result generated by [39]. (b) Result generated by [7]. (c) Result generated by back projection [52]. (d) Our results. (f) The ground truth sharp image. Close-ups are also shown.

[7], and it generates greater detail than [52].

[Translational motion] Figure 4.14 shows an example of translational motion. The moving object is a car moving horizontally. We assume the motion blur within the car is globally invariant and thus techniques for removing globally invariant motion blur can be applied after moving object layer separation. We use the technique proposed in section 4.6 to separate the moving car from static background. Our results is compared with those generated by Fergus *et al.* [39], Ben-Ezra and Nayar [7] and back projection [52]. In this example, the moving car is severely blurred which most of high frequency details are already lost. We demonstrate the limitation of using deconvolution alone which even the motion blur kernel can be estimated

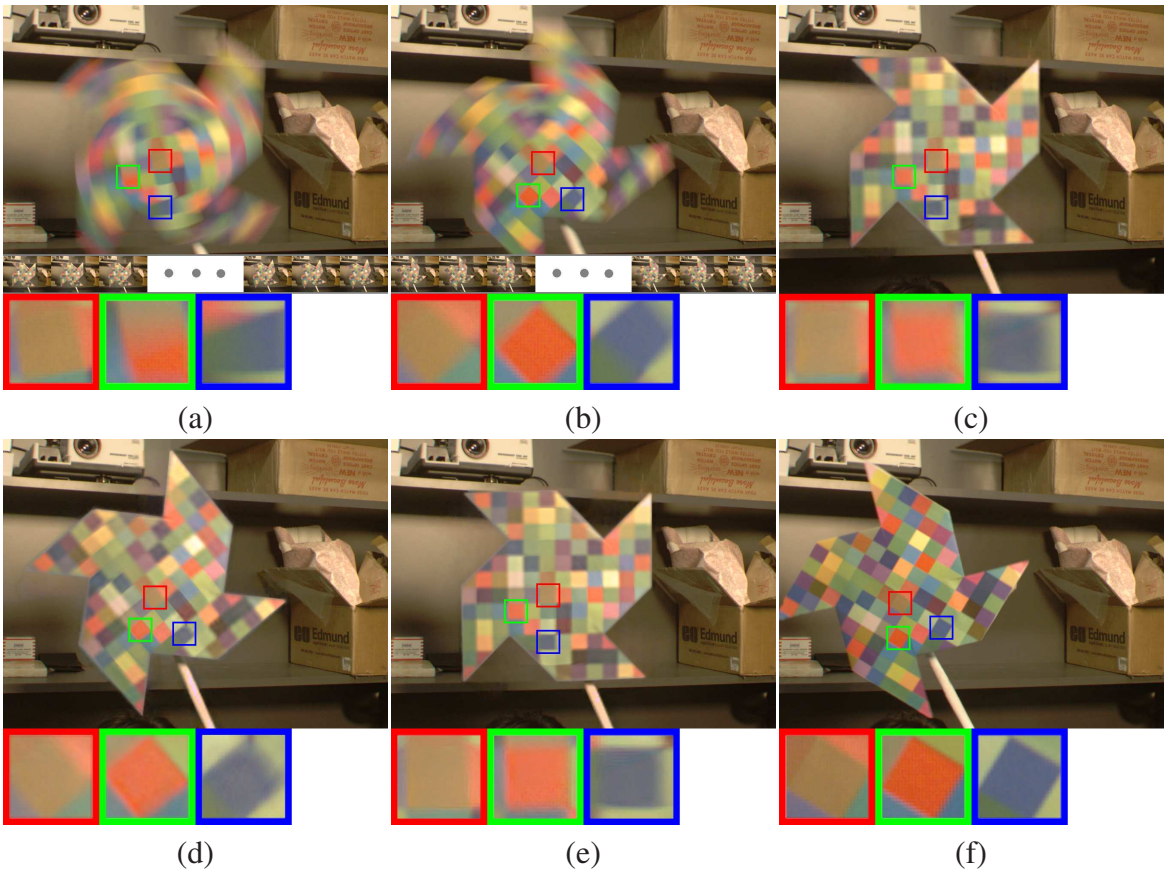


Figure 4.17: Deblurring with and without multiple high-resolution frames. (a)(b) Input images containing both translational and rotational motion blur. (c) Deblurring using only (a) as input. (d) Deblurring using only (b) as input. (e) Deblurring of (a) using both (a) and (b) as inputs. (f) Ground truth sharp image. Close-ups are also shown.

very accurately, unsatisfactory result can still be produced as shown in (c). In this example, the super-resolution result in (d) are better then deconvolution result. But there are still some high-frequency details can not be recovered. Our result is shown in (e) which maintain most low-frequency details recovered by super-resolution and some high frequency details recovered by deconvolution. We notice that some incorrect high frequency details from the static background are also retained in our final result. This is because there were still some incorrect high frequency details from background remained in the separated moving object layer. We believe a better layer separation algorithm that can better separate the foreground and back-

Figure 4.18: Video deblurring with out-of-plane rotational motion. The moving object is a vase with a center of rotation approximately aligned with the image center. First Row: Input video frames. Second Row: Close-ups of a motion blurred region. Third Row: Deblurred video. Fourth Row: Close-ups of deblurred video using the first low-resolution frames as the reference frames. Fifth Row: Close-ups of deblurred video frames using the fifth low-resolution frames as the reference frames. The final video sequence has higher temporal sampling than the original high-resolution video, and is played with frames ordered according to the red lines. Bottom row: Results from space-time super-resolution [34].

ground details could lead to a better result. This example also exhibits a basic limitation of our approach. Since there is significant car motion during the exposure time, most high frequency detail is lost and cannot be recovered by our approach. (f) shows the ground truth captured from another car of same model for comparison.

[Out-of-plane rotational motion] Figure 4.15 shows an example of out-of-plane rotation where occlusion/disocclusion occurs at the object boundary. Our result is compared to Ben-

Ezra and Nayar [7] and back projection [52]. One major advantage of our approach is that we can detect the existence of occlusion/disocclusion of motion blurred moving object. This not only helps to estimate the alpha mask for layer separation, but also helps to eliminate irrelevance regions for back projection. We show our result by choosing the first frame and the last frame as the basic reference frame. This example contains both occlusion and disocclusion simultaneously.

[Zoom-in motion] Figure 4.16 shows another example of motion blur from zoom in effects. Our result is compared to Fergus *et al.* [39], Ben-Ezra and Nayar [7] and back projection [52]. We admit that this is not entirely a fair comparison to Fergus *et al.* [39] since their approach only targets global invariant motion blur. We show the comparison here as to demonstrate the effects of using single blur kernel to deblur spatial varying motion blur. Again, our approach produces a better result with less ringing artifacts and richer in details.

[Deblurring with multiple frames] The benefit of using multiple deconvolutions from multiple high-resolutionframes is exhibited in Figure 4.17, for a pinwheel with both translational and rotational motion. The deblurring result in (c) was computed using only (a) as input. Likewise, (d) is the deblurred result from only (b). Using both (a) and (b) as inputs yields the improved result in (e). This improvement can be attributed to the difference in high-frequency detail that can be recovered from each of the differently blurred images. The ground truth is shown in (f) for comparison.

[Video deblurring with out-of-plane rotational motion] Figure 4.18 demonstrates video deblurring of a vase with out-of-plane rotation. The center of rotation aligns with the image center. The top row displays five consecutive frames of input. The second row shows close-ups of motion blurred region. The middle row shows our results with the first low resolution frame as the basic reference frame. The forth and fifth row shows close-ups of our results with respect to the first and the fifth low resolution frame as the basic reference frame. The information gained from consecutive frames leads to high-quality deblurring results and enhanced temporal

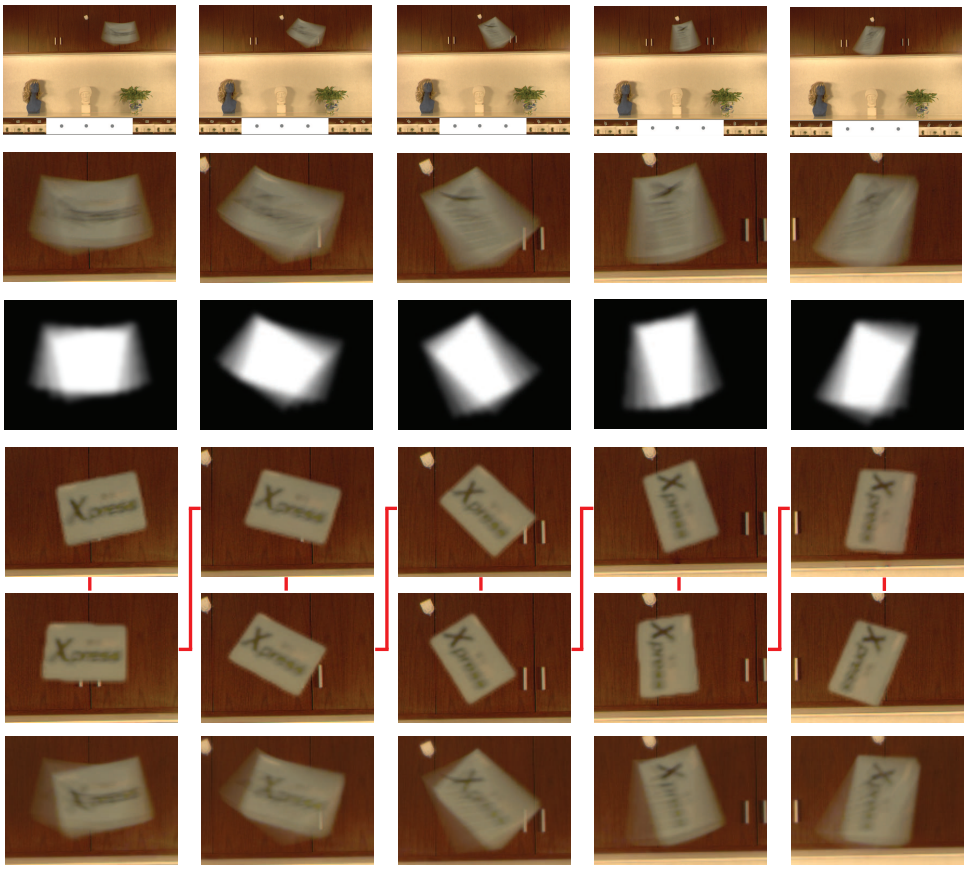


Figure 4.19: Video deblurring with a static background and a moving object. The moving object is a tossed box with arbitrary (in-plane) motion. First Row: Input video frames. Second Row: Close-up of the motion blurred moving object. Third Row: Extracted alpha mattes of the moving object. Fourth Row: The deblurred video frames using the first low-resolution frames as the reference frames. Fifth Row: The deblurred video frames using the third low-resolution frames as the reference frames. The final video with temporal super-resolution is played with frames ordered as indicated by the red lines. Bottom row: Results from space-time super-resolution [34].

consistency. This example also demonstrate the ability of producing multiple solutions as described in section 4.7. To allow temporal super-resolution, we combine the results together as indicated by the red lines in Figure 4.18. We can increase the frame-rate of deblurred high resolution videos up to the same rate as the high-frame-rate low resolution video input.

[Video deblurring with complex in-plane motion] Figure 4.19 demonstrates another video deblurring of a tossed box with arbitrary motion. The top row displays five consecu-

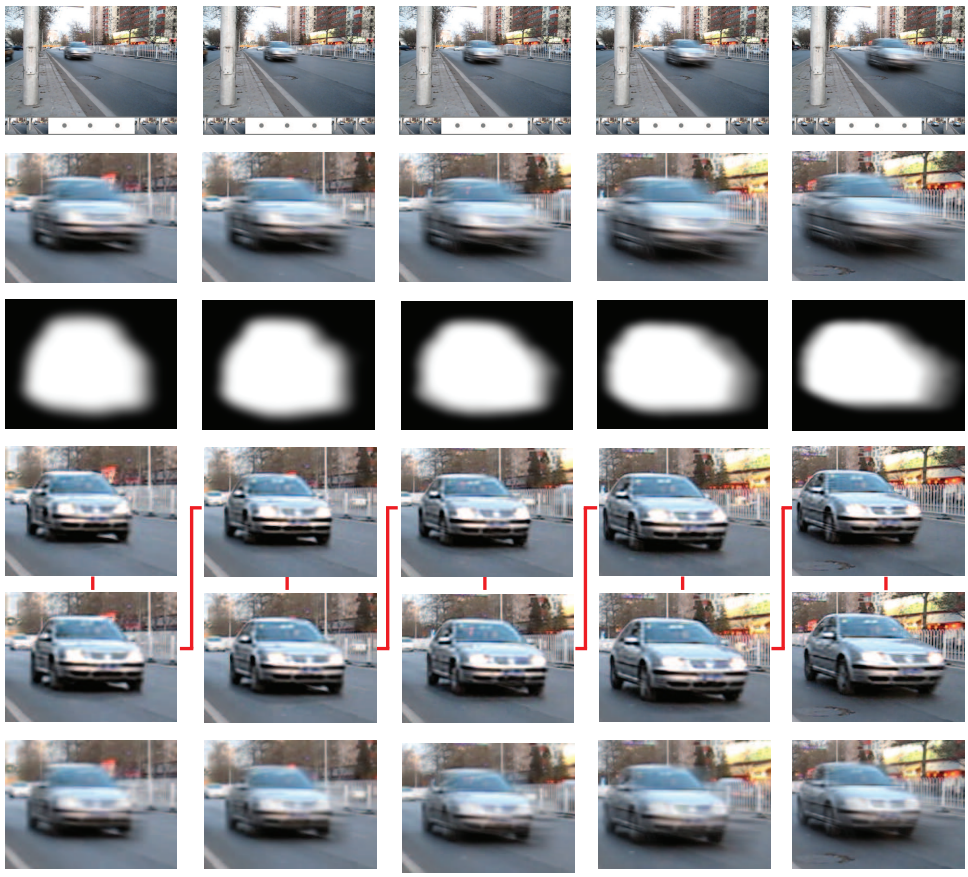


Figure 4.20: Video deblurring in an outdoor scene. The moving object is a car driving towards the camera, which produces both translation and zoom-in blur effects. First Row: Input video frames. Second Row: The extracted alpha mattes of the moving object. Third Row: The deblurred video frames using the first low-resolution frames as the reference frames. Fourth Row: The deblurred video frames using the third low-resolution frames as the reference frames. The final video consists of frames ordered as indicated by the red lines. By combining results from using different low-resolution frames as reference frames, we can increase the frame rate of the deblurred video. Bottom row: Results from space-time super-resolution [34].

tive frames of input. The second row shows close-ups of the motion blurred moving object. The middle row shows our separated mattes for the moving object, and the forth and the last row presents our results with respect to the first and the third low resolution frame as the basic reference frame. Our video deblurring algorithm successfully recovers the text on the tossed box up to certain limit. Similar to the previous video deblurring example, our output is a high frame rate high resolution deblurred video. This result, at the same time, also shows another

limitation, while we can deblur the moving object, the shadows of the moving object are not deblurred, which leads to an inconsistent result. We are still investigating in this direction and we believe more discussions and studies are needed. We leave this as a future research direction.

[Video deblurring with a combination of translational and zoom-in motion] Our final example is shown in Figure 4.20. The moving object is a car moving towards the camera. Both translational effects and zoom-in effects exist in this video deblurring example. The top row displays five consecutive frames of input. The second row shows close-ups of the motion blurred moving object. The middle row shows our separated mattes for the moving object, and the forth and the last row presents our results with respect to the first and the fifth low resolution frame as the basic reference frame. In this example, our focus is to deblur the car in the center of video while other moving objects are leaving untouched. The output is a high frame rate high resolution deblurred video.

4.9 Summary

We have proposed an approach for image/video deblurring using a hybrid camera. Our work has formulated the deblurring process into a Bayesian optimization framework that incorporates optical flow, back projection, kernel refinement, and frame coherency to effectively combine the benefits of both deconvolution and super-resolution. We demonstrate that this approach can produce results that are sharper and cleaner than state-of-the-art techniques.

While our video deblurring algorithm exhibits high-quality results on various scenes, there exist complicated forms of spatially varying motion blur that can be difficult for our method to handle (e.g., motion blur effects caused by object deformation). The performance of our algorithm is also bounded by the performance of several of its components, including optical flow estimation, layer separation and also the deconvolution algorithm. Despite these limitations, we have proposed the first and the only work that can handle spatially varying motion blur

with arbitrary in-plane/out-of-plane rigid motion. This work is also the first to address video deblurring and to increase video frame rates using a deblurring algorithm.

Future research directions for this work include how to improve the deblurring performance through incorporating priors into our framework. Recent deblurring methods have demonstrated the utility of priors, such as the natural image statistics prior and the sparsity prior, for reducing ringing artifacts and for kernel estimation. Another research direction is to improve layer separation by more fully exploiting the available information in the hybrid camera system. Additional future work may also be done on how to recover the background partially occluded by a motion blurred object.

Chapter 5

Summary and Discussion

This chapter concludes this thesis by giving a short summary for each of the previous three chapters discussing soft color segmentation, texture flow estimation, and image/video deblurring. Note that each previously mentioned chapter has a self-contained summary and discussion; this chapter serves to re-iterate those summaries and discussions. In addition, a short discussion on steps and issues to consider when formulating a problem into a Bayesian framework is discussed. Finally, a short description on future research directions concludes this thesis.

5.1 Chapter Summaries

The main goal of this thesis was to demonstrate how various computer vision problems can be addressed using a Bayesian optimization framework. In chapter 1, we discussed the basic properties of the Bayesian formulation and use examples to describe how likelihood probabilities and prior probabilities can be defined. We described common techniques, such as linear regression, alternating optimization (expectation-maximization algorithm) and belief propagation, for solving Bayesian formulated problems. We also gave a brief introduction to our work and its main contributions. Our work on soft color segmentation, texture flow estimation, and image/video deblurring where described in chapter 2, 3 and 4 respectively.

Chapter 2 proposed an alternating optimization frameworks for soft color segmentation. We formulated the problem into a MAP problem whose local optimal solution can be found by using alternating optimization, with update rules defined in chapter 2. We have also discussed the effects of the various parameters in this framework. This soft color segmentation algorithm is a fully automatic method which segments images into regions where region boundaries are fractional instead of binary. Our results are compared against other image segmentation algorithms, including: k-means clustering [32], Mean Shift [26], Expectation-Maximization (EM) [28, 6], Watershed [116], Jseg [29], etc. The usefulness of our algorithm was further demonstrated through various applications, including image matting and color transfer between images.

Chapter 3 proposed a novel texture representation [107] based on filter banks and clustering for texture flow estimation. Our texture flow estimation process was formulated into a discrete labeling problem of a pairwise MRF. The optimal solution was found using belief propagation. The effectiveness of our algorithm is demonstrated through experimental evaluation. Our results are compared with results of Paris *et al.* [82] and Hays *et al.* [48] in real images. We also compared with ground truth in synthetic examples. Both comparisons showed superior results.

Chapter 4 proposed an approach for image/video deblurring using a hybrid camera. Our work formulated the deblurring process as a maximum likelihood (ML) problem, which is a special case of the Bayesian formulation where the prior probability is assumed to be uniform. An iterative method that incorporates optical flow, back-projection, kernel refinement, and frame coherency was used to find a local optimal solution. This approach effectively combined the benefits of both deconvolution and super-resolution, in which ringing artifacts in deblurring process were significantly reduced. We have also extended our approach for spatial varying motion blur and video deblurring. We demonstrated that this approach can produce results that are sharper and clearer than existing state-of-the-art techniques.

5.2 Discussions on Bayesian methods

Having presented three different problems solved using a Bayesian framework and their associated likelihood and prior probabilities, one can realize that the likelihood and prior probabilities are uniquely defined for each problem. In fact, this is one of the advantage of using a Bayesian formulation – it allows researchers to define their own likelihood and prior probabilities based on their understanding on the problem. There are several steps that one can follow to help formulate a problem into a Bayesian framework. These are itemized as follows:

(i) Identify the problem and formulate the goal you want to achieve.

(ii) Understand what observations are obtained from the problem and what assumptions you can made about the problem.

(iii) Define the variables in the problem.

(iv) Write down the Bayesian equations based on the causal relationships between the defined variables.

(v) Define the likelihood probabilities from observations and define the prior probabilities from assumptions. Note that new parameters may need to be introduced in this step.

(vi) Understand the nature of your objective functions and choose a proper method to solve the problem. Some common techniques for solving Bayesian objective functions were described in chapter 1.3.

(vii) Evaluate the results you obtained. If the results are different from your expectations, you may need to reformulate the problem itself, or reconsider your prior probabilities.

The Bayesian method is a mathematical tool to help formulate a problem into a well defined objective function. It is a probabilistic model that describes a problem in terms of beliefs and

degrees of uncertainty. In many situations, the success of a Bayesian model relies on the definition of prior probabilities. As discussed in chapter 1, for the image denoising problem, without prior probabilities the solution would be a trivial solution with $I = I_N$. In many situations, obtaining observations are easy; however, observing the hidden phenomenons or hidden relationships between observations can be challenging. These hidden phenomenons or hidden relationships usually play an important role in the success of an algorithm. To define proper priors for a problem often requires experience and careful thinking. Although there are techniques to help extract these hidden relationships [31, 49], these techniques are mostly based on statistical reasoning in which sufficient amounts of observations are necessary.

5.3 Future Research Directions

There are several future research directions for the work presented in this thesis. One major direction for future work is to consider incorporating better priors into our existing Bayesian formulation of each problem.

For example, in soft color segmentation problem, we can include priors that consider texture pattern or other high level description about the images. Our current approach produces segmentation results without semantic meaning while in many pattern recognition problems, segmentations with semantic meaning are favored.

In texture flow estimation, we can include priors about the changes of color under shadows or shadings. Our current approach assumes the observed "distorted" texture has been normalized with these shadow or shadings effects removed. Through incorporating knowledge about shadows or shadings, our approach would be more reliable in real world scenarios. We can also investigate the possibility of using our proposed texture representation for texture classification.

In the image/video deblurring problem, our current approach is a maximum likelihood approach that does not exploit any prior probability. In the future, we shall investigate how to included priors such such as the natural image statistics/sparisity prior, which are recently

proposed priors for image deblurring. We believe that with these priors included we may produce better results. In addition, we note that the performance of our current approaches is limited by the use of the Richard-Lucy deblurring algorithm [96, 72]. Newer deblurring algorithm, (e.g. [131]) have been recently proposed that could be incorporated into our system to potentially produce better results.

Last but not least, we continue to examine interesting computer vision research problems to see where a Bayesian formulation and optimization can be used to produce good solutions. Current examples under consideration include photometric stereo and optical flow computation.

References

[1] A. Agrawal and R. Raskar. Resolving objects at higher resolution from a single motion-blurred image. In *IEEE Computer Vision and Pattern Recognition*, 2007.

[2] M. Ashikhmin. Synthesizing natural textures. In *ACM Symposium on Interactive 3D Graphics*, 2001.

[3] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, 2002.

[4] J. Bardsley, S. Jefferies, J. Nagy, and R. Plemmons. Blind iterative restoration of images with spatially-varying blur. In *Optics Express*, 2006.

[5] B. Bascle, A. Blake, and A. Zisserman. Motion deblurring and super-resolution from an image sequence. In *European Conference on Computer Vision*, 1996.

[6] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using the expectation-maximization algorithm and its application to content-based image retrieval. In *IEEE International Conference on Computer Vision*, 1998.

[7] M. Ben-Ezra and S.K. Nayar. Motion Deblurring using Hybrid Imaging. In *IEEE Computer Vision and Pattern Recognition*, 2003.

[8] M. Ben-Ezra and S.K. Nayar. Motion-based Motion Deblurring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):689–698, 2004.

[9] O. Ben-Shahar and S. Zucker. The perceptual organization of texture flow: A contextual inference approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):401–417, 2003.

[10] A. Berman, P. Vlahos, and A. Dadourian. Comprehensive method for removing from an image the background surrounding a selected object. *U.S. Patent 6,134,345*, 2000.

[11] J.C. Bezdek and R.J. Hathaway. Convergence of alternating optimization. *Neural, Parallel Science Computing*, 11(4):351–368, 2003.

[12] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, B. Curless, M. Cohen, and S. B. Kang. Using photographs to enhance videos of a static scene. In *Proceedings Eurographics Symposium on Rendering*, 2007.

[13] M. Bigas, E. Cabruja, J. Forest, and J. Salvi. Review of cmos image sensors. *Microelectronics Journal*, 37(5):433–451, 2006.

[14] J. Bigun and J.M.H. du Buf. Geometric image primitives by complex moments in gabor space and the application to texture segmentation. In *IEEE Computer Vision and Pattern Recognition*, 1992.

[15] J. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-021, University of Berkeley, 1997.

[16] M.J. Black and A. Rangarajan. The outlier process: Unifying line processes and robust statistics. In *IEEE Computer Vision and Pattern Recognition*, 1994.

[17] S. Borman and R.L. Stevenson. Super-resolution from image sequences - a review. In *Midwest Symposium on Circuits and Systems*, 1998.

[18] J. Chen and C. K. Tang. Robust dual motion deblurring. In *IEEE Computer Vision and Pattern Recognition*, 2008.

[19] J.L. Chen and A. Kundu. Rotation and gray scale transform invariant texture identification using wavelet decomposition and hidden markov model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):208–214, 1994.

[20] S. Cho, Y. Matsushita, and S. Lee. Removing non-uniform motion blur from images. In *IEEE International Conference on Computer Vision*, 2007.

[21] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *IEEE Computer Vision and Pattern Recognition*, 2001.

[22] Y.Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. *ACM Transactions on Graphics*, 21(3):243–248, 2002.

[23] Y.Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *IEEE Computer Vision and Pattern Recognition*, 2001.

[24] M. Clerc and S. Mallat. The texture gradient equation for recovering shape from texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):536–549, 2002.

[25] F. S. Cohen, Z. Fan, and M. A. Patel. Classification of rotated and scaled texture images using gaussian markov random field models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):192–202, 1991.

[26] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[27] I. Csiszar and G. Tusnady. Information geometry and alternating minimization procedures. *Statistics & Decisions, Supplement Issue*, pages 205–237, 1984.

[28] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximal likelihood from incomplete data via the em algorithm. *Royal Statistics*, B 39:1–38, 1977.

[29] Y. Deng and B.S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001.

[30] N. Dey, L. Blanc-Fraud, C. Zimmer, Z. Kam, P. Roux, J. Olivo-Marin, and J. Zerubia. A deconvolution method for confocal microscopy with total variation regularization. In *IEEE International Symposium on Biomedical Imaging: Nano to Macro*, 2004.

[31] Richard J. Doyle. *Learning causal relations*. Kluwer Academic Publishers, 1986.

[32] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, 2001.

[33] D.F. Dunn, W.E. Higgins, and J. Wakeley. Texture segmentation using 2-d gabor elementary functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):130–149, 1994.

[34] M. Irani E. Shechtman, Y. Caspi. Space-time super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):531–544, 2005.

[35] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *ACM SIGGRAPH*, 2001.

[36] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3):673–678, 2004.

[37] M. Elad and A. Feuer. Superresolution restoration of an image sequence: adaptive filtering approach. *IEEE Transactions on Image Processing*, 8(3):387–395, 1999.

[38] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

[39] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 25(3):787 – 794, 2006.

[40] D.A. Fish, A.M. Brinicombe, E.R. Pike, and J.G. Walker. Blind deconvolution by means of the richardson-lucy algorithm. *Journal of Optical Society America*, 12(1):58 – 65, 1995.

[41] D.A. Forsyth. Shape from texture without boundaries. In *European Conference on Computer Vision*, 2002.

[42] M. Galun, E. Sharon, R. Basri, and A. Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *IEEE International Conference on Computer Vision*, 2003.

[43] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (2nd Edition)*. Prentice Hall, 2002.

[44] S. Gordon, H. Greenspan, and J. Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *IEEE International Conference on Computer Vision*, 2003.

[45] Robert M. Gray. *Entropy and Information Theory*. Springer-Verlag, 2000.

[46] P. Grunwarld. *Minimum Description Length and Maximum Probability*. Kluwer, 2002.

[47] P. C. Hansen, J. G. Nagy, and D. P. OLeary. Deblurring images: Matrices, spectra, and filtering. *Society for Industrial and Applied Mathematic*, 2006.

[48] J. H. Hays, M. Leordeanu, A. A. Efros, and Y.-X. Liu. Discovering texture regularity as a higher-order correspondence problem. In *European Conference on Computer Vision*, 2006.

[49] David Heckerman. A Bayesian Approach to Learning Causal Networks. Technical Report MSR-TR-95-04, 1995.

[50] David Heckerman. A Tutorial on Learning Bayesian Networks. Technical Report MSR-TR-95-06, March 1995.

[51] D. Huawu and D. A. Clausi. Gaussian mrf rotation-invariant features for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):951–955, 2004.

[52] M. Irani and S. Peleg. Improving resolution by image registration. *Computer Vision Graphics and Image Processing*, 53(3):231–239, 1991.

[53] M. Irani and S. Peleg. Motion analysis for image enhancement: Resolution, occlusion and transparency. *Journal of Visual Communication and Image Representation*, 1993.

[54] J. Jia. Single image motion deblurring using transparency. In *IEEE Computer Vision and Pattern Recognition*, 2007.

[55] J. Jia, J. Sun, C.K. Tang, and H.Y. Shum. Bayesian correction of image intensity with spatial consideration. In *European Conference on Computer Vision*, 2004.

[56] N. Joshi, W. Matusik, and S. Avidan. Natural video matting using camera arrays. *ACM Transactions on Graphics*, 25(3):567 – 576, 2006.

[57] R. L. Kashyap and A. Khotanzad. A model-based method for rotation invariant texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):472–481, 1986.

[58] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, 2002.

[59] N. Komodakis. Image completion using global optimization. In *IEEE Computer Vision and Pattern Recognition*, 2006.

[60] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM SIGGRAPH*, 24(3):795 – 802, 2005.

[61] Tod Lauer. Deconvolution with a spatially-variant psf. 4847:167–173, 2002.

[62] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[63] S. Lefebvre and H. Hoppe. Appearance-space texture synthesis. *ACM Transactions on Graphics*, 25(3):541–548, 2006.

[64] A. Levin. Blind motion deblurring using image statistics. In *Neural Information Processing Systems*, 2006.

[65] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 26(3):70, 2007.

[66] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. In *IEEE Computer Vision and Pattern Recognition*, 2006.

[67] A. Levin, P. Sand, T. S. Cho, F. Durand, and W. T. Freeman. Motion-invariant photography. *ACM Transactions on Graphics*, 27(3):71, 2008.

[68] F. Li, J. Yu, and J. Chai. A hybrid camera for motion deblurring and depth map super-resolution. In *IEEE Computer Vision and Pattern Recognition*, 2008.

[69] Y. Liu, W.C. Lin, and J.H. Hays. Near regular texture analysis and manipulation. *ACM Transactions on Graphics*, 23(3):368 – 376, August 2004.

[70] A. Lobay and D.A. Forsyth. Recovering shape and irradiance maps from rich dense texton fields. In *IEEE Computer Vision and Pattern Recognition*, 2004.

[71] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging understanding workshop*, 1981.

[72] L. Lucy. An iterative technique for the rectification of observed distributions. *Astronomy Journal*, 79, 1974.

[73] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *IEEE International Conference on Computer Vision*, 1999.

[74] J. Malik and R. Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, 23(2):149–168, 1997.

[75] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.Y. Shum. Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1150 – 1163, 2006.

[76] M. McGuire, W. Matusik, H. Pfister, J. F. Hughes, and F. Durand. Defocus video matting. *ACM Transactions on Graphics*, 24(3):567–576, 2005.

[77] Morgan McGuire, Wojciech Matusik, and William Yerazunis. Practical, real-time studio matting using dual imagers. In *Eurographics Symposium on Rendering*, 2006.

[78] M. Mirmehdi and M. Petrou. Segmentation of color textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2):142–159, 2000.

[79] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[80] R. Nock and F. Nielsen. Statistical region merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1452–1458, 2004.

[81] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[82] S. Paris, H.M. Briceno, and F.X. Sillion. Capture of hair geometry from multiple images. *ACM Transactions on Graphics*, 23(3):712–719, 2004.

[83] A.J. Patti, M.I. Sezan, and A. Murat Tekalp. Superresolution video reconstruction with arbitrary sampling lattices and nonzero aperture time. *IEEE Transactions on Image Processing*, 6(8):1064–1076, 1997.

[84] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[85] P. Perona. Orientation diffusion. *IEEE Transactions on Image Processing*, 7(3):457–467, 1998.

[86] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3):664–672, 2004.

[87] M. Pietikainen, A. Rosenfeld, and I. Walter. Split-and-link algorithms for image segmentation. *Pattern Recognition*, 15(4):287–298, 1982.

[88] R. Porter and N. Canagrajah. Robust rotation-invariant texture classification: Wavelet, gabor filter and gmrf based schemes. In *IEE Proceedings - Vision, Image, and Signal Processing*, volume 144, 1997.

[89] T. Porter and T. Duff. Compositing digital images. In *ACM SIGGRAPH*, 1984.

[90] T. Randen and J. H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, April 1999.

[91] A. R. Rao and R. Jain. Computerized flow field analysis: Oriented texture fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):693–709, 1992.

[92] A.R. Rao and B.G. Schunck. Computing oriented texture fields. In *IEEE Computer Vision and Pattern Recognition*, 1989.

[93] R. Raskar, A. Agrawal, and J. Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. *ACM Transactions on Graphics*, 25(3):795 – 804, 2006.

[94] A. Rav-Acha and S. Peleg. Two motion blurred images are better than one. *Pattern Recognition Letter*, 26:311–317, 2005.

[95] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001.

[96] W.H. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1), 1972.

[97] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309 – 314.

[98] M. A. Ruzon and C. Tomasi. Alpha estimation in natural images. In *IEEE Computer Vision and Pattern Recognition*, 2000.

[99] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. *ACM Transactions on Graphics*, 27(3):73, 2008.

[100] Q. Shan, W. Xiong, and J. Jia. Rotational motion deblurring of a rigid object from a single image. In *IEEE International Conference on Computer Vision*, 2007.

[101] E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *IEEE Computer Vision and Pattern Recognition*, 2001.

[102] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[103] A.R. Smith and J. F. Blinn. Blue screen matting. In *ACM SIGGRAPH*, 1996.

[104] F. Sroubek, G. Cristobal, and J. Flusser. A unified approach to superresolution and multichannel blind deconvolution. *IEEE Transactions on Image Processing*, 16(9), 2007.

[105] J. Sun, J. Jia, C.K. Tang, and H.Y. Shum. Poisson matting. *ACM Transactions on Graphics*, 23(3):315–321, 2004.

[106] J. Sun, N.N. Zheng, and H.Y. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[107] Y. W. Tai, M.S. Brown, and C.K. Tang. Robust texture flow estimation via dense feature sampling. In *IEEE Computer Vision and Pattern Recognition*, 2007.

[108] Y. W. Tai, J. Jia, and C.K. Tang. Soft color segmentation and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9), 2007.

[109] Y.W. Tai, H. Du, M.S. Brown, and S. Lin. Image/video deblurring using a hybrid camera. In *IEEE Computer Vision and Pattern Recognition*, 2008.

[110] Y.W. Tai, J. Jia, and C.K. Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *IEEE Computer Vision and Pattern Recognition*, 2005.

[111] K.L. Tang, C.K. Tang, and T.T. Wong. Dense photometric stereo using tensorial belief propagation. In *IEEE Computer Vision and Pattern Recognition*, 2005.

[112] M. F. Tappen and W. T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *IEEE International Conference on Computer Vision*, 2003.

[113] D. Tschumperle and R. Deriche. Orthonormal vector sets regularization with pdes and applications. *International Journal of Computer Vision*, 50(12):237–252, 2002.

[114] Z. Tu and S.C. Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):657–673, 2002.

[115] J.K. Udupa, P.K. Saha, and R.A. Lotufo. Relative fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1485–1500, 2002.

[116] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[117] H. Voorhees and T. Poggio. Detecting textons and texture boundaries in natural images. In *IEEE International Conference on Computer Vision*, 1987.

[118] J. Wang, M. Agrawala, and M. Cohen. Soft scissors: An interactive tool for realtime high quality matting. *ACM Transactions on Graphics*, 26(3):9, 2007.

[119] J. Wang and M. Cohen. An iterative optimization approach for unified image segmentation and matting. In *IEEE International Conference on Computer Vision*, 2005.

[120] J. Wang and M.F. Cohen. An iterative optimization approach for unified image segmentation and matting. In *IEEE International Conference on Computer Vision*, 2005.

[121] T.P. Weldon, W.E. Higgins, and D.F. Dunn. Efficient gabor filter design for texture segmentation. *Pattern Recognition*, 29(12):2005–2015, 1996.

[122] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. In *ACM SIGGRAPH*, 2002.

[123] Wiener and Norbert. Extrapolation, interpolation, and smoothing of stationary time series. *New York: Wiley*, 1949.

[124] A.P. Witkin. Scale-space filtering. In *International Joint Conferences on Artificial Intelligence*, 1983.

[125] T.P. Wu, K.L. Tang, C.K. Tang, and T.T. Wong. Dense photometric stereo: A markov random fields approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1830–1846, 2006.

[126] W. R. Wu and S. C. Wei. Rotation and gray-scale transform invariant texture classification using spiral resampling, sub-band decomposition and hidden markov model. *IEEE Transactions on Image Processing*, 5(10):1423–1434, 1996.

[127] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *European Conference on Computer Vision*, 2006.

[128] J. S. Yedidia, W. T. Freeman, and Y. Weiss. *Understanding belief propagation and its generalizations*. Morgan Kaufmann Publishers Inc., 2003.

[129] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[130] L. Yuan, J. Sun, L. Quan, and H.Y. Shum. Image deblurring with blurred/noisy image pairs. *ACM Transactions on Graphics*, 26(3):1, 2007.

[131] L. Yuan, J. Sun, L. Quan, and H.Y. Shum. Progressive inter-scale and intra-scale non-blind image deconvolution. 27(3):1, 2008.

[132] G. Zhang, J. Jia, W. Xiong, T.T. Wong, P.A. Heng, and H. Bao. Moving object extraction with a hand-held camera. In *IEEE International Conference on Computer Vision*, 2007.

[133] J. Zhang, K. Zhou, L. Velho, B. Guo, and H.Y. Shum. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics*, 22(3):295–302, 2003.

[134] L. Zhang and S. M. Seitz. Parameter estimation for mrf stereo. In *IEEE Computer Vision and Pattern Recognition*, 2005.

[135] L. Zhang and S. M. Seitz. Estimating optimal parameters for mrf stereo from a single image pair. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):331 – 342, 2007.

[136] W. Zhao and H. S. Sawhney. Is super-resolution with optical flow feasible? In *European Conference on Computer Vision*, 2002.