Deblurring Using Regularized Locally Adaptive Kernel Regression

Hiroyuki Takeda, *Student Member, IEEE*, Sina Farsiu, *Member, IEEE*, and Peyman Milanfar, *Senior Member, IEEE*

***Abstract*—Kernel regression is an effective tool for a variety of image processing tasks such as denoising and interpolation [1]. In this paper, we extend the use of kernel regression for deblurring applications. In some earlier examples in the literature, such nonparametric deblurring was suboptimally performed in two sequential steps, namely denoising followed by deblurring. In contrast, our optimal solution jointly denoises and deblurs images. The proposed algorithm takes advantage of an effective and novel image prior that generalizes some of the most popular regularization techniques in the literature. Experimental results demonstrate the effectiveness of our method.**

***Index Terms*—Deblurring, denoising, kernel regression, local polynomial, nonlinear filter, nonparametric estimation, spatially adaptive.**

I. Introduction

THE framework of *kernel regression* [2] has been widely used in different guises for solving a variety of pattern detection and discrimination problems [3]. In [1], we described kernel regression as an effective tool for image reconstruction, and established its relation with some popular existing techniques such as *normalized convolution* [4], [5], *bilateral filter* [6], [7], *edge-directed interpolation* [8], and *moving least-squares* [9]. Moreover, we proposed a novel adaptive generalization of kernel regression with excellent results in both denoising and interpolating (for single and multiframe) applications. The image degradation model for all the above techniques mainly consisted of regularly or irregularly sampled data, contaminated with independent and identically distributed (i.i.d.) additive zero mean noise (with otherwise no particular statistical distribution assumed).

In our earlier kernel regression-based image reconstruction methods, an important image degradation source, namely the (camera lens or atmospheric) blur [10], was ignored. In other works, this problem was treated in a two-step process of denoising/deblurring [1], [11]. Such two-step solutions, in general, are suboptimal, and improvements upon them are the subject of this paper.

In general, we have a model for the deblurring problem in Fig. 1, where u is the real scene which is the unknown function of interest, g is the point spread function (PSF) of the blur, ε is an additive white noise, and y is an observation that is the distorted function of u. Following this model, the pixel value y_i at $\mathbf{x}_i$ is expressed as

$$y_i = (g * u)(\mathbf{x}_i) + \epsilon_i, \quad i = 1, \ldots, P \tag{1}$$

where $*$ is the convolution operator, $(g * u)(\mathbf{x}_i)$ is the unknown blurred pixel value at a sampling position $\mathbf{x}_i$, and P is the total number of sampling positions. In matrix notation, we write the blur-free pixels of interest as $\underline{\mathbf{U}} = [u(\mathbf{x}_1), \ldots, u(\mathbf{x}_P)]^T$. Next, we rewrite the model (1) in matrix form as

$$\underline{\mathbf{Y}} = \mathbf{G}\underline{\mathbf{U}} + \underline{\boldsymbol{\varepsilon}} \tag{2}$$

where $\mathbf{Y} \in \mathcal{R}^{L\times M}$ is an observed image, $\mathbf{U} \in \mathcal{R}^{L\times M}$ is the unknown image of interest, and $\boldsymbol{\varepsilon} \in \mathcal{R}^{L\times M}$ is a noise image. The underline indicates that the matrices are lexicographically ordered into a column-stacked vector, e.g., $\underline{\mathbf{Y}} \in \mathcal{R}^{LM\times 1}$. Since the deblurring problem is typically ill-posed, a popular way to estimate the unknown image $\mathbf{U}$ is to use the regularized least square technique [12]

$$\underline{\widehat{\mathbf{U}}}_{\text{RLS}} = \arg\min_{\underline{\mathbf{U}}} \left\|\underline{\mathbf{Y}} - \mathbf{G}\underline{\mathbf{U}}\right\|_2^2 + \lambda C_R(\underline{\mathbf{U}}) \tag{3}$$

where λ is the regularization parameter, and $C_R(\underline{\mathbf{U}})$ is the regularization term. Some representative examples of $C_R(\underline{\mathbf{U}})$ are given in Table I, in which $\boldsymbol{\Upsilon}$ is a high-pass filter (e.g., Laplacian [13]), α is a smoothing parameter, w is the window size, and $\mathbf{S}_{x_1}^{l}$ is the shift operator that shifts an image l pixels in the x_1 direction. In this paper, we replace the first term in (3) with one motivated by kernel regression, which results in a spatially adaptive weighted least square (or least absolute deviation) problem. Additionally, we introduce novel regularization terms which provide spatially adaptive prior information, resulting in a powerful overall setting for deconvolution.

Contributions of this paper are the following. 1) We describe and propose kernel regression as an effective tool for deblurring. 2) We propose a novel image prior which is effective in suppressing both noise and ringing effects. These effects are typical in deblurring applications. We further show that many popular regularization techniques such as digital total-variation (TV), bilateral TV, and Tikhonov regularization [10] in Table I are special cases of this adaptive prior.

Manuscript received October 12, 2007; revised January 13, 2008. This work was supported in part by the U.S. Air Force under Grant F49620-03-1-0387. This work was done while the authors were at University of California Santa Cruz. Software package implementing the proposed method available at http://www.soe.ucsc.edu/~htakeda/AKTV. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Eric Kolaczyk.

H. Takeda and P. Milanfar are with the Electrical Engineering Department, University of California Santa Cruz, Santa Cruz, CA 95064 USA (e-mail: htakeda@soe.ucsc.edu; milanfar@soe.ucsc.edu).

S. Farsiu is with the Department of Ophthalmology, Duke University, Durham, NC 27710 USA (e-mail: sina.farsiu@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2007.918028

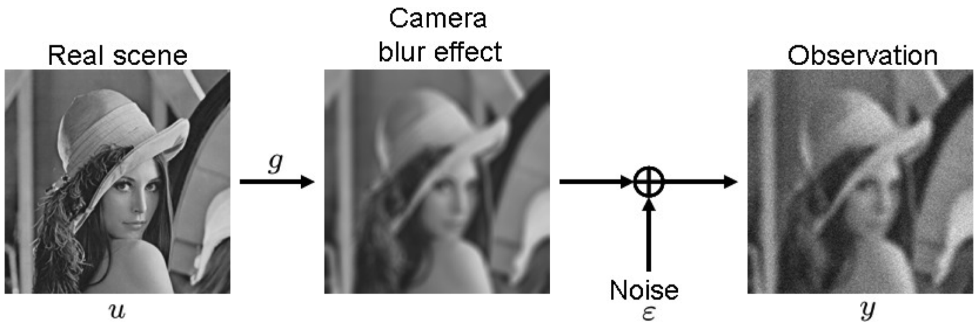


Fig. 1. Data model for the deblurring problem.

TABLE I
REGULARIZATION FUNCTIONS AND THEIR FIRST DERIVATIVES

Tikhonov	Total variation [14]	Bilateral total variation [10]
$\|\Upsilon\underline{\mathbf{U}}\|_2^2$	$\|\Upsilon\underline{\mathbf{U}}\|_1$	$\sum_{l=-w}^{w}\sum_{m=-w}^{w}\alpha^{\lvert l\rvert+\lvert m\rvert}\left\|\underline{\mathbf{U}}-\mathbf{S}_{x_1}^{l}\mathbf{S}_{x_2}^{m}\underline{\mathbf{U}}\right\|_1$

This paper is organized as follows. In Section II, first ignoring the blurring effect, we briefly review the kernel regression framework as described in [1]. Then, we include the blurring effect in the data degradation model (Fig. 1), and introduce a regularized estimator based on the kernel regression framework for the deblurring application. In Section III, we extend the kernel-based deblurring to a data-adaptive method which is the basis for the best results described in this paper. Experiments are presented in Sections IV, and Section V concludes this paper.

II. KERNEL-BASED DEBLURRING

In this section, we briefly review the classical kernel regression approach, and provide some intuition behind it. After that, including the blurring effect, we derive a regularized kernel regression method for the deblurring application.

A. Review

Defining the blurry function as $z(\mathbf{x}) = (g * u)(\mathbf{x})$, we rewrite the data model (1)

$$y_i = z(\mathbf{x}_i) + \epsilon_i, \quad i = 1, \ldots, P, \quad \mathbf{x}_i = [x_{1i}, x_{2i}]^T \tag{4}$$

where y_i is a noise-laden sample at $\mathbf{x}_i$, $z(\cdot)$ is the (hitherto unspecified) *regression function* to be estimated, ε_i is i.i.d. zero mean noise, and P is the total number of samples in a neighborhood (window) of interest. As such, the kernel regression framework provides a rich mechanism for computing point-wise estimates of the regression function with minimal assumptions about global signal or noise models.

While the particular form of $z(\cdot)$ may remain unspecified for now, we can rely on a generic local expansion of the function about a sampling point $\mathbf{x}_i$. Specifically, if $\mathbf{x}$ is near the sample at $\mathbf{x}_i$, we have the Nth order Taylor series[1]

$$\begin{aligned} z(\mathbf{x}_i) &= z(\mathbf{x}) + \{\nabla z(\mathbf{x})\}^T(\mathbf{x}_i - \mathbf{x}) \\ &\quad + \frac{1}{2}(\mathbf{x}_i - \mathbf{x})^T\{\mathcal{H}z(\mathbf{x})\}(\mathbf{x}_i - \mathbf{x}) + \cdots \qquad (5)\\ &= \beta_0 + \boldsymbol{\beta}_1^T(\mathbf{x}_i - \mathbf{x}) + \boldsymbol{\beta}_2^T \operatorname{vech}\left\{(\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^T\right\} \\ &\quad + \cdots \qquad (6)\end{aligned}$$

where ∇ and $\mathcal{H}$ are the gradient (2×1) and Hessian (2×2) operators, respectively, and $\operatorname{vech}(\cdot)$ is the half-vectorization operator which lexicographically orders the lower triangular portion of a symmetric matrix into a column-stacked vector, e.g.,

$$\begin{aligned} \operatorname{vech}\left(\begin{bmatrix} a & b \\ b & c \end{bmatrix}\right) &= [a \quad b \quad c]^T \\ \operatorname{vech}\left(\begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}\right) &= [a \quad b \quad c \quad d \quad e \quad f]^T. \end{aligned} \tag{7}$$

Furthermore, β_0 is $z(\mathbf{x})$, which is the pixel value of interest, and the vectors $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are

$$\boldsymbol{\beta}_1 = \begin{bmatrix} \frac{\partial z(\mathbf{x})}{\partial x_1} & \frac{\partial z(\mathbf{x})}{\partial x_2} \end{bmatrix}^T \tag{8}$$

$$\boldsymbol{\beta}_2 = \begin{bmatrix} \frac{\partial^2 z(\mathbf{x})}{2\partial x_1^2} & \frac{\partial^2 z(\mathbf{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 z(\mathbf{x})}{2\partial x_2^2} \end{bmatrix}^T. \tag{9}$$

Since this approach is based on a *local* representation of say order N, a logical step to take is to estimate the parameters $\{\boldsymbol{\beta}_n\}_{n=0}^{N}$ from all the samples $\{y_i\}_{i=1}^{P}$, while giving the nearby

[1]By using the Taylor series, we are implicitly assuming that the regression function is a locally smooth function up to some order. Of course, other localized representations are also possible and may be advantageous .

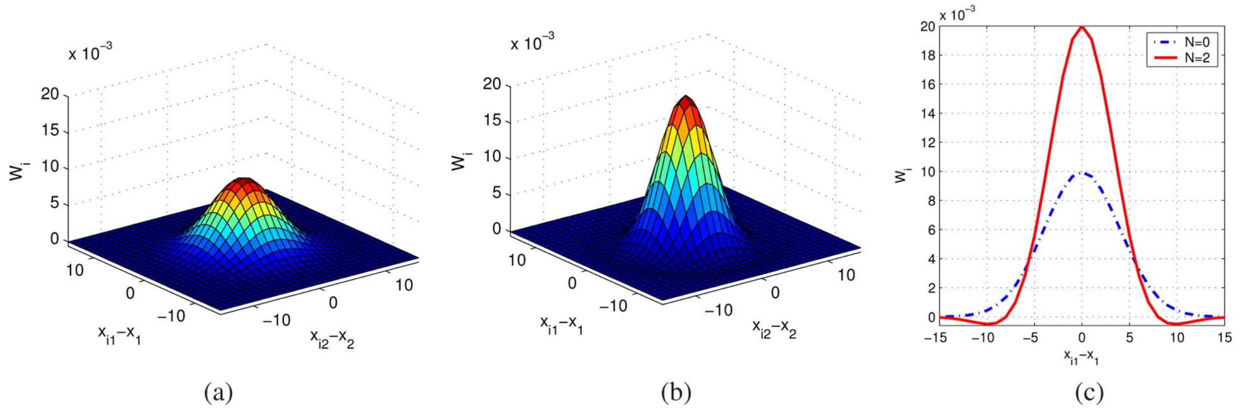


Fig. 2. Equivalent kernel function for the uniformly sampled data case with (a) $N = 0$ and 1, and (b) $N = 2$. The cross sections for each are in (c). The kernel $K_{\mathbf{H}_i}$ in (10) is modeled as a Gaussian with the smoothing matrix $\mathbf{H}_i = \text{diag}[4, 4]$. (a) $N = 0$. (b) $N = 2$. (c) Cross sections.

samples higher weights than samples farther away. A formulation of the fitting problem capturing this idea is to solve the following optimization problem:

$$\min_{\{\boldsymbol{\beta}_N\}_{n=0}^{N}} \sum_{i=1}^{P} \Big| y_i - \beta_0 - \boldsymbol{\beta}_1^T(\mathbf{x}_i - \mathbf{x}) - \boldsymbol{\beta}_2^T \text{vech}\left\{(\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^T\right\} - \cdots \Big|^q K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}) \quad (10)$$

with

$$K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}) = \frac{1}{\det(\mathbf{H}_i)} K\left(\mathbf{H}_i^{-1}(\mathbf{x}_i - \mathbf{x})\right) \quad (11)$$

where N is the regression order, q is the error norm parameter typically set to 2 (for more on the choice of $q = 1$, see [15]), and $K(\cdot)$ is the kernel function (a radially symmetric function such as a Gaussian), and $\mathbf{H}_i$ is the smoothing (2×2) matrix which dictates the “footprint” of the kernel function. The simplest choice of the smoothing matrix is $\mathbf{H}_i = h\mathbf{I}$, where h is called the global smoothing parameter. The shape of the kernel footprint is one of the most important factors that determines the quality of the reconstructed image. For example, it is desirable to use kernels with larger footprints in the smooth areas of the image to reduce the noise effects, while relatively smaller footprints are desirable in the edge and texture areas. More details about the optimization problem above can be found in [1] and [15], and we give details about the choice of the kernel functions in Section III.

To sum up so far, the optimization problem (10), which we term “classic” kernel regression, eventually provides a pointwise estimator of the regression function (q.v. [1], [2], and [15] for derivations). Regardless of the choice of the kernel function (K) and the regression order (N), the estimator always yields a weighted linear combination of nearby samples, that is

$$\hat{z}(\mathbf{x}) = \hat{\beta}_0 = \sum_{i=1}^{P} W_i(K, N, \mathbf{x}_i - \mathbf{x}) y_i, \quad \sum_{i=1}^{P} W_i(\cdot) = 1 \quad (12)$$

where we call W_i the *equivalent kernel* function for y_i. For example, when we choose $N = 0$, the estimator (12) becomes

$$\hat{z}(\mathbf{x}) = \frac{\sum_{i=1}^{P} K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}) y_i}{\sum_{i=1}^{P} K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x})} \quad (13)$$

which is known as the *Nadaraya–Watson* estimator (NWE) [16]. Of course, higher order regressions $(N > 0)$ are also possible. Fig. 2 shows the equivalent kernel function with $N = 0, 1$, and 2 for a uniformly sampled data set.[2] The significant difference between the zeroth and the second order weights is their tails. For $N = 0, 1$, the equivalent kernel is essentially a normalized kernel function as seen in (13). On the other hand, for $N = 2$, the shape of the equivalent kernel is transformed and the weight values in the tail area are negative due to the higher order polynomial in the Taylor series. In general, lower order approximates, such as NWE, result in smoother images (large bias and small variance) as there are fewer degrees of freedom. On the other hand, over-fitting happens in regressions using higher orders of approximation, resulting in small bias and large estimation variance. We also note that smaller values for h (the global smoothing parameter) result in small bias and consequently large variance in estimates. Optimal order and smoothing parameter selection procedures are studied in [9].

In this section, we studied the “classic” kernel regression method, its properties, and showed its usefulness for image restoration purposes. However, as described above, the kernel method ignores the blurring effect, and consequently if deblurring is desired, this must be applied as a postprocessing step to recover the sharpness of the original image. In the following, considering the blurring effect, we derive a regularized deblurring estimator based on the kernel regression framework. After that, in Section III, we present data-adaptive kernel functions which enhance the performance of the deblurring method, and discuss how the deblurring method with the adaptive kernels improves upon other existing methods in the literature. We also show in detail how to implement our deblurring method.

[2]The equivalent kernel function depends on sampling positions. Since this paper focuses on the single frame deblurring problem, we show only the uniformly sampled case where $N = 0$ and 1 give the same weights. The irregularly sampled case and more details can be found in [1].

B. Kernel-Based Deblurring

Considering the blurring effect, instead of z, the function u is the one we wish to estimate. Therefore, in place of representing the blurred function z by a local approximation (Taylor series), we apply the kernel regression framework to u. Following the same procedure as above, the local signal representation of the unknown function u with a Taylor series is

$$u(\mathbf{x}_i) = u(\mathbf{x}) + \{\nabla u(\mathbf{x})\}^T(\mathbf{x}_i - \mathbf{x}) + \frac{1}{2}(\mathbf{x}_i - \mathbf{x})^T\{\mathcal{H}u(\mathbf{x})\}(\mathbf{x}_i - \mathbf{x}) + \cdots \quad (14)$$

Furthermore, in the presence of blur, the blurring effect associates (couples) all the sampling positions $\{\mathbf{x}_i\}_{i=1}^P$ with their neighbors, and this linkage precludes one-by-one estimations of deblurred pixels, such as (12), and necessitates a simultaneous estimation of all the pixels of interest, e.g., (3). Hence, we tightly constrain the reciprocal relationship between neighboring pixel values by a local representation model, i.e., the Taylor series, in order to derive a simultaneous pixel estimator based on the kernel regression framework. Suppose we have a pixel value $u(\mathbf{x}_i)$ and its neighbor $u(\mathbf{x}_j)$ which is located v_1-pixels in the x_1-direction and v_2-pixels in the x_2-direction away from $u(\mathbf{x}_i)$, i.e.,

$$u(\mathbf{x}_j) = u(\mathbf{x}_i + \mathbf{v}) \quad (15)$$

where $\mathbf{v} = [v_1, v_2]^T$. The Taylor series indicates the following relationship between $u(\mathbf{x}_i)$ and $u(\mathbf{x}_j)$:

$$\begin{aligned} u(\mathbf{x}_j) &= u(\mathbf{x}_i + \mathbf{v}) \\ &= u(\mathbf{x}_i) + \{\nabla u(\mathbf{x}_i)\}^T\mathbf{v} + \frac{1}{2}\mathbf{v}^T\{\mathcal{H}u(\mathbf{x}_i)\}\mathbf{v} + \cdots \\ &= u(\mathbf{x}_i) + \begin{bmatrix} u_{x_1}(\mathbf{x}_i) \\ u_{x_2}(\mathbf{x}_i) \end{bmatrix}^T \mathbf{v} \\ &\quad + \frac{1}{2}\begin{bmatrix} u_{x_1^2}(\mathbf{x}_i) \\ 2u_{x_1x_2}(\mathbf{x}_i) \\ u_{x_2^2}(\mathbf{x}_i) \end{bmatrix}^T \text{vech}\left\{\mathbf{v}\mathbf{v}^T\right\} + \cdots \end{aligned} \quad (16)$$

In order to estimate all the pixels simultaneously as an image, we write the local representation (16) at every sampling point $\left(\{\mathbf{x}_i\}_{i=1}^P\right)$ together, and gather the result into lexicographically stacked vector form as

$$\begin{aligned} \begin{bmatrix} u(\mathbf{x}_1 + \mathbf{v}) \\ \vdots \\ u(\mathbf{x}_P + \mathbf{v}) \end{bmatrix} &= \begin{bmatrix} u(\mathbf{x}_1) \\ \vdots \\ u(\mathbf{x}_P) \end{bmatrix} + \begin{bmatrix} u_{x_1}(\mathbf{x}_1) \\ \vdots \\ u_{x_1}(\mathbf{x}_P) \end{bmatrix} v_1 + \begin{bmatrix} u_{x_2}(\mathbf{x}_1) \\ \vdots \\ u_{x_2}(\mathbf{x}_P) \end{bmatrix} v_2 \\ &\quad + \frac{1}{2}\begin{bmatrix} u_{x_1^2}(\mathbf{x}_1) \\ \vdots \\ u_{x_1^2}(\mathbf{x}_P) \end{bmatrix} v_1^2 + \begin{bmatrix} u_{x_1x_2}(\mathbf{x}_1) \\ \vdots \\ u_{x_1x_2}(\mathbf{x}_P) \end{bmatrix} v_1v_2 \\ &\quad + \frac{1}{2}\begin{bmatrix} u_{x_2^2}(\mathbf{x}_1) \\ \vdots \\ u_{x_2^2}(\mathbf{x}_P) \end{bmatrix} v_2^2 + \cdots \end{aligned} \quad (17)$$

For convenience, we denote, for example, the first and second right hand vectors in (17) as the lexicographically ordered desired image ($\underline{\mathbf{U}}$) and its first derivative along the x_1 direction ($\underline{\mathbf{U}}_{x_1}$), respectively. Additionally, the left-hand vector in (17) can be regarded as the shifted version of $\underline{\mathbf{U}}$, and, thus, we simplify the notation in (17) as

$$\begin{aligned} \mathbf{S}_{x_1}^{v_1}\mathbf{S}_{x_2}^{v_2}\underline{\mathbf{U}} &= \underline{\mathbf{U}} + \underline{\mathbf{U}}_{x_1}v_1 + \underline{\mathbf{U}}_{x_2}v_2 \\ &\quad + \underline{\mathbf{U}}_{x_1^2}v_1^2 + \underline{\mathbf{U}}_{x_1x_2}v_1v_2 + \underline{\mathbf{U}}_{x_2^2}v_2^2 + \cdots \end{aligned} \quad (18)$$

where $\mathbf{S}_{x_1}^{v_1}$ and $\mathbf{S}_{x_2}^{v_2}$ are the v_1- or v_2-pixel shift operators along the x_1 or x_2 directions, respectively. Finally, considering an Nth order approximation, we have

$$\begin{aligned} \underline{\mathbf{U}} &= \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\Big(\underline{\mathbf{U}} + \underline{\mathbf{U}}_{x_1}v_1 + \underline{\mathbf{U}}_{x_2}v_2 \\ &\qquad + \underline{\mathbf{U}}_{x_1^2}v_1^2 + \underline{\mathbf{U}}_{x_1x_2}v_1v_2 + \underline{\mathbf{U}}_{x_2^2}v_2^2 + \cdots\Big) \\ &\approx \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{I}_N\mathbb{U}_N \end{aligned} \quad (19)$$

where, for example, when $N = 2$, we have

$$\begin{aligned} \mathbb{I}_2 &= [\mathbf{I} \quad \mathbf{I}_{v_1} \quad \mathbf{I}_{v_2} \quad \mathbf{I}_{v_1^2} \quad \mathbf{I}_{v_1v_2} \quad \mathbf{I}_{v_2^2}] \\ \mathbb{U}_2 &= \left[\underline{\mathbf{U}}^T \quad \underline{\mathbf{U}}_{x_1}^T \quad \underline{\mathbf{U}}_{x_2}^T \quad \underline{\mathbf{U}}_{x_1^2}^T \quad \underline{\mathbf{U}}_{x_1x_2}^T \quad \underline{\mathbf{U}}_{x_2^2}^T\right]^T \end{aligned} \quad (20)$$

and $\mathbf{I}_{v_1} = \text{diag}\{v_1, \ldots, v_1\}$. Naturally, with fewer higher-order terms in this expansion, the smaller shift distances result in a more faithful approximation in (19). This suggests a general (smoothness) prior (regularization term) with weights given by the shift (spatial) distance $\mathbf{v}$ for images smooth to order N as

$$C_R(\mathbb{U}_N) = \sum_{v_1}\sum_{v_2}\left\|\mathbf{W}_u^{(1/q_r)}(\mathbf{v})\left(\underline{\mathbf{U}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{I}_N\mathbb{U}_N\right)\right\|_{q_r}^{q_r} \quad (21)$$

which we call the *adaptive kernel total variation* (AKTV) when $q_r = 1$, and where $\mathbf{W}_u(\mathbf{v})$ is the weight matrix for this term defined as

$$\mathbf{W}_u(\mathbf{v}) = \text{diag}\left\{K_{\mathbf{H}_{(\mathbf{x}_1+\mathbf{v})}}(\mathbf{v}), \ldots, K_{\mathbf{H}_{(\mathbf{x}_P+\mathbf{v})}}(\mathbf{v})\right\}. \quad (22)$$

Here again, K is the kernel function and $\mathbf{H}_{(\mathbf{x}_i+\mathbf{v})}$ is the smoothing matrix at $(\mathbf{x}_i + \mathbf{v})$ computed based on the unknown (estimated) function $\hat{u}(\cdot)$ as described in detail in Section III. The deblurring problem is often ill-posed, in particular, when the width of the PSF is large. Therefore, the regularization term (21) is useful to further restrict the solution space for the signal of interest.

Having introduced the regularization term, we now turn our attention to describing the kernel-based "deblurring" term. Using the local representation (19), the blurred noisy image $\mathbf{Y}$ is expressed as

$$\begin{aligned} \underline{\mathbf{Y}} &= \mathbf{G}\underline{\mathbf{U}} + \underline{\boldsymbol{\varepsilon}} \\ &= \mathbf{G}\mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\Big(\underline{\mathbf{U}} + \underline{\mathbf{U}}_{x_1}v_1 + \underline{\mathbf{U}}_{x_2}v_2 \\ &\qquad + \underline{\mathbf{U}}_{x_1^2}v_1^2 + \underline{\mathbf{U}}_{x_1x_2}v_1v_2 + \underline{\mathbf{U}}_{x_2^2}v_2^2 + \cdots\Big) + \underline{\boldsymbol{\varepsilon}} \\ &\approx \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{G}_N\mathbb{U}_N + \underline{\boldsymbol{\varepsilon}} \end{aligned} \quad (23)$$

with

$$\underline{\mathbf{Y}} = \left[y_1, y_2, \ldots, y_P\right]^T, \quad \underline{\boldsymbol{\varepsilon}} = \left[\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_P\right]^T$$
$$\mathbb{G}_N = [\mathbf{G} \quad \mathbf{GI}_{v_1} \quad \mathbf{GI}_{v_2} \quad \mathbf{GI}_{v_1^2} \quad \mathbf{GI}_{v_1 v_2} \quad \mathbf{GI}_{v_1^2} \quad \ldots] \tag{24}$$

where $\mathbf{G}$ is a $P \times P$ matrix representing the blurring operation given by $g(\cdot)$. Similar to the regularization term (21), the local representation for the blurred noisy image (23) suggests a data fidelity or likelihood term

$$C_L(\mathbb{U}_N) = \sum_{v_1}\sum_{v_2} \left\| \mathbf{W}_z^{(1/q)}(\mathbf{v}) \left(\underline{\mathbf{Y}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{G}_N\mathbb{U}_N \right) \right\|_q^q \tag{25}$$

where q is the error norm parameter for this term, and $\mathbf{W}_z(\mathbf{v})$ is the weight matrix for this likelihood term computed based on the estimated blurred signal $\hat{z}(\cdot) = (g * \hat{u})(\cdot)$ as described in Section III.

In summary, the overall cost function to be minimized is formulated as

$$\begin{aligned} C(\mathbb{U}_N) &= C_L(\mathbb{U}_N) + \lambda C_R(\mathbb{U}_N) \\ &= \sum_{v_1}\sum_{v_2} \left\{ \left\| \mathbf{W}_z^{(1/q)}(\mathbf{v}) \left(\underline{\mathbf{Y}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{G}_N\mathbb{U}_N \right) \right\|_q^q \right. \\ &\quad \left. + \lambda \left\| \mathbf{W}_u^{(1/q_r)}(\mathbf{v}) \left(\underline{\mathbf{U}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{I}_N\mathbb{U}_N \right) \right\|_{q_r}^{q_r} \right\} \end{aligned} \tag{26}$$

where λ is the regularization parameter. We can solve the optimization problem using the steepest descent method (see Appendix A for the gradient term)

$$\widehat{\mathbb{U}}_N^{(\ell+1)} = \widehat{\mathbb{U}}_N^{(\ell)} + \mu \left. \frac{\partial C(\mathbb{U}_N)}{\partial \mathbb{U}_N} \right|_{\mathbb{U}_N = \widehat{\mathbb{U}}_N^{(\ell)}} \tag{27}$$

where μ is the step size.

C. Related Regularization Terms

There are two points worth highlighting about the regularization cost function in (21). First, the regularization term is general enough to subsume several other popular regularization terms existing in the literature, such as the ones in Table I. In particular, note that, if we choose the zeroth regression order ($N = 0$), (21) becomes

$$\sum_{v_1}\sum_{v_2} \left\| \mathbf{W}_u^{(1/q_r)}(\mathbf{v}) \left(\underline{\mathbf{U}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\underline{\mathbf{U}} \right) \right\|_{q_r}^{q_r}. \tag{28}$$

Therefore, for $q_r = 1$, the regularization term (28) can be regarded as digital TV [17] with $\mathbf{W}_u(\mathbf{v}) = \mathbf{I}$, $|v_1| \leq 1$, and $|v_2| \leq 1$. Alternatively, again with $N = 0$, and with $\mathbf{W}_u(\mathbf{v}) = \alpha^{\|\mathbf{v}\|_1}\mathbf{I}$, where $0 \leq \alpha \leq 1$, the term represents the bilateral total variation (BTV) regularization criterion first introduced in [10]. That is to say, the BTV kernel function is defined as

$$K_\alpha(\mathbf{x}_i - \mathbf{x}) = \alpha^{\|\mathbf{x}_i - \mathbf{x}\|_1} \tag{29}$$

where α in this case is the global smoothing parameter. Second, with the choice of the zeroth order ($N = 0$), we would be assuming that the unknown image $\mathbf{U}$ is piecewise constant. On the other hand, the unknown image is assumed piecewise linear and quadratic with $N = 1$ and 2, respectively. In addition, with the locally data-adaptive weights (introduced in Section III), the proposed framework can effectively remove noise and ringing effects, and preserve local edges. Of course, other choices of the kernel function are also possible. In the image restoration literature, the use of data-adaptive kernel functions has recently become very popular. As such, one may opt to use mean-shift [18], [19], or nonlocal mean [20] weights as the kernel function in (26). These methods have implicitly used zeroth-order local representations ($N = 0$), which can also be generalized [21].

III. Locally Adaptive, Data-Dependant Weights

One fundamental improvement on the above method can be realized by noting that the kernel regression estimates using (11), independent of the order (N), are always local *linear* combinations on the data, i.e., (12) with the weights shown in Fig. 2. Hence, they suffer from an inherent limitation due to this local linear action on the data. This is relevant to the kernel-based deblurring estimator as well. In [1], we introduced *data-adaptive kernel functions* which rely on not only the spatial distances ($\mathbf{x}_i - \mathbf{x}$), but also on the radiometric properties of these samples, i.e.,

$$K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}) \Rightarrow K_{\text{adapt}}(\mathbf{x}_i - \mathbf{x}, y_i - y). \tag{30}$$

Moreover, we established key relationships between data-adaptive kernel regression and some existing adaptive filters, such as the bilateral filter, and showed its effectiveness for denoising.

In this section, we briefly describe two types of data-adaptive kernel functions [1]: 1) bilateral kernel and 2) steering kernel. By applying the data-adaptive kernels for the kernel-based deblurring estimator, we arrive at a deblurring method that can effectively suppress both the noise and ringing effects, which are fundamental issues in the deblurring problem. As we will see later in the experimental section, the choice of steering kernels is most effective. Furthermore, we will also explain how to implement the data-adaptive kernel-based deblurring method, in which we estimate the unknown image $\mathbf{U}$ by iteratively updating the image and the data-adaptive weight matrix.

A. Bilateral Kernel Function

An intuitive choice of K_{adapt} is to use separate terms for penalizing the spatial distance between the pixel position of interest and its neighbors' positions $\{\mathbf{x}_i\}$, and the radiometric distance between the corresponding pixels y and $\{y_i\}$

$$K_{\text{bilat}}(\mathbf{x}_i - \mathbf{x}, y_i - y) = K_{\mathbf{H}_s}(\mathbf{x}_i - \mathbf{x})K_{h_r}(y_i - y) \tag{31}$$

with $\mathbf{H}_s = h_s\mathbf{I}$ where h_s and h_r are the global spatial and radiometric smoothing parameters, respectively. Data adaptivity in the bilateral kernel is incorporated by explicitly taking the radiometric distances into account. Fig. 3 illustrates the kernels of the bilateral filter at different positions. (Note: We did not add

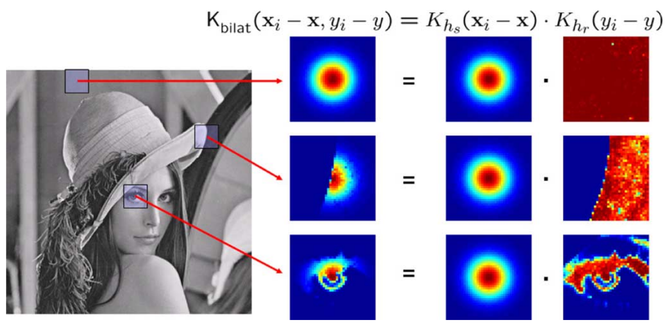

$K_{\text{bilat}}(\mathbf{x}_i - \mathbf{x}, y_i - y) = K_{h_s}(\mathbf{x}_i - \mathbf{x}) \cdot K_{h_r}(y_i - y)$
=
·

Fig. 3. Bilateral kernels in different positions: (top) a flat part, (middle) an edge part, and (bottom) Lena's eye part in the high-SNR case. We did not add noise.

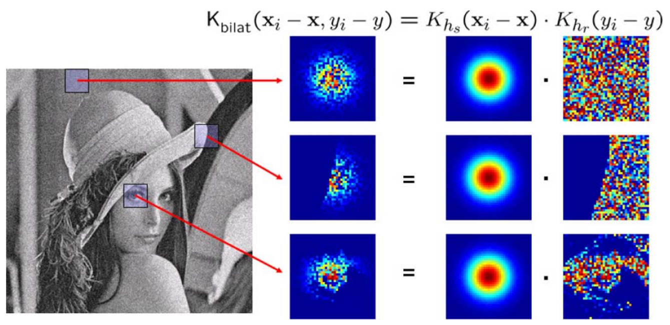

$K_{\text{bilat}}(\mathbf{x}_i - \mathbf{x}, y_i - y) = K_{h_s}(\mathbf{x}_i - \mathbf{x}) \cdot K_{h_r}(y_i - y)$
=
·

Fig. 4. Bilateral kernels in different positions: (top) a flat part, (middle) an edge part, and (bottom) Lena's eye part in the low-SNR case. We added white Gaussian noise with standard deviation 25 (the corresponding SNR is 5.64 [dB]).

any noise here to the Lena image). For example, choosing the bilateral kernel function and $N = 0$ for (12), we have

$$\hat{z}(\mathbf{x}) = \frac{\sum_{i=1}^{P} K_{\mathbf{H}_s}(\mathbf{x}_i - \mathbf{x}) K_{h_r}(y_i - y) y_i}{\sum_{i=1}^{P} K_{\mathbf{H}_s}(\mathbf{x}_i - \mathbf{x}) K_{h_r}(y_i - y)} \tag{32}$$

which is nothing but the well-known bilateral filter [6], [7]. In [21], we showed that the bilateral filter is a special case (zeroth order bilateral kernel regression) of the general kernel regression denoising approach. Establishing such relation helped us to design higher-order bilateral filters with superior performance with respect to the classic definition of the bilateral filter in [6]. Unfortunately, as explained in [1], for very noisy images, the performance of the bilateral kernel techniques significantly degrades due to the direct use of neighboring pixel differences which make the radiometric weight values noisy (see Fig. 4). Such weights become effectively useless for denoising. To overcome this drawback, we proposed in [1] an alternate data-adaptive kernel function: the *steering kernel function*.

B. Steering Kernel Function

The filtering procedure we propose next takes the above ideas one step further, based upon the earlier nonparametric framework. In particular, we observe that the effect of computing $K_{h_r}(y_i - y)$ in (31) is to implicitly measure a function of the local gradient estimated between neighboring values, and to use this estimate to weight the respective measurements. As an example, if a pixel is located near an edge, then pixels on the same side of the edge will have much stronger influence in the filtering. With this intuition in mind, we propose a two-step approach where first an initial estimate of the image gradients is computed using some kind of gradient estimator [say the second order classic kernel regression, e.g., (10) with the choice of K as Gaussian, $q = 2$ and $N = 2$]. Next, this estimate is used to measure the dominant orientation of the local gradients in the image (e.g., [22]). In a second filtering stage, this orientation information is then used to adaptively "steer" the local kernel, resulting in elongated, elliptical contours spread along the directions of the local edge structure. Fig. 5 illustrates a schematic representation of kernel footprints. With these locally adapted kernels, the denoising is effected most strongly along the edges, rather than across them, resulting in strong preservation of details in the final output. To be more specific, the data-adapted steering kernel takes the form

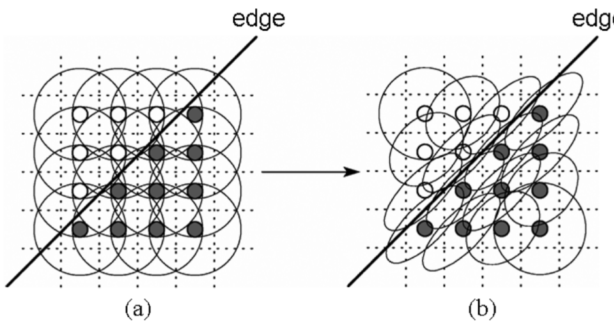


Fig. 5. Kernel spread in a uniformly sampled data set. (a) Kernels in the classic method depend only on the sample density. (b) Data-adapted kernels elongate with respect to the edge.

$$K_{\text{steer}}(\mathbf{x}_i - \mathbf{x}, y_i - y) = K_{\mathbf{H}_i^s}(\mathbf{x}_i - \mathbf{x}) \tag{33}$$

where $\mathbf{H}_i^s$ are now data-adaptive full (2×2) matrices, called *steering matrices*. We define them as

$$\mathbf{H}_i^s = h\mathbf{C}_i^{-(1/2)} \tag{34}$$

where $\mathbf{C}_i$'s are (symmetric) inverse covariance matrices based on differences in the local gray-values. A good choice for $\mathbf{C}_i$'s will effectively spread the kernel function along the local edges as shown in Fig. 5(b).

The local edge structure is related to the gradient covariance (or equivalently, the locally dominant orientation), where a naive estimate of this covariance matrix may be obtained as follows:

$$\widehat{\mathbf{C}}_i = \mathbf{J}^T\mathbf{J} \tag{35}$$

with

$$\mathbf{J} = \begin{bmatrix} \vdots & \vdots \\ z_{x_1}(\mathbf{x}_j) & z_{x_2}(\mathbf{x}_j) \\ \vdots & \vdots \end{bmatrix}, \quad \mathbf{x}_j \in \xi_i \tag{36}$$

where $z_{x_1}(\cdot)$ and $z_{x_2}(\cdot)$ are the first derivatives along x_1 and x_2 axes, respectively, and ξ_i is a local analysis window around the position of interest. The dominant local orientation of the gradients is then related to the eigenvectors of this estimated matrix.

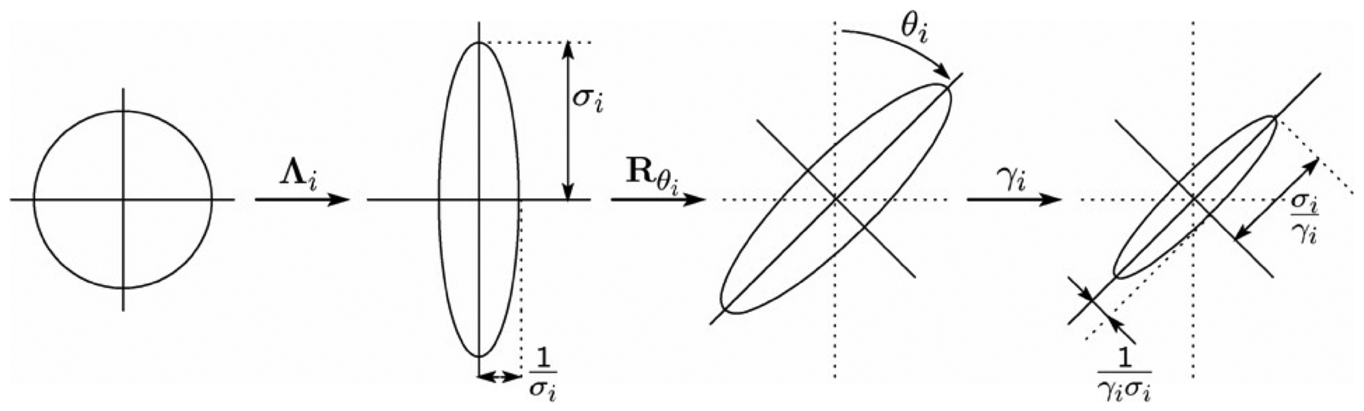


Fig. 6. Schematic representation illustrating the effects of the steering matrix and its component $\left(\mathbf{C}_i = \gamma_i \mathbf{R}_{\theta_i} \boldsymbol{\Lambda}_i \mathbf{R}_{\theta_i}^T\right)$ on the size and shape of the regression kernel.

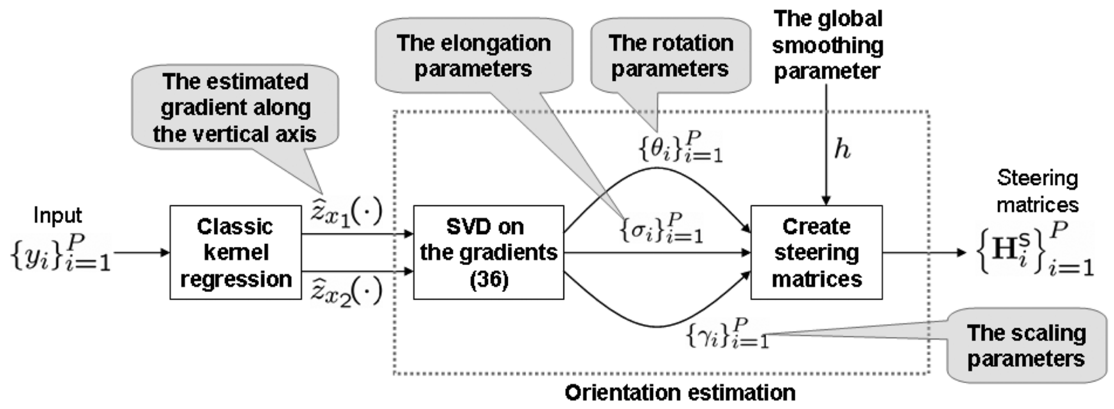


Fig. 7. Block diagram representation of the steering matrix estimation.

Since the gradients depend on the pixel values, and since the choice of the localized kernels in turn depends on these gradients, it, therefore, follows that the "equivalent" kernels for the proposed data-adapted methods form a locally *nonlinear* combination of the data.

While this approach (which is essentially a local principal components method to analyze image (orientation) structure [22]–[24]) is simple and has nice tolerance to noise, the resulting estimate of the covariance may in general be rank deficient or unstable. Therefore, in this case, care must be taken not to take the inverse of the estimate directly.

In order to have a stable estimation for the covariance matrix, we parameterize and regularize it as follow. First, we decompose it into three components (equivalent to eigenvalue decomposition)

$$\mathbf{C}_i = \gamma_i \mathbf{R}_{\theta_i} \boldsymbol{\Lambda}_i \mathbf{R}_{\theta_i^T}$$
$$\mathbf{R}_{\theta_i} = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix}, \quad \boldsymbol{\Lambda}_i = \begin{bmatrix} \sigma_i & 0 \\ 0 & \sigma_i^{-1} \end{bmatrix} \tag{37}$$

where $\mathbf{R}_{\theta_i}$ is a rotation matrix and $\boldsymbol{\Lambda}$ is the elongation matrix. Now the covariance matrix is given by the three parameters γ_i, θ_i, and σ_i, which are the scaling, rotation, and elongation parameters, respectively. Fig. 6 schematically explains how these parameters affect the spreading of kernels. We estimate these parameters as described in [1], and create the steering matrices as illustrated in Fig. 7.

Fig. 8 illustrates examples of the steering kernel function for (a) high- and (b) low-SNR cases (no noise is added for the high SNR case, and we added white Gaussian noise with standard deviation 25, the corresponding SNR being 5.64 [dB]). Note that the footprints are showing the contours of the steering kernels for each center pixel of the window, i.e., illustrating the steering kernels $K_{\mathbf{H}_i^s}(\mathbf{x}_i - \mathbf{x})$ as a function of $\mathbf{x}$ when $\mathbf{x}_i$ and $\mathbf{H}_i$ are fixed at the center of each window. As explained above and shown in Fig. 8(a), the steering kernel footprints capture the local image "edge" structures (large in flat areas, stretched in edge areas, and small in texture areas). Meanwhile, the steering kernel weights ($K_{\mathbf{H}_i^s}(\mathbf{x}_i - \mathbf{x})$ as a function of $\mathbf{x}_i$ with $\mathbf{x}$ held fixed) illustrate the relative size of the actual weights applied to compute the estimate as in (12). We note that even for the highly noisy case [Fig. 8(b)], we can obtain stable estimates of local structure.

C. Implementation

Next, we explain how we implement the kernel-based deblurring with the weight matrices $\mathbf{W}_z(\mathbf{v})$ and $\mathbf{W}_u(\mathbf{v})$ given by the steering kernel function (33). Note that, using the steering kernel (33), the performance strongly depends on the quality of the orientation information. In [1], we proposed an algorithm to iteratively refine the orientation information, and then re-create the steering matrices $\mathbf{H}_i^s$ by the new orientation estimates. Similarly, for deblurring, we propose an iterative method to refine the weight matrix using the steepest descent method (27). To

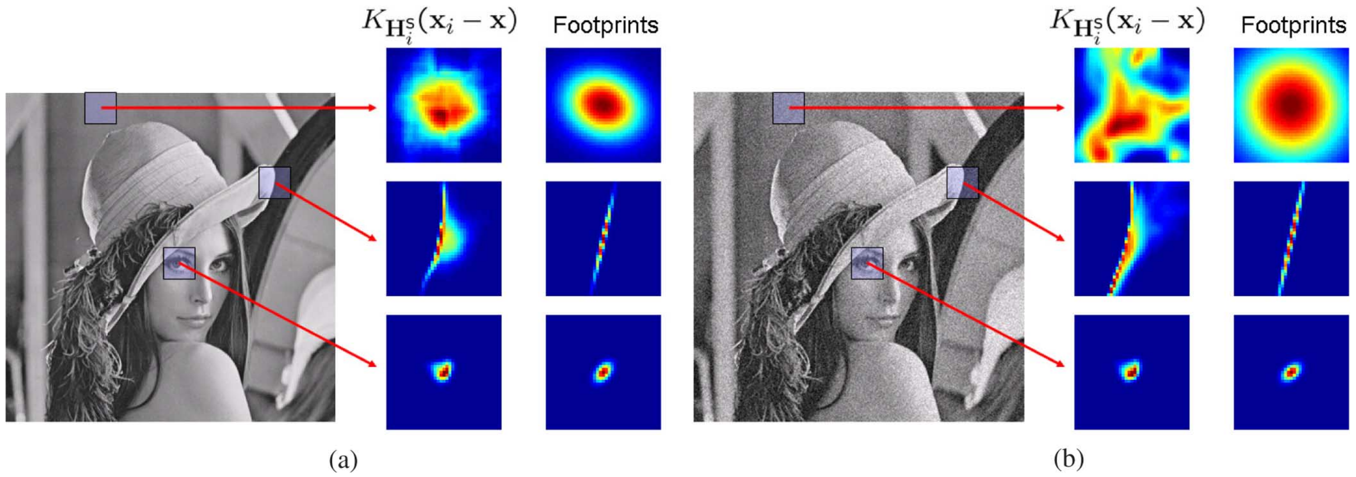
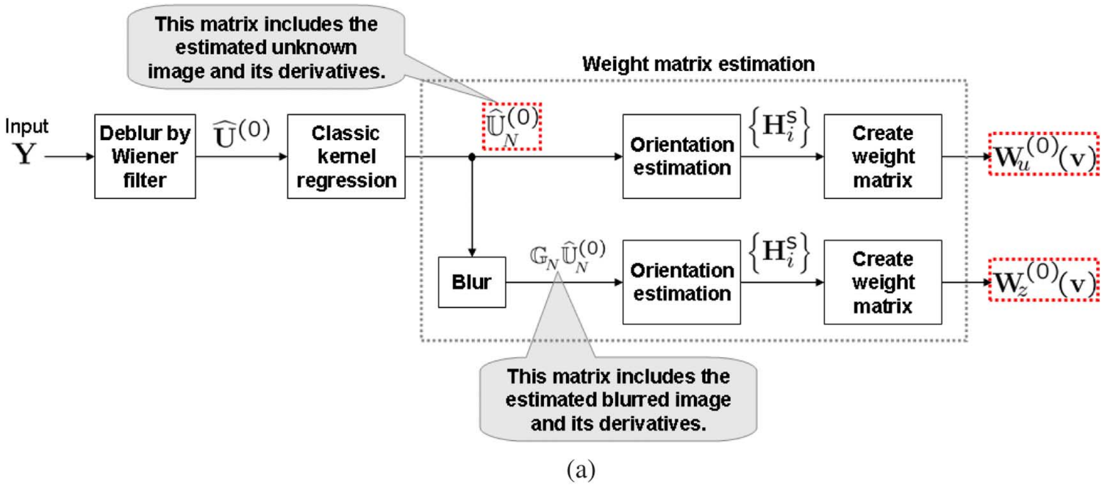


Fig. 8. Steering kernel function and its footprints for (a) high and (b) low SNR cases at flat, edge, and texture areas. We added white Gaussian noise with standard deviation 25 (the corresponding SNR is 5.64 [dB]) for (b). (a) High SNR case. (b) Low SNR case.


This matrix includes the estimated unknown image and its derivatives.
Weight matrix estimation
Input $\mathbf{Y}$
Deblur by Wiener filter
$\widehat{\mathbf{U}}^{(0)}$
Classic kernel regression
$\widehat{\mathbb{U}}_N^{(0)}$
Orientation estimation
$\{\mathbf{H}_i^s\}$
Create weight matrix
$\mathbf{W}_u^{(0)}(\mathbf{v})$
Blur
$\mathbb{G}_N\widehat{\mathbb{U}}_N^{(0)}$
Orientation estimation
$\{\mathbf{H}_i^s\}$
Create weight matrix
$\mathbf{W}_z^{(0)}(\mathbf{v})$
This matrix includes the estimated blurred image and its derivatives.
(a)

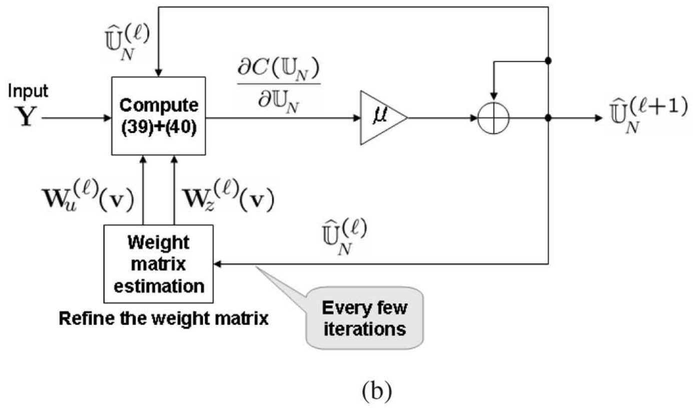

$\widehat{\mathbb{U}}_N^{(\ell)}$
Input $\mathbf{Y}$
Compute (39)+(40)
$\frac{\partial C(\mathbb{U}_N)}{\partial \mathbb{U}_N}$
μ
$\widehat{\mathbb{U}}_N^{(\ell+1)}$
$\mathbf{W}_u^{(\ell)}(\mathbf{v})$
$\mathbf{W}_z^{(\ell)}(\mathbf{v})$
Weight matrix estimation
$\widehat{\mathbb{U}}_N^{(\ell)}$
Refine the weight matrix
Every few iterations
(b)

Fig. 9. Block diagram of the kernel-based deblurring method with the steering weight matrix. (a) Initialization. (b) Iteration.

begin, we first initialize $\widehat{\mathbb{U}}_N$ and the weight matrices $\mathbf{W}_z(\mathbf{v})$ and $\mathbf{W}_u(\mathbf{v})$.[3] To initialize them, we deblur the given image $\mathbf{Y}$ by Wiener filter to have a reasonable estimate of the unknown image $\widehat{\mathbf{U}}^{(0)}$. Next, by applying the second order classic kernel regression ($N = 2$), we have $\widehat{\mathbb{U}}_N^{(0)}$, which contains $\widehat{\mathbf{U}}^{(0)}$ and its first and second derivatives, as defined in (20). Then, using the first derivatives of $\widehat{\mathbf{U}}^{(0)}$, we estimate the rotation, elongation, and scaling parameters, θ_i, σ_i, and γ_i, respectively, and create the steering matrices $\mathbf{H}_i^s$ for every pixel of $\mathbf{Y}$. Finally, we create the weight matrix $\mathbf{W}_u^{(0)}(\mathbf{v})$ in (22) for the regularization term by plugging the steering kernel function (33) with $\mathbf{H}_i^s$'s. As for the likelihood term, we blur $\widehat{\mathbb{U}}_N^{(0)}$ in order to have the estimated blurred image and its first and second derivatives. Following the same procedure, with the first derivatives, we create $\mathbf{W}_z^{(0)}(\mathbf{v})$.

[3]Although we can start with an image whose pixels are all zero, a good initialization greatly reduces the number of iterations.

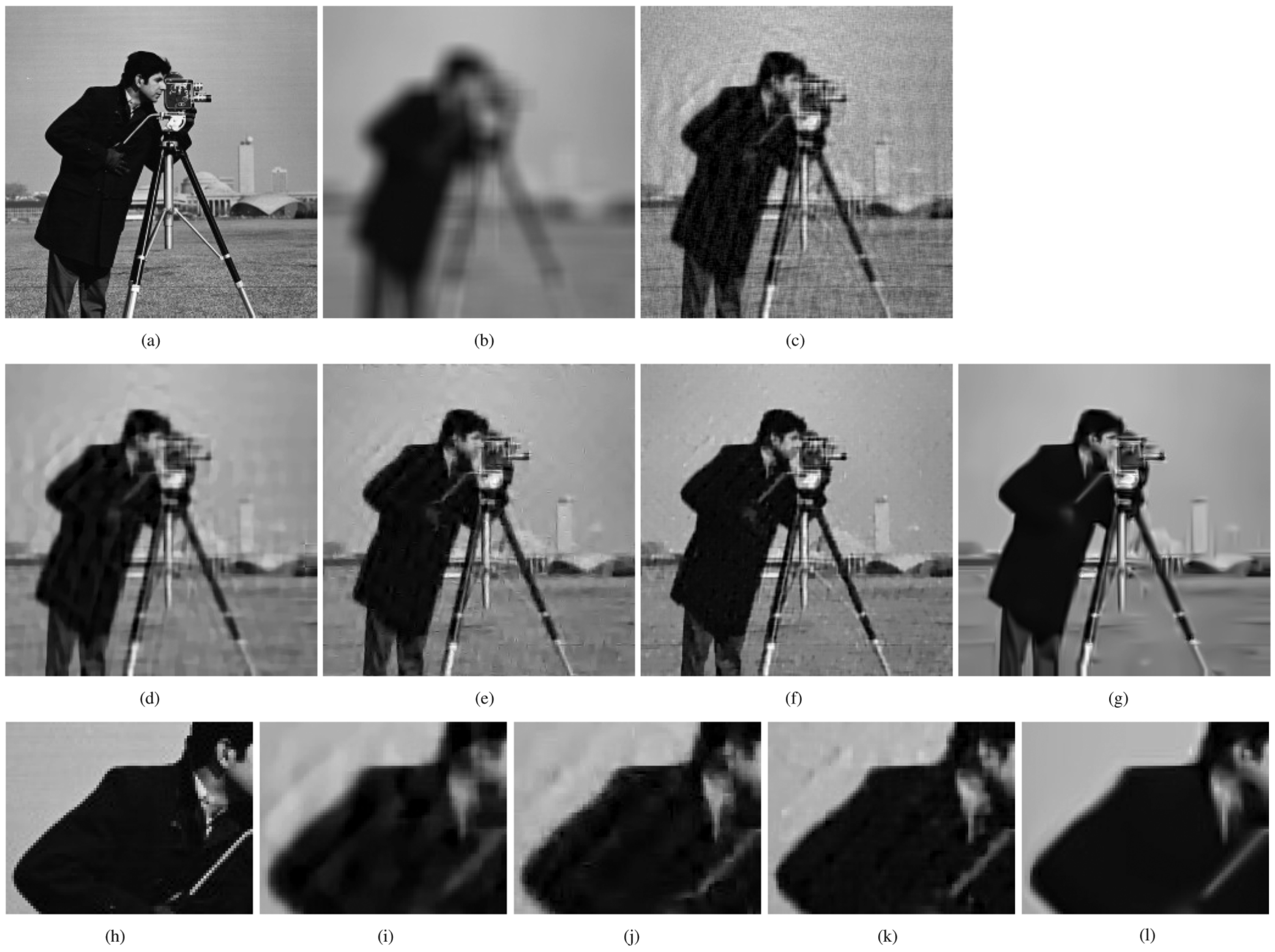


Fig. 10. Single-frame deblurring experiment with the Cameraman image: (a) the original cameraman image, (b) degraded image by blurring with a 19×19 uniform PSF and adding white Gaussian noise ($\mathrm{BSNR} = 40$ [dB]), (c) restored image by Wiener filtering (smoothing parameter 0.0003) [13], (d) restored image by a multistep filter (first denoised by iterative steering kernel regression [1], and deblurred by BTV [10]), (e) restored image by ForWaRD [25], (f) restored image by LPA-ICI [11], and (g) restored image by AKTV ((26) with $q = 2$, $q_r = 1$, $N = 1$, and steering kernels (33)). The corresponding RMSE values for (b)–(g) are 29.67, 17.17, 17.39, 14.37, 13.36, and 14.03, respectively. A selected sections of (a) and (d)–(g) are zoomed in (h)–(l), respectively.

This is the initialization process and Fig. 9(a) illustrates a block diagram of this process. After the initialization, the iteration process begins with $\widehat{\mathbb{U}}_N^{(0)}$, $\mathbf{W}_u^{(0)}(\mathbf{v})$, and $\mathbf{W}_z^{(0)}(\mathbf{v})$. Using (39) and (40) (see Appendix A), we compute the gradient term in (27), and update $\widehat{\mathbb{U}}_N$ and refine the weight matrices ($\mathbf{W}_u^{(0)}(\mathbf{v})$ and $\mathbf{W}_z^{(0)}(\mathbf{v})$) alternately in each iteration.[4] The block diagram of the iteration process is illustrated in Fig. 9(b). As the iteration process continues, the estimated image and its derivatives (and therefore the orientation information) are gradually improved.

IV. Experiments

In this section, we compare the performance of the proposed deblurring algorithm[5] to the state-of-the-art deblurring algorithms in the literature, and demonstrate the competitiveness of our proposed approach with the state of the art.

First, we replicated an experiment from the recent two-step deblurring/denosing technique in [11]. In this experiment, we restored an image which is degraded by application of a severe blur (19×19 uniform PSF) and addition of white Gaussian noise ($\mathrm{BSNR} = 40$ [dB]).[6] Fig. 10(a)–(b) shows the original and the degraded *Cameraman* images. Fig. 10(c)–(g) illustrate the restored images by the Wiener filter [13], a multistep filter (first denoised by iterative steering kernel regression ($N = 2$) [1], and deblurred by BTV [10]), ForWaRD[7] [25], LPA-ICI[8] [11], and AKTV [(26) with $q = 2$, $q_r = 1$, $N = 1$, and steering kernels (33)], respectively. The corresponding RMSE[9] values are (g) 29.67, (h) 17.17, (i) 17.39, (j) 14.37, (k) 13.36, and (l) 14.03. Fig. 10(h)–(l) shows the zoomed images of Fig. 10(a) and (d)–(g), respectively. The smoothing parameters for the

[4]Alternatively, one can consider updating the weight matrices every few iterations.

[5]A MATLAB software package containing all the code used to derive the results reported in this section is available at http://www.soe.ucsc.edu/~htakeda/AKTV.

[6]Blurred signal-to-noise ratio $= 10\log_{10}(\text{blurred signal variance}/\text{noise variance})$[dB].

[7]The software is available at http://www.dsp.rice.edu/software/ward.shtml.

[8]The software is available at http://www.cs.tut.fi/~lasip/.

[9]Root mean square error $= \|\text{True Image} - \text{estimated image}\|_2 = \left\|\underline{\mathbf{U}} - \underline{\widehat{\mathbf{U}}}\right\|_2$.

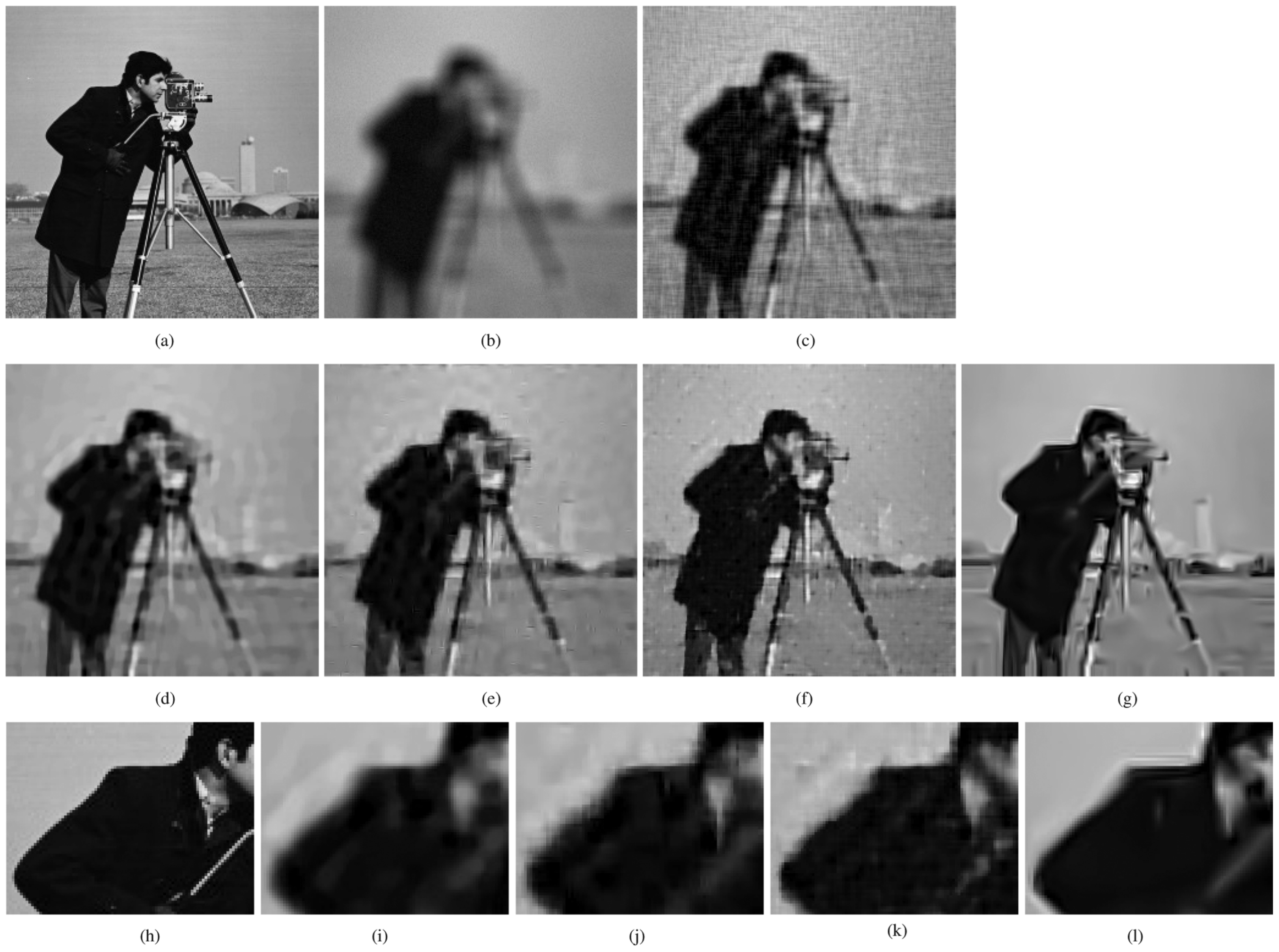


Fig. 11. Single-frame deblurring experiment on the Cameraman image: (a) original cameraman image, (b) degraded image by blurring with a 19×19 uniform PSF and adding white Gaussian noise ($\mathrm{BSNR} = 25$ [dB]), (c) restored image by Wiener filtering (the smoothing parameter 0.004 [13]), (d) restored image by a multistep filter (first denoised by iterative steering kernel regression [1], and deblurred by BTV [10]) (e) restored image by ForWaRD [25], (f) restored image by LPA-ICI [11], and (g) restored image by AKTV ((26) with $q = 2$, $q_r = 1$, $N = 1$, and steering kernels (33)). The corresponding RMSE values for (b)–(g) are 29.82, 21.62, 20.78, 19.44, 18.23, and 17.64, respectively. A selected sections of (a) and (d)–(g) are zoomed in (h)–(l), respectively.

Wiener filter, the multistep filter, LPA-ICI, and AKTV are manually optimized to produce the best RMSE values for each method. The default setup for choosing the parameters was used for the ForWaRD method as suggested by the authors. We see that, in this particular (almost noiseless) experiment, the LPA-ICI algorithm results in an image with slightly better RMSE value than AKTV.

Next, we modified the previous experiment by blurring the *Cameraman* image by the same PSF (19×19 uniform) and adding much more white Gaussian noise to the blurred image resulting in a degraded image of $\mathrm{BSNR} = 25$ [dB]. The resulting image is shown in Fig. 11(b) with the RMSE value of 29.82. Similar to the previous experiment, we restored the *Cameraman* image by the Wiener filter, the multistep filter [1], [10], ForWaRD [25], LPA-ICI [11], and AKTV [(26) with $q = 2$, $q_r = 1$, $N = 1$, and steering kernels (33)]. The resulting images are shown in Fig. 11(c)–(g), respectively, with the RMSE values of (c) 21.62, (d) 20.78, (e) 19.44, (f) 18.23, and (g) 17.64. Fig. 11(h)–(l) shows the zoomed images of Fig. 11(a) and (d)–(g), respectively. Again, we manually tuned the parameters of each method independently to find the restored image with the best RMSE value for each case. In this much noisier example, AKTV resulted in an image with the best RMSE value.

The third experiment is the case of moderate blur and high noise level [5×5 Gaussian PSF with standard deviation (STD) 1.5 and additive white Gaussian noise ($\mathrm{BSNR} = 15$ [dB])]. The original *Lena* image and its degraded version are illustrated in Fig. 12(a)–(b), respectively. Similar to the previous experiments, we have compared the reconstruction performance of the Wiener filter, ForWaRD, LPA-ICI, and AKTV [(26) with $q = 2$, $q_r = 1$, $N = 1$ and steering kernels (33)] in Fig. 12(b)–(e). The corresponding RMSE values are (b) 10.78, (c) 11.18, (d) 7.55, (e) 6.76, and (f) 6.12 Fig. 12(g)–(j) shows the zoomed images of Fig. 12(a) and (d)–(f), respectively.

The last experiment addresses the case of a fair amount of blur and noise level [11×11 Gaussian PSF with $\mathrm{STD} = 1.75$ and additive white Gaussian noise ($\mathrm{BSNR} = 30$ [dB])] and a test image which contains fine details. The original *Chemical Plant* image and its degraded version are illustrated in

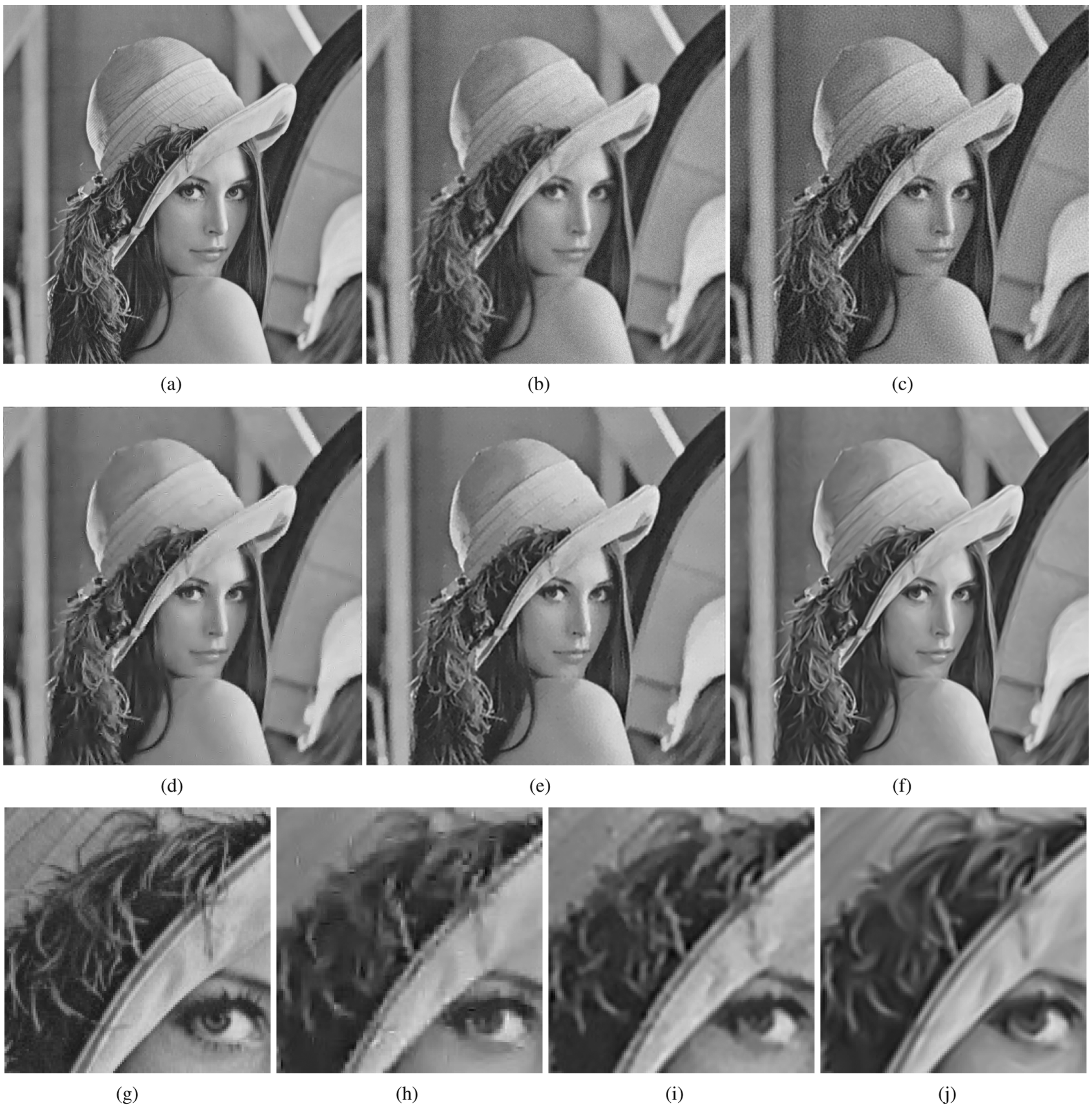


Fig. 12. Single-frame deblurring simulation of the *Lena* image: (a) original Lena image, (b) degraded image by blurring with a 5 × 5 Gaussian PSF $(\mathrm{STD} = 1.5)$ and adding white Gaussian noise $(\mathrm{BSNR} = 15\ [\mathrm{dB}])$, (c) restored image by the Wiener filter method with smoothing parameter of 0.05 [13], (d) restored image by ForWaRD [25], (e) restored image by LPA-ICI [11], and (f) restored image by AKTV ((26) with $q = 2$, $q_r = 1$, $N = 1$, and steering kernels (33)). The corresponding RMSE values for (b)–(f) are 10.78, 11.18, 7.55, 6.76, and 6.12, respectively. A selected sections of (a) and (d)–(f) are zoomed in (g)–(j), respectively.

Fig. 13(a)–(b), respectively. Again, similar to the previous experiments, the restored images by Wiener wilter, ForWaRD, LPA-ICI, and AKTV [(26) with $q = 2$, $q_r = 1$, $N = 1$ and steering kernels (33)] are shown in Fig. 13(c)–(f). Fig. 13(g)–(j) shows the zoomed sections of Fig. 13(a) and (d)–(f), respectively. The corresponding RMSE values are (b) 15.09, (c) 9.29, (d) 8.98, (e) 8.98, and (f) 8.57. In last two experiments, the restored images by AKTV again show the best numerical and visual performances.

V. Conclusion

In this paper, we extended the data-adaptive kernel regression framework of [1], which was earlier used for denoising and interpolating, to include deblurring. Numerical and visual comparison with the state-of-the-art image deblurring techniques showed the superiority of the proposed technique in the low-SNR cases (in which most deblurring algorithms show serious artifacts), while in the high-SNR cases the performance was comparable to the best technique in the literature.

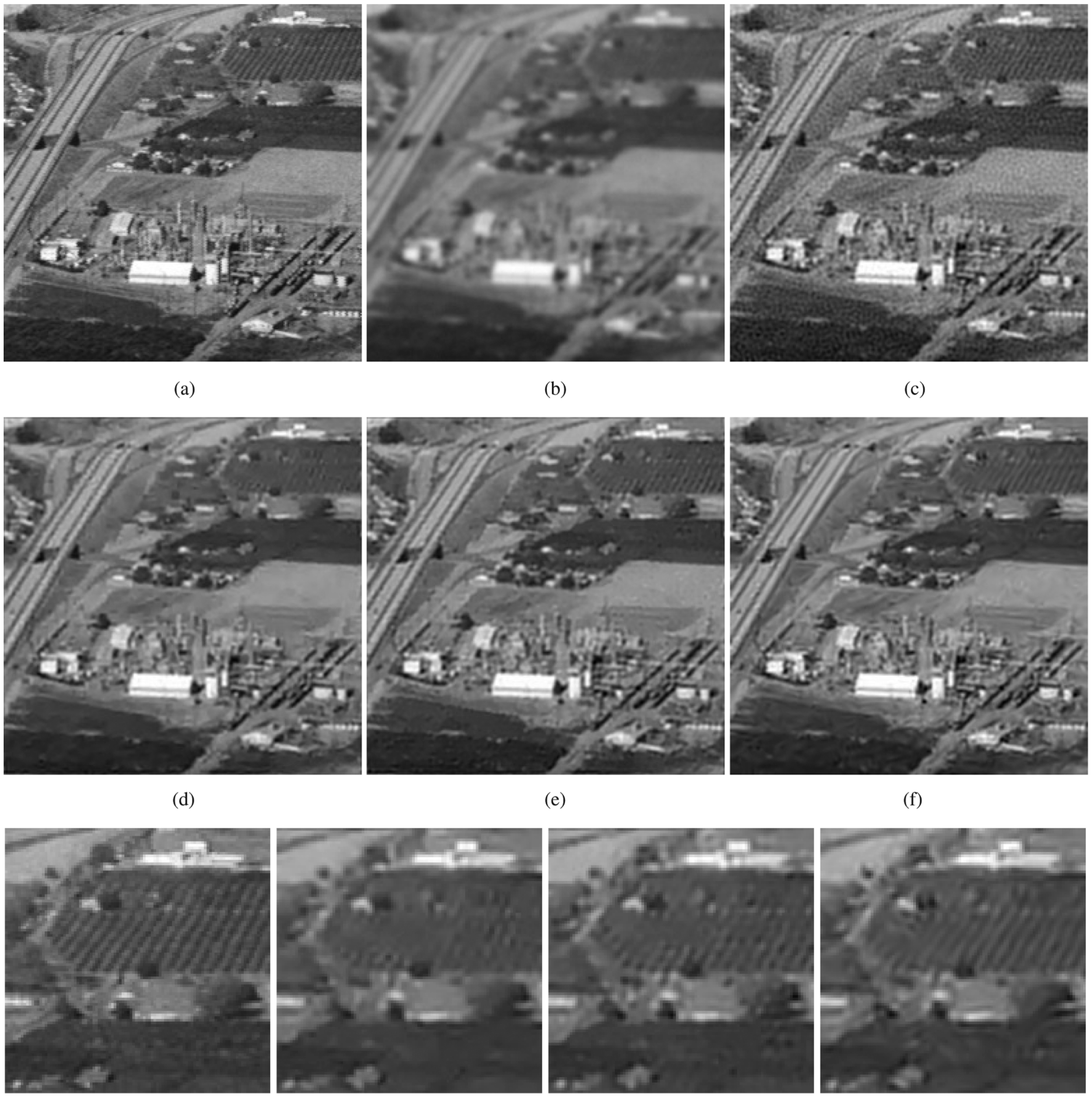


Fig. 13. Single-frame deblurring simulation of the *Chemical Plant* image: (a) original image, (b) degraded image by blurring with a 11×11 Gaussian PSF $(\mathrm{STD} = 1.75)$ and adding white Gaussian noise $(\mathrm{BSNR} = 30\ [\mathrm{dB}])$, (c) restored image by the Wiener filter method with smoothing parameter of 0.01 [13], (d) restored image by ForWaRD [25], (e) restored image by LPA-ICI [11], and (f) restored image by AKTV ((26) with $q = 2, q_r = 1, N = 1$, and steering kernels (33)). The corresponding RMSE values for (b)–(f) are 15.09, 9.29, 8.98, 8.98, and 8.57, respectively. A selected sections of (a) and (d)–(f) are zoomed in (g)–(j), respectively.

Furthermore, the proposed new image prior (21) in its general form subsumes some of the most popular image priors in the literature (Tikhonov, digital TV, BTV). Also, it is possible to create other image priors based on it by choosing local representations other than Taylor series.

Last, we have not provided an obvious advantage in terms of computational complexity in this paper. This issue is one of our important ongoing research problems.

APPENDIX

Steepest Descent: Differentiating the cost function (26), we have

$$\frac{\partial C(\mathbb{U}_N)}{\partial \mathbb{U}_N} = \frac{\partial C_L(\mathbb{U}_N)}{\partial \mathbb{U}_N} + \mu \frac{\partial C_R(\mathbb{U}_N)}{\partial \mathbb{U}_N}. \tag{38}$$

Using (21) and (25), the terms in right hand side are (39) and (40) [see (39) and (40), shown at the top of the next page], where

$$\begin{aligned}\frac{\partial C_L(\mathbb{U}_N)}{\partial \mathbb{U}_N} &= \sum_{v_1}\sum_{v_2}\frac{\partial}{\partial \mathbb{U}_N}\left\|\mathbf{W}_z^{(1/q)}(\mathbf{v})\left(\underline{\mathbf{Y}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{G}_N\mathbb{U}_N\right)\right\|_q^q \\ &= -\sum_{v_1}\sum_{v_2}\mathbb{G}_N^T\mathbf{S}_{x_2}^{v_2}\mathbf{S}_{x_1}^{v_1}\mathbf{W}_z(\mathbf{v}) \\ &\qquad \left\{\mathrm{sign}\left(\underline{\mathbf{Y}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{G}_N\mathbb{U}_N\right)\right. \\ &\qquad\qquad \left.\odot\left|\underline{\mathbf{Y}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{G}_N\mathbb{U}_N\right|^{q-1}\right\}\end{aligned} \tag{39}$$

$$\begin{aligned}\frac{\partial C_R(\mathbb{U}_N)}{\partial \mathbb{U}_N} &= \sum_{v_1}\sum_{v_2}\frac{\partial}{\partial \mathbb{U}_N}\left\|\mathbf{W}_u^{(1/q_r)}(\mathbf{v})\left(\mathbb{E}_0\mathbb{U}_N - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{I}_N\mathbb{U}_N\right)\right\|_{q_r}^{q_r} \\ &= \sum_{v_1}\sum_{v_2}\left(\mathbb{E}_0 - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{I}_N\right)^T\mathbf{W}_u(\mathbf{v}) \\ &\qquad \left\{\mathrm{sign}\left(\underline{\mathbf{U}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{I}_N\mathbb{U}_N\right)\right. \\ &\qquad\qquad \left.\odot\left|\underline{\mathbf{U}} - \mathbf{S}_{x_1}^{-v_1}\mathbf{S}_{x_2}^{-v_2}\mathbb{I}_N\mathbb{U}_N\right|^{q_r-1}\right\}\end{aligned} \tag{40}$$

$\odot$ is the element-by-element product operator for two vectors, and $\mathbb{E}_0$ is a block matrix with identity matrix for the first block and zero matrices for the rest (i.e., $\mathbb{E}_0 = [\mathbf{I} \quad \mathbf{0} \quad \ldots \quad \mathbf{0}]$ and $\underline{\mathbf{U}} = \mathbb{E}_0\mathbb{U}_N$).

References

[1] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 349–366, Feb. 2007.

[2] M. P. Wand and M. C. Jones, *Kernel Smoothing*, ser. Monographs on Statistics and Applied Probability. New York: Chapman & Hall, 1995.

[3] P. Yee and S. Haykin, "Pattern classification as an ill-posed, inverse problem: a regularization approach," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Apr. 1993, vol. 1, pp. 597–600.

[4] H. Knutsson and C.-F. Westin, "Normalized and differential convolution: Methods for interpolation and filtering of incomplete and uncertain data," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, Jun. 16–19, 1993, pp. 515–523.

[5] T. Q. Pham, L. J. v. Vliet, and K. Schutte, "Robust fusion of irregularly sampled data using adaptive normalized convolution," *EURASIP J. Appl. Signal Process.*, 2006, Article ID 83268.

[6] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE Int. Conf. Compute Vision*, Bombay, India, Jan. 1998, pp. 836–846.

[7] M. Elad, "On the origin of the bilateral filter and ways to improve it," *IEEE Trans. Image Process.*, vol. 11, no. 10, pp. 1141–1150, Oct. 2002.

[8] X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE Trans. Image Process.*, vol. 10, no. 10, pp. 1521–1527, Oct. 2001.

[9] N. K. Bose and N. Ahuja, "Superresolution and noise filtering using moving least squares," *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2239–2248, Aug. 2006.

[10] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multiframe super-resolution," *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1327–1344, Oct. 2004.

[11] V. Katkovnik, K. Egiazarian, and J. Astola, "A spatially adaptive nonparametric regression image deblurring," *IEEE Trans. Image Process.*, vol. 14, no. 10, pp. 1469–1478, Oct. 2005.

[12] J. Biemond, R. L. Lagendijk, and R. M. Mersereau, "Iterative methods for image deblurring," *Proc. IEEE*, vol. 78, no. 5, pp. 856–883, May 1990.

[13] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[14] S. Osher and L. I. Rudin, "Feature-oriented image enhancement using shock filters," *SIAM J. Numer. Anal.*, vol. 27, no. 4, pp. 919–940, Aug. 1990.

[15] H. Takeda, S. Farsiu, and P. Milanfar, "Robust kernel regression for restoration and reconstruction of images from sparse, noisy data," in *Proc. Int. Conf. Image Processing*, Atlanta, GA, Oct. 2006, pp. 1257–1260.

[16] E. A. Nadaraya, "On estimating regression," *Theory Probab. Appl.*, pp. 141–142, Sep. 1964.

[17] T. Chan, S. Osher, and J. Shen, "The digital TV filter and nonlinear denoising," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 231–241, Feb. 2001.

[18] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 1, pp. 32–40, Jan. 1975.

[19] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[20] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *SIAM Multiscale Model. Simul.*, vol. 4, no. 2, pp. 490–530, 2005.

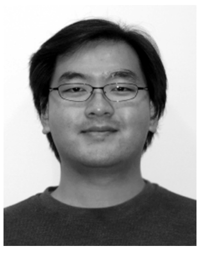
[21] H. Takeda, S. Farsiu, and P. Milanfar, "Higher order bilateral filters and their properties," in *Proc. SPIE Conf. Computational Imaging V*, Mar. 2007, vol. 6498, pp. 64980S-1 to 64980S-9.

[22] X. Feng and P. Milanfar, "Multiscale principal components analysis for image local orientation estimation," presented at the 36th Asilomar Conf. Signals, Syst., Comput., Pacific Grove, CA, Nov. 2002.

[23] R. Mester and M. Hotter, "Robust displacement vector estimation including a statistical error analysis," in *Proc. 5th Inf. Conf. Image Processing and its Applications*, 1995, pp. 168–172.

[24] R. Mester and M. Muhlich, "Improving motion and orientation estimation using an equilibrated total least squares approach," in *Proc. IEEE Int. Conf. Image Process.*, 2001, pp. 929–932.

[25] R. Neelamani, H. Choi, and R. Baraniuk, "Forward: Fourier-wavelet regularized deconvolution for ill-conditioned systems," *IEEE Trans. Signal Process.*, vol. 52, no. 2, pp. 418–433, Feb. 2004.



Hiroyuki Takeda (S'05) received the B.S. degree in electronics from Kinki University, Osaka, Japan, and the M.S. degree in electrical engineering from the University of California, Santa Cruz (UCSC), in 2001 and 2006, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at UCSC.

His research interests are in image and video processing (denoising, interpolation, deblurring, motion estimation, and super-resolution) and inverse problems.

Sina Farsiu (M'05) received the B.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1999, the M.Sc. degree in biomedical engineering from the University of Tehran, Tehran, in 2001, and the Ph.D. degree in electrical engineering from the University of California, Santa Cruz (UCSC), in 2005.

He was a Postdoctoral Scholar at UCSC from January 2006 to February 2007. He is currently with the Department of Ophthalmology, Duke University, Durham, NC, as a Postdoctoral Research Associate. His technical interests include super-resolution, motion estimation, demosaicing, segmentation, adaptive image enhancement, compressive sensing, statistical detection, and imaging through scattering media, spectral domain optical coherence tomography, ophthalmic imaging, adaptive optics, and artificial intelligence.

Peyman Milanfar (SM'98) received the B.S. degree in electrical engineering and mathematics from the University of California, Berkeley, and the S.M. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1988, 1990, and 1993, respectively.

Until 1999, he was a Senior Research Engineer at SRI International, Menlo Park, CA. He is currently a Professor of electrical engineering at the University of California, Santa Cruz. He was a Consulting Assistant Professor of computer science at Stanford University, Stanford, CA, from 1998–2000, and a visiting Associate Professor at Stanford University in 2002. His technical interests are in statistical signal, image, and video processing, and inverse problems.

Dr. Milanfar won a National Science Foundation CAREER award in 2000. He is an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and was an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS from 1998 to 2001. He is a member of the IEEE Image and Multidimensional Signal Processing Technical Committee.