Doctorial Thesis

Motion Blur Removal of Digital Photographs

Sunghyun Cho (조 성 현)

Electrical and Computer Engineering

(Computer Science and Engineering)

Pohang University of Science and Technology

2012

디지털 영상의 모션 블러 제거

Motion Blur Removal of Digital Photographs

Motion Blur Removal of Digital Photographs

by

Sunghyun Cho

Division of Electrical and Computer Engineering
(Computer Science and Engineering program)
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Electrical and Computer Engineering (Computer Science and Engineering program).

Pohang, Korea
December 21, 2011
Approved by

Academic Advisor: Seungyong Lee

Motion Blur Removal of Digital Photographs

Sunghyun Cho

The undersigned have examined this thesis and hereby certify that it is worth of acceptance for a Doctorial degree from POSTECH

December 21, 2011

Committee Chair 이 승 용 (Seal)

Member 한 준 희 (Seal)

Member 홍 기 상 (Seal)

Member 박 종 일 (Seal)

Member 강 문 기 (Seal)

DECE 20052465 조 성 현 Sunghyun Cho, Motion Blur Removal of Digital Photographs. 디지털 영상의 모션 블러 제거, Division of Electrical and Computer Engineering (Computer Science and Engineering program), 2012, 117P, Advisor: Seungyong Lee. Text in English.

ABSTRACT

Motion blur is a common artifact that produces disappointing blurry images with inevitable information loss. It is caused by the nature of imaging sensors that accumulate incoming lights for an amount of time to produce an image. During exposure, if the camera sensor moves or objects move, a motion blurred image will be obtained.

Deblurring is to find the latent sharp image from a given blurred image. Formally, motion blur has been often modeled using a convolution based blur model, where a blurred image is the convolution result between a latent sharp image and a blur kernel. Then, deblurring becomes a deconvolution problem. In *non-blind* deconvolution, the motion blur kernel is given, and the problem is to recover the latent image from a blurry version using the kernel. In *blind* deconvolution, the kernel is unknown and the problem is to estimate the blur kernel as well as the latent image.

While deblurring has been studied extensively my many researchers, it is still very challenging. The major difficulties can be summarized as follows: first there is missing information. Motion blur causes irreversible information loss. Moreover, the blur kernel is unknown in most cases, which makes the problem more difficult. Second, while the convolution based blur model has been widely used, it often does not hold in practice. In real cases, images are usually non-uniformly blurred, e.g., due to object motions and rotational camera shakes. Noise and outliers, e.g., saturated pixels and sensor defects, also severely degrade the performance of deblurring methods.

In this thesis, we present software-based solutions to overcome the difficulties mentioned above. Specifically, the thesis includes the following topics:

Fast uniform motion deblurring from a single blurred image Previous blind deblurring methods need a huge computation time. In this work, we propose a fast blind deconvolution method,

which produces a deblurring result in only a few seconds. The experimental results show that our method is not only faster but also more reliable than previous methods.

Handling outliers in non-blind image deconvolution Outliers, such as sensor defects and saturated pixels, cause severe artifacts in previous methods. In this work, we explicitly model outliers and derive an Expectation-Maximization based non-blind deconvolution method.

Blind deconvolution methods for non-uniform motion blur Although the uniform blur model has been widely used, real-photographed images usually have non-uniform blur caused by object motions and rotational camera shakes. In this work, we propose two methods for handling non-uniform motion blur.

Video motion deblurring Motion blurred video can significantly degrades the performance of computer vision algorithms. We use unblurred frames to achieve very reliable video deblurring results.

Contents

Chapter 1

Introduction

Motion blur is the streaking of rapidly moving objects in a still picture or an image sequence. Sometimes, it is considered as a good feature that makes viewers feel speed of objects. For example, people can feel breathtaking tension on the field of war through a shaken picture of a soldier landing on Omaha beach, Normandy, France during the World War II (Figure 1.1). However, in most cases, motion blur is one of the most annoying artifacts that photographers want to avoid, as it severely degrades image quality with inevitable information loss. Unfortunately, it is unavoidable due to the nature of the camera that accumulates incoming lights. The camera sensors need to be exposed during a certain amount of time to capture a scene, and in a dark environment where the amount of light is not enough, such as in a dark room or at night, the sensors need to be exposed longer. During the exposure, camera shakes or object motions cause motion blur (Figure 1.2).

Deblurring is a problem that finds the latent sharp image from a given blurred image (Figure 1.3). Many previous works have been often modeled motion blur using a convolution operation as:

$$b = k * l + n, \tag{1.1}$$

where b is a blurred image, k is a motion blur kernel or a point spread function (PSF), l is a latent image, and n is unknown noise introduced during image acquisition. $*$ is a convolution operator. Then, deblurring becomes a deconvolution problem, an inverse problem of convolution. According to whether the motion blur kernel is given or not, deconvolution problems are categorized into two

Figure 1.1: A picture of a soldier landing on Omaha beach, Normandy, France in D-day, taken by Robert Capa. Motion blur in this picture reveals breathtaking tension on the war field.

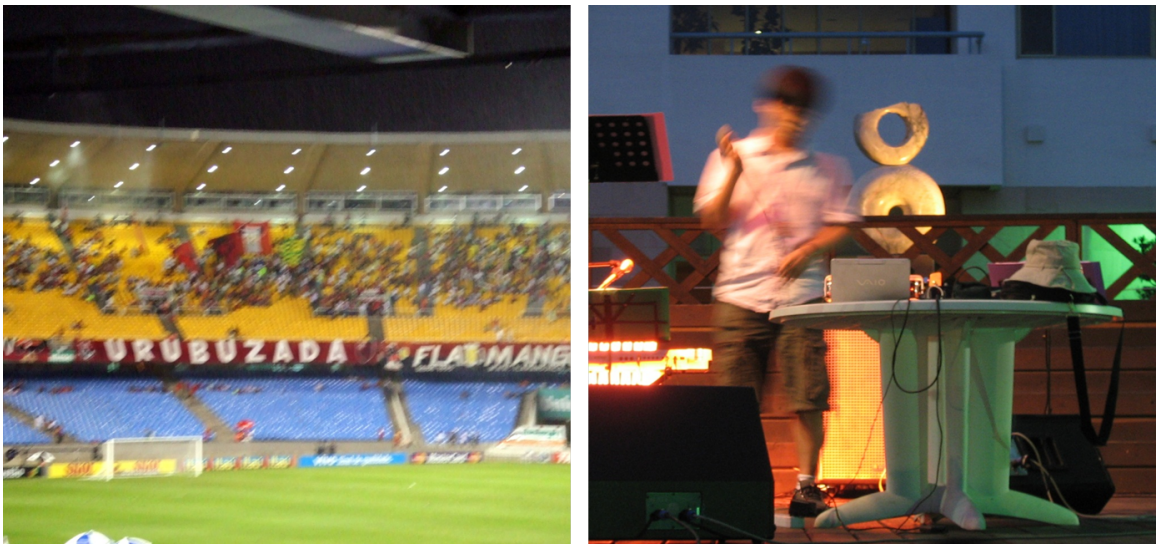


Figure 1.2: Blurred images. Motion blur is usually one of the most annoying artifacts that degrade image quality. Left: blurred by camera shakes, right: blurred by object motion.

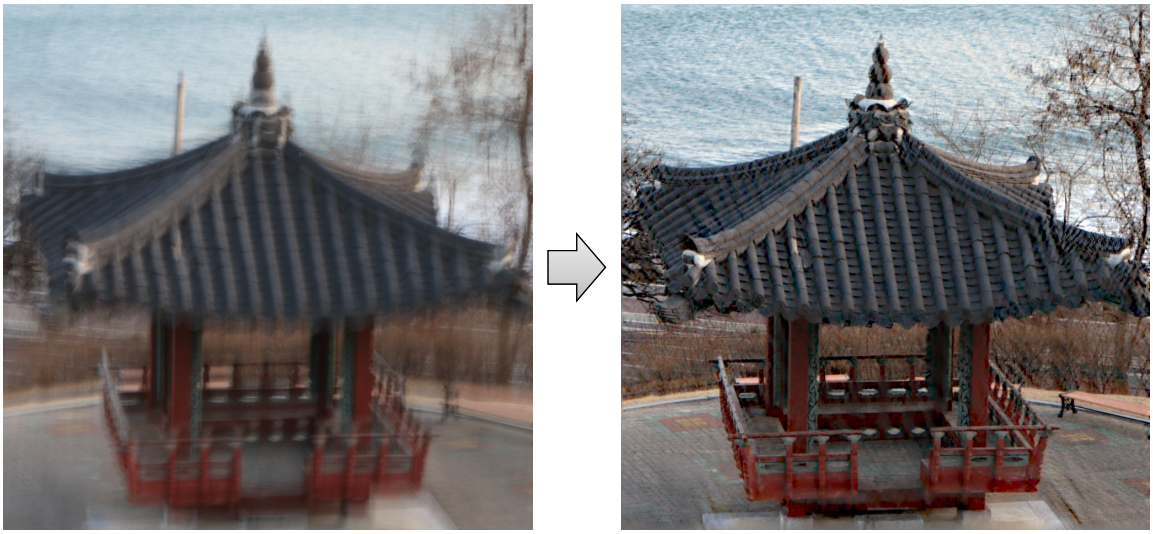


Figure 1.3: Deblurring is a problem that finds the latent sharp image from a given blurred image.

categories. In *non-blind* deconvolution, the motion blur kernel k is given, and the problem is to find the latent sharp image l from the given motion blur kernel k and the blurred image b. In *blind* deconvolution, the kernel k is unknown, and the problem is to find the latent sharp image l and the kernel k from the given blurred image b.

Due to its practical importance, deblurring has been studied extensively by many researchers for decades. For example, the concept of the widely known Wiener filter was first proposed in 1942 [57]. However, it is still a very challenging problem. We can summarize the major difficulties in deblurring as follows: First, in non-blind deconvolution, due to the irreversible information loss caused by motion blur, it is impossible to perfectly restore original information. Second, in blind deconvolution, the number of unknowns (k and l) exceeds the number of observations (b), thus the problem is a severely under-constrained problem. Moreover, blind deconvolution inherently includes problems caused by non-blind deconvolution as many blind deconvolution systems use on non-blind deconvolution as a component. Third, while the convolution based blur model (Eq.

(1.1)), where every pixel in an image is blurred in the same way, has been widely used for previous works, a real blurred image are usually non-uniformly blurred, i.e., an image is degraded by locally different blurs, e,g., due to object motions and camera rotations. Lastly, noise and outliers, e.g., saturated pixels and sensor defects, can severely degrade the performance of blind and non-blind deconvolution methods.

In this thesis, we propose software-based solutions to overcome some of the difficulties in blind and non-blind deconvolution. We first propose blind and non-blind deconvolution methods for uniform motion blur. The proposed methods are highly practical in terms of quality and computational cost compared to previous methods. We also propose two blind deconvolution methods for non-uniform motion blur, one of which is designed for handling non-uniform blur caused by object motions and depth difference, and the other one is designed for handling non-uniform blur caused by non-translational camera motions. Lastly, we propose a deblurring method for blurry video frames. The followings are detailed introduction of the contributions of this thesis.

Fast uniform motion deblurring from a single blurred image Thanks to the recent progress in deblurring, there have been proposed several blind deconvolution methods that can produce high-quality results. However, they heavily depend on complex nonlinear optimization techniques and need more than several minutes, and even in some cases, more than an hour for deblurring a single image of moderate size, which makes the methods far from practical. In Chapter 2, we propose a fast blind deconvolution method, which can estimate a blur kernel from a given blurred image in only a few seconds. To achieve high speed, the proposed method adopts a prediction scheme, which predicts gradient maps of the latent sharp image efficiently and effectively. By combining the prediction scheme and a simple non-blind deconvolution method, we avoid complex nonlinear optimization while still being able to estimate the latent image reliably. We also introduce a kernel estimation method based on image gradient values instead of pixel values, which can estimate a blur kernel more reliably and quickly. The experimental results show that the proposed method is not

only faster but also more reliable than the previous methods.

Handling outliers in non-blind image deconvolution Previous non-blind deconvolution methods usually assume a small amount of noise, which can be modeled by a Gaussian distribution. However, this assumption does not hold in practice. For example, sensor defects and saturated pixels break this assumption, and cause severe ringing artifacts. In Chapter 3, we propose an outlier handling method in non-blind image deconvolution. To handle outliers, we first analyze the sources of outliers, which break the previous blur model, and explicitly model outliers in the image formation process. Based on the new model, we derive an Expectation-Maximization (EM) based alternating non-blind deconvolution method.

Blind deconvolution methods for non-uniform motion blur Although uniform motion blur model, where every pixel in an image is blurred in the same way, has been widely used for previous works, in real cases, photographed images usually have non-uniform motion blur caused by depth difference, moving objects, and non-translational camera shakes such as rotational motions. In Chapter 4, we propose a blind deconvolution method for handling non-uniform motion blur based on segmentation. The proposed method assumes that a blurred image is blurred by piecewise uniform blurs, which can be caused by moving objects and depth difference. The method simultaneously estimates multiple motions, uniform motion blur kernels, and the associated image segments from multiple blurred images.

While the method in Chapter 4 is able to handle non-uniform motion blur caused by moving objects and depth difference, it cannot handle non-uniform motion blur caused by non-translational camera shakes. In order to remove such motion blur, we propose a blind deconvolution method based on the projective motion blur model [54] in Chapter 5. The projective motion blur model uses a set of homographies to describe motion blur caused by camera shakes, i.e., a blurred image is the sum of differently warped versions of a latent image by different homographies. The proposed method alternatingly estimates homographies from given blurred images to estimate motion blur.

Compared to the segmentation-based blind deconvolution method mentioned above, the proposed method cannot handle non-uniform motion blur caused by object motions, but is more proper to handle non-uniform motion blur caused by camera shakes.

Video motion deblurring As well as images, video frames can be also severely degraded by motion blur caused by hand shakes. While it is hard to notice motion blur in video frames when just watching a video, motion blur can significantly degrade the quality of computer vision algorithms such as video stabilization techniques. It is, however, hard to remove motion blur from video frames using the existing methods, which are designed for deblurring images, because such methods cannot reliably deblur every single frame. To overcome this, we propose a method for removing non-uniform motion blur from a video sequence in Chapter 6. The proposed method avoids direct kernel estimation and deconvolution to video frames, and instead, uses unblurred information from neighboring frames. As a result, our method is able to recover sharp video frames without introducing ringing artifacts even when moving objects present.

1.1 Notations

In the remaining of this thesis, we shall denote images using a few different notations. As images can be represented as a 2-dimensional functions, we will use $f(x, y)$ for representing an image, where x and y are scalar values for horizontal and vertical indices, respectively. As we do not need to separately deal with x and y in most cases, we shall more frequently denote an image by $f(\mathbf{x})$ where $\mathbf{x}$ is a 2D vector such that $\mathbf{x} = (x, y)^T$. For brevity, we shall omit the pixel indices (x, y) and use simply f as already used in Chapter 1 whenever possible. We will also use a vector form for representing images, as it is more convenient to derive the actual optimization process in many cases. In such cases, we will use $\mathbf{f}$ for representing a vector consisting of all pixel values of f.

Chapter 2

Fast Uniform Motion Deblurring from a Single Blurred Image

Single-image blind deconvolution is an ill-posed problem because the number of unknowns exceeds the number of observed data. Early approaches imposed constraints on motion blur kernels and used parameterized forms for the kernels [13, 11, 59, 44]. Recently, several methods were proposed to handle a more general motion blur given a single image [18, 28, 48]. While these methods can produce excellent deblurring results, they necessitate intensive computation. It usually takes more than several minutes for the methods to deblur a single image of moderate size.

Most blind deconvolution methods take an iterative process that alternatingly optimizes the motion blur kernel and the latent image. In the process, the blur kernel is obtained from the estimated latent image and the given blurred image. The kernel is then used to estimate the latent image by applying non-blind deconvolution to the given blurred image. The new estimated latent image is used for kernel estimation in the next iteration. The intensive computation of previous methods stems from the complicated methods used for kernel estimation and latent image estimation. Optimization involving large matrices and vectors is needed for kernel estimation, and sophisticated optimization techniques are necessary to handle non-blind deconvolution with non-linear priors.

This chapter presents a fast blind deconvolution method that produces a deblurring result from a single image in only a few seconds. The high speed of our method is enabled by accelerating both

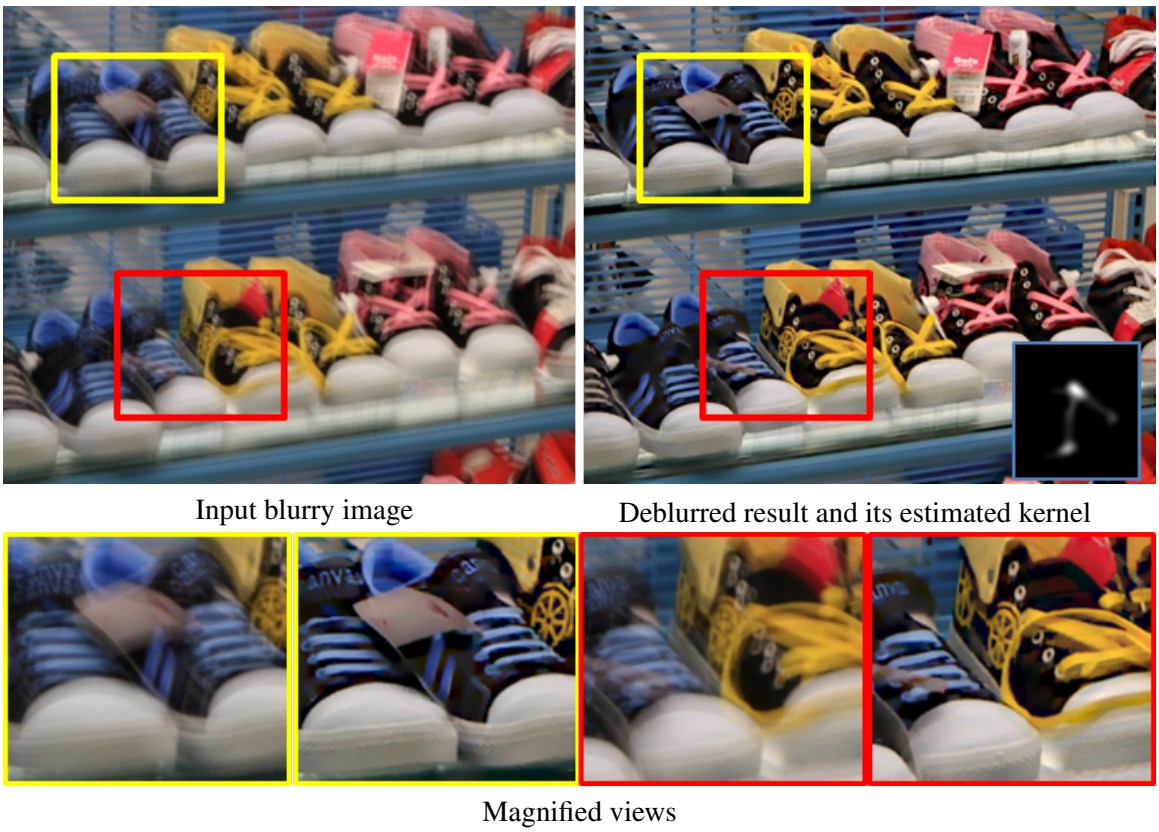


Figure 2.1: Fast single image deblurring. Our method produces a deblurring result from a single image very quickly. Image size: 713×549. Motion blur kernel size: 27×27. Processing time: 1.078 seconds.

kernel estimation and latent image estimation steps in the iterative deblurring process. Our method produces motion deblurring results with comparable quality to previous work, but runs an order of magnitude faster. A C++ implementation of our method usually takes less than one minute to deblur an image of moderate size, which is about 20 times faster than the C-language implementation of the previous method [48]. A GPU implementation of our method further reduces the processing time to within a few seconds, which is fast enough for practical applications of deblurring.

To accelerate latent image estimation, we introduce a novel prediction step into the iterative deblurring process. Strong edges are predicted from the estimated latent image in the prediction step

and then solely used for kernel estimation. This approach allows us to avoid using computationally inefficient priors for non-blind deconvolution to estimate the latent image. Small deconvolution artifacts could be discarded in the prediction step of the next iteration without hindering kernel estimation. For non-blind deconvolution, we use a simple method with a Gaussian prior, which can be computed quickly using Fourier transforms. Since only simple image processing techniques are used in the prediction step, the combination of simple non-blind deconvolution and prediction can efficiently obtain an estimated latent image used for kernel estimation.

For kernel estimation, we formulate the optimization function using image derivatives rather than pixel values. We take a conjugate gradient (CG) method to solve the numerical system derived from the optimization function, and use Fourier transforms to calculate the gradient needed for the CG method. Working with image derivatives allows us to reduce the number of Fourier transforms from twelve to two, saving $5/6$ of the computation required for calculating the gradient. In addition, we show that the condition number of the numerical system using image derivatives is smaller than that using pixel values, which enables the CG method to converge faster. Consequently, the numerical optimization process for kernel estimation is significantly accelerated in our method.

The contributions of this chapter can be summarized as follows.

- We propose a fast method for single-image blind deconvolution, which deblurs an image of moderate size within a few seconds.
- We accelerate latent image estimation in the iterative deblurring process, without degrading the accuracy of kernel estimation, by combining a prediction step with simple non-blind deconvolution.
- We achieve significant acceleration of the numerical optimization for kernel estimation with formulation using image derivatives rather than pixel values.
- The acceleration technique for kernel estimation can be adopted to other deblurring methods for performance improvement.

2.1 Related Work

Blind deconvolution is a severely ill-posed problem as the number of unknowns (a latent image and a blur kernel) exceeds the number of observed data (a blurred image). To reduce the ill-posedness of blind deconvolution, many previous approaches made restrictive assumptions on a motion blur, such as a parametrized linear motion in a single or multiple images. Chen et al. [13] used two consecutively blurred images to estimate a motion blur. Rav-Acha and Peleg [44] used horiznotally and vertically blurred images to help reconstruct details. For a single image input, Yitzhaky et al. [59] made an isotropy assumption to estimate a 1D directional motion blur. Shortly afterwards, they have extended their method to account for a high frequency motion blur caused by vibration that is observed along with the major 1D directional motion blur [58]. Ji and Liu [27] proposed a method to handle more general types of blur, such as 1D accelerated motions. Due to the assumption of simple parametric blur kernels, these approaches cannot handle the high diversity of a 2D motion blur.

Recently, several algorithms were proposed to handle a more general motion blur. Fergus et al. [18] used a variational Bayesian method with natural image statistics to estimate a non-parametric uniform blur kernel. However, with the complexity of their statistical model, it takes a relatively long time to estimate a PSF, even on a small image patch. Jia [28] used an alpha matte that describes transparency changes caused by a motion blur for kernel estimation. The method largely depends on the quality of the input alpha matte. Yuan et al. [60] used two images for motion deblurring, of which one is noisy but has sharp edges, and the other is motion blurred. Shan et al. [48] introduced an effective deblurring method based on alternating estimation of a blur kernel and a latent image. Their method however still needs several minutes to deblur an image. Levin et al. [37] analyzed previous blind deconvolution methods and quantitatively evaluated the methods of Fergus et al. [18] and Shan et al. [48] using a data set. We refer readers to [37] for a review of blind motion deblurring algorithms.

Hardware approaches have also been introduced for image deblurring. Ben-Ezra and Nayar [8] proposed a hybrid imaging system, where a high-resolution camera captures the blurred frame and a low-resolution camera with a faster shutter speed is used to estimate the camera motion. Levin et al. [36] introduced a system for capturing a uniformly blurred image by controlling the camera motion even if the objects have different 1D movements.

There have been image deblurring methods that include image filtering similar to our prediction step. Money and Kang [42] used a shock filter to find sharp edges, and estimated a motion blur kernel using the filtered image. As a simple parametric blur kernel is assumed, this method cannot handle a more complicated blur. Joshi et al. [32] predicted sharp edges using edge profiles and estimated motion blur kernels from the predicted edges. However, their goal is to remove small blurs, which can be described with single peaks. Independently of our work, the PhD dissertation of Joshi [30] mentioned as future work a blind deconvolution method that uses prediction in an iterative process. However, the dissertation does not contain details of the method, and the preliminary results show a far lower quality of deblurring than ours.

2.2 Fast Single Image Blind Deconvolution

2.2.1 Single Image Blind Deconvolution

A successful approach for blind deconvolution is alternating optimization of a latent image l and a blur kernel k in an iterative process. Based on the blur model in Eq. (1.1), we respectively solve the equations similar to

$$l' = \underset{l}{\operatorname{argmin}}\{\|b - k * l\|^2 + \phi_{\text{latent}}(l)\}, \tag{2.1}$$

$$k' = \underset{k}{\operatorname{argmin}}\{\|b - k * l\|^2 + \phi_{\text{kernel}}(k)\} \tag{2.2}$$

in the latent image and kernel estimation steps of the process. In Eqs. (2.1) and (2.2), $\|b - k * l\|^2$ is the data fitting term and ϕ_{latent} and ϕ_{kernel} are regularization terms. For an image f, the norm $\|f\|$ is usually defined as $\|f\|^2 = \sum_{\mathbf{x}} |f(\mathbf{x})|^2$ where $\mathbf{x}$ is a 2D vector for a pixel position. For ϕ_{latent}, Chan

and Wong [11] used total variation, and Jia [28] used a term that prefers a two-tone image for the transparency map. Shan et al. [48] used image derivatives as well as pixel values for the data fitting term while a natural image prior is used for ϕ_{latent}. Several regularization terms have been used for ϕ_{kernel}, such as L^1 and L^2-norms, total variation, and a uniform prior that means no regularization.

The main purpose of the iterative alternating optimization is to progressively refine the motion blur kernel k. The final deblurring result is obtained by the last non-blind deconvolution operation that is performed with the final kernel k and the given blurred image b. The intermediate latent images estimated during the iterations have no direct influence on the deblurring result. They only affect the result indirectly by contributing to the refinement of kernel k.

The success of previous iterative methods comes from two important properties of their latent image estimation, *sharp edge restoration* and *noise suppression in smooth regions*, which enable accurate kernel estimation. Although we assume the blur is uniform over the given image, a more accurate blur kernel can be obtained around sharp edges. For example, we cannot estimate a blur kernel on a region with a constant intensity. Since a natural image usually contains strong edges, a blur kernel can be effectively estimated from the edges reconstructed in latent image estimation. Noise suppression in smooth regions is also important because such regions usually occupy much larger areas than strong edges in a natural image. If noise has not been suppressed in smooth regions, the data fitting term in Eq. (2.2) would be significantly affected by the noise, compromising the accuracy of kernel estimation from strong edges.

To achieve sharp edge restoration and noise suppression when solving Eq. (2.1), previous methods usually perform computationally expensive non-linear optimization. In addition, kernel estimation using Eq. (2.2) is computationally demanding as it involves a number of operations on huge matrices and vectors. As a result, both latent image estimation and kernel estimation require heavy computation in the iterative process of previous blind deconvolution methods.

2.2.2 Fast Blind Deconvolution

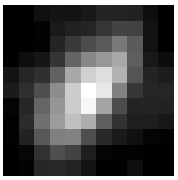
In this chapter, we present a fast blind deconvolution technique by reducing the computational overhead for latent image estimation and kernel estimation. For accelerating latent image estimation, we assume that the latent image has enough strong edges, and explicitly pursue sharp edge restoration and noise suppression using image filters, instead of taking a computationally expensive non-linear prior in Eq. (2.1). For kernel estimation, we accelerate the numerical optimization process of Eq. (2.2) by excluding pixel values in the formulation.

In our method, latent image estimation is divided into two parts: simple deconvolution and prediction. Given a blurred image b and kernel k, we first remove the blur to obtain an estimate l of the latent image using simple and fast deconvolution with a Gaussian prior. Due to the characteristics of a Gaussian prior, L would contain smooth edges and noise in smooth regions. In the prediction step, we obtain a refined estimate l' by restoring sharp edges and removing noise from l with efficient image filtering techniques. As a result, l' provides a high-quality latent image estimation needed for accurate kernel estimation, in spite of the poor quality of simple deconvolution (Figure 2.2).

For kernel estimation, we use a CG method to solve Eq. (2.2). During the solution process, we need to calculate the gradient of the energy function many times. The gradient calculation requires heavy computation, involving multiplications of huge matrices and vectors. Fortunately, the multiplications correspond to convolution operations and can be accelerated using fast Fourier transforms (FFTs). However, when we perform FFTs in sequence, we should properly handle image boundaries, which prohibits direct concatenation of FFTs. By formulating the energy function in Eq. (2.2) with only image derivatives, we can simplify the boundary handling and reduce the number of FFTs significantly. The formulation also makes the numerical system derived from Eq. (2.2) well-conditioned, providing a fast convergence.

(a) blurred image

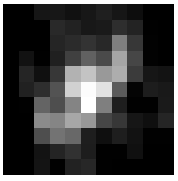
(b) deconvolution result



(e) previous kernel

(c) prediction result

(d) deconvolution of Shan et al.



(f) updated kernel

Figure 2.2: Effect of prediction. Our simple deconvolution with (a) and (e) produces a poor result (b). However, the prediction result (c) from (b) has only sharp edges while ringing artifacts have been removed. The kernel (f) estimated using (c) shows a refinement toward the real kernel, shown in Figure 2.4. Comparison of (c) with (d) obtained directly from (a) and (e) shows that combination of simple deconvolution with prediction can produce a similar result to sophisticated deconvolution, for the purpose of the input for kernel estimation. For visualization, the image in (c) has been restored from the predicted gradient maps by Poisson reconstruction.

2.2.3 Process Overview

Figure 2.3 shows the overall process of our blind deconvolution method. To progressively refine the motion blur kernel k and the latent image l, our method iterates three steps: prediction, kernel estimation, and deconvolution. Recall that our latent image estimation is divided into prediction and

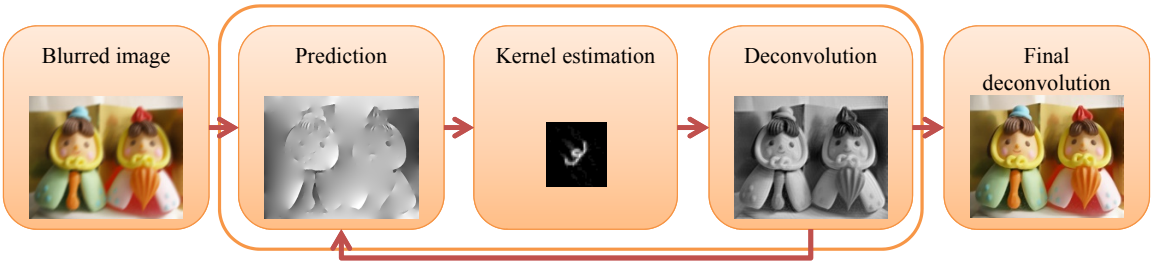


Figure 2.3: Overview of our deblurring process.

deconvolution. We place prediction at the beginning of the loop to provide an initial value of l for kernel estimation, where the input of the prediction is the given blurred image b.

In the prediction step, we compute gradient maps $\{p_x, p_y\}$ of l along the x and y directions which predict salient edges in l with noise suppression in smooth regions. Except at the beginning of the iteration, the input of the prediction step is the estimate of l obtained in the deconvolution step of the previous iteration. In the kernel estimation step, we estimate k using the predicted gradient maps $\{p_x, p_y\}$ and the gradient maps of b. In the deconvolution step, we obtain an estimate of l using k and b, which will be processed by the prediction step of the next iteration.

To make the estimations of k and l more effective and efficient, our method employs a coarse-to-fine scheme. At the coarsest level, we use the down-sampled version of b to initialize the process with the prediction step. After the final estimate of l has been obtained at a coarse level, it is up-sampled by bilinear interpolation and then used for the input of the first prediction step at the next finer level. In our experiments, we performed seven iterations of the three steps at each scale. As detailed in Section 2.3, this coarse-to-fine scheme enables handling of large blurs for which prediction with image filtering may not suffice to capture sharp edges.

In the coarse-to-fine iterative process for updating k and l, we use the gray-scale versions of b and l. After the final k has been obtained at the finest level, i.e., with the input image size, we perform the final deconvolution with k on each color channel of b to obtain the deblurring result.

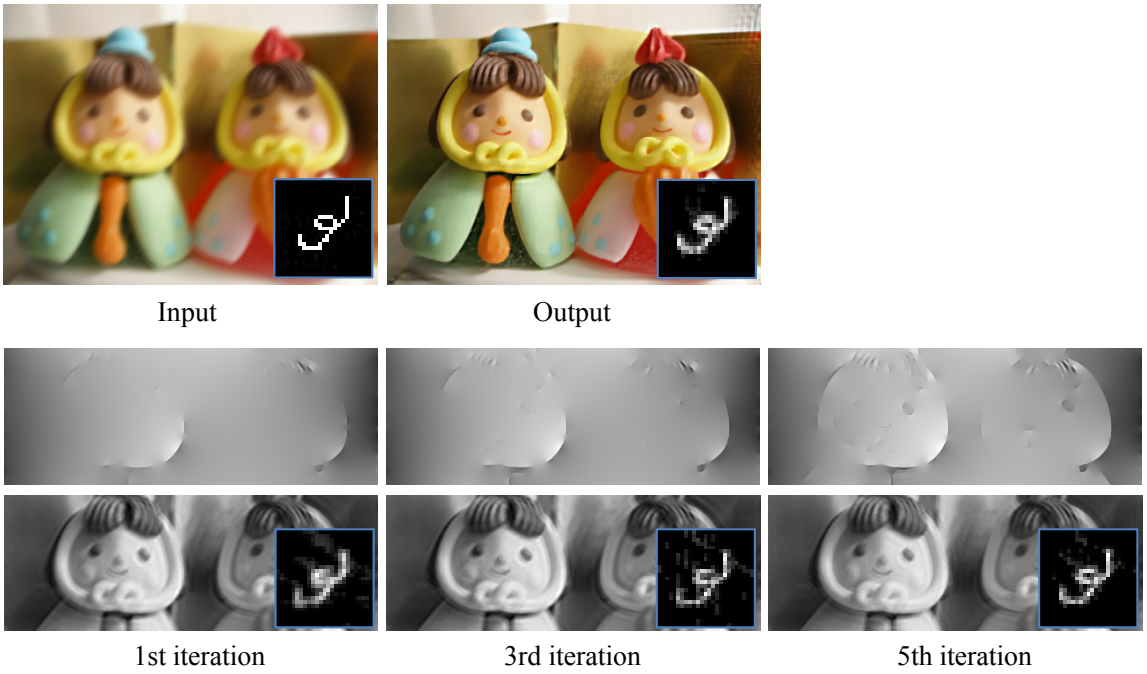


Figure 2.4: Deblurring process example. The left image on the top row shows the input blurred image and the original motion blur kernel. The right image on the top row shows the deblurring result and the final estimated kernel. The three columns on the bottom row show the predicted gradient maps $\{p_x, p_y\}$ (top) and the deconvolution results with estimated kernels (bottom) after different numbers of iterations. For visualizing $\{p_x, p_y\}$, we used Poisson reconstruction. Note that the deblurring process itself does not use Poisson reconstruction of predicted gradients. All intermediate results are shown at the finest scale.

Although any non-blind deconvolution technique can be used for this step, we use the technique proposed in [48], which efficiently obtains high quality deconvolution results. Figure 2.4 shows an example of our deblurring process with intermediate estimates of k and l.

2.3 Fast Latent Image Estimation

Prediction In the prediction step, we estimate the image gradient maps $\{p_x, p_y\}$ of the latent image l in which only the salient edges remain and other regions have zero gradients. Consequently, in the kernel estimation step, only the salient edges have influences on optimization of the kernel

because convolution of zero gradients is always zero regardless of the kernel.

We use a shock filter to restore strong edges in l. A shock filter is an effective tool for enhancing image features, which can recover sharp edges from blurred step signals [43]. The evolution equation of a shock filter is formulated as

$$f_{t+1}(\mathbf{x}) \quad = \quad f_t(\mathbf{x}) - \text{sign}(\Delta f_t(\mathbf{x}))\|\nabla f_t(\mathbf{x})\|dt, \tag{2.3}$$

where f_t is an image at time t, and Δf_t and ∇f_t are the Laplacian and gradient of f_t, respectively. dt is the time step for a single evolution. $\|\cdot\|$ is the L^2-norm of a vector. sign(u) is a function that returns either $+1$ or -1 with respect to the sign of u.

Our prediction step consists of bilateral filtering, shock filtering, and gradient magnitude thresholding. We first apply bilateral filtering [55] to the current estimate of l to suppress possible noise and small details. A shock filter is then used to restore strong edges of l. The result l' of shock filtering contains not only high-contrast edges but also enhanced noise. We remove the noise by computing and thresholding the gradient maps $\{\partial l'/\partial x, \partial l'/\partial y\}$ of l'. The truncated gradient maps $\{p_x, p_y\}$ give the final output of the prediction step.

The support size of bilateral filtering is fixed as 5×5, and the spatial sigma σ_s is set to 2.0. The range sigma σ_r is a user parameter, which is related to the noise level of the input blurred image b. When b contains much noise, we use a large value for σ_r. For shock filtering, we perform a single evolution of Eq. (2.3) with the time step dt. At the beginning of the iterative deblurring process, we use large values for σ_r and dt to clearly restore strong sharp edges in l. As the iteration goes, we gradually decrease the values by multiplying 0.9 at each iteration. For most of our experiments, we used 0.5 and 1.0 for the initial values of σ_r and dt, respectively.

The threshold for truncating gradients is determined as follows. To estimate an $m \times m$ kernel, we need the information of blurred edges in at least m different directions. We construct the histograms of gradient magnitudes and directions for $\{\partial l'/\partial x, \partial l'/\partial y\}$. Angles are quantized by $45°$, and gradients of opposite directions are counted together. Then, we find a threshold that keeps at least

rm pixels from the largest magnitude for each quantized angle. We use 2 for r by default. To include more gradient values in $\{p_x, p_y\}$ as the deblurring iteration progresses, we gradually decrease the threshold determined at the beginning by multiplying 0.9 at each iteration.

Deconvolution In the deconvolution step, we estimate the latent image l from a given kernel k and the input blurred image b. We solve the following optimization problem

$$\underset{l}{\operatorname{argmin}} \left\{ \sum_{\partial_*} \omega_* \left\| k * \partial_* l - \partial_* b \right\|^2 + \alpha \left\| \frac{\partial l}{\partial x} \right\|^2 + \alpha \left\| \frac{\partial l}{\partial y} \right\|^2 \right\}, \tag{2.4}$$

where $\partial_* \in \{\partial_o, \partial/\partial x, \partial/\partial_y, \partial^2/\partial x^2, \partial^2/\partial x \partial y, \partial^2/\partial y^2\}$ denotes the partial derivative operator in different directions and orders, $\omega_* \in \{\omega_0, \omega_1, \omega_2\}$ is a weight for each partial derivative, and α is a weight for the regularization terms. ∂_o is the zeroth order partial differential operator. Each ∂_* forms a map $(k * \partial_* l - \partial b)$, and we define $\|f\|^2 = \sum_{\mathbf{x}} |f(\mathbf{x})|^2$ for a map f. The first term in the energy is based on the blur model of [48], which uses image derivatives for reducing ringing artifacts. The regularization terms $\|\partial l/\partial x\|^2 + \|\partial l/\partial y\|^2$ prefer l with smooth gradients, as mentioned in [35]. Eq. (2.4) can be optimized very fast by pixel-wise division in the frequency domain, which needs only two FFTs. For ω_*, we used the values given in [48]. We empirically found that $\alpha = 0.1$ works well in most cases.

Optimizing Eq. (2.4) may not produce high-quality results, compared to sophisticated deconvolution methods (e.g., [35, 61, 48]), and the results can contain smoothed edges and ringing artifacts. However, due to the prediction step that sharpens edges and discards small details, this simple deconvolution does not hinder accurate estimation of a blur kernel in our iterative process.

Large blurs Our prediction method may fail to correctly predict sharp edges for large blurs. However, our coarse-to-fine scheme enables us to avoid direct prediction of edges from a largely blurred image. We first predict sharp edges in a low resolution image, where the extents of blurs have been narrowed and most edges can be predicted without severe localization errors. At a higher resolution, we start prediction of sharp edges with an upsampled version of the *deconvolved* image obtained at

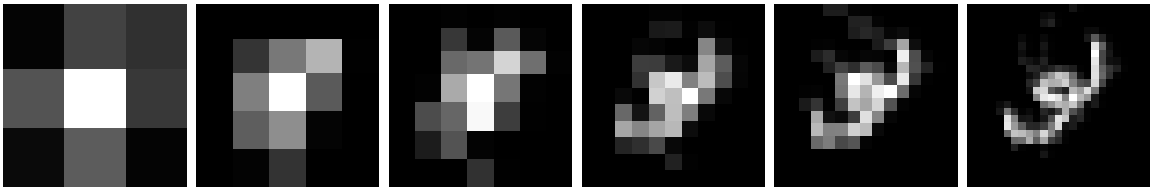


Figure 2.5: Estimated kernels at different scales for the deblurring example in Figure 2.4. As the scale becomes finer, a more detailed structure of the kernel is recovered.

a coarser resolution, which contains reduced amounts of blurs. In the iterative process at a specific scale, sharp edge prediction is applied to the deconvolution result obtained by an updated kernel in the previous iteration, progressively improving the prediction accuracy. With the multi-scale iterative process, we can estimate kernels for large blurs using prediction with small size bilateral and shock filters. This multi-scale iterative process is also the main difference from [32] and [42], which enables our method to estimate complex large motion blurs that cannot be handled by the previous methods. Figure 2.5 shows estimated kernels at different scales in our multi-scale process.

Speed comparison For estimating latent image gradient maps to be used for kernel estimation, our method requires two FFTs for deconvolution and simple image filtering operations for prediction. Bilateral filtering with a small support size and shock filtering, as well as gradient thresholding, can be performed very fast. In contrast, the highly efficient deconvolution method of Shan et al. [48], which is based on variable substitution, commonly needs 30 to 60 FFTs, i.e., 15 to 30 times more FFTs than our method. Obviously, our method with simple convolution and prediction is much faster than the latent image estimation with a sophisticated deconvolution technique.

2.4 Fast Kernel Estimation

To estimate a motion blur kernel using the predicted gradient maps $\{p_x, p_y\}$, we solve the following optimization problem

$$\underset{k}{\operatorname{argmin}} \left\{ \sum_{(p_*, b_*)} \omega_* \| k * p_* - b_* \|^2 + \beta \| k \|^2 \right\}, \tag{2.5}$$

where $\omega_* \in \{\omega_1, \omega_2\}$ denotes a weight for each partial derivative, and β is a weight for the Tikhonov regularization. p_* and b_* vary among

$$\begin{aligned} (p_*, b_*) \quad \in \quad & \left\{ \left(p_x, \frac{\partial b}{\partial x} \right), \left(p_y, \frac{\partial b}{\partial y} \right), \left(\frac{\partial p_x}{\partial x}, \frac{\partial^2 b}{\partial x^2} \right), \left(\frac{\partial p_y}{\partial y}, \frac{\partial^2 b}{\partial y^2} \right), \right. \\ & \left. \left(\frac{1}{2} \left(\frac{\partial p_y}{\partial x} + \frac{\partial p_x}{\partial y} \right), \frac{\partial^2 b}{\partial x \partial y} \right) \right\}. \end{aligned} \tag{2.6}$$

Our energy function in Eq. (2.5) is similar to [48]. A difference is that we use only the image derivatives, not including the pixel values, in the energy function. In addition, similar to Yuan et al. [60], our energy function includes a Tikhonov regularization term, instead of the L^1-norm of k used in [48]. In our experiments, we used the values given in [48] for ω_* and set β to 5.

We can write Eq. (2.5) in a matrix form,

$$\underset{\mathbf{k}}{\operatorname{argmin}} \left\{ \| \mathbf{A}\mathbf{k} - \mathbf{b} \|^2 + \beta \| \mathbf{k} \|^2 \right\}, \tag{2.7}$$

where $\mathbf{A}$ is a matrix consisting of five p_*'s, $\mathbf{k}$ is a vector representing the motion blur kernel k, and $\mathbf{b}$ is a vector consisting of five b_*'s. To minimize Eq. (2.7), we use a CG method. Then, the gradient of Eq. (2.7), defined by

$$2\mathbf{A}^T\mathbf{A}\mathbf{k} + 2\beta\mathbf{k} - 2\mathbf{A}^T\mathbf{b}, \tag{2.8}$$

should be evaluated many times in the minimization process.

Computation of Eq. (2.8) is time consuming due to the vast size of $\mathbf{A}$. When the sizes of l and k are $n \times n$ and $m \times m$, respectively, the size of $\mathbf{A}$ is $5n^2 \times m^2$. Thus, direct computation of $\mathbf{A}\mathbf{k}$

needs heavy computational and storage overhead. Although the size of $\mathbf{A}^T\mathbf{A}$ is $m^2 \times m^2$, which is relatively small, precomputing $\mathbf{A}^T\mathbf{A}$ is still time consuming. Each element of $\mathbf{A}^T\mathbf{A}$ requires the computation of dot product for two shifted versions of five p_*'s.

However, since $\mathbf{Ak}$ corresponds to spatially-invariant convolution operations between five p_*'s and k, we can accelerate the computation by FFTs. Specifically, computing $\mathbf{Ak}$ needs six FFTs: one $\mathcal{F}(k)$ and five $\mathcal{F}^{-1}\left[\omega_*\mathcal{F}(p_*)\mathcal{F}(k)\right]$'s, where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the forward and inverse Fourier transforms, respectively. Each frequency component is multiplied in a pixel-wise manner. Note that we can precompute $\mathcal{F}(p_*)$ before starting the CG method. Similarly, computing $\mathbf{A}^T\mathbf{y}$ can be accelerated by performing six FFTs, where $\mathbf{y} = \mathbf{Ak}$. As a result, it costs a total of 12 FFTs to compute the gradient defined by Eq. (2.8) at each iteration of the CG method. In addition to $\mathcal{F}(p_*)$, $\mathbf{A}^T\mathbf{b}$ can be computed with FFTs in the preprocessing step.

To further accelerate the computation by reducing the number of FFTs, we concatenate evaluations of $\mathbf{Ak}$ and $\mathbf{A}^T\mathbf{y}$ to directly compute $\mathbf{A}^T\mathbf{Ak}$. Then, $\mathbf{A}^T\mathbf{Ak}$ can be computed by

$$\mathcal{F}^{-1}\left[\sum_{p_*}\omega_*\overline{\mathcal{F}(p_*)}\mathcal{F}(p_*)\mathcal{F}(k)\right] \tag{2.9}$$

with appropriate cropping and flipping operations, where $\overline{\mathcal{F}(p_*)}$ is the complex conjugate of $\mathcal{F}(p_*)$. In Eq. (2.9), $\sum_{p_*}\omega_*\overline{\mathcal{F}(p_*)}\mathcal{F}(p_*)$ can be precomputed before CG iteration. Thus only two FFTs are needed for computing the gradient, saving 10 FFTs.

This efficient computation benefits from using only image derivatives but no pixel values in Eq. (2.5). If we include pixel values in Eq. (2.5), the computation of $\mathbf{Ak}$ with FFTs will have boundary artifacts due to the periodicity property of Fourier transforms. Then, we should handle the boundary artifacts before computing $\mathbf{A}^T\mathbf{y}$. This intervening boundary handling prohibits direct computation of $\mathbf{A}^T\mathbf{Ak}$ using Eq. (2.9). In contrast, since our method uses only image derivatives, we can avoid the boundary artifacts by simply padding the boundaries of the derivative images p_*'s with zeros before computing $\mathbf{Ak}$. We set the width of a padded image as a power of prime numbers, 2, 3, 5, and 7, which is greater than or equal to $(n + m - 1)$, where n and m are the widths of the input

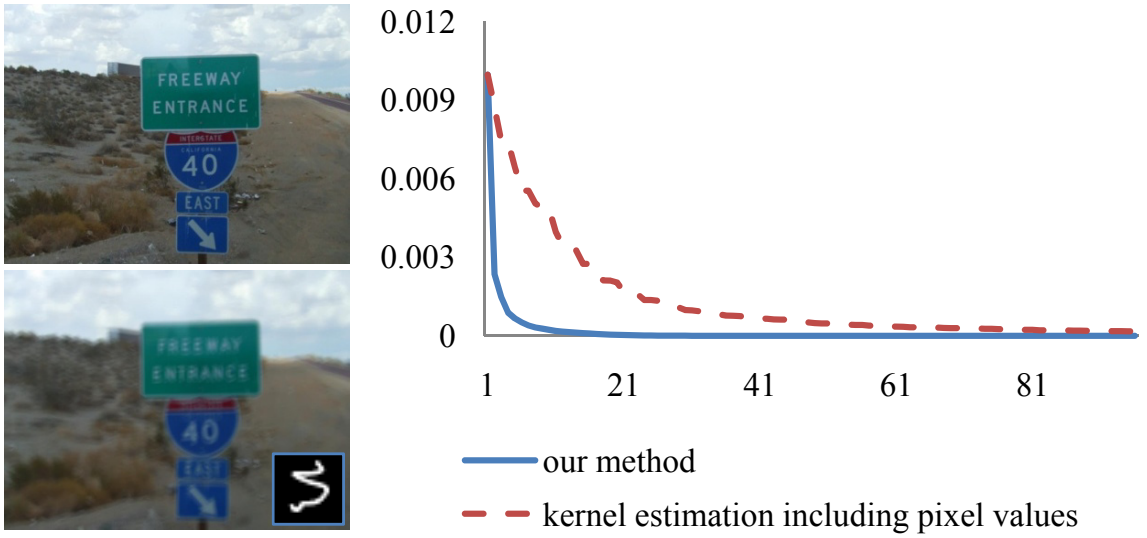


Figure 2.6: Convergence speed of the numerical method in kernel estimation. Left: original image and a blurred version used for experiment. Right: error of the intermediate kernel estimate (vertical) vs. number of iterations in a CG method. Our method converges faster than the formulation including pixel values. The error of a kernel estimate is measured by the sum of pixel-wise squared differences from the original kernel.

image and the kernel, respectively. The height of a padded image is determined similarly. FFTs can be computed quickly for such image sizes.

Motion blur kernels are generally assumed normalized and containing no negative components. After optimizing Eq. (2.5), we set elements with values smaller than $1/20$ of the biggest one to zero. Then, the remaining non-zero values are normalized so that their sum becomes one.

Convergence speed In numerical optimization, the number of iterations for convergence, or convergence speed, is important. We compared our method for kernel estimation against a kernel estimation method using pixel values. As the energy functions of the both methods consist of quadratic terms, they both converge to the global optimum. Interestingly, our method for kernel estimation shows faster convergence than the method including pixel values. In Figure 2.6, we used the CG method to estimate the kernel for a synthetically blurred image. The graphs in Figure 2.6 show that

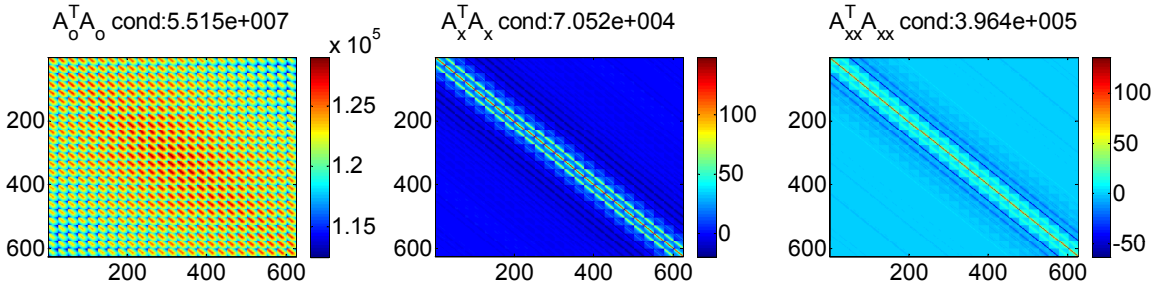


Figure 2.7: Visualization of matrix $\mathbf{A}_*^T\mathbf{A}_*$. From left to right, $\mathbf{A}_o^T\mathbf{A}_o$, $\mathbf{A}_x^T\mathbf{A}_x$, and $\mathbf{A}_{xx}^T\mathbf{A}_{xx}$ computed with the example in Figure 2.6.

the kernel estimation error of our method drastically decreases only in a few iterations while the error decreases slowly in the case of using pixel values.

The fast convergence of our method results from the well-conditioned structure of the matrix $\mathbf{A}^T\mathbf{A}$ in Eq. (2.8). $\mathbf{A}^T\mathbf{A}$ can be represented by

$$\mathbf{A}^T\mathbf{A} \;=\; \sum_{*} \omega_* \mathbf{A}_*^T\mathbf{A}_*, \tag{2.10}$$

where $\mathbf{A}_*^T\mathbf{A}_*$ is defined by $(\mathbf{A}_*^T\mathbf{A}_*)_{(i,j)} = (\mathbf{l}_*^i)^T(\mathbf{l}_*^j)$. $\mathbf{l}_*^i$ is the vector representation of $\partial_* l$ after shifted by the amount depending on i. Image derivatives are usually close to zero except on edge pixels. Hence, for a derivative image p_*, the values in $\mathbf{A}_*^T\mathbf{A}_*$ are large only around the diagonal and become small rapidly for off-diagonal regions. In contrast, if we involve pixel values for kernel estimation, $\mathbf{A}_o^T\mathbf{A}_o$ from the latent image l will be included in the summation of Eq. (2.10). As pixel values are mostly non-zero over an image, $\mathbf{A}_o^T\mathbf{A}_o$ have large values except for far off-diagonal regions.

Figure 2.7 visualizes $\mathbf{A}_o^T\mathbf{A}_o$, $\mathbf{A}_x^T\mathbf{A}_x$, and $\mathbf{A}_{xx}^T\mathbf{A}_{xx}$ computed with the example image in Figure 2.6. Condition numbers of the matrices are 5.5155×10^7, 7.0516×10^4 and 3.9636×10^5, respectively. Clearly, $\mathbf{A}_x^T\mathbf{A}_x$, and $\mathbf{A}_{xx}^T\mathbf{A}_{xx}$ have diagonally dominant structures and smaller condition numbers than $\mathbf{A}_o^T\mathbf{A}_o$. As a result, we can reduce the number of iterations in the CG method used for kernel estimation by excluding pixel values in the energy function.

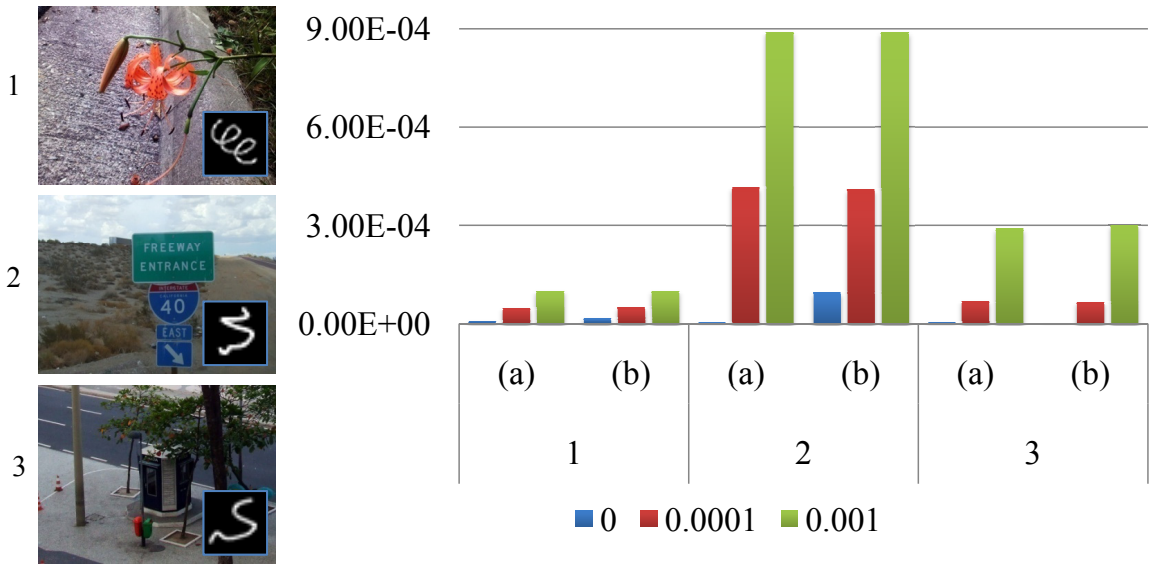


Figure 2.8: Kernel estimation error. Left: test images with blur kernels. Right: errors of the estimated kernels with (a) our method and (b) the method including pixel values. Errors are measured by the sum of pixel-wise squared differences between the estimated and original kernels. Different colors denote different amounts of added noise. For all test cases, (a) and (b) show similar errors.

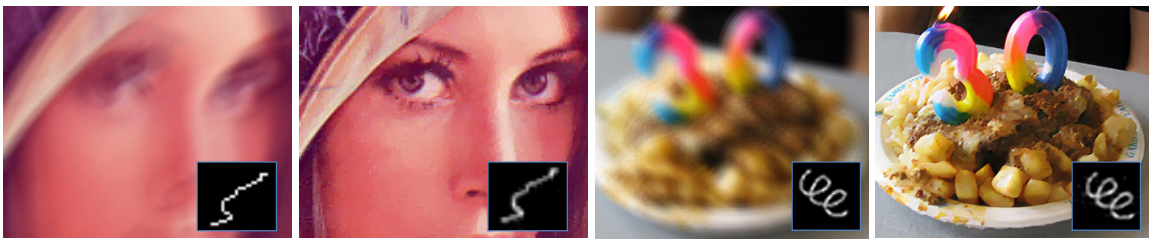


Figure 2.9: Deblurring of synthetic examples. The first and third images show blurred images with applied motion blur kernels. The second and fourth images show our deblurring results with estimated kernels. The kernel sizes of the left and right examples are 29×25 and 21×21, respectively.

Speed comparison For speed comparison, we consider a kernel estimation method, whose energy function is the same as [48], except that the L^2-norm is used for a kernel prior. We call the method

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

Figure 2.10: Deblurring results of real photographs. Top row: input blurred images. Bottom row: deblurring results with estimated kernels.

Shan-L2. Note that the original version in [48] needs more computation, since it uses the L^1-norm for a kernel prior. Also, according to the authors' document on additional programming details, which is available on their website, their kernel estimation step directly assembles huge matrices, which results in memory shortage and excessive computation. While our method needs two FFTs per CG iteration, Shan-L2 needs 14 FFTs, as it uses six images, one pixel value image plus five derivative images. Moreover, since it uses pixel values, it needs more iterations than ours, as analyzed above. In our experiments, our method runs five CG iterations, while Shan-L2 needs about 30 iterations to obtain a similar accuracy. Other previous methods usually use only pixel values for kernel estimation. In this case, convolutions cannot be concatenated due to the boundary problem, so they need four FFTs to compute a gradient of their energy function. They would also need more iterations than ours. To sum up, our kernel estimation is more than 40 times faster than Shan-L2 and more than 10 times faster than other methods using only pixel values.

Additionally, we found that the kernel estimation in [48] is more time-consuming than their latent image estimation. Even if Shan-L2 is used instead of the original version, kernel estimation

requires about 10 times more FFTs than latent image estimation. Consequently, when we set [48] as a baseline, kernel estimation contributes more to acceleration than latent image estimation in our method.

Accuracy To test the accuracy of our kernel estimation method, we experimented with synthetic examples, where we added Gaussian noise with different variances. For the examples, we estimated the kernels using our method and the other method including pixel values. Figure 2.8 shows the errors of estimated kernels. Although our method does not use pixel values, it exhibits similar accuracy to the case of including pixel values. Furthermore, our method gives better accuracy for some cases due to the well-conditioned system obtained by excluding pixel values.

Application to other methods Our acceleration technique for optimizing Eq. (2.5) can be used for other energy functions of kernel estimation. If the gradient of an energy function can be represented by a similar form to Eq. (2.8), the optimization process can be accelerated by our technique. For example, in the kernel estimation of [60], we can use the gradient maps of a noisy image for Eq. (2.5), instead of $\{p_x, p_y\}$, and accelerate kernel estimation with our technique. Similarly, our acceleration technique can be used for kernel estimation of [28, 48].

2.5 Results

In our experiments, the kernel size was specified by the user and did not have much influence on the accuracy of kernel estimation if the size was large enough to contain the estimated kernel. Although our method contains several parameters, we mainly controlled the kernel size, the range sigma σ_r of bilateral filtering for prediction, and the parameters of the final deconvolution operation which are adopted from [48].

To demonstrate the quality of the estimated motion blur kernels, we first show deblurring results with synthetic examples. In Figure 2.9, the estimated motion blur kernels have almost the same

Image	Size		Processing time (sec.)		
	Image	Kernel	A	B	C
a	972×966	65×93	2.188	3.546	5.766
b	1024×768	49×47	1.062	1.047	2.125
c	483×791	35×39	0.343	0.235	0.578
d	858×558	61×43	0.406	0.297	0.703
e	846×802	35×49	0.516	0.406	0.922

Table 2.1: Processing times of the deblurring examples in Figure 2.10. A: kernel estimation. B: final deconvolution. C: total processing time.

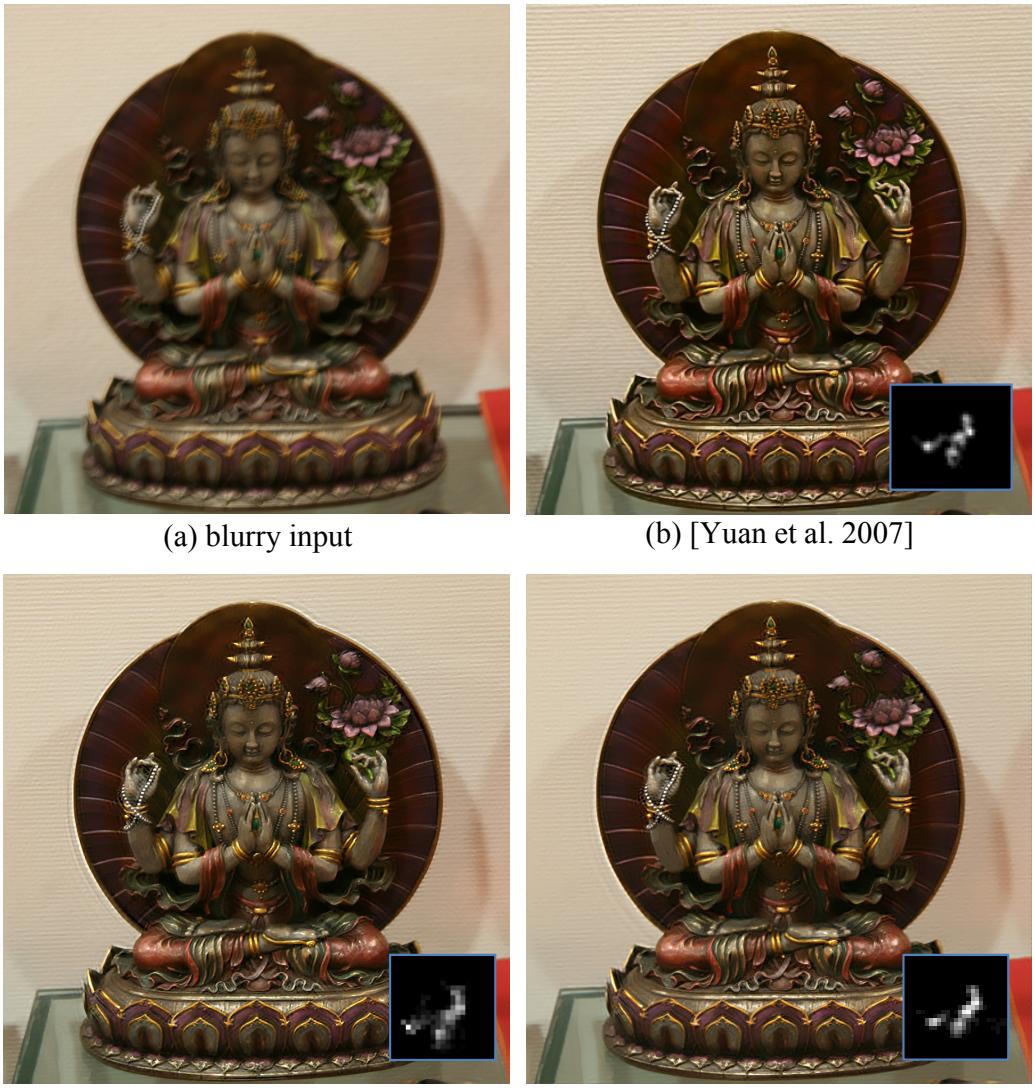
shapes as the original. The deblurring results show that the fine details of the original latent images have been accurately recovered.

Figures 2.1 and 2.10 show deblurring results of real photographs. The photographs contain complex structures and different camera motions. In the deblurring results, sharp edges have been significantly enhanced, revealing the object shapes and structures more clearly. The estimated motion blur kernels show reasonable shapes.

Table 2.1 shows the processing times for the deblurring examples in Figure 2.10. We implemented our method with GPU acceleration using BSGP [26]. For FFT, we used a FFT library provided with CUDA. Since BSGP is similar to C-language and easy to use, GPU implementation was almost straightforward. Our testing environment was a PC running MS Windows XP 32bit version with Intel Core2 Quad CPU 2.66GHz, 3.25GB RAM, and an NVIDIA GeForce GTX 280 graphics card. Even for kernels of large sizes, our method can remove blurs from input images of moderate sizes within a few seconds. Actually, for the photographs in Figures 2.10(c), 2.10(d), and 2.10(e), it took less than one second.

Figure 2.11 compares our deblurring results with previous methods. A pair of blurred and noisy images is needed in [60] while a given single image can be deblurred in [48]. Although our method uses a single image and the computation is much faster than the previous methods, the quality of the results is comparable due to our accurate kernel estimation.

Table 2.2 compares the processing times of our method and [48]. For comparison, we also



(a) blurry input (b) [Yuan et al. 2007]

(c) [Shan et al. 2008] (d) our method

Figure 2.11: Comparison with previous deblurring methods.

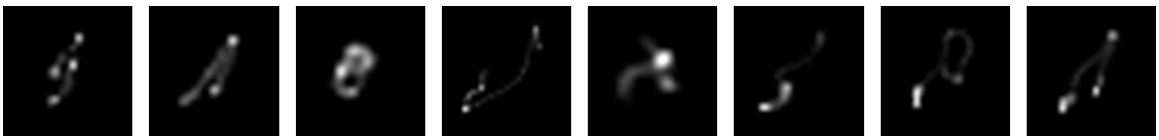
implemented our method using C++ without GPU acceleration. To measure the computation time of [48], we used the executable provided by the authors on the internet[1]. We used the four images included in the internet distribution with the parameters provided by the authors. The processing

[1] http://www.cse.cuhk.edu.hk/~leojia/projects/motion_deblurring/index.html

Image	Size		Processing time (sec.)		
	Image	Kernel	A	B	C
Picasso	800×532	27×19	360	20	0.609
statue	903×910	25×25	762	33	0.984
night	836×804	27×21	762	28	0.937
red tree	454×588	27×27	309	11	0.438

Table 2.2: Processing time comparison. A: [Shan et al. 2008]. B: our method with C++ implementation. C: our method with GPU acceleration.

Ground truth images of no blur



Ground truth blur kernels

Figure 2.12: Test set of Levin et al. [37].

times of our method include the time for the final deconvolution, which is performed by our C++ or GPU implementation of the method in [48]. To exclude additional deconvolution time from the comparison, we did not use the additional ringing suppression method of [48] for both their and our results. Table 2.2 shows that with a C++ implementation, our deblurring method is about 20 times faster than the executable of [48], which is a C-language implementation.

2.5.1 Quantitative evaluation on deblurring quality

For evaluating deblurring quality of our method, we performed quantitative comparison of our method with previous deblurring methods using the data set of [37] (Figure 2.12). The data set

consists of 32 test cases, which are made from four image patches and eight blur kernels. Each test case consists of one sharp image with no blur, one blurred image, and a ground truth motion blur kernel. Please refer to [37] for more details about the data set. For evaluation with each test case, we followed the method used in [37]. First, we produced the deconvolution result using the ground truth kernel. We used the sparse deconvolution method of [35]. Then, we ran our method to estimate a motion blur kernel. For all test cases, we consistently used the default values described in this chapter for the other parameters except for kernel sizes, including the range sigma $\sigma_r = 0.5$. We produced two deconvolution results using the estimated kernel for comparison of the deblurring quality as well as the accuracy of kernel estimation. One is obtained using the deconvolution method of Shan et al. [48] and the other is obtained by the sparse deconvolution method. As in [37], we measured the ratio of deconvolution errors from the sharp image between the ground truth and our estimated kernels. Since a deconvolution result can be misaligned due to the translation of a motion blur kernel, we aligned the center of a motion blur kernel to the center of the kernel image before performing deconvolution.

For comparison, we performed the same experiment with Fergus et al.'s method [18] and Shan et al.'s method [48] using the authors' executables available on internet. We tried several different values of parameters for Shan et al.'s method and chose the parameter values that generated the best results for all test cases. Specifically, we used 0.618, 0.04, 0.1, and 0.08 for the parameters *m*ultiScaleRatio, *n*oiseStr, *d*eblurStrength, and *k*CutRatio of the authors' executable, respectively. To obtain deconvolution results from the kernels estimated by Fergus et al.'s method, we used the sparse deconvolution method. For the estimated kernels of Shan et al.'s method, we produced two deconvolution results for each test case using the sparse deconvolution method and the original deconvolution method of Shan et al., as done with our method.

In our experiments, we could not get the error values reported by Levin et al. [37]. We obtained much larger deconvolution error values even in the cases of the ground truth kernels. We guess it is because the way we measured the error values differs from Levin et al.'s way, although we could

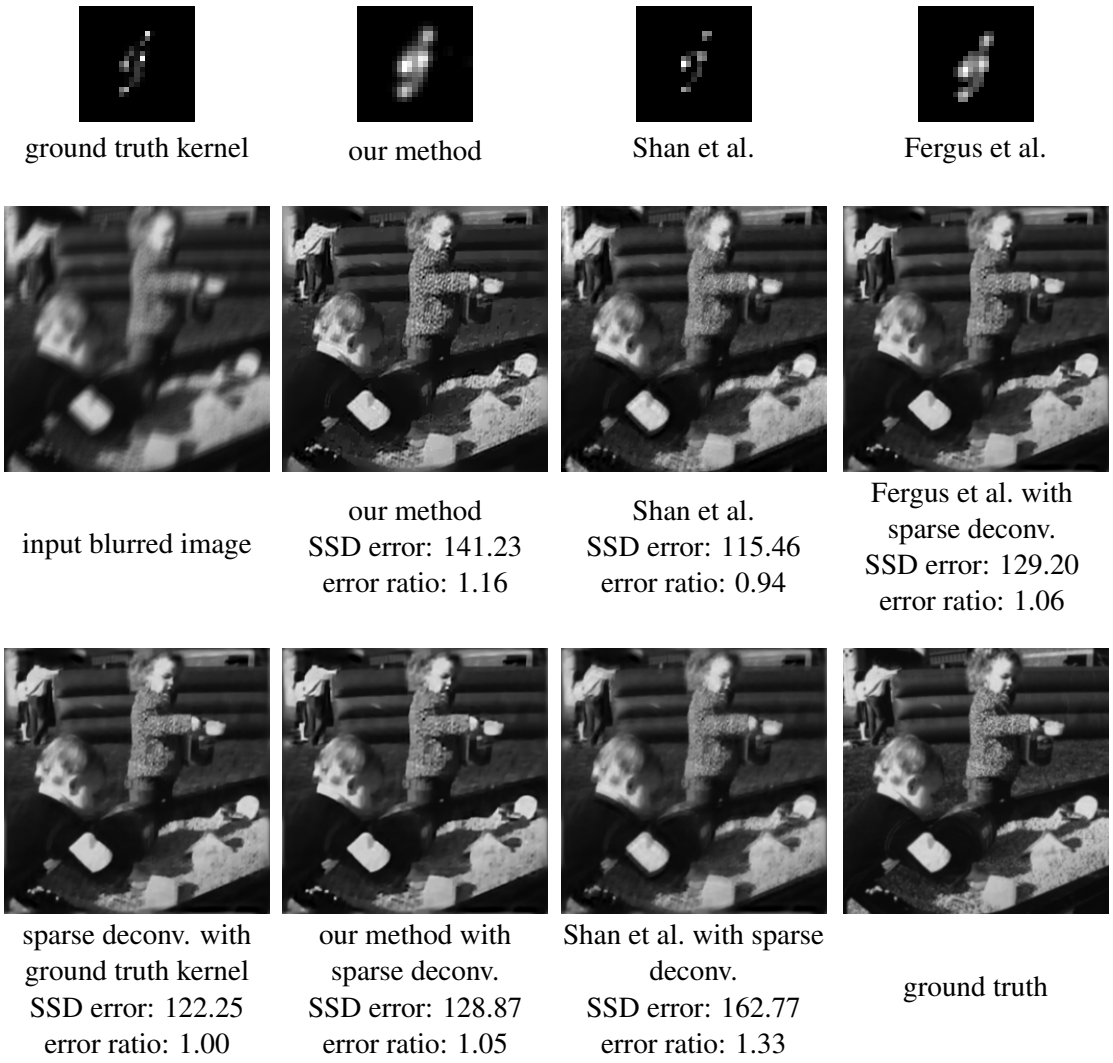


Figure 2.13: Comparison with previous methods using the data set of Levin et al. [37].

not figure out the exact reason. The experimental results, however, still show the effectiveness of our method over the previous methods.

Figure 2.13 shows some results and their associated error values. We also plot the cumulative histograms of deconvolution error ratios in the same way as [37] (Figure 2.14). For example, in the histograms, a bin of 3 shows the percentage of test cases whose deconvolution error ratios are below 3. The histograms shows that our method achieved satisfying results for all test cases. Kernels estimated by our method shows similar shapes to the ground truth kernels, although they

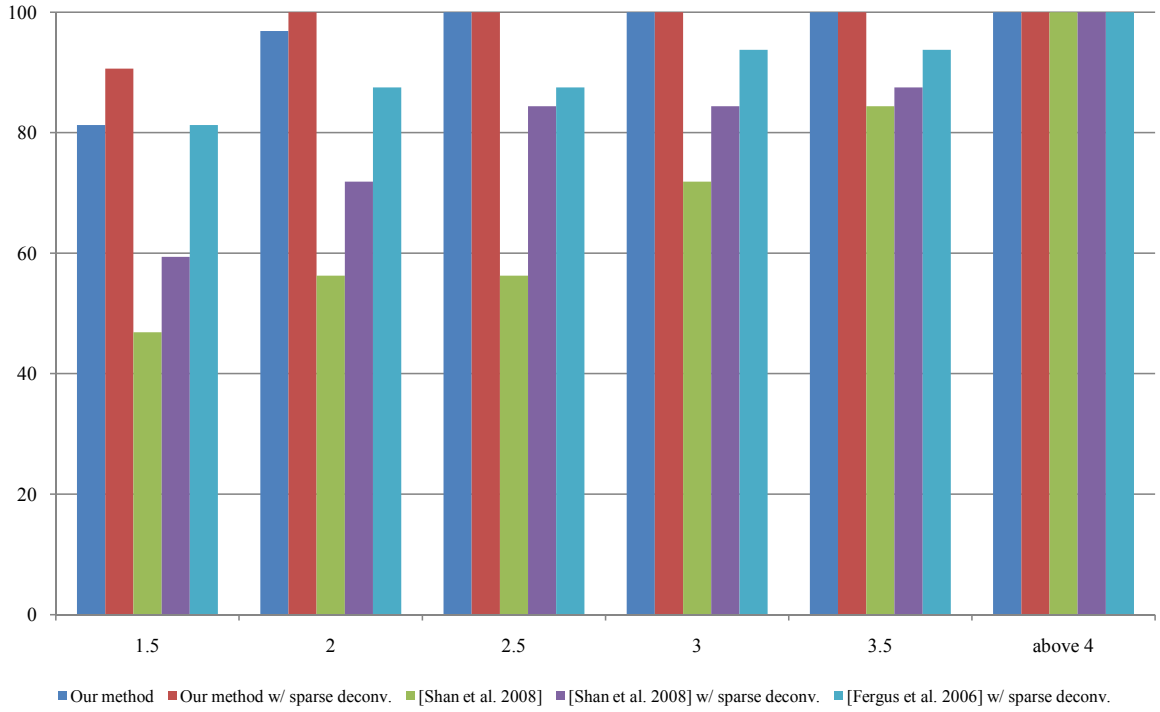


Figure 2.14: Cumulative histograms of error ratios.

are a little more blurry. The blurriness comes from the strategy of our method that tries to recover sharp edges even when the original unblurred edges are not really sharp. In our experiments, our method estimated blur kernels more robustly than Fergus et al.'s and Shan et al.'s methods. Due to accurately estimated kernels, both with the two deconvolution methods, our method generated deblurring results comparable to the sparse deconvolution results with the ground truth kernels.

2.6 Discussion

In this chapter, we proposed a fast blind deconvolution method, which provides enough speed for consumers to use deblurring in practice. Our method is based on the intuition that blurred strong edges can provide reliable information for the faithful estimation of a motion blur kernel. With the intuition, we could specify the required properties of an estimated latent image used for kernel

estimation in an iterative deblurring process. By explicitly achieving the properties with simple image filters in a multi-scale approach, we could avoid using computationally expensive priors for handling complex and large blurs. We hope that this intuition would help in solving more difficult deblurring problems, such as spatially-varying motion blurs.

Limitations Our prediction depends on local features rather than global structures of the image. If an image has strong local features inconsistent with other image regions, our method may fail to find a globally optimal solution. For example, such inconsistent features can be caused by saturated pixels. Our kernel estimation uses the Tikhonov regularization term, which is simpler than a sparsity prior used in [18]. Although less accurate kernels could be obtained with this simple term, the experiments in this chapter demonstrate that the accuracy of our kernel estimation is reasonable.

We assume the latent image contains enough sharp edges that can be used for kernel estimation. While most photographs include people, buildings, and natural scenery, which usually have sharp edges, this assumption may not always hold, e.g., with a photograph of furry objects. In this case, kernel estimation would be less accurate and the deblurring result could be over-sharpened.

Our method shares common limitations with other uniform motion deblurring methods. Due to the limitation of the blur model based on convolution, Outliers such as saturated pixels from strong lights and severe noise would not be properly handled. While out method assumes a uniform motion blur, real photographs are often degraded by spatially-varying blurs caused by moving objects and non-translational camera shakes. In Chapter 3, we will study how to handle outliers for non-blind deconvolution, and in Chapters 4 and 5, we will cover our solutions for non-uniform motion blur.

Chapter 3

Handling Outliers in Non-Blind Image Deconvolution

The task for blind image deconvolution is to infer both the blur kernel k and the latent image l from a single input blurred image b, which is severely ill-posed. Blind deconvolution approaches intensively use non-blind deconvolution, the process of estimating the latent image l given the blurred image b and the estimated blur kernel k, while inferring k and for generating the final output l [18, 15, 48, 37, 14, 31]. This makes non-blind deconvolution a key component in the deblurring pipeline.

Various non-blind deconvolution approaches have been proposed in the literature, ranging from classic Wiener filter to modern optimization approaches with image priors. However, in practice these methods often produce severe ringing artifacts even when the blur kernel is known or well estimated (Figure 3.1). We argue that this is mainly because the linear blur model in Eq. (1.1) does not consider non-linear outliers that often exist in real imaging process. For instance, one common outlier is saturated pixels. When we shoot a low lighting scene with a long exposure time, a few bright spots in the scene will be saturated (Figure 3.1a). The intensities of these pixels can no longer be modeled using Eq. (1.1) since a non-linear clipping function has been applied at the corresponding sensor locations. Other types of outliers include dead pixels of sensors, hot pixels, non-linear in-camera processing, and so on. These outliers are common in real imaging systems,

but are ignored in previous deconvolution methods.

In this chapter, we develop an algorithm that explicitly handles outliers in the deconvolution process. We first analyze how various types of outliers violate the linear blur assumption and consequently cause severe ringing artifacts to the result image. We then propose a new deconvolution method which contains an explicit component for outlier modeling. In our approach, we classify image pixels into two main categories: inlier pixels which satisfy the linear blur model and can be well-recovered using traditional deconvolution, and outlier pixels which cannot be explained by the linear model. We employ an Expectation-Maximization (EM) method to iteratively refine the outlier classification and the latent image.

To evaluate the effectiveness of the proposed method, we compare our approach with the state-of-the-art deconvolution methods on both synthetic and real-world examples. The results show that by explicitly detecting and handling outliers in the deconvolution process, severe ringing artifacts which are common to previous methods can be effectively reduced in our approach.

3.1 Related Work

As non-blind deconvolution is an important component in many blind deconvolution approaches, it has been extensively studied in image processing and computer vision fields. Classical methods include Wiener filtering, Kalman filtering, constrained least squares filtering and the Richardson-Lucy algorithm [45, 40, 57] The readers are referred to the comprehensive survey of Banham and Katsaggelos [2] for more details of these methods.

In order to suppress ringing artifacts and restore image features effectively, various image priors and regularization schemes have been proposed for deconvolution, such as total variation (TV) regularization [47], sparse image prior [35], and natural image statistics [48]. In previous work, these image priors are usually combined with an L^2-norm based data fidelity term, which is derived from a Gaussian noise model. This combination leads to high quality results when the observed image contains only a small amount of noise that can be well approximated by a Gaussian distribution. To

handle impulsive noise such as salt-and-pepper noise, Bar et al. [4] proposed a data fidelity term based on an L^1-norm, which can be derived from the assumption of a Laplacian distribution for noise. However, since these priors and data fidelity terms are derived from specific noise models, they cannot effectively suppress artifacts caused by other types of noise and outliers. In contrast, our method restores image details and suppresses ringing artifacts better than these approaches, as we will demonstrate later.

To suppress ringing artifacts, Yuan et al. [61] proposed a coarse-to-fine progressive deconvolution approach, where bilateral regularization is iteratively applied at each scale for restoring sharp edges while avoiding ringing. This method thus tries to implicitly handle outliers by directly suppressing artifacts. We show that ringing artifacts can be more efficiently removed by *explicitly* modeling the outliers in the deconvolution process. To the best of our knowledge, we are the first to systematically model outliers for non-blind deconvolution.

For handling saturated pixels in non-blind deconvolution, Harmeling et al. [23] proposed a method that detects saturated pixels by thresholding input blurry images, and masks out them from the deconvolution process. While the method recovers saturated pixels better than previous deconvolution methods, using a single threshold to detect saturated pixels in the input image is erroneous, as there is no guidance on how to find the optimal threshold value. In contrast, our method does not involve a threshold, and is more reliable and accurate as demonstrated later.

3.2 Outlier Analysis

In this section we analyze how various types of outliers violate the linear blur model and cause artifacts in previous approaches. All deconvolution results in this section were generated using the sparse prior based method proposed by Levin et al. [35].

Saturated/Clipped pixels As camera sensors have limited dynamic ranges, pixels receiving more photons than the maximum capacity will be saturated and the corresponding pixel intensities will

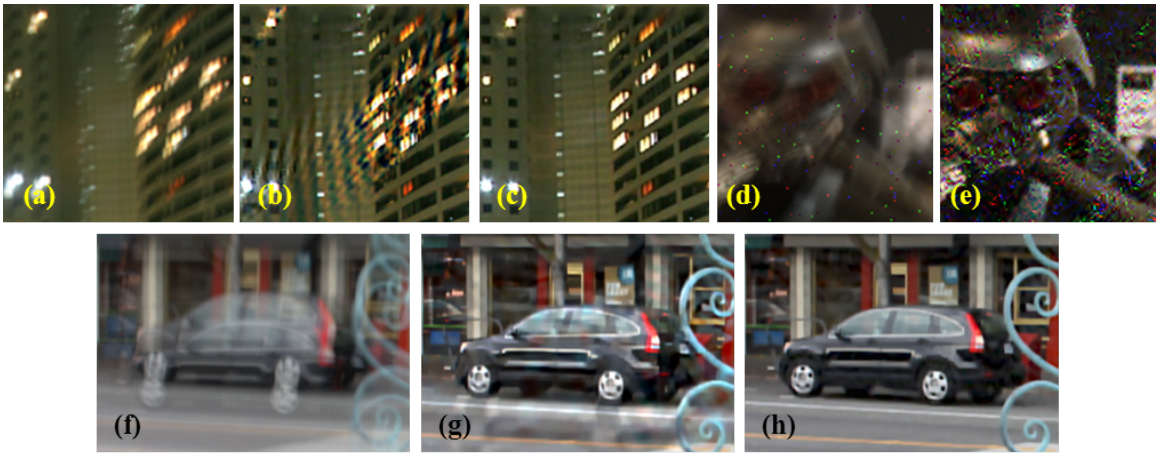


Figure 3.1: Illustrations of deconvolution outliers. (a) Input blurred HDR image with high intensity value clipping. (b) Deconvolution result of (a). (c) Deconvolution result of (a) without applying intensity clipping. (d) Input blurred image with added uniform noise. (e) Deconvolution result of (d). (f) Input blurred image with non-linear response curve applied. (g) Deconvolution result of (f). (h) Deconvolution result of (f) without applying the response curve.

be clipped. This is a common scenario when shooting a night image with long exposure, where the majority of the scene is dark but there are some bright lights. The non-linear clipping violates the linear blur model, leading to ringing artifacts around these spots. To demonstrate this, we synthetically blurred an HDR image using a known kernel, and then clipped pixel values that are larger than a threshold, as shown in Figure 3.1a. Directly applying deconvolution produces severe ringing artifacts shown in Figure 3.1b. Note that in this case the blur kernel is accurate and no noise is added. For comparison, we applied the same deconvolution method without clipping the input pixel values, which yields a high quality result in Figure 3.1c. It clearly suggests that saturated pixels can be a main source of ringing artifacts. Besides saturation, parts of an image, such as shadow areas, could be underexposed and intensities in those regions could be clipped to black. Such clipping also breaks the linear blur model, and may cause ringing artifacts too.

Non-Gaussian noise An image can be degraded by noise from various sources, and they are typically non-Gaussian [4]. In addition, the camera sensor may contain defective pixels causing

bright or dark spots in an image [16]. For low quality inputs, there may be complicated compression artifacts. Previous methods often assume a specific type of noise, thus is not robust enough to handle all cases. Figure 3.1d shows a synthetic example where uniform noise was added to the input image. Severe artifacts can be seen in the deconvolution result (Figure 3.1e), as the deconvolution method assumes a Gaussian noise model.

Nonlinear camera response curve Digital cameras all have built-in image processing units, and some of the processing steps are highly nonlinear. For instance, a camera usually applies a non-linear response curve to map the scene radiance to pixel intensity. This will violate the linear blur model, especially around high contrast edges where pixels on the two sides of an edge are separated far away on the response curve. Figure 3.1f shows a synthetic example of applying a non-linear response curve to a blurred raw image, and Figure 3.1g shows the deconvolution result, where artifacts are obvious around strong edges. Figure 3.1h shows the deconvolution result without applying the response curve, which is artifact-free.

3.3 Deconvolution with Outlier Handling

Our goal is to develop a robust deconvolution algorithm which can perform reliably well when outliers present. Among different types of outliers, nonlinear in-camera processing can be avoided by using raw camera output, or reduced by first applying an inverse response curve obtained from camera calibration [20] to the input image. However, other outliers are extremely hard to remove using image processing techniques. Our key observation is that although these outliers vary in nature, their impacts to the deconvolution process are common: First, they cause the linear blur model to fail. Second, they cause the noise in Eq. (1.1) to be non-Gaussian. We thus propose a deconvolution algorithm that specifically handles these two violations. For simplicity we will derive our algorithm under the assumption that the blur kernel is spatially invariant. In practice we can easily extend it to handle non-uniform blur kernels.

The key idea of our approach is to classify observed pixel intensities into two categories: *inliers*

whose formation satisfies Eq. (1.1); and *outliers* whose formation does not, which include clipped pixels and those from other sources. For classification, we introduce a binary map m such that $m(\mathbf{x}) = 1$ if the observed intensity $b(\mathbf{x})$ is an inlier, and $m(\mathbf{x}) = 0$ otherwise. To find the most probable latent image l given the blurred image b with outliers and the blur kernel k, we exclude the outliers from the deconvolution process using the inlier map m. Since we do not know the true value of m, we propose an EM method which alternatingly computes the expectation of m and performs deconvolution using the expectation.

To establish a more accurate blur model including outliers, we assume that a noise-free blurred image is captured by sensors and then the captured intensities are clipped into the dynamic range of a camera. Noise and outliers are added to the clipped blurred image within the dynamic range. This model can be represented as:

$$b \quad = \quad c(k * l) + n, \tag{3.1}$$

where c is a clipping function. When u is within the dynamic range, $c(u) = u$; otherwise, $c(u)$ returns the maximum or minimum intensity of the dynamic range. We assume that the additive noise n is spatially independent, and it follows a Gaussian distribution only for inliers. For outliers, we assume that the observed intensity at an outlier pixel is completely independent of k and l, and may have an arbitrary value in the dynamic range. Recall that outliers can come from multiple sources that are hard to model accurately, as we discussed in Section 3.2. Therefore we assume a *uniform distribution* for outliers, without superimposing any strong priors. In the following, we formulate our objective function, and derive the detailed process for deconvolution using this blur and noise model.

Mathematically, finding the most probable latent image l can be formulated as a maximum a posteriori (MAP) estimation problem such that:

$$l_{\mathrm{MAP}} \quad = \quad \underset{l}{\operatorname{argmax}}\, p(l|k, b). \tag{3.2}$$

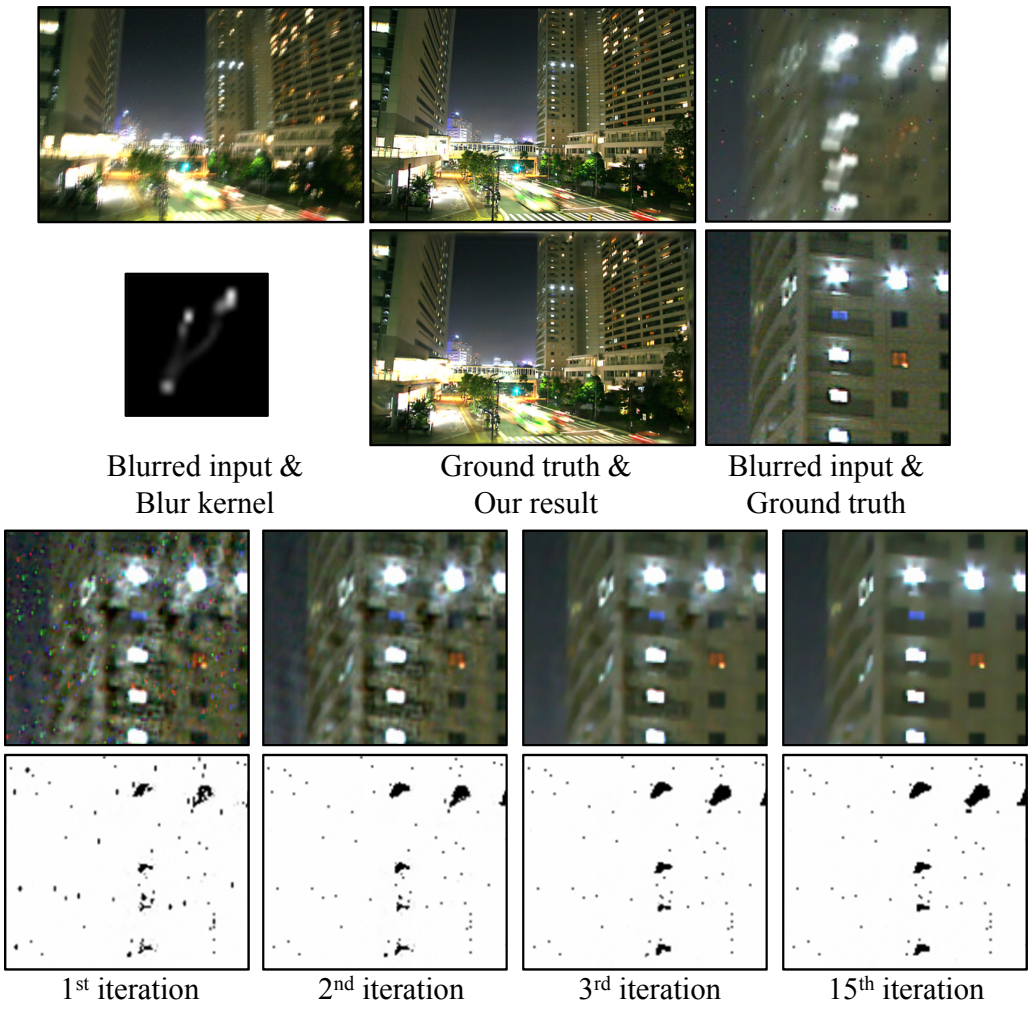


Figure 3.2: Deconvolution example of our method with intermediate results. The input image is synthetically blurred and high intensity values are clipped. Uniform noise is also added. The right four columns show magnified patches of intermediate estimates of l and the weights w^m computed for the red channel at EM iterations. Due to saturated pixels and noise, the estimate at the first iteration shows severe artifacts, but as iteration goes, artifacts are removed, and details are recovered.

According to the Bayes' theorem:

$$\begin{aligned} l_{\text{MAP}} &= \underset{l}{\operatorname{argmax}}\, p(b|k,l)p(l) \\ &= \underset{l}{\operatorname{argmax}} \sum_{m\in\mathcal{M}} p(b,m|k,l)p(l) \\ &= \underset{l}{\operatorname{argmax}} \sum_{m\in\mathcal{M}} p(b|m,k,l)p(m|k,l)p(l) \end{aligned} \tag{3.3}$$

where $\mathcal{M}$ is the space of all possible configurations of m. We define the latent image prior $p(l)$ as

$$p(l) \quad = \quad \exp(-\lambda\phi(l))/Z_p, \tag{3.4}$$

where Z_p is a normalization constant. Using the sparse prior [35], we set

$$\phi(l) \quad = \quad \sum_x \sum_y \left\{ \left| \frac{\partial l(x,y)}{\partial x} \right|^\alpha + \left| \frac{\partial l(x,y)}{\partial y} \right|^\alpha \right\}. \tag{3.5}$$

We use $\alpha = 0.8$ in our system as done in [35].

Since we assumed that noise is spatially independent, i.e., $p(b|m,k,l) = \prod_{\mathbf{x}} p(b(\mathbf{x})|m,k,l)$. Then, based on our noise model, we define $p(b(\mathbf{x})|m,k,l)$ as:

$$p(b(\mathbf{x})|m,k,l) \quad = \quad \begin{cases} \mathcal{N}(b(\mathbf{x})|f(\mathbf{x}),\sigma) & \text{if } m(\mathbf{x}) = 1 \\ C & \text{if } m(\mathbf{x}) = 0 \end{cases} \tag{3.6}$$

where $f = k * l$, $\mathcal{N}$ is a Gaussian distribution, and σ is the standard deviation. C is a constant defined as the inverse of the width of the dynamic range in the input image.

For the classification prior $p(m|k,l)$, we assume that m is also spatially independent, i.e., $p(m|k,l) = \prod_{\mathbf{x}} p(m(\mathbf{x})|k,l) = \prod_x p(m(\mathbf{x})|f(\mathbf{x}))$. We then define $p(m(\mathbf{x})|f(\mathbf{x}))$ based on the value of $f(\mathbf{x})$ as:

$$p(m(\mathbf{x}) = 1|f(\mathbf{x})) \quad = \quad \begin{cases} P_{\text{in}} & \text{if } f(\mathbf{x}) \in \text{DR} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where DR is the dynamic range, and $P_{\text{in}} \in [0,1]$ is the probability that $b(\mathbf{x})$ is an inlier. For example, by setting $P_{\text{in}} = 0.9$, we assume that 90% of non-clipped observed pixels $b(\mathbf{x})$ are inliers. According to our blur model in Eq. (3.1), when $f(\mathbf{x})$ is out of DR, the observed intensity $b(\mathbf{x})$ cannot be an inlier. It is either a clipped value or an outlier of another type, thus $m(\mathbf{x})$ should be always 0 in this case. Note that we do not limit the dynamic range of $l(\mathbf{x})$ in Eq. (3.1), so $f(\mathbf{x})$ can go beyond DR. In our implementation, we use the normalized range, DR $= [0,1]$.

Solving Eq. (3.3) is challenging, since the large number of possible configurations for m makes marginalizing $p(b,m|k,l)$ intractable. We thus adopt an EM method [10] to solve it. Instead of

marginalizing likelihood $p(b, m|k, l)$ with respect to m, our EM method tries to find an optimal l, which maximizes the complete-data posterior $p(l|b, m, k) \propto p(b, m|k, l)p(l)$. As we do not know the true value of m, we cannot use the complete-data likelihood $p(b, m|k, l)$ when maximizing $p(l|b, m, k)$. Our EM method thus evaluates the expectation of $p(b, m|k, l)$ and uses it for finding the optimal l.

The expectation of $p(b, m|k, l)$ and the estimate of l are updated by performing the E and M steps alternatingly. Specifically, the E step finds the posterior distribution $p(m|b, k, l)$ of m using the current estimate l^{o} of l, and then evaluates the expectation $Q(l, l^{\mathrm{o}})$ of the complete-data log likelihood $p(b, m|k, l)$ under $p(m|b, k, l)$. Note that the evaluated $Q(l, l^{\mathrm{o}})$ is not a value, but a functional with respect to l whose parameters have been determined using $p(m|b, k, l)$. The subsequent M step revises the estimate of l by maximizing the complete-data log posterior, which is defined as $Q(l, l^{\mathrm{o}}) + \log p(l)$. In the following, we will derive the details of the two steps.

E step Using the current estimate l^{o} of l, the expectation $Q(l, l^{\mathrm{o}})$ is defined by:

$$\begin{aligned} Q(l, l^{\mathrm{o}}) &= E[\log p(b, m|k, l)] && (3.8)\\ &= E[\log p(b|m, k, l) + \log p(m|k, l)], && (3.9) \end{aligned}$$

where E is the expectation under $p(m|b, k, l^{\mathrm{o}})$. We put Eqs. (3.6) and (3.7) into Eq. (3.9), assuming if $f^o(\mathbf{x})$ is in or out of DR, so is $f(\mathbf{x})$, respectively. Then, up to a constant, we have:

$$\begin{aligned} Q(l, l^{\mathrm{o}}) &= E\left[\sum_{\mathbf{x}} m(\mathbf{x}) \log \mathcal{N}(b(\mathbf{x})|f(\mathbf{x}), \sigma)\right] && (3.10)\\ &= -\sum_{\mathbf{x}} \frac{E[m(\mathbf{x})]}{2\sigma^2} |b(\mathbf{x}) - f(\mathbf{x})|^2 && (3.11) \end{aligned}$$

where $E[m(\mathbf{x})]$ is given by:

$$E[m(\mathbf{x})] = p(m(\mathbf{x}) = 1|b, k, l^{\mathrm{o}}). \quad (3.12)$$

By the Bayes' theorem, the posterior $p(m(\mathbf{x})|b, k, l^{\mathrm{o}})$ is

$$p(m(\mathbf{x})|b, k, l^{\mathrm{o}}) = \frac{p(b(\mathbf{x})|m(\mathbf{x}), k, l^{\mathrm{o}})p(m(\mathbf{x})|k, l^{\mathrm{o}})}{p(b(\mathbf{x})|k, l^o)}, \quad (3.13)$$

where

$$p(b(\mathbf{x})|k,l^o) \;=\; \sum_{m(\mathbf{x})=0}^{1} p(b(\mathbf{x})|m(\mathbf{x}),k,l^{\mathrm{o}})p(m(\mathbf{x})|k,l^{\mathrm{o}}). \tag{3.14}$$

Again, by putting Eqs. (3.6) and (3.7) into Eq. (3.13), we have:

$$E[m(\mathbf{x})] \;=\; \begin{cases} \dfrac{\mathcal{N}(b(\mathbf{x})|f^{\mathrm{o}}(\mathbf{x}),\sigma)P_{\text{in}}}{\mathcal{N}(b(\mathbf{x})|f^{\mathrm{o}}(\mathbf{x}),\sigma)P_{\text{in}}+CP_{\text{out}}} & \text{if } f^{\mathrm{o}}(\mathbf{x}) \in \text{DR} \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

where $f^{\mathrm{o}} = k * l^{\mathrm{o}}$ and $P_{\text{out}} = 1 - P_{\text{in}}$. As $E[m(\mathbf{x})]$ completely determines $Q(l,l^{\mathrm{o}})$ as a functional of l, $E[m(\mathbf{x})]$ is the only value that is actually computed in the E step.

In Eq. (3.6), $p(b(\mathbf{x})|m(\mathbf{x})=0,k,l^{\mathrm{o}})$ always follows a uniform distribution. If $f^o(\mathbf{x})$ is out of DR, we could expect $b(\mathbf{x})$ to be close to one bound of DR, which can be better represented by a non-uniform distribution. However, in this case, $p(m(\mathbf{x})=1|k,l^{\mathrm{o}})$ becomes zero and $p(b(\mathbf{x})|m(\mathbf{x})=0,k,l^{\mathrm{o}})$ has no effect on $E[m(\mathbf{x})]$. Therefore, we use a uniform distribution in Eq. (3.6) for all cases of outlier pixels, including clipped pixels.

M step The M step finds the revised estimate l^{n} such that

$$l^{\mathrm{n}} \;=\; \underset{l}{\operatorname{argmax}}\{Q(l,l^{\mathrm{o}}) + \log p(l)\}, \tag{3.16}$$

which is equivalent to minimizing:

$$\sum_{\mathbf{x}} w^m(\mathbf{x})|b(\mathbf{x}) - (k*l)(\mathbf{x})|^2 + \lambda\phi(l), \tag{3.17}$$

where $w^m(\mathbf{x}) = E[m(\mathbf{x})]/2\sigma^2$. To minimize Eq. (3.17), we use the iteratively reweighted least squares (IRLS) method used in previous deconvolution approaches [35]. First, we introduce pixelwise weights w^h and w^v such that $w^h(\mathbf{x}) = |\partial l(x,y)/\partial x|^{\alpha-2}$ and $w^v(\mathbf{x}) = |\partial l(x,y)/\partial y|^{\alpha-2}$. Then, Eq. (3.17) can be approximated by:

$$\sum_{\mathbf{x}} w^m(\mathbf{x})|b(\mathbf{x}) - (k*l)(\mathbf{x})|^2 + \lambda\psi(l), \tag{3.18}$$

where

$$\psi(l) \quad = \quad \sum_x \sum_y \left\{ w^h(x,y) \left| \frac{\partial l(x,y)}{\partial x} \right|^2 + w^v(x,y) \left| \frac{\partial l(x,y)}{\partial y} \right|^2 \right\}. \tag{3.19}$$

For fixed w^h and w^v, Eq. (3.18) becomes a quadratic function with respect to l, which can be effectively minimized using the conjugate gradient (CG) method. We thus can minimize Eq. (3.17) by alternating between updating (w^h, w^v) and minimizing Eq. (3.18).

The above EM procedure is easy to understand if we consider the meaning and role of $E[m(\mathbf{x})]$. If the observed intensity $b(\mathbf{x})$ is likely to be an inlier, $E[m(\mathbf{x})]$ computed in the E step is close to one, and vice versa. These values are then used as pixel weights in the actual deconvolution process, which is performed in the M step. As a result, only inliers with large weights are used for deconvolution in the M step, while outliers with low weights are excluded. In Section 3.4, we provide more analysis on this outlier handling scheme.

Algorithm Algorithm 1 shows the overall process for our EM-based deconvolution algorithm. For the initial estimate l^0, we use the result of the deconvolution using a Gaussian prior, by setting all weights to one. Then, we perform the EM iterations. In the E step, we update weights w^m, w^h and w^v given the currently estimated l. In the M step, we minimize Eq. (3.18) using the updated weights. We empirically found that N_iters $= 15$ and $\sigma = 5/255$ work well in most cases, and we adjusted λ according to the amount of noise in the input image. We consistently used $P_{\text{in}} = 0.9$ for generating all results in this chapter. To minimize Eq. (3.18), we run the CG method with 25 iterations. Figure 3.2 shows an example of applying our method to a night city image, and its intermediate results for the latent image l with the associated weights w^m.

3.4 Analysis on Outlier Handling

In this section we provide more insightful analysis on how our method can recover outlier pixels. We also demonstrate why we need to model clipped pixels explicitly in the classification prior $p(m|k,l)$.

Algorithm 1 EM Deconvolution for Outlier Handling

procedure DECONVOLUTION(b, k)
  Let $w^m(\mathbf{x})$, $w^h(\mathbf{x})$ & $w^v(\mathbf{x}) \leftarrow 1$ for all $\mathbf{x}$
  Set l^{o} by minimizing Eq. (3.18)
  for iter $\leftarrow 1$, N_iters **do**
    E step updates w^m, w^h and w^v using l^{o}
    M step updates l^{n} by minimizing Eq. (3.18)
    $l^{\text{o}} \leftarrow l^{\text{n}}$
  end for
  return l^{o}
end procedure

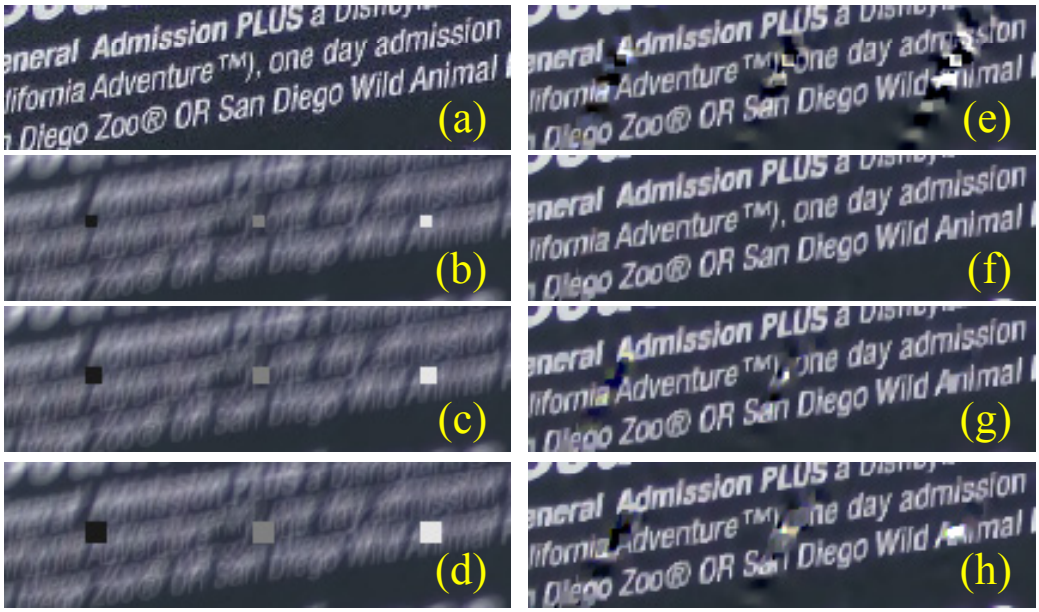


Figure 3.3: Recovering outlier pixels. (a) ground truth image. (b, c, d) blurred images with outliers of different sizes and intensities. (e) sparse deconvolution result of (b). (f, g, h) our deconvolution results of (b, c, d), respectively.

3.4.1 Recovering missing information

In a blurred image b, due to the scattering nature of blur, a pixel $\mathbf{x}$ contains partial information of the original intensities of neighboring pixels. If $\mathbf{x}$ is an inlier, $w^m(\mathbf{x}) = E[m(\mathbf{x})]/2\sigma^2$ is non-zero. Then, the observed intensity $b(\mathbf{x})$ at $\mathbf{x}$ contributes to recovering pixel values around $\mathbf{x}$ by minimizing Eq. (3.18), while the original intensity $l(\mathbf{x})$ at $\mathbf{x}$ is recovered from its own and neighboring pixel

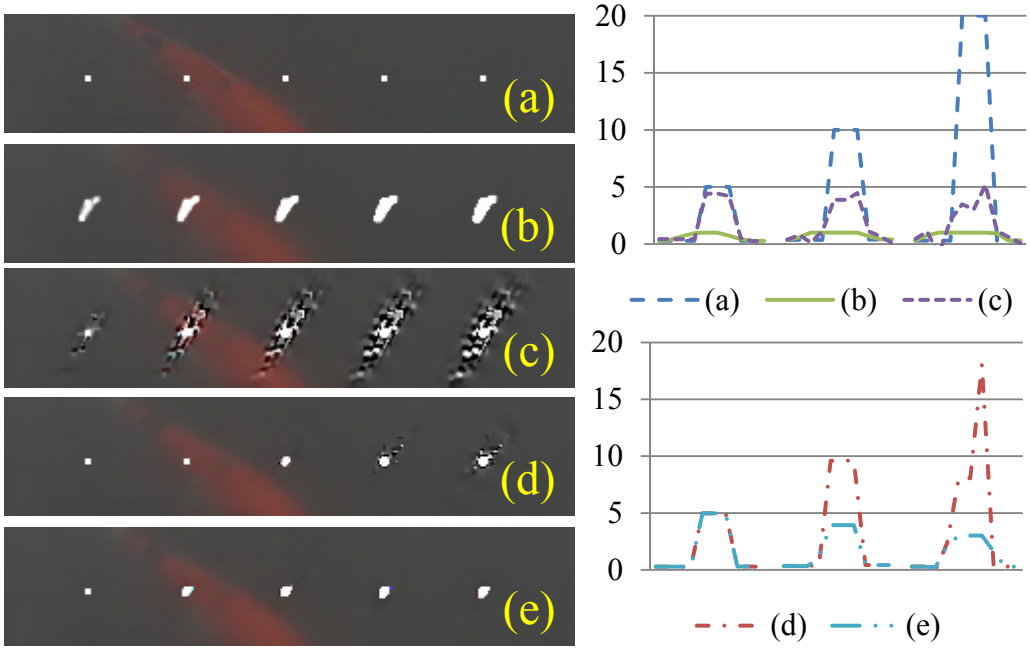


Figure 3.4: Recovering high intensity pixels. (a) ground truth image with pixels of different high intensities. From left to right, the original intensities of the white dots are 5, 10, 20, 40, and 80. (b) blurred image of (a), where high intensities are clipped. (c) sparse deconvolution result of (b). (d) our deconvolution result with only clipped pixel handling. (e) our result with both clipped pixel handling and outlier handling. Right: pixel intensities of each image along a scan line crossing the left three clipped pixels.

information. If $\mathbf{x}$ is an outlier, $l(\mathbf{x})$ can still be reconstructed using the information in the neighboring pixels around $\mathbf{x}$, although $b(\mathbf{x})$ does not contribute to recovering pixel values because $w^m(\mathbf{x})$ is close to zero. As a result, in our method outliers are not just smoothed out, their original intensities may be recovered actually if the neighboring pixels maintain enough amount of information about them.

To verify this observation, we conducted an experiment shown in Figure 3.3. We synthetically blurred the original image with a known kernel and added outlier pixels of different intensities and sizes. The deconvolution results show that, although the blurred image is contaminated by outliers, underlying textures can be recovered well for small outliers. For large outliers, neighboring pixels have less information about the true intensities of the outlier pixels. These pixels cannot be accurately recovered in our method, and can only be smoothed out.

Clipped pixels can also be recovered in the same way. In Figure 3.4, we synthetically added high intensity pixels with different intensity values into the original image. We then blurred the image with a known kernel, and clipped high intensity pixels. To study the effect of clipped pixel handling separately from non-Gaussian noise handling, we turn off the non-Gaussian noise handling by setting $P_{\text{in}} = 1$. The deconvolution result in Figure 3.4d shows that added pixels can be better recovered when their original intensities are lower. As the original intensity goes higher, more neighboring pixels around the added pixel are saturated as well. After intensity clipping, the scattered portion of the original high intensity in the neighboring pixels is also lost, which makes recovering the original intensity to be impossible. Ringing artifacts are introduced in this case.

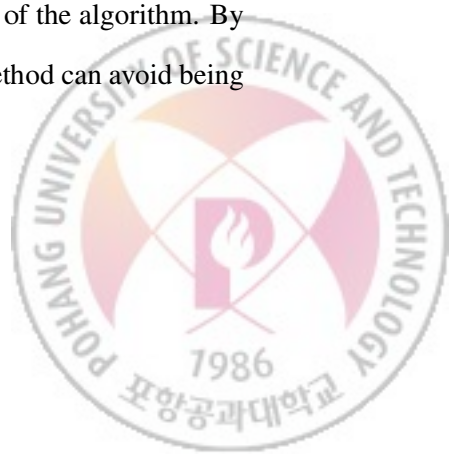
Can the deconvolution algorithm generate reasonable results even if original high intensity pixels cannot be recovered from their neighbors? The answer is yes. Since the pixels around the high intensity pixels also violate the linear blur model, they will be treated as outliers. By dealing with them properly in the EM algorithm, we can reduce the ringing artifacts, even though we cannot recover the true intensities. Figure 3.4e shows that our full deconvolution method can successfully recover a good latent image from the input image.

3.4.2 Effects of modeling clipped pixels

Since clipped pixels also break the linear blur model, one may wonder if we can drop the dynamic range check in Eq. (3.7), and let the clipped pixels be handled together with other outliers. This leads to another EM procedure which is similar to the one derived in Section 3.3, where the M step remains the same, and in the E step, $E[m(\mathbf{x})]$ is computed as

$$E[m(\mathbf{x})] \quad = \quad \frac{\mathcal{N}(b(\mathbf{x})|f(\mathbf{x})^{\text{o}}, \sigma)P_{\text{in}}}{\mathcal{N}(b(\mathbf{x})|f(\mathbf{x})^{\text{o}}, \sigma)P_{\text{in}} + CP_{\text{out}}}. \tag{3.20}$$

In Eq. (3.15), non-zero $E[m(\mathbf{x})]$ is obtained only when $f^{\text{o}}(\mathbf{x}) \in$ DR. In contrast, in Eq. (3.20), $E[m(\mathbf{x})]$ is always non-zero regardless of $f^{\text{o}}(\mathbf{x})$, even though it can be small for a clipped pixel $\mathbf{x}$. The difference looks small, but it has a significant impact on the performance of the algorithm. By completely excluding the clipped pixels from the optimization process, our method can avoid being



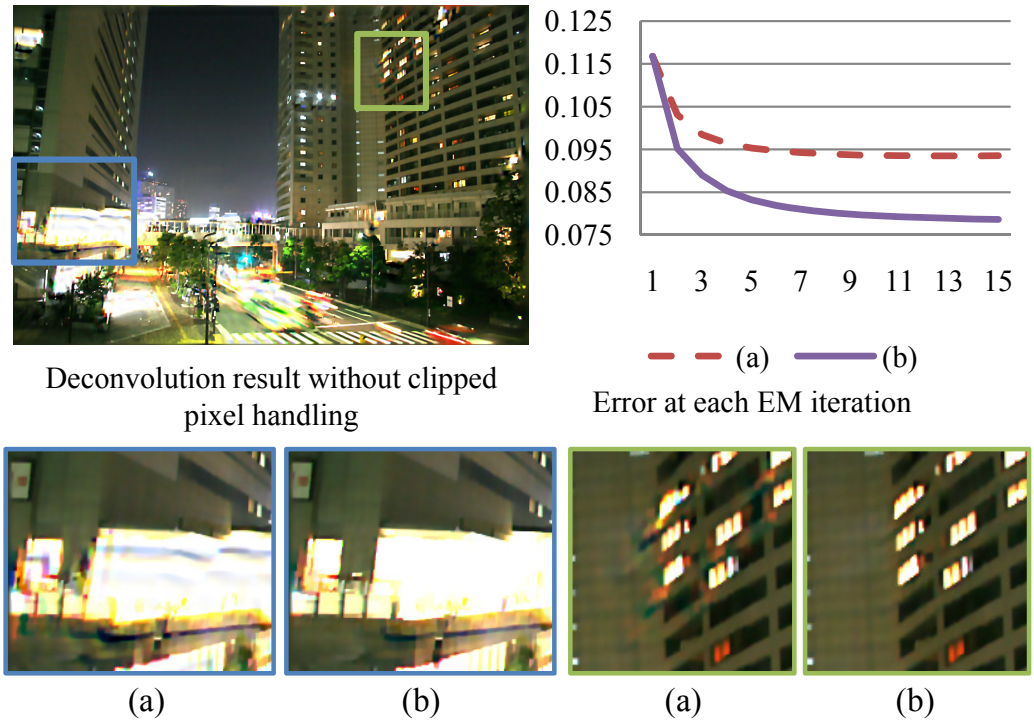


Figure 3.5: Deconvolution result of the blurred image in Figure 3.2 obtained without modeling clipped pixels. Bottom row: magnified patches of (a) the deconvolution result without modeling clipped pixels and (b) the result in Figure 3.2. Upper right: plots of root mean squared errors (RMSE) from the ground truth at EM iterations for the methods without and with modeling clipped pixels.

distracted by the corrupted information from them, and converge to a better solution quickly.

Figure 3.5 shows an example where we apply Eq. (3.20) in the E step. Due to the proposed outlier handling mechanism, the deconvolution result does not have significant artifacts overall, except some moderate ringing artifacts around saturated regions (Figure 3.5a). However, using Eq. (3.15) in the E step still leads to a better result (Figure 3.5b). This is consistent with the plots of error values in Figure 3.5, which shows that the algorithm converges to a more accurate result just in a few iterations using Eq. (3.15). This leads to the conclusion that for images with saturated pixels, which is a common scenario in night photographing, explicitly modeling clipped pixels can help the proposed algorithm achieve better results more efficiently.

Figure 3.6: Deconvolution results of a synthetically blurred image. From left to right: blurred input with uniform noise and saturated pixels, sparse deconvolution, L^1-norm based deconvolution, our method, and the ground truth image.

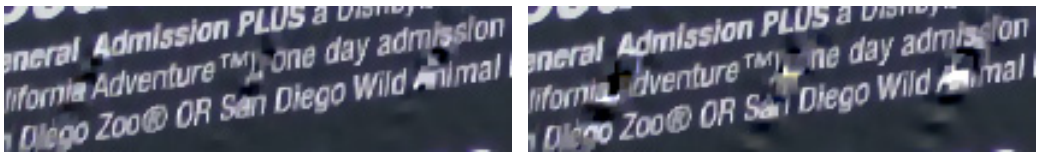


Figure 3.7: Deconvolution results of (b) and (c) of Figure 3.3 using L^1-norm based deconvolution.

3.5 Results

We implemented our method in Matlab. Our testing environment is a PC running MS Windows 7 64bit version with Intel Core i7 CPU and 12GB RAM, but we did not use the multi-core facility. The average computation time is about six minutes for images of one mega-pixels. Our project webpage[1] provides high-resolution versions of the images shown in this section and additional examples, as well as Matlab source code of our method.

Figure 3.6 shows a synthetic example. The input image was synthetically blurred and high intensity pixels were clipped. Uniform noise was also added. We deconvolved the blurred image using a sparse prior [35], L^1-norm based deconvolution [4], and our method. The L^1-norm based deconvolution method uses an L^1-norm data fidelity term, which is known to be robust to outliers, with a sparse image prior. For comparison, we implemented L^1-norm based deconvolution using an

[1] http://cg.postech.ac.kr/research/deconv_outliers

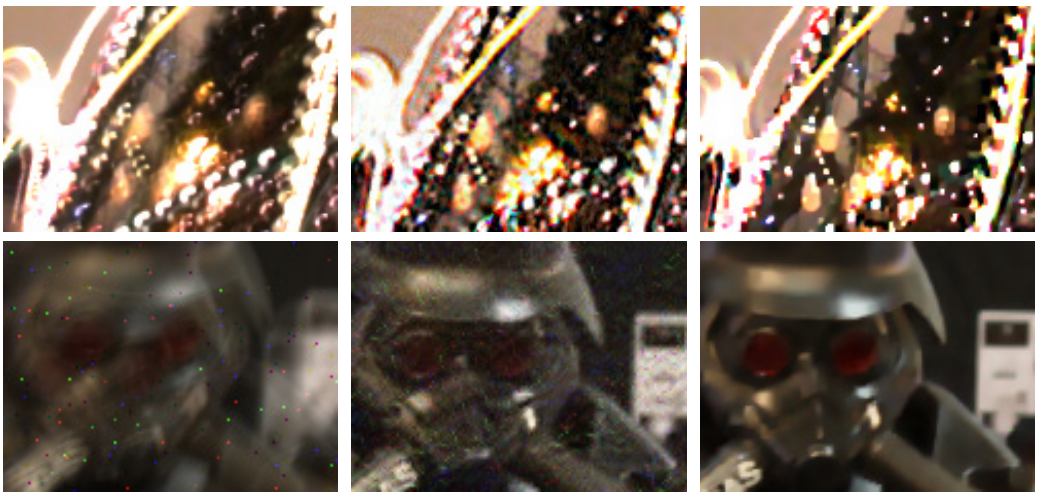


Figure 3.8: Comparison between Yuan et al.'s method [61] and ours. From left to right: Synthetically blurred inputs with uniform noise and saturated pixels, Yuan et al.'s method, and our method. Our results show less ringing artifacts than Yuan et al.'s results.

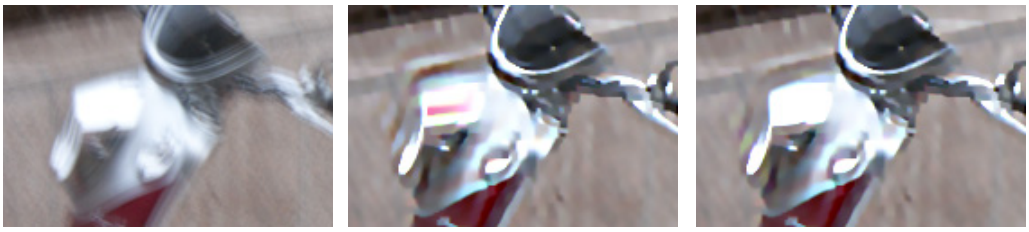


Figure 3.9: Comparison between Harmeling et al.'s method [23] and ours. From left to right: real blurred input with saturated pixels, Harmeling et al.'s method, and our method. Saturated pixels are better recovered in our result.

IRLS method. In Figure 3.6 we can see the result of sparse deconvolution contains artifacts caused by both uniform noise and clipped pixels. The result of L^1-norm based deconvolution shows no artifacts from uniform noise, but still has artifacts around saturated pixels. In contrast, our result contains no artifacts for both uniform noise and saturated pixels. We also measured peak-signal-to-noise ratios (PSNR) of the results. From the blurred input to our result in Figure 3.6 (left to right), PSNR values are 17.15, 16.46, 18.56, and 21.91. Our result achieves a much higher PSNR than the others.

While we added outliers with the size of one pixel to the blurred image in Figure 3.6, outliers

can be larger in practice. In Figure 3.7, we applied L^1-norm based deconvolution to the blurred images in Figure 3.3, to compare its performance on outlier handling with our method. The results show that our method suppresses large outliers better than L^1-norm based deconvolution. This is because our blur model better describes outliers than a Laplace distribution, which is the basis of the L^1-norm data fidelity term.

Figure 3.8 shows a comparison between the deconvolution method of Yuan et al. [61] and ours. As our model explicitly handles outliers, our results have no visible artifacts around saturated pixels and other outliers. In contrast, the results of Yuan et al.'s method present noticeable artifacts, and less details are recovered in their results.

Figure 3.9 shows a comparison between the deconvolution method of Harmeling et al. [23] and ours. Harmeling et al.'s method detects saturated pixels by thresholding input blurry images, which can be erroneous as there is no guidance on the optimal threshold value. In contrast, our method detects saturation on the latent image without involving a threshold, and better recovers saturated pixels.

Figure 3.10 shows deconvolution results of real blurred images. To obtain blur kernels for real images, we used the blind deconvolution method of Cho and Lee [14]. Since outliers may degrade the quality of kernel estimation, we cropped rectangular regions without obvious outliers (e.g., saturated pixels) from input images, and estimated blur kernels using them. We then applied deconvolution methods to the input images using the estimated kernels. The results show that our method handles outliers most effectively. In summary, our experimental results show that our deconvolution method leads to significantly less artifacts, such as ringing around saturated pixels or high-frequency noise, than previous approaches.

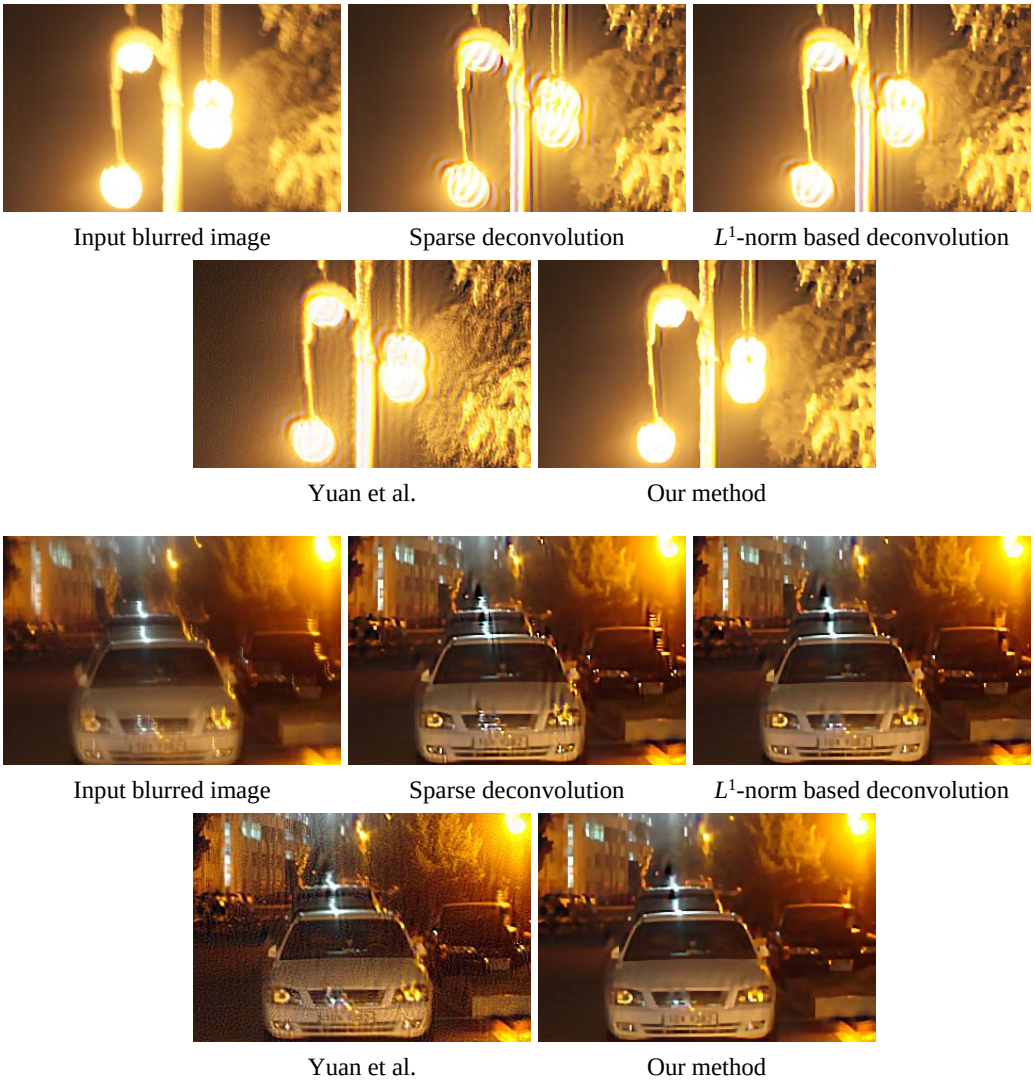


Figure 3.10: Deconvolution results of real blurred images. Our results show less ringing artifacts than the others.

3.6 Discussion

In this chapter, we analyzed various types of outliers, which cause severe ringing artifacts in previous deconvolution methods. Based on the outlier analysis, we proposed a robust, EM-based non-blind deconvolution method, which explicitly detects outliers and properly handles them in the deconvolution process. We further provided detailed analysis on outlier pixel recovery and clipped

pixel handling of our approach. We evaluated our method with both synthetic and real-world blur images, and compared the results with other state-of-the-art methods.

Previous studies have pointed out that real blurs are often non-uniform and spatially varying [37]. While our method was derived under the assumption that the blur is spatially invariant. the method can be easily extended to handle non-uniform blurs. This can be achieved by replacing the convolution operation in our formulation with a non-uniform blurring operation, e.g., the blurring operation using a projective motion chain [54] or the segmentation based blur model in Chapter 4.

Chapter 4

Non-Uniform Motion Deblurring from Multiple Blurred Images

While motion deblurring has been studied by many researchers, most methods solve the problem under an assumption that there is only a single motion blur kernel for the entire image as Eq. (1.1) describes. However, in real-world cases, photographed images often have spatially-varying motion blurs due to multiple relative motions caused by moving objects or depth variations from the camera (Fig. 4.1).

This chapter proposes an algorithm for removing such spatially-varying motion blurs from images. The removal of spatially-varying motion blurs involves three different problems: estimation of motions, segmentation into regions of homogeneous motions, and estimation of motion blur kernels. We jointly solve these problems by an energy minimization approach. Our algorithm uses a regularized form of the energy function and minimizes it by an alternating optimization technique. From two or more input images, images are restored by removing spatially-varying motion blurs.

The primary contributions of the work in this chapter are twofold. First, it proposes a new approach to restoring images that are contaminated by spatially-varying motion blurs. Second, in addition to the restoration, associated motion blur kernels, segmentation, and motions are simultaneously obtained. The proposed method has wider applicability compared with the prior approaches that only assume a single motion blur kernel.

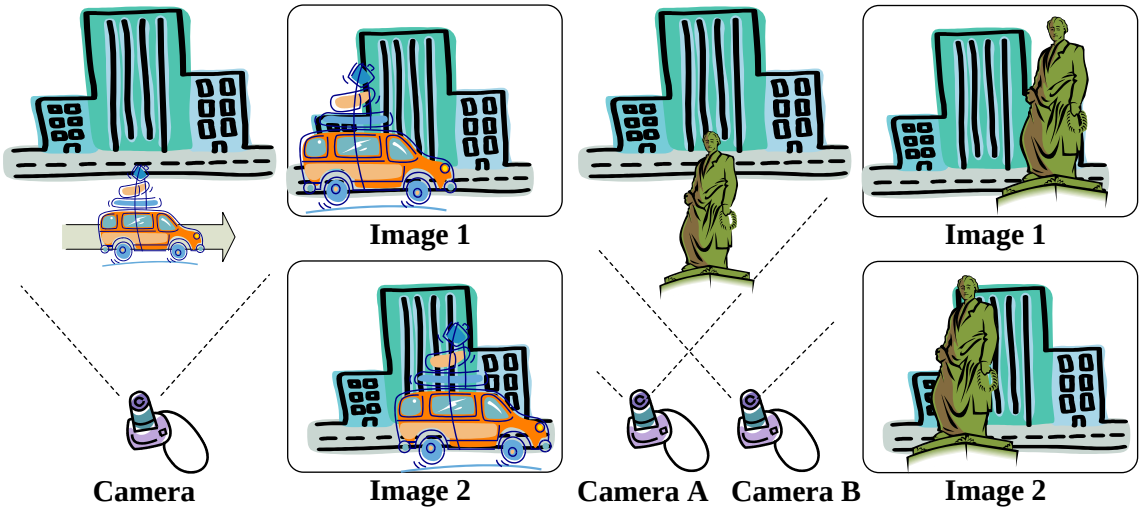


Figure 4.1: Left: different motions due to a moving object. Right: different motions due to a depth difference.

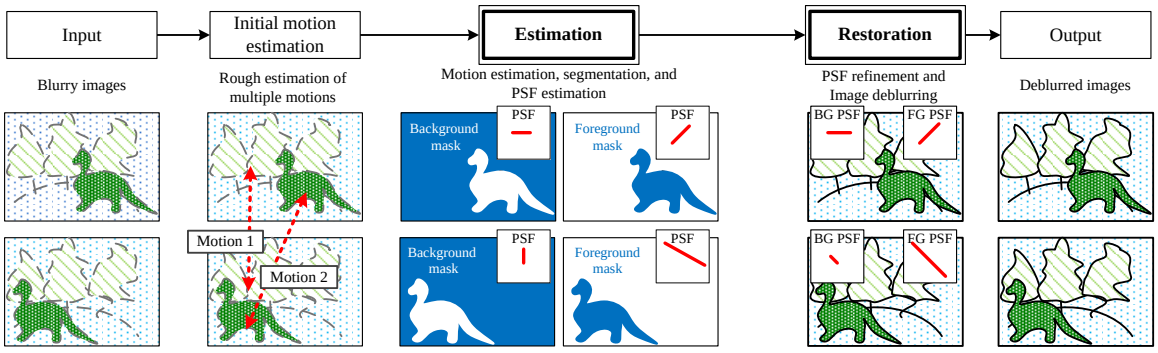


Figure 4.2: Overview of the proposed method. Given blurry images as input, the algorithm finds the initial guess for the multiple motion estimation using Bhat et al.'s method [9]. In the estimation step, accurate motion estimation, segmentation, and motion PSF estimation are performed. With these estimates, the restoration step deblurs input images with refinement of the motion PSF estimates.

4.1 Related Work

To handle non-uniform blur caused by moving objects and depth variation, a few segmentation based approaches have been proposed. Levin [34] proposed a method to restore a blurry image that contains one motion blurred object and a sharp background. The algorithm uses the statistical

distributions of image gradients to estimate motion blurs and to locate a blurry foreground object. Bardsley et al. [5] used a phase diversity method to find motion PSFs in which segmentation is performed by splitting an image into regions that have homogeneous motion PSFs.

In the multiple image framework, Chen et al. [13] proposed a method for recovering the original image from two consecutively blurred images. The method adopts an energy minimization technique that computes the best translation of the camera from the two consecutive images. Similar to Chen et al.'s method, Rav-Acha and Peleg [44] used two blurry images (one is horizontally and the other is vertically blurred) for deblurring by simultaneously estimating parameters of image displacements using iterative energy minimization. Bascle et al. [6] proposed a method to remove motion blur and generate high-resolution images from an image sequence by finding pixelwise motion. Bar et al. [3] developed a unified framework of segmentation and blind restoration by treating the image segmentation and restoration as a coupled problem. However, these methods are limited to uniform motion blur not spatially-varying blur.

4.2 Overview

Our goal is to restore motion blurred images in which spatially-varying motion blurs are contained. Figure 4.2 illustrates the overview of our method. Our algorithm takes multiple blurry images of roughly the same scene as input, e.g., consecutive frames from a video. To simplify the explanation, we describe our algorithm in the case of two input images although the algorithm can naturally work with more than two input images. Given the input images, our method first finds an initial guess of multiple different motions that appear in the images. For this step, we use Bhat et al.'s motion fitting method [9]. Since the accurate motion estimation is performed in the following step, the initial estimation result need not necessarily be precise.

Once the initial estimation of motion parameters is obtained, the first stage of the proposed method simultaneously refines the motion estimates, performs image segmentation into regions of

homogeneous motions, and estimates the corresponding motion PSFs. The estimated PSFs are further refined in the following restoration stage, where we perform the refinement and image restoration at the same time, using a method similar to Jin et al.'s work [29].

There are a few assumptions in the proposed method. First, a piecewise affine motion between image segments is assumed. Second, we assume that a motion PSF is dominated by a linear component.

4.3 Formulation

4.3.1 Motion Blur Model

Let b be an image that contains a motion blur and l be the corresponding sharp image in which the motion blur does not exist. Assuming that a PSF is dominated by a linear component, these two images b and l are related by the following equation

$$b(\mathbf{x}) = k_{\mathbf{v}} * l(\mathbf{x}) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} l(\mathbf{x} + \mathbf{v}t) dt, \tag{4.1}$$

where $\mathbf{x}$ is a 2D vector representing a pixel position, $\mathbf{v}$ is a 2D motion vector corresponding to the relative motion between a camera and the scene, and $k_{\mathbf{v}}$ is a motion blur kernel parametrized by the motion vector $\mathbf{v}$.

In a captured image, multiple different motions and spatially-varying motion blurs may exist. To represent n different relative motions and motion blurs, we use the subscript i such as $\mathbf{v}_i$ and $k_{\mathbf{v}_i}$, where $0 \leq i < n$, to indicate the i-th motion. We also introduce a segmentation mask vector m_i to indicate the i-th image region that corresponds to the i-th relative motion. In m_i, we use floating-point numbers instead of binary numbers, and all elements are set in the range $[0, 1]$. For example, if $m_i(\mathbf{x}) = 1$, the pixel $\mathbf{x}$ belongs to the i-th motion. As $m_i(\mathbf{x})$ decreases, the pixel $\mathbf{x}$ is less likely to belong to the i-th motion. Using this notation, the motion blur caused by the i-th motion can be described as

$$m_i(\mathbf{x}) b(\mathbf{x}) \quad = \quad m_i(\mathbf{x}) \left\{ k_{\mathbf{v}_i} * l(\mathbf{x}) \right\}. \tag{4.2}$$

Assume that there are two sharp images l_0 and l_1 of roughly the same scene. Image segments of these two images are associated by piecewise affine motions as

$$m_i(\mathbf{x})l_0(\mathbf{x}) \quad = \quad m_i(\mathbf{x})l_1(A_i(\mathbf{x})), \tag{4.3}$$

where A_i is an affine transform of the i-th motion that maps a pixel position from l_0 to l_1. When these image segments are blurred by spatially-varying motion PSFs, Eq. (4.3) changes to

$$\begin{aligned} m_i(\mathbf{x})b_0(\mathbf{x}) &= m_i(\mathbf{x})\left\{k_{\mathbf{v}_{0i}} * l_0(\mathbf{x})\right\}, \qquad \text{and} \\ m_i(\mathbf{x})b_1(A_i(\mathbf{x})) &= m_i(\mathbf{x})\left\{k^{A_i}_{\mathbf{v}_{1i}} * l_1(A_i(\mathbf{x}))\right\}, \end{aligned} \tag{4.4}$$

where b_0 and b_1 are blurry images, and $k^{A_i}_{\mathbf{v}_{1i}}$ is the i-th blur kernel parametrized by a vector $\mathbf{v}^{A_i}_{1i}$. $\mathbf{v}_{0i}$ represents the i-th motion blur parameter in image b_0. Likewise, $\mathbf{v}_{1i}$ denotes the i-th motion blur parameter of image b_1. In the following, we denote k_{0i} and $k^{A_i}_{1i}$ to represent $k_{\mathbf{v}_{0i}}$ and $k^{A_i}_{\mathbf{v}_{1i}}$, respectively. For a 2D motion vector $\mathbf{v}$, $\mathbf{v}^A$ indicates a vector warped by a transform A without the translation component.

From Eqs. (4.3) and (4.4), we obtain

$$m_i(\mathbf{x})b_1(A_i(\mathbf{x})) \quad = \quad m_i(\mathbf{x})\left\{k^{A_i}_{1i} * l_0(\mathbf{x})\right\}. \tag{4.5}$$

Because of occlusions or image boundaries, image areas that do not have corresponding regions in the other image occur. For these image areas, segments cannot be defined. We call these occluded pixels and occluded regions.

4.3.2 Estimation Step

This section describes the algorithm for simultaneously estimating multiple relative motions, segmentation, and motion PSFs.

Since motion blurs are commutative, applying motion PSFs k_{0i} and $k^{A_i}_{1i}$ to the i-th segments of

$b_1(A_i(\mathbf{x}))$ and $b_0(\mathbf{x})$ respectively will produce the same result, which can be written as

$$\begin{aligned} & m_i(\mathbf{x}) \left\{ k_{1i}^{A_i} * b_0(\mathbf{x}) \right\} = m_i(\mathbf{x}) \left\{ k_{1i}^{A_i} * k_{0i} * l(\mathbf{x}) \right\} \\ = & \, m_i(\mathbf{x}) \left\{ k_{0i} * k_{1i}^{A_i} * l(\mathbf{x}) \right\} = m_i(\mathbf{x}) \left\{ k_{0i} * b_1(A_i(\mathbf{x})) \right\} \end{aligned} \tag{4.6}$$

where $l = l_0$. In the following, to simplify the notation, we will use l to represent l_0. Assuming that the input images have Gaussian noise, from Eq. (4.6), we can obtain the following least-squares problem

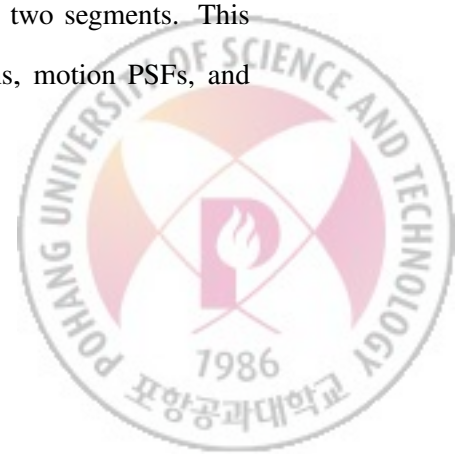
$$\sum_i \sum_{\mathbf{x}} m_i(\mathbf{x}) \left| k_{0i} * b_1(A_i(\mathbf{x})) - k_{1i}^{A_i} * b_0(\mathbf{x}) \right|^2 \quad = \quad 0. \tag{4.7}$$

Therefore, we expect that minimizing the left term of Eq. (4.7) yields the solutions of k and A, if masks m_i are properly set. However, due to the non-linearity of the equation, the solution that minimizes the term cannot be determined uniquely. To assure the uniqueness of the optimal solution, we use additional constraints similar to Jin et al.'s method [29]. For masks m_i, setting all the elements to zero can produce a numerically optimal result when minimizing the left term of Eq. (4.7), however, this approach should be avoided. In addition, it is better to preserve edges in mask images while smooth transition of mask values should be observed in blurry image regions. Combining all these constraints, we design a regularized form of the energy function as follows.

$$\begin{aligned} E(\mathbf{v}, \mathbf{A}, \mathbf{m}) = & \underbrace{\sum_i \sum_{\mathbf{x}} m_i(\mathbf{x}) \left| k_{0i} * b_1(A_i(\mathbf{x})) - k_{1i}^{A_i} * b_0(\mathbf{x}) \right|^2}_{error\ term} \\ + & \underbrace{\alpha \sum_i \left\{ \|\mathbf{v}_{0i}\|^2 + \|\mathbf{v}_{1i}^{A_i}\|^2 \right\}}_{motion\ blur\ term} + \underbrace{\beta \sum_{\mathbf{x}} \left| 1 - \sum_i m_i(\mathbf{x}) \right|^2}_{occlusion\ term} \\ + & \underbrace{\gamma \sum_i \sum_{\mathbf{x}} \left\{ g(\mathbf{x}) \left| \frac{\partial m_i(x,y)}{\partial x} \right|^2 + g(\mathbf{x}) \left| \frac{\partial m_i(x,y)}{\partial y} \right|^2 \right\}}_{edge\ term}, \end{aligned} \tag{4.8}$$

where $\mathbf{v}, \mathbf{A}$ and $\mathbf{m}$ are the sets of $\mathbf{v}_{ji}$, A_i, and m_i, respectively. α, β, and γ are weighting factors.

In Eq. (4.8), the error term computes the weighted differences between two segments. This is the core term in the energy function for estimating the multiple motions, motion PSFs, and



segmentation. The motion blur term assures the uniqueness of the motion PSFs by penalizing to excessively long motion vectors. The occlusion term accounts for the occluded pixels that are measured by $(1 - \sum_i m_i(\mathbf{x}))$. The edge term makes mask estimates rapidly change their values around edges, but slowly change them in the smooth regions. To evaluate transitions of mask values, spatial gradients of mask values are computed. g is a weighting function, which is defined as $g(\mathbf{x}) = 1/(\|\nabla b_0(\mathbf{x})\| + \delta)$. δ is a small positive constant used to prevent division by zero. The masks are constrained by $\sum_i m_i(\mathbf{x}) \leq 1 \ \wedge \ m_i(\mathbf{x}) \geq 0$ for all $\mathbf{x}$.

We optimize the energy function in Eq. (4.8) using an alternating optimization approach and gradient-based minimization techniques. The implementation details about the optimization are described in Section 4.4.

4.3.3 Restoration Step

The estimated PSFs in the previous stage are *relative* PSFs among images that are not necessarily *optimal* for generating sharp images via deconvolution. We call the PSFs that associate a sharp image with the blurry image the *optimal PSFs* to distinguish from the relative PSFs. This section describes an image restoration method that also computes optimal PSFs.

As described by Jin et al. [29], the optimal PSFs and the relative PSFs are related as

$$\hat{\mathbf{v}}_{0i} = \sqrt{s^2+1}\mathbf{v}_{0i} - s\mathbf{v}_{1i}^{A_i}, \quad \text{and} \tag{4.9}$$

$$\hat{\mathbf{v}}_{1i}^{A_i} = -s\mathbf{v}_{0i} + \sqrt{s^2+1}\mathbf{v}_{1i}^{A_i}, \tag{4.10}$$

where s is an unknown scalar. From these relations, the image b_0 can be deblurred to yield a sharp image l, except for the occluded regions, by minimizing the following function with respect to l and s:

$$\sum_i \sum_{\mathbf{x}} m_i(\mathbf{x}) \left\{ \left| b_0(\mathbf{x}) - k_{0i(s)} * l(\mathbf{x}) \right|^2 + \left| b_1(A_i(\mathbf{x})) - k_{1i(s)}^{A_i} * l(\mathbf{x}) \right|^2 \right\} + \phi_{\text{latent}}(l), \tag{4.11}$$

where $k_{0i(s)}$ and $k_{1i(s)}^{A_i}$ are blur kernels parametrized by motion blur vectors $\hat{\mathbf{v}}_{0i}$ and $\hat{\mathbf{v}}_{1i}^{A_i}$ computed by Eqs. (4.9) and (4.10), and $\phi_{\text{latent}}(l)$ is a regularization function. We used a regularization function

introduced by Geman et al. [19], $\phi_{\text{latent}}(l) = \zeta \sum_{\mathbf{x}} \|\nabla l(\mathbf{x})\|^2/(1+\|\nabla l(\mathbf{x})\|^2)$, where ζ is a weighting factor for the regularization term. ζ should be relatively small to avoid strong bias to $\phi_{\text{latent}}(l)$.

As mentioned earlier, the above method only works for images that do not have occluded regions. To handle the occluded regions, we assume that the occluded regions belong to the neighboring and dominant region of the homogeneous motion. This assumption holds for images where the dominant motion is observed in the background of the scene, e.g., by camera motion. Using this assumption, a new mask m^o is built by a dilation operation to assign the new mask values in the occluded regions. Once the new mask m^o is obtained, Eq. (4.11) is slightly modified so that the occluded regions are correctly handled;

$$\begin{aligned} E_{\text{restore}}(l,s) \quad = \quad & \sum_i \sum_{\mathbf{x}} \left\{ m_i^o(\mathbf{x}) \left| b_0(\mathbf{x}) - k_{0i(s)} * l(\mathbf{x}) \right|^2 + m_i(\mathbf{x}) \left| b_1(A_i(\mathbf{x})) - k_{1i(s)}^{A_i} * l(\mathbf{x}) \right|^2 \right\} \\ & + \phi_{\text{latent}}(l), \end{aligned} \tag{4.12}$$

By minimizing Eq. (4.12), the deblurred image l corresponding to b_0 and optimal PSFs k are obtained.

In the next section, we will cover how to optimize the energy functions defined in this section.

4.4 Implementation

4.4.1 Computation of Estimation Step

In the estimation step, motions $\mathbf{A}$, image segments $\mathbf{m}$, and non-uniform motion PSFs k (which are parametrized by $\mathbf{v}$) are estimated using Eq. (4.8). To jointly optimize these parameters, we use an alternating optimization approach for three sets of parameters $\{m_i\}$, $\{k_i\}$ and $\{A_i\}$. One of these parameter sets is updated at one step, and the others are updated in the following steps. This iteration continues until the energy function converges. We use a gradient-based unconstrained non-linear optimization method, `fminunc` of Matlab optimization toolbox, to optimize motion PSFs $\{k_i\}$, and a gradient projection method [46] to optimize segmentation masks $\{m_i\}$. For estimating motions $\{A_i\}$, we use a registration method described by Baker et al. [1].

For more convenient derivation of the gradients with respect to each variable, we first rewrite Eq. (4.8) in a matrix form as:

$$E(\mathbf{v},\mathbf{A},\mathbf{m}) = \sum_i \mathbf{d}_i^T\mathbf{M}_i\mathbf{d}_i + \alpha\sum_i\left\{\|\mathbf{v}_{0i}\|^2 + \|\mathbf{v}_{1i}^{A_i}\|^2\right\} + \beta\left\|\mathbf{1}-\sum_i\mathbf{m}_i\right\|^2 + \gamma\sum_i\left\{(\mathbf{D}_x\mathbf{m}_i)^T\mathbf{G}(\mathbf{D}_x\mathbf{m}_i) + (\mathbf{D}_y\mathbf{m}_i)^T\mathbf{G}(\mathbf{D}_y\mathbf{m}_i)\right\}, \quad (4.13)$$

where $\mathbf{d}_i$ is a vector consisting of the values of $\left(k_{0i} * b_1(A_i(\mathbf{x})) - k_{1i}^{A_i} * b_0(\mathbf{x})\right)$. $\mathbf{m}_i$ is a vector representation of m_i, and $\mathbf{M}_i$ is a diagonal matrix whose diagonal elements are the components of $\mathbf{m}_i$. $\mathbf{G}$ is a diagonal matrix whose diagonal elements consist of $g(\mathbf{x})$, and $\mathbf{1}$ is a vector whose all elements are set to 1. $\mathbf{D}_x$ and $\mathbf{D}_y$ are convolution matrices corresponding to partial derivative (or finite difference) operators $\partial/\partial x$ and $\partial/\partial y$, respectively.

Segmentation masks Segmentation masks m_i are estimated by optimizing the terms of Eq. (4.8) except the motion blur term. The gradient of m_i is then computed from Eq. (4.13) as

$$\nabla_{\mathbf{m}_i}E(\mathbf{v},\mathbf{A},\mathbf{m}) = \mathbf{d}_i^T\mathbf{d}_i - 2\beta(\mathbf{1}-\sum_j\mathbf{m}_j) + 2\gamma[\mathbf{D}_x^T\mathbf{G}\mathbf{D}_x + \mathbf{D}_y^T\mathbf{G}\mathbf{D}_y]\mathbf{m}_i. \quad (4.14)$$

Motion PSFs Motion PSFs can be estimated by optimizing the error term and the motion blur term of Eq. (4.8). By computing gradients of motion blur parameters $\mathbf{v}_{0i}$ and $\mathbf{v}_{1i}^{A_i}$ in a similar way to Jin et al.[29], we can obtain

$$\nabla_{\mathbf{v}_{0i}}E(\mathbf{v},\mathbf{A},\mathbf{m}) = 2\left(\mathbf{M}_i\dot{\mathbf{K}}_{0i}\nabla\mathbf{b}_1^{A_i}\right)^T\mathbf{d}_i + \alpha\mathbf{v}_{0i}, \quad \text{and} \quad (4.15)$$

$$\nabla_{\mathbf{v}_{1i}^{A_i}}E(\mathbf{v},\mathbf{A},\mathbf{m}) = -2\left(\mathbf{M}_i\dot{\mathbf{K}}_{1i}^{A_i}\nabla\mathbf{b}_0\right)^T\mathbf{d}_i + \alpha\mathbf{v}_{1i}^{A_i}, \quad (4.16)$$

where $\nabla\mathbf{b}_0 = [\mathbf{D}_x\mathbf{b}_0 \;\; \mathbf{D}_y\mathbf{b}_0]$ is a matrix consisting of two column vectors of image gradients, $\mathbf{b}_0$ is a vector representation of b_0, and $\dot{\mathbf{K}}_{\mathbf{v}}$ is a matrix representation of convolution with a kernel $\dot{k}_{\mathbf{v}}$ defined as

$$\dot{k}_{\mathbf{v}} * l = -\int_{-\infty}^{\infty}\frac{t}{\sqrt{2\pi}}e^{-t^2/2}l(\mathbf{x}+\mathbf{v}t)dt. \quad (4.17)$$

Motion parameters Motion parameters A_i can be computed by optimizing the error term of Eq. (4.8). To optimize the error term, $k_{0i} * b_1(A_i(\mathbf{x}))$ and $k_{1i}^{A_i} * b_0(\mathbf{x})$ should be computed at each iteration. However, if we assume that current motion parameters A_i are close enough to the solution, we can approximate the minimization by image registration between $b'_1(\mathbf{x}) = k_{0i} * b_1(A_i(\mathbf{x}))$ and $b'_0(\mathbf{x}) = k_{1i}^{A_i} * b_0(\mathbf{x})$. Consequently, motion parameters can be computed by Lucas-Kanade based registration method between b'_0 and b'_1 with weighting factors m_i. For this computation, we used a weighted inverse-compositional method described by Baker et al. [1].

Hierarchical estimation For large images, estimation of these parameters is computationally intensive, and obtaining an optimal solution becomes difficult due to the number of unknown parameters. To avoid this problem, we used a pyramidal approach for the computation in a coarse-to-fine manner. To assign initial values for the estimation in a finer level, segmentation masks are enlarged by a bilinear interpolation method. An estimate of the motion blur parameter at a pixel is multiplied by two and propagated to the corresponding four pixels at the next level of the pyramid. Likewise, the estimated motion parameters are propagated to the next level of the pyramid with the translation parameters multiplied by two.

4.4.2 Computation of Restoration Step

The image restoration and PSF refinement are also performed by a gradient-based optimization. We use a gradient projection method to force pixel values of l to be within the valid range, and an unconstrained gradient descent method to optimize the parameter s of Eq. (4.12). We describe the details of optimizing Eq. (4.12) for obtaining the deblurred image l and optimal PSFs.

As done for the estimation step, we rewrite Eq. (4.12) in a matrix form as:

$$E_{\text{restore}}(\mathbf{l}, s) \quad = \quad \sum_i (\mathbf{d}_{0i}^T \mathbf{M}_i^o \mathbf{d}_{0i} + \mathbf{d}_{1i}^T \mathbf{M}_i \mathbf{d}_{1i}) + \phi_{\text{latent}}(\mathbf{l}). \tag{4.18}$$

where $\mathbf{d}_{0i}$ and $\mathbf{d}_{1i}$ are vectors consisting of the values of $(b_0 - k_{0i(s)} * l)$ and $\{b(A_i(\mathbf{x})) - k_{1i(s)}^{A_i} * l(\mathbf{x})\}$, respectively. $\mathbf{M}_i^o$ and $\mathbf{M}_i$ are diagonal matrices corresponding to m_i^o and m_i, respectively.

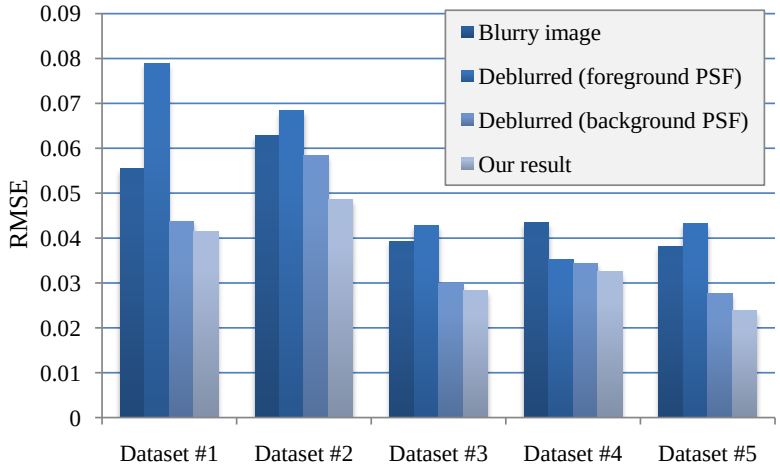


Figure 4.3: Comparison of RMSEs from the ground truth using synthetic datasets that contain background and foreground motion PSFs. From left to right bars in each dataset, RMSE of the input blurry image, the deblurred image using the background PSF, the deblurred image using the foreground PSF, and our result are shown.

Image deblurring From Eq. (4.18), the gradient of the deblurred image $\mathbf{l}$ can be written as

$$\nabla_{\mathbf{l}} E_{\text{restore}}(\mathbf{l}, s) = -2\mathbf{K}_{0i(s)}^{T}\mathbf{M}_i^o\mathbf{d}_{0i} - 2\big(\mathbf{K}_{1i(s)}^{A_i}\big)^{T}\mathbf{M}_i\mathbf{d}_{1i} + \nabla_{\mathbf{l}}\phi_{\text{latent}}(\mathbf{l}), \tag{4.19}$$

where $\mathbf{K}_{0i(s)}$ and $\mathbf{K}_{1i(s)}^{A_i}$ are convolution matrices corresponding to $k_{0i(s)}$ and $k_{1i(s)}^{A_i}$, respectively.

Motion PSFs To obtain the optimal PSFs, the gradient of parameter s in Eq. (4.18) can be computed as

$$\frac{\partial E_{\text{restore}}(\mathbf{l}, s)}{\partial s} = -2\mathbf{d}_{0i}^{T}\mathbf{M}_i^o(\dot{\mathbf{K}}_{0i(s)}\nabla\mathbf{l})(q\mathbf{v}_{0i} - \mathbf{v}_{1i}^{A_i}) - 2\mathbf{d}_{1i}^{T}\mathbf{M}_i(\dot{\mathbf{K}}_{1i(s)}^{A_i}\nabla\mathbf{l})(q\mathbf{v}_{1i}^{A_i} - \mathbf{v}_{0i}), \tag{4.20}$$

where $q = s/\sqrt{s^2+1}$.

4.5 Results

To evaluate the proposed algorithm, we used both synthetic and real-world images. The input images are first converted into grayscale images to perform the estimation step. Once the parameters are obtained, the restoration step is performed for each color band to produce the final result. Throughout our experiments, we set parameters in Eq. (4.8) as $(\alpha, \beta, \gamma, \delta) = (10^{-2}, 10^{-3}, 2.0 \times 10^{-5}, 10^{-4})$, and the weighting factor of ρ in Eq. (4.12) as $\zeta = 10^{-5}$. The image intensity is normalized in the range $[0, 1]$.

We ran our algorithm over five synthetically-blurred images. For images of 640×480 pixels, average computation time is about six minutes for the estimation step and 40 seconds for the restoration step. Root mean squared errors (RMSE) from the ground truth (original sharp image) were measured in normalized intensity (Figure 4.3). From left to right for each dataset, the first column shows RMSE of the blurry image, and the second and third columns show RMSEs computed from the restored images using the foreground PSF and the background PSF respectively. The right-most column shows the RMSE of our result. Images restored by only foreground PSFs show errors even larger than that of blurred images. Also, the images restored only by background PSFs show slightly larger errors than our results. This is because the area of the background region is usually large enough to make the RMSE numerically small. However, perceptual differences are large due to the large error in the foreground region.

Four different real-world scenes that contain spatially-varying motion blurs were deblurred by our method (Figure 4.4). The first and second rows show the blurry input images. The third and fourth rows show the segmentation masks that correspond to the background and foreground, respectively. Brighter pixels indicate the higher confidence in being the part of background/foreground. Pixels that do not belong to either the foreground or the background are estimated to be in the occluded regions. In the fifth row, estimated PSFs are shown in a tabular form, e.g, 1-A corresponds

to the PSF of Input 1, segment A (background). The following two rows show the deblurred results restored from input 1 and 2. For the deblurring, the estimated non-uniform PSFs are used for the corresponding image regions. For better visualization, the bottom row shows magnified image portions.

Input images containing three different motions were tested to show that our method is able to handle more than two different motions (Figure 4.5). These three motions correspond to the background, a pencil vase, and a cigarette pack, respectively, because of depth differences among them. The cigarette pack is the nearest object to a camera, and the pencil vase lies between the cigarette pack and the background. Our method successfully segments the scene, estimates the motion PSFs, and deblurs the input images. The bottom row of the figure shows the magnified views of the image portions for a better comparison.

Figure 4.6 depicts the power of handling spatially-varying motion PSFs. The foreground object is blurred by its own motion while background is also blurred by a camera motion. The image deblurred only with the background PSF shows severe artifacts on the foreground object due to the inconsistency with the foreground PSF (Figure 4.6). On the other hand, the image deblurred only with the foreground PSF yields a blurry background (Figure 4.6c). Our method is able to avoid this problem because it uses non-uniform motion PSFs for image restoration (Figure 4.6).

4.6 Discussion

In this chapter, we proposed a new method for removing non-uniform motion blurs from images caused by moving objects and depth variation. To achieve this goal, the problem is formulated as simultaneous estimation of multiple motions, segmentation, and spatially-varying motion PSFs. The problem is solved by optimizing a regularized form of energy function. Furthermore, the estimated PSFs are refined to achieve the restoration of images. We have evaluated the proposed method using a variety of synthetic and real-world images and validated the effectiveness of the algorithm.

Although we used only two images as input, the algorithm can naturally take more than two

images too. In this case, the estimation step is performed for each pair of images, and the restoration step is performed by simultaneously optimizing the energy functions of all pairs of images.

Limitations Our algorithm has a few limitations. One is that it shows some artifacts around the boundaries of different motions in restored images. Blurry regions on boundaries of the foreground object still remain (Fig. 4.6). This artifact is inevitable due to missing information of hidden pixels behind the foreground objects. Second, like existing segmentation algorithms, our segmentation is not performed well on textureless regions because it is difficult to determine the motion in such regions. Therefore, the proposed method works better if the input images are more textured.

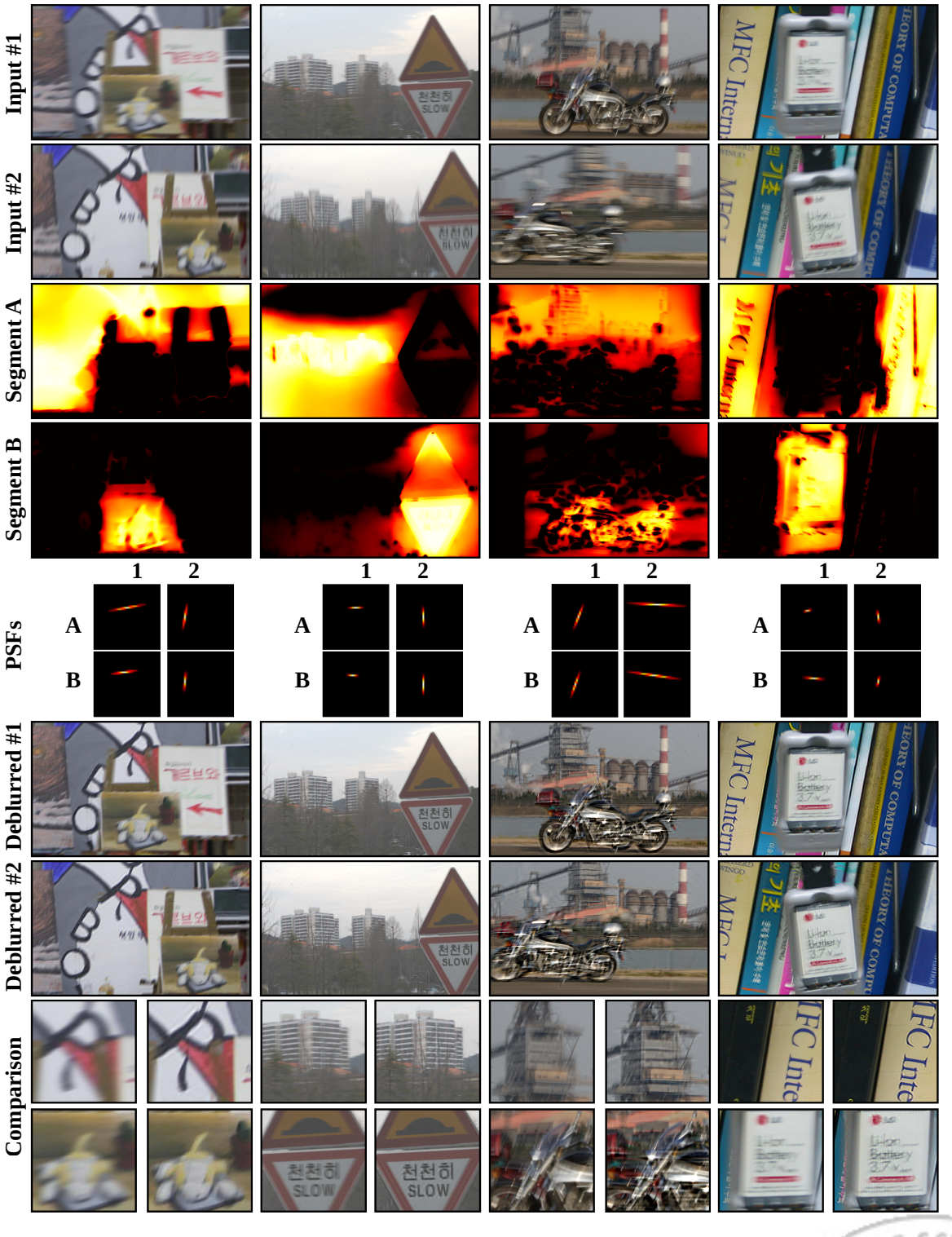


Figure 4.4: Results of our method. Among the four scenes, the left-most column shows the dataset generated synthetically, while the others are photographed from real-world scenes. The top two rows show blurry input images. Next two rows show segmentation masks followed by estimated motion PSFs. Below the motion PSFs, deblurred results are shown. The bottom row shows magnified views of original blurry images and the deblurred images for a better visualization purpose.

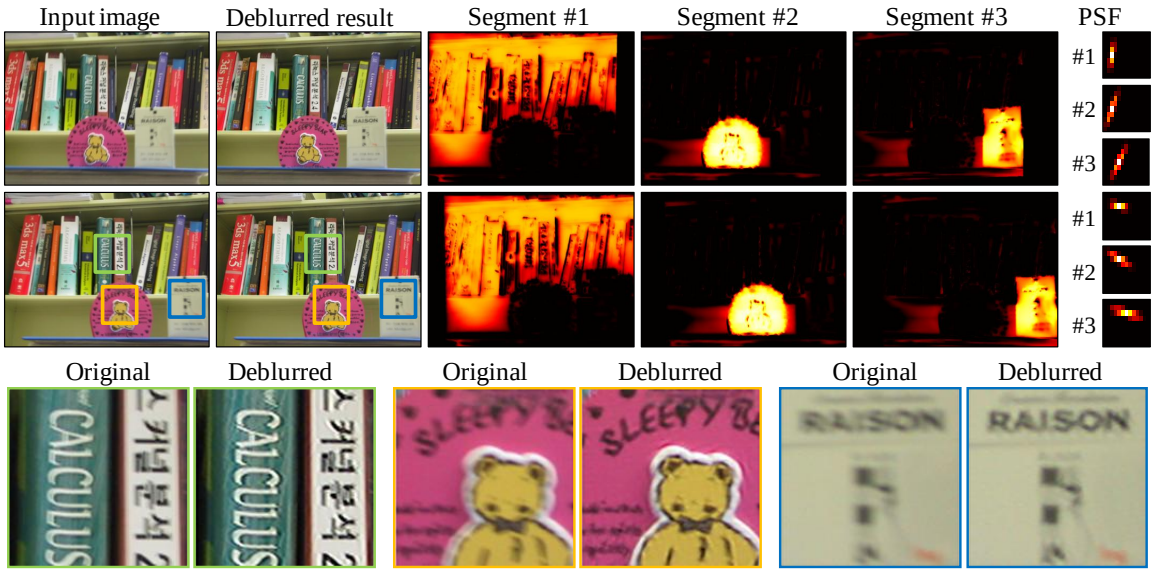


Figure 4.5: Result of deblurring images that contain three different motion and motion PSFs.

(a) One of two input images

(b) Deblurred by background PSF

(c) Deblurred by foreground PSF

(d) Our result (non-uniform PSF)

Figure 4.6: Comparison of deblurred results. (a) shows one of the two input images. (b) shows the deblurred result using the background motion PSF, and (c) shows the result using the foreground motion PSF. In (b) and (c), either foreground or background are not restored well due to the inaccurate PSFs. (d) shows the result of our method. Since our method uses non-uniform motion PSFs for restoring images, the result shows less artifacts compared with (b) and (c).

Chapter 5

Non-Uniform Motion Deblurring for Camera Shakes

While we proposed a method for removing non-uniform blur caused by moving objects and depth variation in Chapter 4, non-translational camera shakes such as rotational motions of a camera also cause non-uniform motion blur as shown by Levin et al. [37]. Such non-uniform blur caused by camera shakes is hard to handle using segmentation based approaches because such blur is globally smooth but different for every pixel.

To handle non-uniform blur caused by camera shakes, Tai et al. [54] introduced a projective motion blur model, which uses a small number of homographies to fully describe the motions of camera shakes in 3D. This blur model can describe translational, rotational and even zooming blur, which is often observed in a video. However, they proposed only a non-blind deblurring algorithm, which limits the usage of the model. Some recent techniques have tried to estimate the motion PSF of this model using coded exposure [53], video sequence [38], and motion sensor [31], but they need either special hardware or additional assumption on motion blur due to the difficulty of estimating a set of homographies.

As the projective motion blur model has clear benefits, natural questions would be: *Is it possible to accurately estimate non-uniform motion blur from a blurred image or images based on the model?*

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

Figure 5.1: Non-uniform motion deblurring: (a)&(b) Input images, (c) Our non-uniform motion deblurring result, (d) Uniform motion deblurring result [48].

How? In this chapter, we try to answer these questions by introducing registration based non-uniform motion PSF estimation algorithm. Our idea is to borrow the matured image registration techniques to the PSF estimation problem as the projective blur model uses homographies. Based on this idea, we design a PSF estimation algorithm by transforming it into a set of registration problems. To show the effectiveness of the algorithm, we apply it into a blind deblurring framework following a previous work [12], which uses multiple blurred images as input. Figure 5.1 shows an example of our inputs and their result.

In summary, the contributions of this chapter are as follows.

- We present a fully automatic non-uniform deblurring algorithm which utilizes the non-uniform motion blur model in [54]. We also discuss issues regarding our current approach and limitations.

- While there exist algorithms for non-uniform motion deblurring, those algorithms either require special hardware [53, 38, 31] or limit the freedom of camera motion in 3D world [56, 22, 24]. On the other hand, our approach is fully automatic, and allows the full 3D motion parameters of camera shake to be estimated.

- We bridge the gap between motion estimation and motion deblurring. While a PSF in motion

deblurring represents motions of camera, current deblurring algorithms can only be used to estimate a PSF, but cannot be used to estimate generic motion. Our approach shows that there is a possibility to use generic motion estimation, such as the Lucas-Kanade algorithm, for PSF estimation.

5.1 Related Work

For handling non-uniform motion blur caused by non-translational camera shakes such as rotational motions of a camera, a few approaches have been proposed. Shan et al. [49] handled a spatially varying blur by restricting the relative motion between the camera and the scene to be a global in-plane rotation. Tai et al. [52] used a hybrid camera [7] to estimate per-pixel PSFs using an auxiliary video camera. Dai and Wu [17] used an alpha matte to estimate a spatially varying motion PSF. However, these approaches rely on the conventional kernel based PSF model for spatially varying motion blurs, and have limitations. Not only does the model require significant storage to represent per-pixel PSFs, but it also significantly complicates the deblurring process by forcing each pixel to be processed independently.

After Tai et al. [54] introduced the projective motion blur model, several follow-up approaches have been proposed. Whyte et al. [56] proposed a non-uniform blur model, which assumes the motion blur model contains only rotational motions of a camera. Gupta et al. [22] estimate non-uniform blur consists of in-plane translations and rotations. Hirsch et al. [24] combines a patch-wise uniform motion blur model [25] and the rotational model in [56] to estimate a non-uniform PSF efficiently. These methods are restrictive with only three degrees of freedom for camera motion in 3D world. There is also a work that can fully estimate all of the six degrees of freedom of camera motion in 3D by using motion inertia sensors [31]. In this chapter, we try to explore a solution which can take the full advantage of the projective motion blur model with neither limiting the degree of freedom of camera motion nor using any special hardware.

5.2 Non-uniform Motion Blur Model

As the convolution operation is a weighted sum of translated versions of an image, the conventional motion blur model in Eq. (1.1) can be expressed as:

$$b(\mathbf{x}) = \sum_i w_i l(T_i(\mathbf{x})) + n(\mathbf{x}) \tag{5.1}$$

where T_i is a translational transform such that $T_i(\mathbf{x}) = \mathbf{x} + (x_i, y_i)^T$. It represents the in-plane translational motion of a camera at time t_i during the exposure period, which can be parametrized by a 2D vector. w_i is a weight representing the proportion, i.e., $\sum_i w_i = 1$, of time spent by the camera in that pose.

When a camera shake contains only translational motions, Eqs. (1.1) and (5.1) are simply different representations of the same model. However, if a camera shake includes non-translational motions, Eq. (1.1) cannot be immediately extended to represent non-uniform spatially varying motion blurs in b. In contrast, in Eq. (5.1), we can easily replace T_i by a general homography P_i, and obtain a new non-uniform motion blur model for a camera shake [54]:

$$b(\mathbf{x}) = \sum_i w_i l(P_i(\mathbf{x})) + n(\mathbf{x}) \tag{5.2}$$

Comparing Eq. (5.2) with Eq. (5.1), we have a significantly larger number of variables to estimate. A translation T_i has two independent parameters but a homography P_i contains eight parameters. While the conventional blind motion deblurring problem has already been known to be ill-posed, a blind motion deblurring algorithm for this new blur model is even more challenging. Fortunately, as we will present in the next section, we can transfer this ill-posed problem into a well-posed image registration problem that allows us to solve the blind motion deblurring problem effectively. This intuitive formulation and the solution process are the major contributions of this chapter.

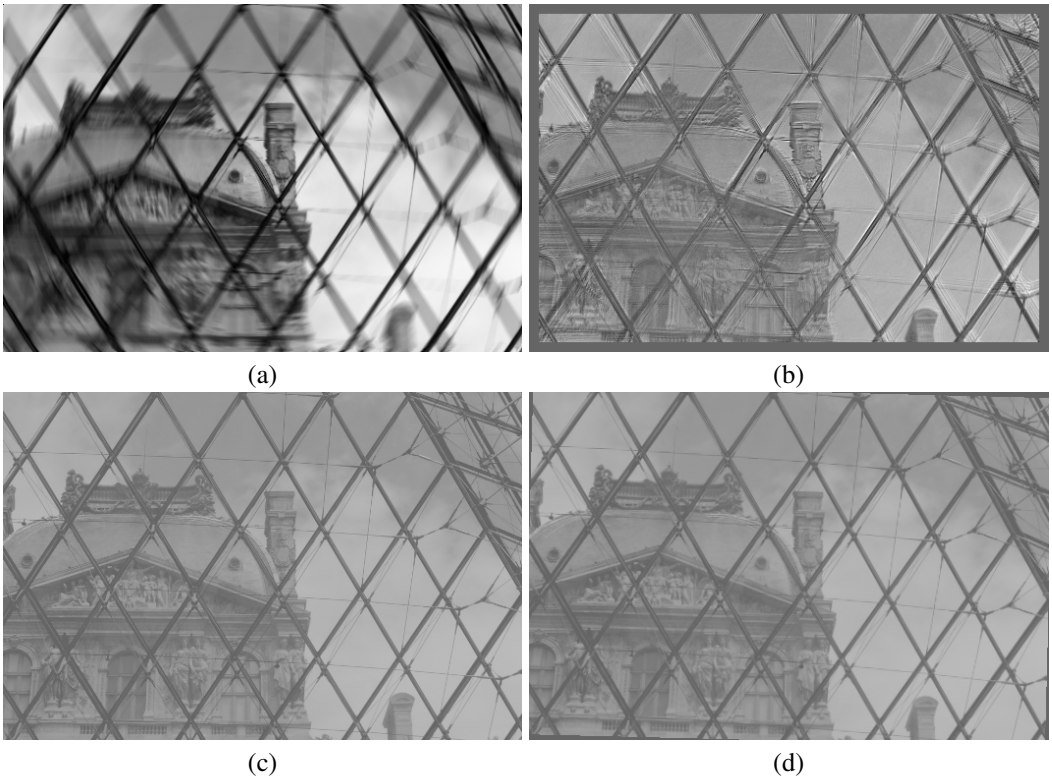


Figure 5.2: Image registration for kernel update. (a) Input motion blurred image, (b) Residual image e_i^t, (c) Attenuated latent image $w_i l^t$, (d) Registered image $w_i l^t(P_i(\mathbf{x}))$. Image brightness in (b), (c), and (d) were enhanced for visualization.

5.3 PSF Estimation

In this section, we describe our non-uniform PSF estimation method, which estimates a non-uniform PSF from a given blurred image and a latent sharp image. Such estimation is an important component of many blind deblurring systems, which alternatingly update a latent image and a PSF [48, 14, 24]. In the next section, we will apply the PSF estimation method into a blind deblurring system, which uses multiple blurred images.

Given a latent image l, we want to estimate PSF parameters ($\mathbf{P}$, $\mathbf{w}$) using the motion blur model in Eq. (5.2). We define the objective function for PSF estimation as:

$$\underset{\mathbf{P},\mathbf{w}}{\operatorname{argmin}} \left\{ \sum_{\mathbf{x}} |b(\mathbf{x}) - \sum_i w_i l(P_i(\mathbf{x}))|^2 + \phi_{\text{homo}}(\mathbf{P}) + \phi_{\text{weight}}(\mathbf{w}) \right\}, \tag{5.3}$$

where ϕ_{homo} and ϕ_{weight} are priors for the regularization of **P**, and **w**, respectively. **P** is the set of homographies $\{P_i\}$ and **w** is a vector such that $\mathbf{w} = (w_1, w_2, \cdots, w_p)^T$ where p is the number of homographies.

Without loss of generality, we can rearrange the order of homographies in Eq. (5.2) and obtain

$$b(\mathbf{x}) - \sum_{j \neq i} w_j l(P_j(\mathbf{x})) \quad = \quad w_i l(P_i(\mathbf{x})) + n(\mathbf{x}), \tag{5.4}$$

where the left-hand side is the residual image e_i for the homography P_i. Based on Eq. (5.4), we can transform the PSF estimation problem into an image registration problem by finding P_i that minimizes the difference between $w_i l(P_i(\mathbf{x}))$ and e_i (Figure 5.2). That is, we solve

$$\underset{P_i}{\operatorname{argmin}} \left\{ \sum_{\mathbf{x}} |e_i(\mathbf{x}) - w_i l(P_i(\mathbf{x}))|^2 + \phi_{\text{homo}}(P_i) \right\}, \tag{5.5}$$

where

$$e_i(\mathbf{x}) \quad = \quad b(\mathbf{x}) - \sum_{j \neq i} w_j l(P_j(\mathbf{x})). \tag{5.6}$$

This minimization can be handled using a Lucas-Kanade based image registration method [1]. Hence, for fixed l and w, we can estimate P by obtaining P_i one by one using Eq. (5.5). To regularize the image registration process, we set the prior $\phi_{\text{homo}}(P_i) = \lambda_{\text{homo}} \|\mathbf{P}_i - \mathbf{I}\|^2$, where $\mathbf{P}_i$ is the 3×3 homography matrix corresponding to P_i and **I** is the identity matrix. λ_{homo} is the relative weight for the prior, and in our implementation, we use $\lambda_{\text{homo}} = 1$, which we empirically found worked well in our experiments.

To obtain **w**, we rewrite Eq. (5.2) as

$$\mathbf{b} \quad = \quad \mathbf{L}\mathbf{w} + \mathbf{n}, \tag{5.7}$$

where **b** and **n** are vector representations of b and n, respectively. **L** is defined as:

$$\mathbf{L} \quad = \quad \begin{bmatrix} \mathbf{l}_1 & \mathbf{l}_2 & \cdots & \mathbf{l}_p \end{bmatrix} \tag{5.8}$$

where p is the number of homographies, and $\mathbf{l}_i$ is a vector representation of a transformed latent image $l(P_i(\mathbf{x}))$. We solve Eq. (5.7) by

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{b} - \mathbf{L}\mathbf{w}\|^2 + \phi_{\text{weight}}(\mathbf{w}), \tag{5.9}$$

where we set the prior $\phi_{\text{weight}}(\mathbf{w}) = \lambda_{\text{weight}}\|\mathbf{w}\|^2$ to avoid singularity of matrix inverse. Since the number of pixels of an image is much larger than the number of homographies, Eq. (5.9) can be solved effectively using a non-negative least square method, subject to $\sum_i w_i = 1$, by computing

$$\mathbf{w} \quad = \quad \left(\mathbf{L}^T\mathbf{L} + \lambda_{\text{weight}}\mathbf{I}\right)^{-1}\mathbf{L}^T\mathbf{b}. \tag{5.10}$$

where $\mathbf{I}$ is an identity matrix. In our implementation, we use the relative weight $\lambda_{\text{weight}} = 0.1$, which we empirically found worked well in most cases.

PSF estimation with known camera intrinsic parameters If the information about the camera is available, we can further restrict the parameters of homographies $\mathbf{P}$ by rewriting each $\mathbf{P}_i$ as follow:

$$\mathbf{P}_i \quad = \quad \mathbf{C}(\mathbf{R}_i + \mathbf{T}_i)\mathbf{C}^{-1}, \tag{5.11}$$

where $\mathbf{C}$ is the camera intrinsic matrix, which can be obtained from the images' EXIF tags, $\mathbf{R}_i$ is the rotational matrix, and $\mathbf{T}_i$ is the translational matrix. As a result, six parameters (three for $\mathbf{R}_i$ and three for $\mathbf{T}_i$) are sufficient to describe each homography $\mathbf{P}_i$. Based on this decomposition, we can estimate a homography in a more efficient and reliable way using the Lucas-Kanade algorithm as described in [51]. Moreover, we can further reduce the number of parameters in a homography by making an assumption (e.g., no z-axis translation or no out-of-plane rotation) about the camera motion. For more details on the derivation of the Lucas-Kanade algorithm using camera intrinsic parameters, we refer readers to [51].

5.4 Deblurring with Multiple Blurred Images

In this section, we describe a blind motion deblurring algorithm which uses the PSF estimation method described in Section 5.3. For blind motion deblurring, we use multiple blurred images

because using multiple images is known to be more robust and produce results of higher quality than single image deblurring. Also as pointed out by [37], single image deblurring needs more than simple maximum a posteriori (MAP) estimation based alternating optimization because sparsity priors prefer blurry images than sharp ones and such alternating optimization can result in the trivial solution of a delta function PSF. For example, Cho and Lee [14] used a prediction scheme and Fergus et al. [18] used a variational Bayesian approach to overcome this problem. We will more discuss about single image deblurring based on our PSF estimation in Section 5.6.

When we use multiple blurred images as input, the images contain information of the same scene even though their amount and distribution of blurs may differ. The major advantage of this setting is that we can benefit from the mutual information of the scene in the input images. Such information helps to avoid the trivial solution problem in single image deblurring. It effectively suppresses ringing artifacts and image noise and artifacts caused by saturated regions during latent image restoration, and we can achieve higher accuracy of PSF estimation with a better estimation of the latent image.

Mathematically, we want to solve the following optimization problem:

$$\underset{l,\mathbf{P},\mathbf{w}}{\operatorname{argmin}} \left\{ \sum_{j} \sum_{\mathbf{x}} |b_j(\mathbf{x}) - \sum_{i} w_{(j,i)} l(P_{(j,i)}(\mathbf{x}))|^2 + \phi_{\text{latent}}(l) + \phi_{\text{homo}}(\mathbf{P}) + \phi_{\text{weight}}(\mathbf{w}) \right\}, \quad (5.12)$$

where j is the index for input images, and ϕ_{latent} is the prior for the regularization of l.

A common approach to this type of optimization is to divide the problem into two closely related subproblems, and to solve each subproblem alternatingly and iteratively. In this chapter, we follow this direction and divide the problem into PSF ($\mathbf{P}$, $\mathbf{w}$) estimation and latent image (l) restoration.

PSF estimation with multiple blurred images We use the method described in Section 5.3 to obtain the motion PSF individually for each input image. Suppose we have two motion blurred input images, b_1 and b_2. In the first iteration, we use the second image b_2 as the latent image l^1 to estimate the initial PSF $(P^1_{(1,i)}, w^1_{(1,i)})$ of the first image b_1. The single image deblurring result of b_1

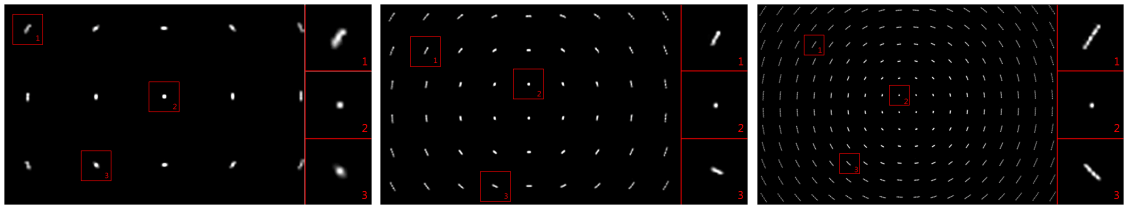


Figure 5.3: Estimated motion PSFs at different scales. For visualization purpose, we construct the PSFs by applying the homographies to the impulse values at regularly sampled pixel locations. Note that the density and the shape of a PSF become refined from a coarse to the finer scales.

is then used to estimate the initial PSF $(P^1_{(2,i)}, w^1_{(2,i)})$ of the second image b_2. After we obtain the PSFs of both images, b_1 and b_2, we estimate the latent image l^2 with the latent image restoration method described later. Then, the algorithm continues using l^2 as the latent image for estimating the PSF of each image in the next iteration.

Latent image restoration with multiple blurred images To obtain the latent image l with fixed $(\mathbf{P}^t, \mathbf{w}^t)$, we need to solve

$$\underset{l}{\operatorname{argmin}} \left\{ \sum_j \sum_{\mathbf{x}} \left| b_j(\mathbf{x}) - \sum_i w^t_{(j,i)} l(P^t_{(j,i)}(\mathbf{x})) \right|^2 + \phi_{\text{latent}}(l) \right\}. \tag{5.13}$$

This is a non-blind motion deblurring problem, which can be solved effectively using the approach in [54]. In our implementation, we set $\phi_{\text{latent}}(l)$ as

$$\phi_{\text{latent}}(l) \quad = \quad \lambda_{\text{latent}} \sum_x \sum_y \left\{ \left| \frac{\partial l(x,y)}{\partial x} \right|^\alpha + \left| \frac{\partial l(x,y)}{\partial y} \right|^\alpha \right\}, \tag{5.14}$$

where λ_{latent} is the relative weight and we use $\lambda_{\text{latent}} = 0.005$ in our implementation. As done in Chapter 3, we use $\alpha = 0.8$, which gives a sparsity prior [35] to suppress ringing artifacts and image noise amplification in the latent image restoration process. Eq. (5.13) can be effectively solved using an iterative reweighted least square method [35].

Multi-scale implementation In order to avoid poor local minima, and to increase the stability of the algorithm, we adopt a multi-scale approach using a Gaussian pyramid. The upsampling factor

s from one level to the next is two. We estimate the motion PSF and the latent image from coarse to fine levels, where the PSF and latent image estimated at a coarse level are upsampled and used as initial values for optimization in the next finer level. At the coarsest level, we use $\mathbf{P}_i = \mathbf{I}$ and $w_i = 1/p$ for initial homographies and weights ($\mathbf{P}^0$,$\mathbf{w}^0$). For the initial latent image l^1 at the coarsest level, we use the downsampled version of another image among multiple input images, as described in Section 5.4.

At the coarsest level, we use only a few number (e.g., four) of homographies to approximate the motion PSF. When we move from a coarse to the finer level, we quadruple the number of homographies used for representing the motion PSF. Since the width and height of an image are doubled, we need to quadruple the sampling rate of homographies in order to catch up the increased sampling rate of image details. Increasing the sampling rate of homographies also allows us to achieve finer refinement of the estimated motion PSF (Figure 5.3).

Suppose $\{\cdots, (\mathbf{P}_i, w_i), \cdots\}$ is the set of estimated motion PSFs at a coarse level. Each $\mathbf{P}_i$ is a 3×3 homography matrix corresponding to P_i. The initial value of $(\mathbf{P}, \mathbf{w})$ at the next level will be $\{\cdots, (\mathbf{P}_i^1, \frac{w_i}{4}), (\mathbf{P}_i^2, \frac{w_i}{4}), (\mathbf{P}_i^3, \frac{w_i}{4}), (\mathbf{P}_i^4, \frac{w_i}{4}), \cdots\}$, where

$$\mathbf{P}_i^j = \begin{bmatrix} s & 0 & t_x^j \\ 0 & s & t_y^j \\ 0 & 0 & 1 \end{bmatrix} \mathbf{P}_i \begin{bmatrix} \frac{1}{s} & 0 & 0 \\ 0 & \frac{1}{s} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.15}$$

$(t_x^j, t_y^j) \in \{(0,0), (\frac{1}{2}s, 0), (0, \frac{1}{2}s), (\frac{1}{2}s, \frac{1}{2}s)\}$ is the translation for the increased sampling rate. Before upsampling, we drop the homographies with weights less than $0.01 \times \max\{w_i\}$ to maintain a small number of homographies.

5.5 Results

We first use a synthetic example to compare the deblurring result of our method with previous methods [56, 14, 48, 18] (Figure 5.4). For comparison with [56], we *did not* implement their variational Bayesian approach. Instead, we constrain the parameters of our homography model to contain only

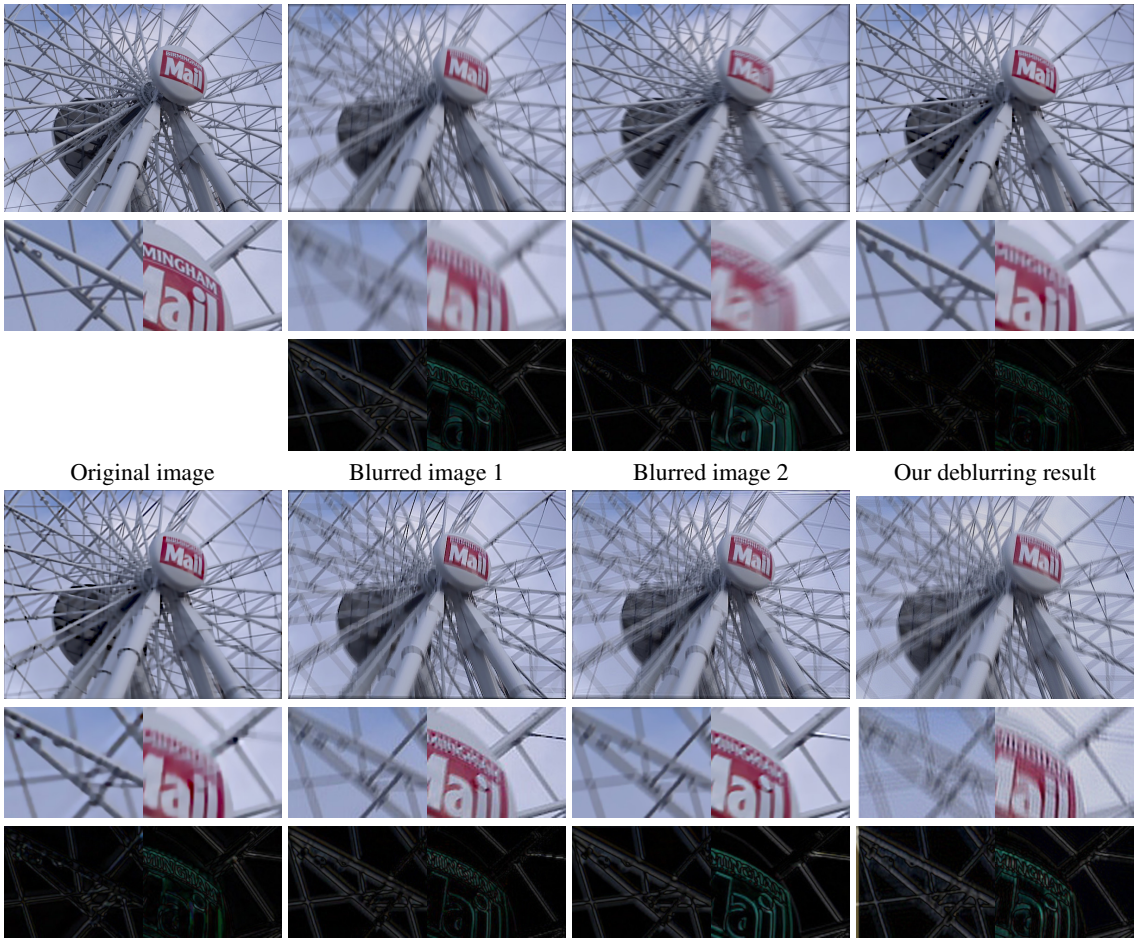


Figure 5.4: Comparison of deblurring results using a synthetic example. The first PSF on the left contains y-translation with x-axis and z-axis rotations, and the second PSF on the right contains x-translation with y-axis and z-axis rotations.

rotational motions. Hence, the results of [56] shown in this chapter are *our* results but with their model, which is more restrictive than our model. The methods of Cho and Lee [14], Shan et al. [48], and Fergus et al. [18] are for single image blind motion deblurring. While our approach uses multiple input images, we run their methods on each input image, and pick the best result. We know that this comparison is not entirely fair, but still expect it can give some useful information.

(a) Input image 1 (b) Input image 2 (c) Our method (d) Our result using blur model of Whyte et al.

Figure 5.5: Comparison of our method with [56]. The magnified views clearly show that our general non-uniform blur model using homographies helps generate better deblurring results than the restrictive model of [56] using only rotations.

Method	PSNR
Our method	32.5257
Whyte et al. [56]	31.3315
Cho and Lee [14]	30.5990
Shan et al. [48]	30.6280
Fergus et al. [18]	30.2384

Table 5.1: Peak signal-to-noise ratio (PSNR) of the deblurring results in Figure 5.4

In Figure 5.4, the motion blurs of the two input blurred images contain rotational (both in-plane and out-of-plane) and translational motions. We show the deblurring results with magnified views and difference images from the ground truth image. Table 5.1 compares the restoration errors quantitatively using PSNR. Our approach produces visually and quantitatively the least amount of errors compared to previous methods. The ground truth PSFs and the estimated PSFs are also shown in Figure 5.4 to demonstrate the accuracy of our PSF estimation. We show more real world examples in Figure 5.6.

5.6 Discussion

In this chapter, we proposed a deblurring method which can effectively handle non-uniform motion blur effects in real photographs. Our major contribution is the blind motion deblurring algorithm

Figure 5.6: Deblurring examples with real images. The first and second columns show the input images, on which the estimated PSFs are overlaid. The third column shows the deblurring results.

which transforms the PSF estimation problem into an image registration problem. While our current implementation uses multiple motion blurred images as input, we believe that this framework can be extended to single image deblurring if it is equipped with a more robust registration method. The following discuss some issues related to our work.

Degrees of Freedom in Camera Motion The motion blur model using homographies allows eight degrees of freedom. In practice, the camera motion in 3D world has only six degrees of freedom for rotations and translations. The additional two degrees of freedom correspond to the unknowns in camera intrinsic parameters. Whyte et al. [56] commented that using only three degrees of freedom for rotation is sufficient to handle non-uniform spatially varying motion deblurring.

While this comment is correct for some scenario such as distant scenes, it becomes invalid when the magnitude of a camera motion is large. Joshi et al. [31] measured a camera motion using motion inertia sensors and validated that the camera motion indeed has six degrees of freedom. Their deblurring algorithm requires the information from all six degrees of freedom among which translations cannot be ignored. In our experiments, we compared our results with eight, six, and three degrees of freedom. The results with eight and six degrees of freedom are similar, but the results with only three degrees of freedom could degrade significantly, as shown in our examples when the effects of translations are large.

Using homographies alone without weights The original formulation in [54] uses homographies without weights assigned to them. In their work, each homography carries the same weight and the variation of PSF intensities is handled by the different occurrence counts of homographies. We note that these two representations are essentially the same. However, by including weights in the motion blur model, we can significantly reduce the number of homographies and save the computation. Nevertheless, we still need a sufficient number of homographies to model a non-uniform motion PSF. The minimum number of required homographies depends on the variations of motion PSFs. Typically, we found that about 30 homographies are sufficient.

Registration between blurred images Previous approaches using multiple images for deblurring (e.g., [60, 12]) require a preprocessing step to register input images. However, registration among blurry images is difficult due to ambiguities in blurry edges. In contrast, our method embeds the registration step into the PSF estimation process, and is free from this preprocessing step.

Other possible solutions Although we proposed a novel method to estimate non-uniform motion PSFs using image registration, we note that there might be other solutions, e.g., using hardware approaches [53, 38, 31] or the variational Bayesian approach [56]. The method in [56], however,

Input image Result

Figure 5.7: Single image deblurring result.

requires quantization of the solution space and is not scalable to a large number of variables. Therefore, they reduced the number of variables by making an assumption that the camera motion contains only rotations (both in-plane and out-of-plane). In contrast, our approach does not need to make a restrictive assumption about the camera motion. Furthermore, we can handle the camera intrinsics as unknowns by using general homographies. Hence, our approach can be considered more general than [56].

Single image versus multiple images One major drawback of our method is the requirement of multiple input images. While this requirement may restrict the applications of our approach, using multiple images offers several advantages over single image. The main advantage is the robustness. In single image deblurring, as discussed in [37], alternating optimization for a MAP solution favors the delta function solution. In multiple image deblurring, we can effectively avoid such a trivial solution because the solution does not satisfy the constraints from multiple images. Using multiple images for deblurring is also effective in suppressing artifacts, such as amplified noise and ringing, and better handles saturated regions. We have also found that our algorithm is robust to parameter

values, while many previous deblurring methods using a single image usually require careful and extensive tuning of parameters.

Figure 5.7 shows a single image deblurring result of our algorithm. We used the sharp edge prediction method in [14] to avoid the delta function solution. However, the result in Figure 5.7 still contains artifacts similar to the results provided in [56]. We believe that these artifacts mainly come from the instabilities of using a single image for PSF estimation and deblurring. Because non-uniform motion deblurring should resolve a large number of unknowns that may not be robustly determined by a single image input, it seems reasonable to use multiple input images for obtaining better deblurring results.

Practical issues and limitations Our approach uses multiple motion blurred images as input. If the input images do not contain the same scene information, e.g., different poses of a person and change of white balance due a time gap between two captures, our algorithm would fail. However, nowadays, digital cameras usually allow a user to capture several images in a row while holding down the shutter button. Such camera setup enables avoiding the possible problems caused by a long time gap between consecutive captures.

Chapter 6

Video Motion Deblurring

Camera motion is one of the most important factors that differentiate professional videos from ones captured by amature users. Professional videos are often shot using special equipments such as dollies or steadicams to achieve smooth camera motion, while armature ones are often shot by hand-held cameras with significant camera shake. The impact of a shaky camera to the captured video is twofold: first, it introduces temporal jitter to the video content which is unpleasant to watch; and second, it blurs video frames significantly at times when the camera shake is intense.

Various video stabilization systems [39, 21] have been proposed recently to effectively smooth the camera motion in a shaky video. Although these approaches can successfully stabilize the video content and achieve highly smooth camera motion, it leaves the blurriness caused by original camera motion untouched. As a result, the blurry video frames become the most noticeable artifact in a stabilized video. The blurry video frames also pose difficulties for video stabilization approaches to achieve high quality stabilization results. Most stabilization systems rely on feature point tracking to plan the camera motion, and feature tracking over blurry frames is often not reliable. Restoring sharp frames from blurry ones caused by camera motion, which we dub *video motion deblurring*, is thus critical to generating a high quality stabilization result from a shaky input sequence.

Given the recent advances in image deblurring research, one straightforward idea for video motion deblurring is to first identify blurry frames, and then apply single or multiple image deblurring techniques to them. Unfortunately, we found in our study that existing image deblurring approaches

Figure 6.1: A comparison of deblurring a motion-blurred frame using various approaches. (a) A blurry frame of the "cars" video sequence. (b) Input. (c) Single image deblurring. (d) Multi-frame deblurring. (e) Our method. (f) Similar image region from a nearby sharp frame.

are incapable of generating satisfactory results for video deblurring. An example is shown in Figure 6.1, where we apply a recent single image deblurring method [14] and a multi-frame deblurring approach [38] to restore the blurry frame shown in Figure 6.1a. Both results contain significant, unacceptable artifacts due to the following reasons. First, the blur kernels in a video are both spatially and temporally varying, introduced by both camera motion and object motion. Previous methods cannot reliably estimate accurate blur kernels on every frame, which is the absolute requirement for them to generate good deconvolution results. Second, even with good kernel estimation, deconvolution is still sensitive to various factors such as noise and saturated pixels, and can easily introduce severe ringing artifacts, as suggested by Yuan et al. [61]. Last but not least, video deblurring demands temporal coherence. Directly applying an image deblurring method to individual frames often generates temporally incoherent results.

In this chapter, we present an effective approach for video motion deblurring, which avoids applying direct kernel estimation and deconvolution to video frames. Our method is built upon the key observation that camera shake is caused by high frequency, irregular hand motion. It causes the same image content to appear sharper on some frames when the motion intensity is small, and more blurry on others when the motion intensity is large. With proper alignment and motion compensation, sharp regions can be used to directly restore their corresponding regions in blurry frames.

In our approach, we first estimate a parametric, homography-based motion model for each frame

as an approximation to the real motion, which can be far more complicated due to parallax. We then use the approximate motion model to define the *luckiness* of a pixel to measure its sharpness. To deblur a video frame, we search for luckier pixels in nearby frames and use them to replace less lucky ones in the current frame. The pixel correspondence across frames is obtained by using the estimated homographies, followed by a local search to compensate for the inaccuracy of the motion model. To compare a sharp patch to a blurry one, we use forward convolution to blur the sharp patch with the estimated blur function of the blurry patch. When copying lucky pixels to a blurry frame, we adopt an idea of patch-based texture synthesis [33] to better preserve object structures. Finally, we apply a temporal filter to the deblurred frames to maintain temporal coherence.

Our method is designed to remove only the blur introduced by camera motion on static objects in the video. This is in lieu with the design goal of video stabilization approaches which usually can stabilize the background only. For moving objects, since their blur is typically dominated by object motion which is preserved in the final video, the blur on them rarely stands out as a noticeable artifact. On the contrary, we found that human perception is much more sensitive to the blur on background regions which are supposed to be still. Our method can effectively remove the most annoying background blur and can work reliably well when moving foreground objects present, as we will demonstrate in Section 6.4.

6.1 Related Work

Li et al. [38] propose a system to create sharp panoramas from motion-blurred videos. Their system uses homographies as the motion model between adjacent frames, which leads to spatially-varying blur kernels. The motion and duty cycle parameters are estimated along with latent images in an energy minimization formulation. Our method adopts a similar homography-based motion model as the approximate blur model. However, unlike Li et al.'s system, we do not treat the homography-based model as an accurate one, and explicitly handle the model inaccuracy by incorporating local search of matching patches. Our method is thus able to work well on videos where the motions

among frames cannot be parametrized solely using homographies. Another important difference is that we do not use deconvolution to restore latent images. Instead we directly use sharp pixels from nearby frames to avoid common artifacts that deconvolution may introduce, such as ringing artifact for a moving object shown in Figure 6.1d.

Besides these deconvolution based techniques, there is a video deblurring work that avoids direct kernel estimation and deconvolution, which are restrictive and not robust. Matsushita et al. [41] proposed a practical video deblurring method in their video stabilization system. They detected sharp frames using the statistics of image gradients, and interpolated sharp frames to increase the sharpness of blurry frames. However, their frame-to-frame pixel alignment method uses only the camera motion represented by homographies, and does not consider either the effect of blur kernels or the alignment errors introduced by using homographies only. This alignment inaccuracy in their method inevitably degrades the quality of deblurred frames, as we will show in Section 6.4.

6.2 Motion-blurred Video Frames

In this section, we first analyze why our assumption on the existence of sharp image regions in a motion-blurred video is valid. We then describe an approximate blur model which we will use in our deblurring approach.

6.2.1 Sources of sharp regions

Our method targets on removing motion blurs introduced by hand-held cameras. Hand shake is an irregular motion which changes velocity in a short period of time. The captured video frames sample the hand shake motion at a much higher frame rate. If the video lasts for a reasonable amount of time (typically a few seconds), there are bound to be *lucky* frames captured when the camera motion is almost steady (e.g., turning points of camera motion), resulting in sharp frames, as well as blurry ones which were captured with large camera motion. Figure 6.2 shows an example. We select one representative feature point on the background, shown as the yellow dot in Figure 6.2a, and

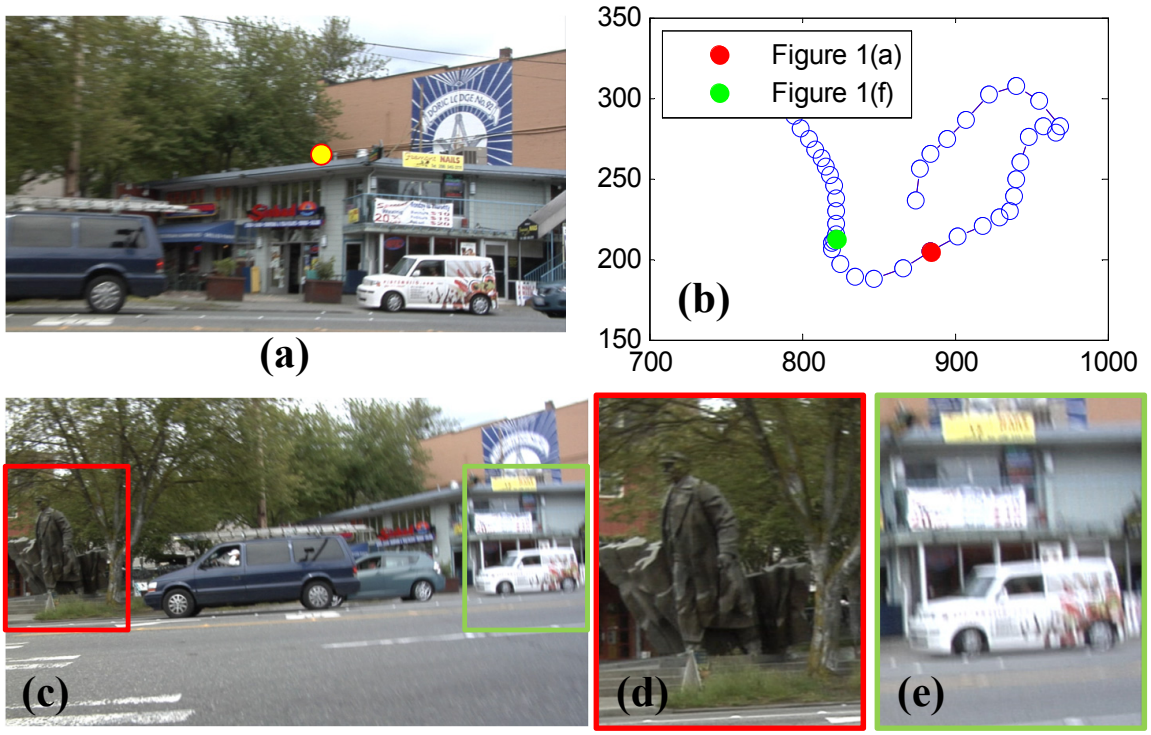


Figure 6.2: Sources of sharp regions in a motion-blurred video. (a) A feature point to be tracked for the "cars" sequence. (b) The trajectory of the feature point in the entire video. (c) One frame of the "cars" sequence with camera rotation (d) A local region near the rotation center. (e) A local region far from the rotation center.

plot its trajectory in the image space in Figure 6.2b. Since the background is static, this trajectory represents the camera motion very well. It clearly shows that the velocity of the camera motion changes dramatically in the course of the video, resulting in both sharp (Figure 6.1f) and blurry (Figure 6.1a) frames at different times.

In addition to sharp and blurry frames, local image regions can be both sharp and blurry in the same frame, due to spatial-varying blur kernels. Figure 6.2c shows a video frame captured with camera rotation, where the rotation center is around the middle of the left image boundary. An image region near the rotation center is sharp (Figure 6.2d) while a region far away from the rotation center is more blurry (Figure 6.2e). This example suggests that we should consider local sharpness in video frames for deblurring.

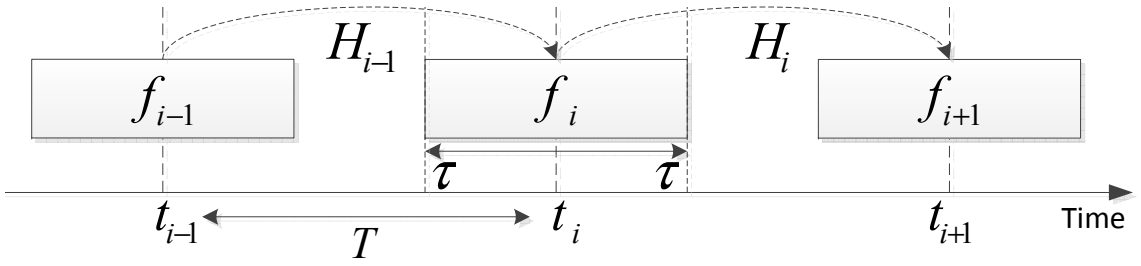


Figure 6.3: Illustration of the approximate blur model.

We would like to point out that this assumption fails if the camera motion is dominated by a carefully directed large motion throughout the entire video, such as constant panning. In practice we found that most amateur videos usually contain various camera motions such as walking, zooming, and panning, thus our assumption holds well on a wide range of videos.

6.2.2 Approximate blur model

Our method approximates the blurs of video frames using homographes. As shown in Figure 6.3, suppose the duty cycle of the camera is 2τ, and the exposure time interval for frame f_i is $[t_i - \tau, t_i + \tau]$. Let the time interval $t_i - t_{i-1} = T$, and the latent image of f_i be l_i. We first assume that the motion between adjacent frames can be approximated by a homography, i.e., $l_{i+1}(\mathbf{x}) \approx l_i(P_i(\mathbf{x}))$ where P_i is a homography that maps a 2D pixel position to another 2D pixel position. We also assume that the velocity of the motion is constant between adjacent frames, then we have:

$$f_i(\mathbf{x}) \quad = \quad \frac{1}{2\tau}\int_{t=0}^{\tau}\left[l_i(P_{i-1}^t(\mathbf{x})) + l_i(P_i^t(\mathbf{x}))\right]dt. \tag{6.1}$$

P_{i-1}^t and P_i^t are parametrized by homography matrices $\mathbf{P}_{i-1}^t$ and $\mathbf{P}_i^t$, respectively, which are defined as:

$$\mathbf{P}_{i-1}^t = \frac{T-t}{T}\mathbf{I} + \frac{t}{T}\mathbf{P}_{i-1}^{-1}, \qquad \mathbf{P}_i^t = \frac{T-t}{T}\mathbf{I} + \frac{t}{T}\mathbf{P}_i. \tag{6.2}$$

As an image can be expressed in a discrete vectorized form, we can rewrite Eq. (6.1) as:

$$\mathbf{f}_i \quad = \quad \frac{1}{2\tau}\int_{t=0}^{\tau}\left[\widehat{\mathbf{P}}_{i-1}^t\mathbf{l}_i + \widehat{\mathbf{P}}_i^t\mathbf{l}_i\right]dt, \tag{6.3}$$

where $\mathbf{f}_i$ and $\mathbf{l}_i$ are vectors representing f_i and l_i, respectively, and $\widehat{\mathbf{P}}$ is a linear transformation matrix derived from a homography P. For a video frame whose number of pixels is n, $\mathbf{f}$ and $\mathbf{l}$ are n-dimensional, and the size of $\widehat{\mathbf{P}}$ is $n \times n$. Discretizing Eq. (6.3) gives us:

$$\mathbf{f}_i \doteq b_i(\mathbf{l}_i) = \frac{1}{1+2\tau}\left[\sum_{d=1}^{\tau}\left(\widehat{\mathbf{P}}_{i-1}^{d}\mathbf{l}_i + \widehat{\mathbf{P}}_{i}^{d}\mathbf{l}_i\right) + \mathbf{l}_i\right], \tag{6.4}$$

where T becomes the sampling rate in $[t_i, t_{i+1}]$ which we fix at 20 in our implementation, and τ becomes the number of samples that fall into the duty cycle. We call b_i the *blur function* for the i-th frame.

This blur model is similar to the one developed in the multi-image deblurring method for panorama generation [38], which also uses homography as the underlying motion model. However, our work only treats this model as an approximation in order to deal with more complicated videos with parallax, while it was treated as an accurate model in the previous method. To explicitly handle the modeling errors, our method employs an additional local search step for aligning image regions of different frames, as we will describe in detail in Section 6.3.1.

6.2.3 Luckiness measurement

With the motion model in Section 6.2.2, we introduce a measurement of *luckiness* for a pixel in a video frame, which describes the absolute displacement of the pixel among adjacent frames. For a pixel $\mathbf{x}$ in frame f_i, its luckiness is defined as:

$$\alpha_i(\mathbf{x}) \quad = \quad \exp\left(\frac{-\|(P_{i-1}^{-1}(\mathbf{x}) - \mathbf{x}\|^2 + \|P_i(\mathbf{x}) - \mathbf{x}\|^2}{\sigma^2}\right), \tag{6.5}$$

where σ is a constant which we set at 10 pixels in our implementation. Eq. (6.5) computes the displacement of pixel $\mathbf{x}$ when the camera moves from frame f_{i-1} to f_{i+1} through f_i. When the frame-to-frame motion of $\mathbf{x}$ is small, $P_{i-1}^{-1}(\mathbf{x})$ and $P_i(\mathbf{x})$ are close to $\mathbf{x}$, thus $\alpha_i(\mathbf{x})$ is close to 1, indicating that the image patch centered at $\mathbf{x}$ is likely to be sharp. Otherwise $\alpha_i(\mathbf{x})$ is small, indicating that the patch is likely to contain large motion blur. The luckiness α_i of a whole frame f_i is simply defined as the average value of all $\alpha_i(\mathbf{x})$ for pixels in f_i.

6.2.4 Blur function estimation

There are two parameters that need to be estimated for the blur function b_i in Eq. (6.4) before we can use it for deblurring: the homography P_i and the duty cycle τ. To estimate the homography P_i, we first apply feature tracking using a standard KLT approach [50], and use the tracked feature points to compute the initial homographies. We then refine the homographies using Lucas-Kanade registration [1] between frames.

In our deblurring approach, we need to estimate homographies not only between adjacent frames, but also between any two frames in a local temporal window $W_i = [i - m, i + m]$, where m is set to 5 in our implementation. We denote the homography from frame i to frame j as P_{ij}, where $j \in W_i$. Obviously $P_{i,i+1} = P_i$ in Figure 6.3. Homographies among non-adjacent frames are initialized and refined in the same way as P_i.

Note that unlike the previous approach [38], we do not further update the homographes when we deblur the video frames. This is because we only treat the homography as an approximate motion model. Small errors in their estimation are handled by local search of matching patches in the deblurring process.

To compute τ, we first select a set of frame pairs, where each pair have a large difference in the luckiness measurement, so that the accuracy of blur functions can be effectively tested. Let (f_i^k, f_j^k), $k = 1, ..., N$, be N pairs of frames with $j \in W_i$, where the frame luckiness difference, $\alpha_i - \alpha_j$, is larger than a threshold (0.6α in our system, where $\alpha = \max_i\{\alpha_i\}$). We then seek the optimal τ which minimizes:

$$E(\tau) \quad = \quad \sum_{k=1}^{N} \left\| b_j^k(\widehat{\mathbf{P}}_{ij}^k \mathbf{f}_i^k) - \mathbf{f}_j^k \right\|^2 . \tag{6.6}$$

Intuitively, we first align f_i^k with f_j^k using the homography $\widehat{\mathbf{P}}_{ij}$, then blur the aligned sharp frame $\widehat{\mathbf{P}}_{ij}^k \mathbf{f}_i^k$ using the blur function b_j^k of f_j^k, and compute the sum of squared differences between the synthetically blurred frame and the observed f_j^k. The value of Eq. (6.6) depends on the blur functions b_j^k for all k: if b_j^k is accurate, it should be small, and vise versa. Since b_j^k is determined by Eq. (6.4)

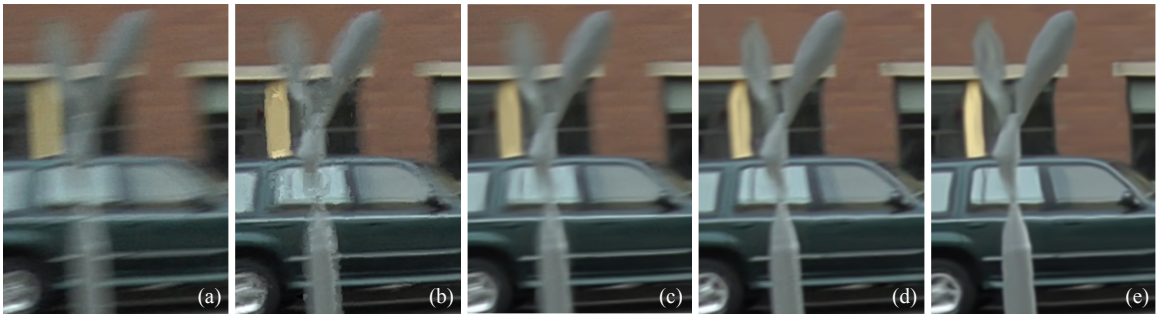


Figure 6.4: Illustration of the impact of each step in our algorithm on the final result. (a) A region in the input blurry frame. (b) Copying center pixels of matched patches from nearby frames. (c) Adding weighted patch averaging. (d) Adding the lucky patch prior in Eq. (6.8). (e) Adding the frame processing order.

for a given value of τ, Eq. (6.6) becomes an energy function of τ. We minimize Eq. (6.6) using a brute-force search as τ can only take a limited set of integer values from 1 to $\lfloor T/2 \rfloor$.

6.3 Video Frame Deblurring

6.3.1 Single frame deblurring

Once we have obtained the blur function b_i for frame f_i, we could compute the latent frame l_i using a deconvolution method that can handle non-uniform blurs represented by homographies [38]. However, this straightforward approach generates less satisfactory results in practice. We instead use lucky pixels in nearby frames to restore l_i, avoiding artifacts from deconvolution.

Let $p_{i,\mathbf{x}}$ be an $n \times n$ image patch in frame f_i centered at a pixel $\mathbf{x}$. In our implementation, $n = 11$. For $p_{i,\mathbf{x}}$, we find another $n \times n$ patch $p_{j,\mathbf{y}}$ in frame f_j, where $j \in W_i$ and $\alpha_{j,\mathbf{y}} > \alpha_{i,\mathbf{x}}$, such that:

$$\mathbf{p}_{j,\mathbf{y}} = \underset{j,\mathbf{y}}{\operatorname{argmin}} \left\| b_i \left(\widehat{\mathbf{P}}_{ji} \mathbf{p}_{j,\mathbf{y}} \right) - \mathbf{p}_{i,\mathbf{x}} \right\|^2 \tag{6.7}$$

where $\mathbf{p}_{i,\mathbf{x}}$ and $\mathbf{p}_{j,\mathbf{y}}$ are vector representations of $p_{i,\mathbf{x}}$ and $p_{j,\mathbf{y}}$, respectively. In Eq. (6.7), the patch difference is computed as the sum of squared differences of corresponding pixels. We consider $p_{j,\mathbf{y}}$ as a good estimate of the latent patch of $p_{i,\mathbf{x}}$, since it is a sharper patch that has similar appearance

to $p_{i,\mathbf{x}}$ when blurred using b_i. Before searching patches in f_j, we warp f_j using the homography P_{ji} to compensate for the camera motion between the two frames. Given that the homography-based motion model is a good estimate, we limit the search range for $\mathbf{y}$ in an $m \times m$ window which is centered at $P_{ji}^{-1}(\mathbf{x}) = P_{ij}(\mathbf{x})$. In our implementation, $m = 21$. Ideally, if P_{ji} is the accurate motion model between f_i and f_j, we can simply set the search range m to be 1, i.e., comparing $p_{i,\mathbf{x}}$ only with $p_{j,\mathbf{H}_{ij}\mathbf{x}}$. However, in practice due to parallax and object motions, the real motion among frames is generally more complicated than a single homography. Using a larger search range m allows us to compensate for the error in our motion model using homographies.

To deblur a whole frame, we search the best matching patch $p_{j,\mathbf{y}}$ for each patch $p_{i,\mathbf{x}}$ in f_i. Then, a simple way to restore l_i is to copy the center pixel $\mathbf{y}$ of $p_{j,\mathbf{y}}$ onto each pixel $\mathbf{x}$ in f_i. However, this simple copying may incur misalignments of object structures in l_i, as each pixel is copied individually without enforcing spatial coherence (Figure 6.4b). To solve this problem, we adopt a patch-based texture synthesis approach [33]. Except for boundary pixels, each pixel $\mathbf{x}$ in f_i belongs to n^2 patches $p_{i,\mathbf{x}'}$ of f_i, where $\mathbf{x}'$ is in the $n \times n$ neighborhood of $\mathbf{x}$. When these n^2 patches are overlaid by their best matching patches, $\mathbf{x}$ is given n^2 candidate pixel values, one from each patch. The final value of $\mathbf{x}$ is computed as a weighted average of the n^2 values, where the luckiness of each candidate pixel is used as the weight to reduce the loss of pixel sharpness in the averaging. Figure 6.4c shows that with this approach, we can better preserve the object structures than simple copying.

The temporal windows W_i used for patch search contains the frame f_i itself. If a patch $p_{i,\mathbf{x}}$ belongs to a moving object in f_i, the object motion would incur a different blur from the blur function b_i, and the true blur function for $p_{i,\mathbf{x}}$ is usually dominated by the object motion. In this case, the best option would be to keep the pixel value of $\mathbf{x}$ intact to preserve the object motion. Our method achieves this in a natural way, by allowing $\mathbf{p}_{i,\mathbf{x}}$ to compare with $b_i(\mathbf{p}_{i,\mathbf{x}})$. Due to the blur function difference, any patch in another frame f_j cannot minimize the fitting error in Eq. (6.7) well. On the other hand, the fitting error between $\mathbf{p}_{i,\mathbf{x}}$ and $b_i(\mathbf{p}_{i,\mathbf{x}})$ is relatively small, since $p_{i,\mathbf{x}}$ is already

severely blurred by the object motion, and $b_i(\mathbf{p}_{i,\mathbf{x}})$ is a slightly smoothed version of $\mathbf{p}_{i,\mathbf{x}}$, which has similar appearance to $p_{i,\mathbf{x}}$. Thus $p_{i,\mathbf{x}}$ becomes the best candidate for itself if $\mathbf{x}$ lies inside a moving foreground object.

6.3.2 Improved deblurring using luckiness

To further improve the sharpness of deblurred frames, we add a lucky patch prior to Eq. (6.7) so that sharper patches are preferred over blurrier ones:

$$\mathbf{p}_{j,\mathbf{y}} \quad = \quad \underset{j,\mathbf{y}}{\operatorname{argmin}} \left\{ \left\| b_i \left(\widehat{\mathbf{P}}_{ji} \mathbf{p}_{j,\mathbf{y}} \right) - \mathbf{p}_{i,\mathbf{x}} \right\|^2 + \lambda \|1 - \alpha_{j,\mathbf{y}}\|^2 \right\}. \tag{6.8}$$

We use a small value for the weight λ, which is 0.01 in our implementation. This allows the patch match term to dominate Eq. (6.8) and the lucky patch prior to have effect only when the values of the patch match term are similar. Figure 6.4d shows that the lucky patch prior helps improve the sharpness of a restored frame.

We also use the luckiness to determine the processing order of frames when applying the frame deblurring method to the whole video sequence. We first sort all frames based on their luckiness values, and start processing from the luckiest frame first. For luckier frames, most of the pixels remain unchanged after deblurring. As the luckiness values of frames become lower, more pixels will be updated by sharper pixels from already processed frames. We treat the luckiness as another color channel and update it in the same way as we update RGB values of pixels. As a result, sharp pixels are propagated from luckier frames to less lucky ones. Figure 6.4e shows the final deblurring result using the proposed frame processing order. It is clearly sharper than Figure 6.4d which was generated without using the processing results of other frames.

6.3.3 Temporal coherence

In the deblurring method proposed above, each frame is processed independently from other frames, and temporal coherence is not guaranteed among neighboring frames. To improve the temporal coherence of the final result, a straightforward approach would be to use 3D volumetric patches in the

patch search process using Eq. (6.8). To do this we can extend a 2D patch into neighbor frames to construct a 3D patch, using homographies to align corresponding pixels. However, searching with 3D patches significantly increases the required computation, which reduces the practical applicability of the approach. More importantly, using only homographies cannot give us accurate registration between frames, thus it is difficult to find a 3D sharp patch that can fit well with a given blurred 3D patch, since both 3D patches contain alignment errors internally.

To improve temporal coherence, we apply temporal smoothing to the corresponding pixels among frames. Let $\mathbf{x}$ be a pixel in a frame f_i and $p_{i,\mathbf{x}}$ be a patch in f_i centered at $\mathbf{x}$. For each frame f_j, $j \in W_i$, we find a pixel $\mathbf{y}$ in f_j whose patch $p_{j,\mathbf{y}}$ is most similar to $p_{i,\mathbf{x}}$ after homography P_{ji} is applied:

$$\mathbf{y} \quad = \quad \underset{\mathbf{y}}{\operatorname{argmin}} \|\widehat{\mathbf{P}}_{ji}\mathbf{p}_{j,\mathbf{y}} - \mathbf{p}_{i,\mathbf{x}}\|^2. \tag{6.9}$$

We use the same patch size and search range as Eq. (6.8). As the result of the temporal search, we have $2M + 1$ values, one from each frame f_j, $j \in W_i$. We then blend the $2M + 1$ values by weighted averaging to obtain temporally smooth value of $\mathbf{x}$. The weight $w_{j,\mathbf{y}}$ for a value from f_j is determined using the luckiness and the patch similarity:

$$w_{j,\mathbf{y}} \quad = \quad \alpha_{j,\mathbf{y}} \exp\left(-\|\widehat{\mathbf{P}}_{ji}\mathbf{p}_{j,\mathbf{y}} - \mathbf{p}_{i,\mathbf{x}}\|^2/\sigma^2\right). \tag{6.10}$$

Figure 6.5 shows the result of temporal smoothing, where the deblurred results of two adjacent frames have been made coherent.

While temporal smoothing improves temporal coherence, it may smooth out image features and reduce the effect of frame deblurring. To restore importance features, we apply shock filter [43] to the resulting frames from temporal smoothing. Since the frames are already temporally smooth, shock filter generates a temporally coherent result even though independently applied to each frame. Figure 6.5 shows that sharp features are effectively recovered by shock filter, while keeping temporal coherence.

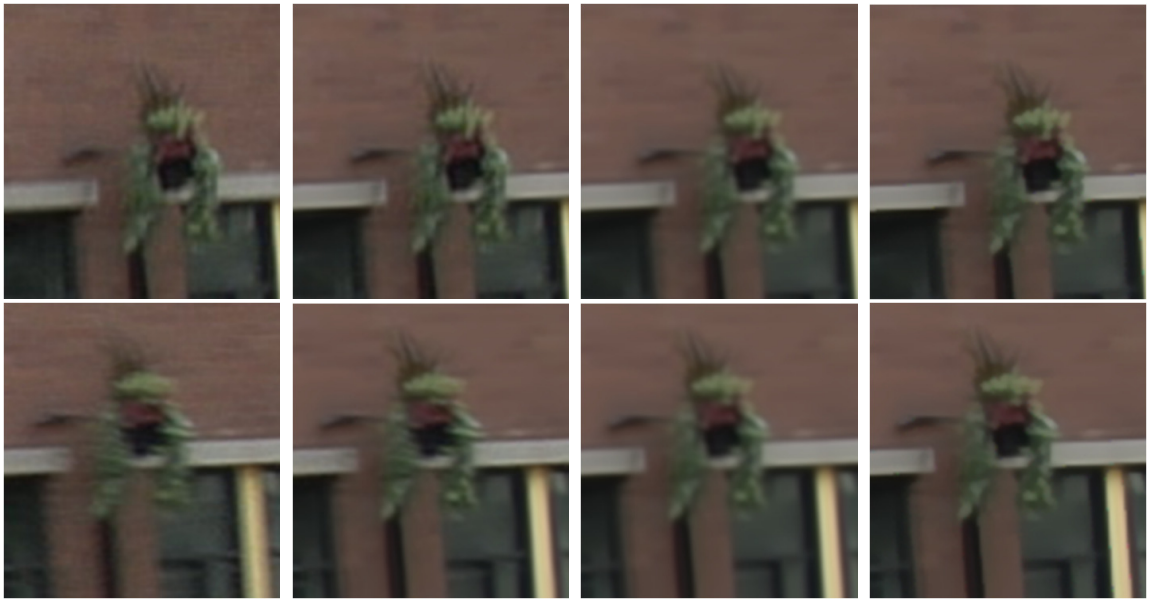


Figure 6.5: Temporal coherence of deblurred frames. The top and bottom rows show regions of two adjacent frames. From left to right: input blurry frames, results of frame deblurring, adding temporal smoothing, and adding shock filtering.

Figure 6.6: Examples of deblurred video frames. Top: Input blurry frames. Bottom: Our deblurring results.

6.4 Results and Comparisons

We have applied our method to a variety of example videos shot by hand-held cameras. For better comparison, we also provide stabilized versions of the input and deblurred videos, which allow us to concentrate solely on the blur artifact without being distracted by shaky camera motions. As

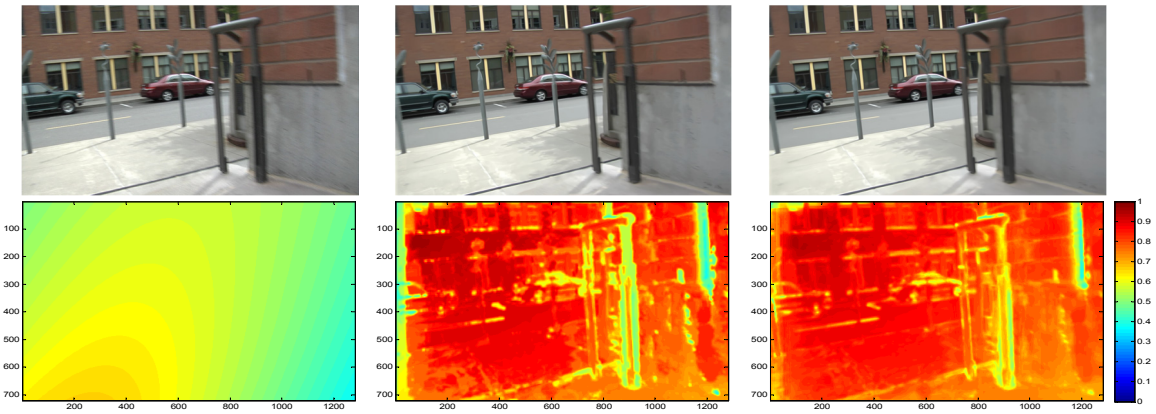


Figure 6.7: Visualization of pixel luckiness values at each step of our algorithm. From left to right: input blurry frame; after frame-wise restoration; after temporal smoothing.

can be seen in the examples, the stabilized input videos contain annoying blur artifact caused by original camera shakes, where video frames unpredictably become blurry for a short time before coming back to normal. Our method can largely remove this artifact and generate more temporal-coherently sharp stabilized videos.

Figure 6.6 shows input blurry frames from different videos and the deblurring results. These examples demonstrate that our method can successfully restore sharp frames from blurry input. Especially, the middle and right examples contain moving objects and significant depth differences among objects. Even for such cases, our method recovers sharp details without noticeable artifacts.

Figure 6.7 visualizes pixel luckiness values of one frame at each step of our algorithm. Initially, the luckiness values of all pixels, computed by Eq. (6.5), are low, indicating the frame is blurry. After frame-wise restoration, the updated luckiness values are high for almost all pixels. After temporal smoothing, the luckiness values have become smooth due to weighted averaging. Figure 6.8 shows frame luckiness values of the entire video. The input video has dynamically changing frame luckiness values due to irregular camera motion. The frame-wise restoration step improves the frame luckiness for all blurry frames. The final temporal smoothing step produces smoothly changing frame luckiness values.

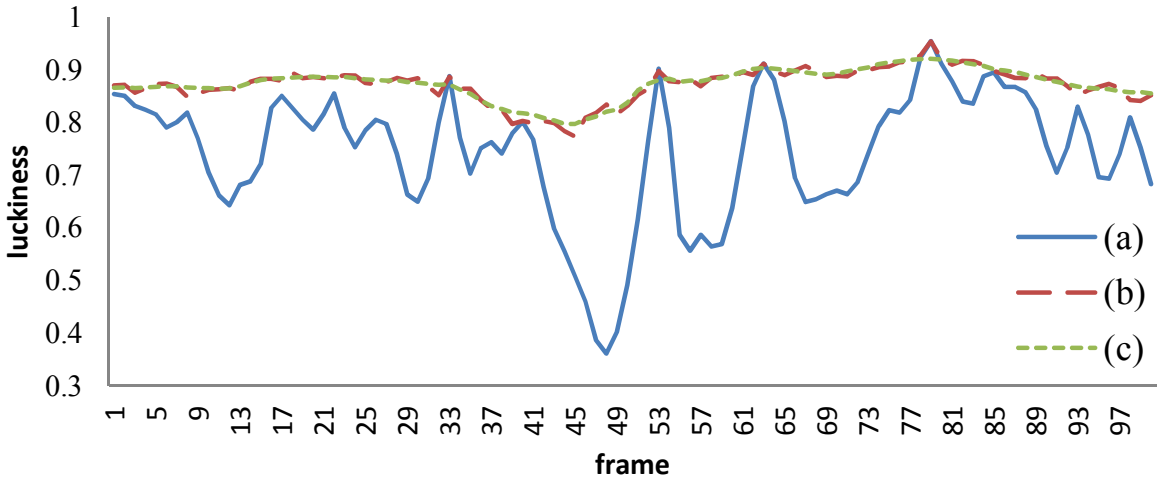


Figure 6.8: Plot of frame luckiness values of the input video in Figure 6.7 at each step of our algorithm. (a) Input blurry video. (b) After frame-wise restoration. (c) After temporal smoothing.

Figure 6.9: Comparison with a previous method. (a) Blurry frame. (b) Matsushita et al. [41] (c) Our result.

Figure 6.9 shows a comparison with the interpolation-based video deblurring method of Matsushita et al. [41]. As their method uses simple frame interpolation without considering the characteristics of underlying motion blurs, the deblurring result still appears to be blurry. In contrast, our method takes into account the estimated motion blur function in Eq. (6.8), which leads to a much sharper result.

Figure 6.10 shows a comparison between our method and multi-frame deblurring. Similar to Li et al. [38], we first estimate homographies between frames, and use them to align every three

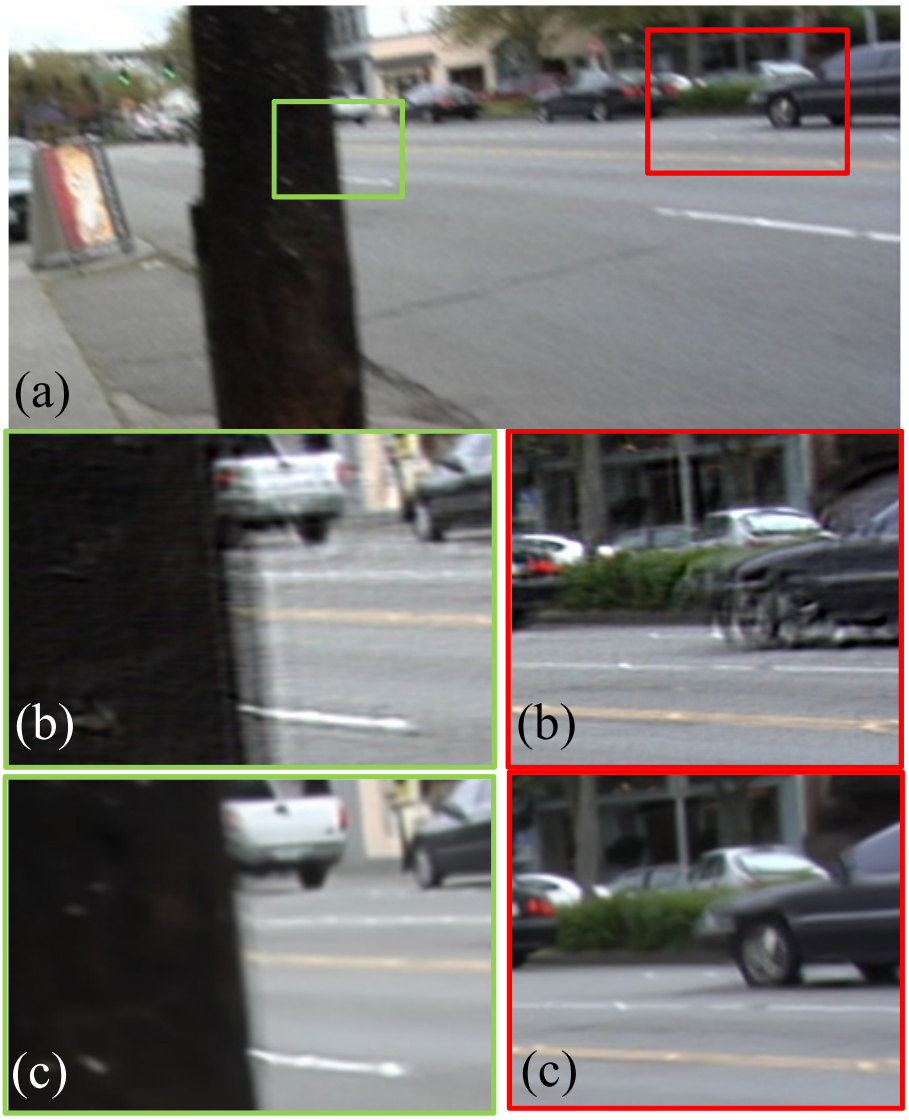


Figure 6.10: Comparison with the multi-frame deblurring approach. (a) Blurry frame. (b) Multi-frame deblurring result. (c) Our result.

consecutive frames. We then approximate spatially varying motion blur kernels using the estimated homographies between frames, and finally apply a multi-image deconvolution method [54]. Our implementation of multi-frame deblurring does not exactly match the method of Li et al. [38]. Nevertheless, the implementation shares the core limitations with their method, i.e., homography-based motion model and no handling of moving objects, and is expected to provide reasonable

Figure 6.11: Examples of failure cases. (a) and (c) Input video frame. (b) and (d) Our result.

comparison.

In Figure 6.10a, the input video presents strong parallax due to the significant depth difference between the front electricity pole and the background street, thus the adjacent frames cannot be aligned well using a single homography. Consequently, the deblurring result shows noticeable artifacts around the front pole. Furthermore, pixels in moving objects cannot be aligned using a single homography either, which leads to severe ringing artifacts around the moving car. In contrast, our method generates better results in both cases, as it naturally handles errors from the homography-based motion model and moving objects.

6.4.1 Failure cases

Our approach has a few limitations. First, our method relies on the image structures in individual patches for finding their corresponding sharp patches. For a severely blurred patch, the fine image structure in it may be destroyed completely, and our method may fail to find the proper sharp patch in nearby frames. Figure 6.11a-b shows such an example. Although the input frame is largely improved by our method, we can see noticeable artifact around a fine image structure, highlighted by a green arrow.

Currently our method has difficulty to handle saturated pixels. Due to clipping of pixel values, the motion blur model in Eq. (6.3) does not hold for saturated pixels, and our method cannot properly match a sharp patch with a blurry one in the presence of saturated pixels. Figure 6.11c shows such an example, where the traffic lights cause pixels to be saturated around them. Our method fails to improve the regions around the traffic lights, as shown in Figure 6.11d, although other parts of the frame is largely improved.

Our method is an example-based approach which relies on using sharp patches from nearby frames to restore blurry ones. If the camera motion is constantly large and there is no sharp patches available, our method cannot restore blurry frames, just leaving the input video untouched. A similar situation is that only part of the scene has sharp patches available, such as a video where the camera pans at the beginning, then stops at the end. Combining our method with a deconvolution-based approach using multi-frames may help in solving this problem.

6.5 Discussion

The camera motion in a hand-held sequence often causes some portion of video frames to be more blurry than others. To completely remove all artifacts introduced by the noisy camera motion, a video motion deblurring approach is needed in conjunction with a video stabilization technique. In this chapter, we present a practical video deblurring method which avoids using deconvolution to restore sharp frames. Since our solution only involves forward convolution and patch-based image

synthesis, it is robust enough to handle a wide range of videos.

The multi-frame deblurring method of Li et al. [38] estimates the duty cycle of a camera to determine the motion blurs of frames. Their method uses a different duty cycle for each pair of adjacent frames, while a globally constant τ is used for the duty cycle in our method. However, the duty cycle is a characteristic of a camera, and can be reasonably assumed to be constant. We found that a single duty cycle is enough for our motion blur model to help searching the matched sharp pixels for blurry ones.

In patch-based texture synthesis [33], the patch search and averaging process is performed iteratively to generate a better texture. In contrast, our frame deblurring method performs the process only once in restoring sharp frames. To improve the restored frames, we experimented with a similar iterative process to texture synthesis, where the result frames in the previous iteration were used for searching sharp patches. However, the improvement due to the iterative process was marginal and not enough to justify the additional computational cost. The main reason was that the results of patch search for blurry pixels did not largely change with iterations, as the best matching pixels in the first iteration would remain as strong candidates in the later iterations.

Chapter 7

Conclusion and Future Work

7.1 Summary

In this thesis, we have proposed software-based approaches for removing motion blur, which overcome the previous limitations, such as the ill-posedness of blind deconvolution, outliers, and non-uniform blurs.

In Chapter 2, we presented a blind deconvolution method for removing uniform motion blur, which is fast enough for practical purposes. The method performs latent image estimation and blur kernel estimation steps alternatingly. To achieve high speed, we accelerated both estimation steps. For fast latent image estimation, we introduced a novel prediction step, and for fast kernel estimation, we used image derivative values instead of pixel values.

In Chapter 3, we defined a new blur model reflecting outliers, and derived an outlier handling method for non-blind image deconvolution. We explicitly modelled outliers and saturated pixels in our blur model, and adopted the EM method for solving the problem.

In Chapter 4, we presented a segmentation-based blind deconvolution method for non-uniform motion blur caused by depth difference and moving objects. The method assumes that input images are blurred by piecewise uniform motion blurs, and alternatingly performs motion estimation, segmentation, and motion blur kernel estimation.

In Chapter 5, we proposed a blind deconvolution method for removing non-uniform motion blur

caused by non-translational camera shakes. We adopted a new blur representation of Tai et al. [54], which uses a set of homographies. Our method transforms a blur kernel estimation problem into image registration problems to estimate non-uniform motion blur.

Finally, we presented a method for removing non-uniform motion blur caused by camera motion from video frames in Chapter 6. The proposed method does neither directly estimate blur kernel nor perform deconvolution. Instead, the method removes motion blur from blurry frames using sharp patches from their neighboring frames.

7.2 Future Work

Outlier handling in blind deconvolution The outlier handling method in Chapter 3 handles outliers only in non-blind deconvolution, not in kernel estimation. However, we have discovered that outliers have significant impact on kernel estimation too. As future work, we would like to extend our outlier modeling to the kernel estimation step, in order to develop a blind deconvolution method that is robust enough to handle various outliers.

Single image blind deconvolution for non-uniform motion blur While the methods presented in Chapters 4 and 5 need multiple input images of the same scene, such inputs are sometimes not available. In order to cover such cases, we would like to extend our non-uniform deblurring methods to single image deblurring as future work. We expect that the prediction scheme presented in Chapter 2 would help for non-uniform motion blur as well. Incorporating the outlier handling method presented in Chapter 3 to the non-uniform deblurring methods for making the methods more reliable would be also interesting future work.

Video deblurring The method proposed in Chapter 6 still has a few limitations that we would like to resolve in future. Particularly, we plan to derive a more complicated patch matching algorithm which can handle saturated pixels. We also plan to incorporate deconcolution technique into our method, so that we can improve moving objects by separating the camera motion from the object

motion, and remove the effect of the former. Introducing deconvolution can also help in cases where there is no sharp patches available for parts of the background scene.

Currently our method is implemented in C++ and runs on a single thread. Deblurring a HD size video frame on a PC with Intel Core i7 CPU takes about one minute. However, our approach is easily parallelizable, as the search of sharp patches for blurry pixels can be carried out independently. We expect that GPU implementation would significantly improve the performance.

요 약 문

모션 블러는 영상의 품질을 심각하게 훼손하는 현상 중 하나로 영상을 흐릿하게 만들며 정보의 손실을 일으킨다. 카메라 센서는 영상을 획득하기 위해, 적정량의 빛을 받아들여야만 하며, 이를 위해서는 일정 시간 이상 센서가 노출되어야만 한다. 센서가 노출되어 있는 동안 카메라의 흔들림이나 촬영 대상이 되는 물체의 흔들림에 의해 블러가 발생하게 된다.

디블러링은 카메라나 물체의 움직임으로 인해 흐릿해진 영상으로부터 원래의 선명한 영상을 복원해 내는 문제이다. 모션 블러는 일반적으로 컨볼루션 연산에 기반한 블러 모델을 이용하여 모델링되었다. 이 모델에 따르면 블러 영상은 선명한 영상과 블러 커널의 컨볼루션 연산의 결과로 생성되는데, 이 때문에 디블러링은 컨볼루션의 역연산인 디컨볼루션 문제로 치환이 된다. 또한 디컨볼루션 문제는 흔들린 영상이 주어져 있을 때, 블러 커널과 선명한 영상을 모두 찾아 내는 블라인드 디컨볼루션 문제와 흔들린 영상과 블러 커널을 알 때, 선명한 영상을 찾아 내는 논블라인드 디컨볼루션 문제로 구분이 된다.

디블러링 문제는 그 수요로 인해 많은 연구자들에 의해 오랫동안 연구되어 왔으나, 여전히 매우 풀기 어려운 문제이다. 이 문제가 어려운 이유는 다음과 같이 정리될 수 있다. 첫 번째로 모션 블러는 돌이킬 수 없는 정보의 손실을 초래하며, 이로 인해 본래의 정보를 모두 복원할 수가 없다. 또한, 대부분의 경우 블러 커널을 알지 못하기 때문에, 선명한 영상의 복원을 위해서는 블러 커널을 같이 추정해야 한다는 점이 디블러링 문제를 더욱 풀기 어렵게 한다. 두 번째로 컨볼루션 연산 기반의 블러 모델은 광범위하게 사용되어 왔지만, 이 모델은 실제 흔들린 사진에는 종종 부합되지 않는다. 실제 사진의 경우, 카메라의 회전이나 물체의 움직임으로 인해 컨볼루션 연산으로는 표현할 수 없는 비균일 모션 블러가 흔히 발생하게 된다. 카메라 센서의 결함이나 빛포화 현상과 같은 노이즈와 이상값 역시 디블러링의 성능을

크게 저하시키는 요인이다.

본 논문에서는 상기 디블러링의 문제점들을 극복하기 위한 소프트웨어 기반의 해결책들을 제시하고자 한다. 구체적으로 본 논문에서는 다음의 주제들을 다룬다.

균일 모션 블러를 위한 고속 블라인드 디컨볼루션 방법 기존 블라인드 디블러링 방법들은 디블러링을 위해 매우 긴 연산 시간을 소모했다. 본 연구에서는 이를 획기적으로 줄이는 방법을 제시한다. 또한 연산 속도외에 디블러링 성능 또한 기존 방법보다 안정적임을 실험 결과를 통해 보여준다.

논블라인드 디컨볼루션에서의 이상값 처리 센서 결함과 빛 포화 현상 등에 의해 발생하는 이상값은 기존 논블라인드 디컨볼루션에서 심각한 악영향을 초래한다. 본 연구에서는 이상값이 발생하는 과정을 모델링하여 이상값의 악영향을 피하는 Expectation-Maximization 방법 기반의 논블라인드 디컨볼루션 방법을 제시한다.

비균일 모션 블러를 위한 블라인드 디컨볼루션 방법 균일 모션 블러 모델이 광범위하게 사용되어 왔지만, 실제 사진의 경우에는 물체의 움직임이나 카메라의 회전으로 발생하는 떨림에 의해 비균일 모션 블러가 종종 발생하게 된다. 본 연구에서는 이러한 비균일 모션 블러를 다루기 위한 두 개의 서로 다른 방법을 제시한다.

비디오 프레임의 디블러링 흔들린 비디오 영상은 컴퓨터 비전 알고리즘의 성능에 심각한 악영향을 끼친다. 본 연구에서는 비디오 프레임들에서 흔들리지 않은 선명한 프레임을 이용해 흔들린 프레임의 블러를 제거하는 방법을 제시한다.

Bibliography

[1] BAKER, S., AND MATTHEWS, I. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV) 56*, 3 (2004), 221–255.

[2] BANHAM, M. R., AND KATSAGGELOS, A. K. Digital image restoration. *Signal Processing Magazine, IEEE 14*, 2 (Mar. 1997), 24–41.

[3] BAR, L., SOCHEN, N., AND KIRYATI, N. Variational pairing of image segmentation and blind restoration. In *Proc. ECCV 2004* (2004), vol. 2, pp. 166–177.

[4] BAR, L., SOCHEN, N., AND KIRYATI, N. Image deblurring in the presence of impulsive noise. *International Journal of Computer Vision (IJCV) 70*, 3 (December 2006), 279–298.

[5] BARDSLEY, J., JEFFERIES, S., NAGY, J., AND PLEMMONS, R. Blind iterative restoration of images with spatially varying blur. *Optics Express 14* (2006), 1767–1782.

[6] BASCLE, B., BLAKE, A., AND ZISSERMAN, A. Motion deblurring and superresolution from an image sequence. In *Proc. ECCV 1996* (1996), vol. 2, pp. 573–582.

[7] BEN-EZRA, M., AND NAYAR, S. Motion deblurring using hybrid imaging. In *Proc. CVPR 2003* (Jun 2003), vol. I, pp. 657–664.

[8] BEN-EZRA, M., AND NAYAR, S. K. Motion-based motion deblurring. *IEEE Trans. Pattern Analysis Machine Intelligence 26*, 6 (2004), 689–698.

[9] BHAT, P., ZHENG, K., SNAVELY, N., AGARWALA, A., AGRAWALA, M., COHEN, M., AND CURLESS, B. Piecewise image registration in the presence of multiple large motions. In *Proc. CVPR 2006* (2006), vol. 2, pp. 2491–2497.

[10] BISHOP, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

[11] CHAN, T. F., AND WONG, C.-K. Total variation blind deconvolution. *IEEE Trans. Image Processing 7*, 3 (1998), 370–375.

[12] CHEN, J., YUAN, L., TANG, C.-K., AND QUAN, L. Robust dual motion deblurring. In *Proc. CVPR 2008* (2008), pp. 1–8.

[13] CHEN, W.-G., NANDHAKUMAR, N., AND MARTIN, W. N. Image motion estimation from motion smear - a new computational model. *IEEE Trans. Pattern Analysis Machine Intelligence 18*, 4 (1996), 412–425.

[14] CHO, S., AND LEE, S. Fast motion deblurring. *ACM Trans. Graphics 28*, 5 (2009), article no. 145.

[15] CHO, S., MATSUSHITA, Y., AND LEE, S. Removing non-uniform motion blur from images. In *Proc. ICCV 2007* (2007), pp. 1–8.

[16] COVINGTON, M. A. *Digital SLR Astrophotography*. Cambridge University Press, 2007.

[17] DAI, S., AND WU, Y. Motion from blur. In *Proc. CVPR 2008* (2008), pp. 1–8.

[18] FERGUS, R., SINGH, B., HERTZMANN, A., ROWEIS, S. T., AND FREEMAN, W. Removing camera shake from a single photograph. *ACM Trans. Graphics 25*, 3 (2006), 787–794.

[19] GEMAN, S., AND MCCLURE, D. Bayesian image analysis: An application to single photon emission tomography. In *Proc. of the Statistical Computing Section, American Statistical Association* (1985), vol. 54, pp. 281–289.

[20] GROSSBERG, M., AND NAYAR, S. Modeling the space of camera response functions. *IEEE Trans. Pattern Analysis Machine Intelligence 26*, 10 (Oct 2004), 1272–1282.

[21] GRUNDMANN, M., KWATRA, V., AND ESSA, I. Auto-directed video stabilization with robust l1 optimal camera paths. In *Proc. CVPR 2011* (2011).

[22] GUPTA, A., JOSHI, N., ZITNICK, L., COHEN, M., AND CURLESS, B. Single image deblurring using motion density functions. In *Proc. ECCV 2010* (2010).

[23] HARMELING, S., SRA, S., HIRSCH, M., AND SCHÖLKOPF, B. Multiframe blind deconvolution, super-resolution, and saturation correction via incremental EM. In *Proc. ICIP 2010* (2010), pp. 3313–3316.

[24] HIRSCH, M., SCHULER, C., HARMELING, S., AND SCHOLKOPF, B. Fast removal of non-uniform camera shake. In *Proc. ICCV 2011* (2011).

[25] HIRSCH, M., SRA, S., SCHÖLKOPF, B., AND HARMELING, S. Efficient filter flow for space-variant multiframe blind deconvolution. In *Proc. CVPR 2010* (2010).

[26] HOU, Q., ZHOU, K., AND GUO, B. BSGP: bulk-synchronous GPU programming. *ACM Trans. Graphics 27*, 3 (2008), article no. 19.

[27] JI, H., AND LIU, C. Motion blur identification from image gradients. In *Proc. CVPR 2008* (2008), pp. 1–8.

[28] JIA, J. Single image motion deblurring using transparency. In *Proc. CVPR 2007* (2007), pp. 1–8.

[29] JIN, H., FAVARO, P., AND CIPOLLA, R. Visual tracking in the presence of motion blur. In *Proc. CVPR 2005* (2005), vol. 2, pp. 18–25.

[30] JOSHI, N. *Enhancing photographs using content-specific image priors*. PhD thesis, UCSD, September 2008.

[31] JOSHI, N., KANG, S. B., ZITNICK, L., AND SZELISKI, R. Image deblurring with inertial measurement sensors. *ACM Trans. Graphics 29*, 3 (2010).

[32] JOSHI, N., SZELISKI, R., AND KREIGMAN, D. PSF estimation using sharp edge prediction. In *Proc. CVPR 2008* (2008), pp. 1–8.

[33] KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. Texture optimization for example-based synthesis. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)* (August 2005), 795–802.

[34] LEVIN, A. Blind motion deblurring using image statistics. In *Advances in Neural Information Processing Systems* (2006), pp. 841–848.

[35] LEVIN, A., FERGUS, R., DURAND, F., AND FREEMAN, W. T. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graphics 26*, 3 (2007), article no. 70.

[36] LEVIN, A., SAND, P., CHO, T. S., DURAND, F., AND FREEMAN, W. T. Motion-invariant photography. *ACM Trans. Graphics 27*, 3 (2008), article no. 71.

[37] LEVIN, A., WEISS, Y., DURAND, F., AND FREEMAN, W. Understanding and evaluating blind deconvolution algorithms. In *Proc. CVPR 2009* (2009), pp. 1–8.

[38] LI, Y., KANG, S. B., JOSHI, N., SEITZ, S., AND HUTTENLOCHER, D. Generating sharp panoramas from motion-blurred videos. In *Proc. CVPR 2010* (2010), pp. 2424–2431.

[39] LIU, F., GLEICHER, M., WANG, J., JIN, H., AND AGARWALA, A. Subspace video stabilization. *ACM Trans. Graphics 30*, 1 (2010), 4:1–4:10.

[40] LUCY, L. An iterative technique for the rectification of observed distributions. *Astronomical Journal 79*, 6 (1974), 745–754.

[41] MATSUSHITA, Y., OFEK, E., GE, W., TANG, X., AND SHUM, H.-Y. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Analysis Machine Intelligence 28*, 7 (2006), 1150–1163.

[42] MONEY, J. H., AND KANG, S. H. Total variation minimizing blind deconvolution with shock filter reference. *Image and Vision Computing 26*, 2 (2008), 302–314.

[43] OSHER, S., AND RUDIN, L. I. Feature-oriented image enhancement using shock filters. *SIAM Journal on Numerical Analysis 27*, 4 (1990), 919–940.

[44] RAV-ACHA, A., AND PELEG, S. Two motion-blurred images are better than one. *Pattern Recognition Letters 26* (2005), 311–317.

[45] RICHARDSON, W. Bayesian-based iterative method of image restoration. *J. Opt. Soc. Am. 62*, 1 (1972).

[46] ROSEN, J. B. The gradient projection method for nonlinear programming: Part I. Linear constraints. *Journal of the Society for Industrial and Applied Mathematics 8*, 1 (1960), 181–217.

[47] RUDIN, L. I., OSHER, S., AND FATEMI, E. Nonlinear total variation based noise removal algorithms. *Physica. D 60* (Nov. 1992), 259–268.

[48] SHAN, Q., JIA, J., AND AGARWALA, A. High-quality motion deblurring from a single image. *ACM Trans. Graphics 27*, 3 (2008), article no. 73.

[49] SHAN, Q., XIONG, W., AND JIA, J. Rotational motion deblurring of a rigid object from a single image. In *Proc. ICCV 2007* (2007).

[50] SHI, J., AND TOMASI, C. Good features to track. In *Proc. CVPR 1994* (1994), pp. 593–600.

[51] SZELISKI, R., AND SHUM, H.-Y. Creating full view panoramic image mosaics and texture-mapped models. *SIGGRAPH 97* (1997), 251–258.

[52] TAI, Y.-W., DU, H., BROWN, M. S., AND LIN, S. Image/video deblurring using a hybrid camera. In *Proc. CVPR 2008* (2008), pp. 1–8.

[53] TAI, Y.-W., KONG, N., LIN, S., AND SHIN, S.-Y. Coded exposure imaging for projective motion deblurring. In *Proc. CVPR 2010* (2010).

[54] TAI, Y.-W., TAN, P., AND BROWN, M. S. Richardson-lucy deblurring for scenes under a projective motion path. *IEEE Trans. Pattern Analysis Machine Intelligence 33*, 8 (2011), 1603–1618.

[55] TOMASI, C., AND MANDUCHI, R. Bilateral filtering for gray and color images. In *Proc. ICCV 1998* (1998), pp. 839–846.

[56] WHYTE, O., SIVIC, J., ZISSERMAN, A., AND PONCE, J. Non-uniform deblurring for shaken images. In *Proc. CVPR 2010* (2010).

[57] WIENER, N. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press, 1964.

[58] YITZHAKY, Y., BOSHUSHA, G., LEVI, Y., AND KOPEIKA, N. S. Restoration of an image degraded by vibrations using only a single frame. *Optical Engineering 38*, 8 (2000).

[59] YITZHAKY, Y., MOR, I., LANTZMAN, A., AND KOPEIKA, N. S. Direct method for restoration of motion-blurred images. *Journal of Opt. Soc. Am. A. 15*, 6 (1998), 1512–1519.

[60] YUAN, L., SUN, J., QUAN, L., AND SHUM, H.-Y. Image deblurring with blurred/noisy image pairs. *ACM Trans. Graphics 26*, 3 (2007), article no. 1.

[61] YUAN, L., SUN, J., QUAN, L., AND SHUM, H.-Y. Progressive inter-scale and intra-scale non-blind image deconvolution. *ACM Trans. Graphics 27*, 3 (2008), article no. 74.

Curriculum Vitae

Education

2005. 9. – 2012. 2. Ph.D. in Computer Science and Engineering, POSTECH
Advisor: Prof. Seungyong Lee
Thesis title: Motion Blur Removal of Digital Photographs

2001. 3. – 2005. 8. B.S. in Computer Science and Engineering, POSTECH
double major in Mathematics
Magna Cum Laude

Experience

2010. 7. - 2010.11. Intern at Adobe Systems
Creative Technologies Lab. Seattle, WA, US

2006. 8. - 2007. 2. Intern at Microsoft Research Asia
Visual Computing Group. Bejing, China

Activities

Journal reviewer
IEEE Transactions on Image Processing
The Visual Computer Journal
Journal of Computer Science and Technology
Arabian Journal for Science and Engineering

Conference reviewer
The 13th IEEE International Conference on Computer Vision (ICCV 2011)
The Visual Communications and Image Processing (VCIP 2011)
The 5th Pacific-Rim Symposium on Image and Video Technology (PSIVT 2011)
The 4th Pacific-Rim Symposium on Image and Video Technology (PSIVT 2010)

Awards

2011. 7 Nominated for KCGS (Korea Computer Graphics Society) Young Researcher Award

2009. 2 포항공과대학교 미래정보기술사업단 BK21 국제부문 우수실적상

2009. 2 HCI 2009 Best Paper Award

2008. 10 Microsoft Research Asia 2008-2009 Graduate Research Fellowship Award

2005. 8 Graduated from POSTECH with honor (Magna Cum Laude)

Publications

1. Sunghyun Cho, Hojin Cho, Yu-Wing Tai, Young Su Moon, Junguk Cho, Shihwa Lee, Seungyong Lee, *Lucas-Kanade Image Registration Using Camera Parameters*, To appear in IS&T/SPIE Electronic Imaging 2012.
2. Hojin Cho, Sunghyun Cho, Young Su Moon, Junguk Cho, Shihwa Lee, Seungyong Lee, *Analysis of Practical Coverage of Uniform Motions for Approximating Real Camera Shakes*, To appear in IS&T/SPIE Electronic Imaging 2012.
3. Sunghyun Cho, Jue Wang, Seungyong Lee, *Handling Outliers in Non-blind Image Deconvolution*, In Proc. of the 13th IEEE International Conference on Computer Vision (ICCV 2011), pp. 1–8, Barcelona, Spain, November 2011. (oral presentation)
4. Sunghyun Cho, Hojin Cho, Yu-Wing Tai, Seungyong Lee, *Non-uniform Motion Deblurring for Camera Shakes using Image Registration*, In Proc. of ACM SIGGRAPH 2011 Talks, article no. 62, Vancouver, Canada, August 2011.
5. Sunghyun Cho, Hyunjun Lee, Seungyong Lee, *Image Decomposition Using Deconvolution*, In Proc. of 17th IEEE International Conference on Image Processing (ICIP 2010), pp. 1965–1968, Hong Kong, September 2010.
6. Ghibom Bhak, Soonkoo Lee, Jae Woo Park, Sunghyun Cho, Seung R. Paik, *Amyloid hydrogel derived from curly protein fibrils of alpha-synuclein*, Biomaterials, vol. 31, no. 23, pp. 5986–5995, August 2010.
7. Sunghyun Cho, Seungyong Lee, *Fast Motion Deblurring*, ACM Transactions on Graphics (SIGGRAPH ASIA 2009), vol. 28, no. 5, article no. 145, December 2009.
8. Sunghyun Cho, Hanul Choi, Yasuyuki Matsushita, Seungyong Lee, *Image Retargeting Using Importance Diffusion*, In Proc. of 16th IEEE International Conference on Image Processing (ICIP 2009), pp. 977–980, Cairo, Egypt, November 2009.
9. Sunghyun Cho, Yasuyuki Matsushita, Seungyong Lee, *Removing Non-Uniform Motion Blur from Images*, In Proc. of 11th IEEE International Conference on Computer Vision (ICCV 2007), pp. 1–8, Rio de Janeiro, Brazil, October 2007.

10. 장철훈, 조성현, 이승용, *전문가의 카메라 움직임을 이용한 비디오 리슈팅*, 한국 컴퓨터그래픽스학회 하계 학술대회 논문집, 2010. (extended abstract)
11. 조성현, 이승용, *에지 예측을 기반으로 한 효율적인 영상 디블러링*, HCI 2009 (최우수논문상)
12. 손민정, 조성현, 이승욱, 구본기, 이승용, *스타일화된 얼굴 일러스트레이션*, 한국 컴퓨터그래픽스학회 논문지, vol. 14, no. 2, pp. 27–33, 2008.
13. 손민정, 조성현, 이승욱, 구본기, 이승용, *스타일화된 얼굴 일러스트레이션*, 한국 컴퓨터그래픽스학회 하계 학술대회 논문집, 2008.
14. 조성현, Yasuyuki Matsushita, 이승용, *중요도 확산을 이용한 영상 리타겟팅*, 한국정보과학회 학술대회, 2008.
15. 조성현, Yasuyuki Matsushita, 이승용, *복수의 영상에서 비균일 모션 블러 제거*, 한국 컴퓨터그래픽스학회 하계 학술대회 논문집, 2007. (extended abstract)