Super-Resolution Without Explicit Subpixel Motion Estimation

Hiroyuki Takeda, *Student Member, IEEE*, Peyman Milanfar, *Senior Member, IEEE*, Matan Protter, and Michael Elad, *Senior Member, IEEE*

***Abstract*—The need for precise (subpixel accuracy) motion estimates in conventional super-resolution has limited its applicability to only video sequences with relatively simple motions such as global translational or affine displacements. In this paper, we introduce a novel framework for adaptive enhancement and spatiotemporal upscaling of videos containing complex activities without explicit need for accurate motion estimation. Our approach is based on multidimensional kernel regression, where each pixel in the video sequence is approximated with a 3-D local (Taylor) series, capturing the essential local behavior of its spatiotemporal neighborhood. The coefficients of this series are estimated by solving a local weighted least-squares problem, where the weights are a function of the 3-D space-time orientation in the neighborhood. As this framework is fundamentally based upon the comparison of neighboring pixels in both space and time, it implicitly contains information about the local motion of the pixels across time, therefore rendering unnecessary an explicit computation of motions of modest size. The proposed approach not only significantly widens the applicability of super-resolution methods to a broad variety of video sequences containing complex motions, but also yields improved overall performance. Using several examples, we illustrate that the developed algorithm has super-resolution capabilities that provide improved optical resolution in the output, while being able to work on general input video with essentially arbitrary motion.**

***Index Terms*—Denoising, frame rate upconversion, interpolation, kernel, local polynomial, nonlinear filter, nonparametric, regression, spatially adaptive, super-resolution.**

I. Introduction

THE emergence of high definition displays in recent years (e.g., 720×1280 and 1080×1920 or higher spatial resolution, and up 240 Hz in temporal resolution), along with the proliferation of increasingly cheaper digital imaging technology has resulted in the need for fundamentally new image processing algorithms. Specifically, in order to display relatively low quality content on such high resolution displays, the need for better space-time upscaling, denoising, and deblurring algorithms has become an urgent market priority, with correspondingly interesting challenges for the academic community. The existing literature on enhancement and upscaling (sometimes called super-resolution[1]) is vast and rapidly growing in both the single frame case [1], [2] and the multiframe (video) case [3]–[11], and many new algorithms for this problem have been proposed recently. Yet, one of the most fundamental roadblocks has not been overcome. In particular, in order to be effective, essentially all the existing multiframe super-resolution approaches must perform (sub-pixel) accurate motion estimation [3]–[12]. As a result, most methods fail to perform well in the presence of complex motions which are quite common. Indeed, in most practical cases where complex motion and occlusions are present and not estimated with pinpoint accuracy, existing algorithms tend to fail catastrophically, often producing outputs that are of even worse visual quality than the low-resolution inputs. Meanwhile, important strides have been made in the motion estimation aspects as well; [13] by Baboulez *et al.* can be cited as one recent example.

In this paper, we address the challenging problem of spatiotemporal video super-resolution in a fundamentally different way, which removes the need for explicit subpixel accuracy motion estimation. We present a methodology that is based on the notion of consistency between the estimated pixels, which is derived from the novel use of kernel regression [14], [15]. Classical kernel regression is a well-studied, nonparametric point estimation procedure. In our earlier work [15], we generalized the use of these techniques to spatially adaptive (steering) kernel regression, which produces results that preserve and restore details with minimal assumptions on local signal and noise models [16]. Other related nonparametric techniques for multidimensional signal processing have emerged in recent years as well. In particular, the concept of normalized convolution [17], and the introduction of support vector machines [18] are notable examples. In the present work, the steering techniques in [15] are extended to 3-D where, as we will demonstrate, we can perform high fidelity space-time upscaling and super-resolution. Most importantly, this is accomplished without the explicit need for accurate motion estimation.

In a related recent work [19], we have generalized the nonlocal means (NLM) framework [20] to the problem of super-resolution. In that work, measuring the similarity of image *patches* across space and time resulted in "fuzzy" or probabilistic estimates of motion. Such estimates also avoided

Manuscript received November 25, 2008; revised May 01, 2009. First published May 26, 2009; current version published August 14, 2009. This work was supported in part by the AFOSR under Grant FA9550-07-1-0365 and in part by the United States—Israel Binational Science Foundation under Grant 2004199. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Pier Luigi Dragotti.

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

M. Protter and M. Elad are with the Department of Computer Science Technion, Israel Institute of Technology, Haifa 32000, Israel (e-mail: matanprcs.technion.ac.il; elad@cs.technion.ac.il).

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.2009.2023703

[1]To clarify the use of the words super-resolution and upscaling, we note that if the algorithm does not receive input frames that are aliased, it will still produce an output with a higher number of pixels and/or frames (i.e., "upscaled"), but which is not necessarily "superresolved."

the need for explicit motion estimation and gave relatively larger weights to more similar patches used in the computation of the high resolution estimate. Another recent example of a related approach appears in [21] where Danielyan *et al.* have presented an extension of the block-matching 3-D filter (BM3D) [22] for video super-resolution, in which the explicit motion estimation is also avoided by classifying the image patches using block matching. The objectives of the present work, our NLM-based approach [19], and Video-BM3D [21] just mentioned are the same: namely, to achieve super-resolution on general sequences, while avoiding explicit (subpixel-accurate) motion estimation. These approaches represent a new generation of super-resolution algorithms that are quite distinctly different from all existing super-resolution methods. More specifically, existing methods have required highly accurate subpixel motion estimation and have thus failed to achieve resolution enhancement on arbitrary sequences.

We propose a framework which encompasses both video denoising, spatiotemporal upscaling, and super-resolution in 3-D. This framework is based on the development of locally adaptive 3-D filters with coefficients depending on the pixels in a local neighborhood of interest in space-time in a novel way. These filter coefficients are computed using a particular measure of similarity and consistency between the neighboring pixels which uses the local geometric and radiometric structure of the neighborhood.

To be more specific, the computation of the filter coefficients is carried out in the following distinct steps. First, the local (spatiotemporal) gradients in the window of interest are used to calculate a covariance matrix, sometimes referred to as the "local structure tensor" [23]. This covariance matrix, which captures a locally dominant orientation, is then used to define a local metric for measuring the similarity between the pixels in the neighborhood. This local metric distance is then inserted into a (Gaussian) kernel which, with proper normalization, then defines the local weights to be applied in the neighborhood.

The above approach is based on the concept of *Steering Kernel Regression* (SKR), earlier introduced in [15] for 2-D signals (images). A specific extension of these concepts to 3-D signals for the express purpose of video denoising and resolution enhancement are the main subjects of this paper. As we shall see, since the development in 3-D involves the computation of orientation in space-time [24], motion information is implicitly and reliably captured. Therefore, unlike conventional approaches to video processing, 3-D SKR does not require explicit estimation of (modestly sized but essentially arbitrarily complex) motions, as this information is implicitly captured within the locally "learned" metric. It is worth mentioning in passing here that the approach we take, while independently derived, is in the same spirit as the body of work known as *Metric Learning* in the machine learning community, e.g., [25].

Naturally, the performance of the proposed approach is closely correlated with the quality of estimated space-time orientations. In the presence of noise, aliasing, and other artifacts, the estimates of orientation may not be initially accurate enough, and as we explain in Section II-D, we, therefore, propose an iterative mechanism for estimating the orientations, which relies on the estimate of the pixels from the previous iteration.

To be more specific, as shown in Fig. 6, we can first process a video sequence with orientation estimates of modest quality. Next, using the output of this first step, we can re-estimate the orientations, and repeat this process several times. As this process continues, the orientation estimates are improved, as is the quality of the output video. It is important to note that the numerical stability of this process has been empirically observed. The overall algorithm we just described will be referred to as the 3-D Iterative Steering Kernel Regression (3-D ISKR).

As we will see in the coming sections, the approach we introduce here is ideally suited for implicitly capturing relatively small motions using the orientation tensors. However, if the motions are somewhat large, the resulting (3-D) local similarity measure, due to its inherent local nature, will fail to find similar pixels in nearby frames. As a result, the 3-D kernels essentially collapse to become 2-D kernels centered around the pixel of interest within the same frame. Correspondingly, the net effect of the algorithm would be to do frame-by-frame 2-D upscaling. For such cases, as discussed in Section II-C, some level of explicit motion estimation is unavoidable to reduce temporal aliasing and achieve resolution enhancement. However, as we will illustrate in this paper, this motion estimation can be quite rough (accurate to within a whole pixel at best). This rough motion estimate can then be used to "neutralize" or "compensate" for the large motion, leaving behind a residual of small motions, which can be implicitly captured within the 3-D orientation kernel. In summary, our approach can accommodate a variety of complex motions in the input videos by a two-tiered approach: (i) large displacements are neutralized by rough motion compensation either globally or block-by-block as appropriate, and (ii) 3-D ISKR handles the fine-scale and detailed rest of the possibly complex motion present.

The contributions of this paper are as follows: 1) We introduce steering kernel regression in space-time as an effective tool for video processing and super-resolution, which does not require explicit, (sub-pixel) accurate motion estimation, 2) we develop the iterative implementation of this algorithm to enhance its performance, and 3) we include the concept of rough motion compensation to widen the range of applicability of the method to sequences with quite general and complex motions.

This paper is structured as follows. In Section II, first we briefly describe the fundamental concepts behind the SKR framework in 2-D and present the extension of the SKR framework to 3-D including discussions of how our method captures local complex motions and performs rough motion compensation, and explicitly describe its iterative implementation. In Section III, we provide some experimental results with both synthetic and real video sequences, and we conclude this paper in Section IV.

II. Space-Time Steering Kernel Regression

In this section, we first review the fundamental framework of *kernel regression* [16] and its extension, the steering kernel regression (SKR) [15], in 2-D. Then, we extend the steering

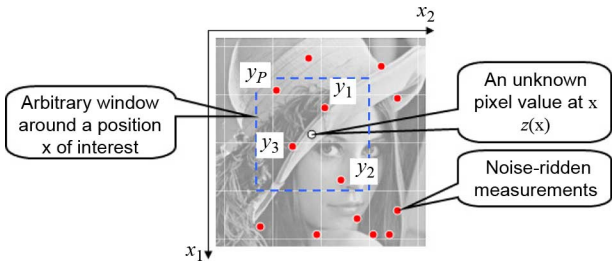


Fig. 1. Data model for the kernel regression framework.

approach to 3-D and discuss some important aspects of the 3-D extension.

A. Review of Steering Kernel Regression in 2-D

1) Classic Kernel Regression: The KR framework defines its data model as

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

where y_i is a noisy sample at $\mathbf{x}_i$ (Note: x_{1i} and x_{2i} are spatial coordinates), $z(\cdot)$ is the (hitherto unspecified) *regression function* to be estimated, ε_i is an i.i.d. zero mean noise, and P is the total number of samples in an arbitrary "window" around a position $\mathbf{x}$ of interest as shown in Fig. 1. 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, we can develop 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

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

where ∇ and $\mathcal{H}$ are the gradient (2×1) and Hessian (2×2) operators, respectively, and $\mathrm{vech}(\cdot)$ is the half-vectorization operator that lexicographically orders the lower triangular portion of a symmetric matrix into a column-stacked vector. Furthermore, β_0 is $z(\mathbf{x})$, which is the signal (or pixel) value of interest, and the vectors $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are

$$\begin{aligned} \boldsymbol{\beta}_1 &= \left[\frac{\partial z(\mathbf{x})}{\partial x_1}, \quad \frac{\partial z(\mathbf{x})}{\partial x_2}\right]^T \\ \boldsymbol{\beta}_2 &= \frac{1}{2}\left[\frac{\partial^2 z(\mathbf{x})}{\partial x_1^2}, \quad 2\frac{\partial^2 z(\mathbf{x})}{\partial x_1 \partial x_2}, \quad \frac{\partial^2 z(\mathbf{x})}{\partial x_2^2}\right]^T. \end{aligned} \tag{3}$$

Since this approach is based on *local* signal representations, a logical step to take is to estimate the parameters $\{\boldsymbol{\beta}_n\}_{n=0}^N$ using all the neighboring samples $\{y_i\}_{i=1}^P$ while giving the nearby samples higher weights than samples farther away. A (weighted) least-square formulation of the fitting problem capturing this idea is

$$\begin{aligned} \min_{\{\boldsymbol{\beta}_n\}_{n=0}^N} \sum_{i=1}^{P} \Big[& y_i - \beta_0 - \boldsymbol{\beta}_1^T (\mathbf{x}_i - \mathbf{x}) \\ & - \boldsymbol{\beta}_2^T \mathrm{vech}\left\{(\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^T\right\} - \cdots \Big]^2 K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}) \end{aligned} \tag{4}$$

with

$$K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}) = \frac{1}{\det(\mathbf{H}_i)} K\left(\mathbf{H}_i^{-1}(\mathbf{x}_i - \mathbf{x})\right) \tag{5}$$

where N is the regression order, $K(\cdot)$ 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 $\mathbf{H}_i = h\mathbf{I}$ for every sample, where h is called the *global smoothing parameter*. The shape of the kernel footprint is perhaps the most important factor in determining the quality of estimated signals. For example, it is desirable to use kernels with large footprints in the smooth local regions to reduce the noise effects, while relatively smaller footprints are suitable in the edge and textured regions to preserve the signal discontinuity. Furthermore, it is desirable to have kernels that adapt themselves to the local structure of the measured signal, providing, for instance, strong filtering along an edge rather than across it. This last point is indeed the motivation behind the *steering* KR framework [15] which we will review in Section II-A2.

Returning to the optimization problem (4), regardless of the regression order and the dimensionality of the regression function, we can rewrite it as the weighted least squares problem

$$\hat{\mathbf{b}} = \arg\min_{\mathbf{b}} \left[(\mathbf{y} - \mathbf{A}\mathbf{b})^T \mathbf{K} (\mathbf{y} - \mathbf{A}\mathbf{b}) \right] \tag{6}$$

where

$$\begin{aligned} \mathbf{y} &= [y_1, y_2, \ldots, y_P]^T \\ \mathbf{b} &= \left[\beta_0, \boldsymbol{\beta}_2^T, \ldots, \boldsymbol{\beta}_N^T\right]^T \end{aligned} \tag{7}$$

$$\begin{aligned} \mathbf{K} = \mathrm{diag}\Big[& K_{\mathbf{H}_1}(\mathbf{x}_1 - \mathbf{x}), K_{\mathbf{H}_2}(\mathbf{x}_2 - \mathbf{x}) \\ & \ldots, K_{\mathbf{H}_P}(\mathbf{x}_P - \mathbf{x}) \Big] \end{aligned} \tag{8}$$

and see equation (9), shown at the bottom of the next page, with "diag" defining a diagonal matrix. Using the notation above, the optimization (4) provides the weighted least square estimator

$$\hat{\mathbf{b}} = \left(\mathbf{A}^T \mathbf{K} \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{K} \mathbf{y} \tag{10}$$

and the estimate of the signal (i.e., pixel) value of interest β_0 is given by a weighted *linear* combination of the nearby samples

$$\begin{aligned} \hat{z}(\mathbf{x}) = \hat{\beta}_0 = \mathbf{e}_1^T \hat{\mathbf{b}} &= \sum_{i=1}^{P} W_i(K, \mathbf{H}_i, N, \mathbf{x}_i - \mathbf{x})\, y_i \\ \sum_{i=1}^{P} W_i(\cdot) &= 1 \end{aligned} \tag{11}$$

where $\mathbf{e}_1$ is a column vector with the first element equal to one and the rest equal to zero, and we call W_i the *equivalent kernel* weight function for y_i (q.v. [15] or [16] for more detail). For example, for zeroth order regresion (i.e., $N = 0$), the estimator (11) becomes

$$\hat{z}(\mathbf{x}) = \hat{\beta}_0 = \frac{\sum_{i=1}^{P} K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x})\, y_i}{\sum_{i=1}^{P} K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x})} \tag{12}$$

which is the so-called *Nadaraya-Watson* estimator (NWE) [26].

What we described above is the "classic" kernel regression framework, which as we just mentioned, yields a pointwise estimator that is always a local *linear* combination of the neighboring samples. As such, it suffers from an inherent limitation. In the next sections, we describe the framework of *steering* KR in two and three dimensions, in which the kernel weights themselves are computed from the local window, and, therefore, we arrive at filters with more complex (nonlinear) action on the data.

2) Steering Kernel Regression: The steering kernel framework is based on the idea of robustly obtaining local signal structures (i.e., discontinuities in 2- and 3-D) by analyzing the radiometric (pixel value) differences locally, and feeding this structure information to the kernel function in order to affect its shape and size.

Consider the (2×2) smoothing matrix $\mathbf{H}_i$ in (5). As explained in Section II-A1, in the generic "classical" case, this matrix is a scalar multiple of the identity with the global scalar parameter h. This results in kernel weights which have equal effect along the x_1- and x_2-directions. However, if we properly choose this matrix, the kernel function can capture local structures. More precisely, we define the smoothing matrix as a symmetric matrix

$$\mathbf{H}_i^{\mathrm{s}} = h\mathbf{C}_i^{-1/2} \tag{13}$$

which we call the *steering* matrix and where, for each given position $\mathbf{x}_i$, the matrix $\mathbf{C}_i$ is estimated as the local covariance matrix of the neighborhood spatial gradient vectors. A naive estimate of this covariance matrix may be obtained by

$$\hat{\mathbf{C}}_i = \mathbf{J}_i^T\mathbf{J}_i \tag{14}$$

with

$$\mathbf{J}_i = \begin{bmatrix} z_{x_1}(\mathbf{x}_1) & z_{x_2}(\mathbf{x}_1) \\ \vdots & \vdots \\ z_{x_1}(\mathbf{x}_P) & z_{x_2}(\mathbf{x}_P) \end{bmatrix} \tag{15}$$

where $z_{x_1}(\cdot)$ and $z_{x_2}(\cdot)$ are the first derivatives along x_1- and x_2-axes, and P is the number of samples in the local analysis window around a sampling position $\mathbf{x}_i$. However, the naive estimate may in general be rank deficient or unstable. Therefore, instead of using the naive estimate, we obtain the covariance matrices by using the (compact) singular value decomposition (SVD) of $\mathbf{J}_i$. A specific choice of $\mathbf{C}_i$ using the SVD for the 2-D case is discussed in [15], and we will show $\mathbf{C}_i$ for the 3-D case in Section II-B.

With the above choice of the smoothing matrix and a Gaussian kernel, we now have the steering kernel function as

$$K_{\mathbf{H}_i^{\mathrm{s}}}(\mathbf{x}_i - \mathbf{x}) = \frac{\sqrt{\det(\mathbf{C}_i)}}{2\pi h^2} \times \exp\left\{-\frac{1}{2h^2}\left\|\mathbf{C}_i^{1/2}(\mathbf{x}_i - \mathbf{x})\right\|_2^2\right\}. \tag{16}$$

Fig. 2 shows visualization of the 2-D steering kernel function for a noise-free and noisy image of Lena (we added white Gaussian noise with standard deviation 25, the corresponding PSNR[2] being 20.16 [dB]). As shown in Fig. 2, the steering kernel weights[3] (which are the normalized $K_{\mathbf{H}_i^{\mathrm{s}}}(\mathbf{x}_i - \mathbf{x})$ 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 (11). We note that even for the highly noisy case, we can obtain stable estimates of local structure.

At this point, the reader may be curious to know how the above formulation would work for the case where we are interested not only in denoising, but also upscaling the images. We discuss this novel aspect of the framework in detail in Section II-D.

B. Space-Time (3-D) Steering Kernel Regression

So far, we presented SKR in 2-D. In this section, we introduce the time axis and present *Space-Time* SKR to process video data. As mentioned in the introductory section, we explain how this extension can yield a remarkable advantage in that space-time SKR does not necessitate explicit (sub-pixel) motion estimation.

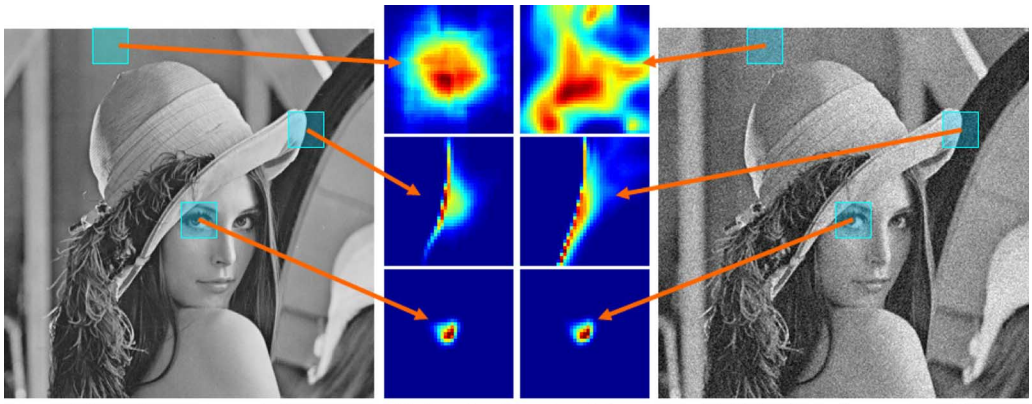
First, introducing the time axis, we have the 3-D data model as

$$y_i = z(\mathbf{x}_i) + \varepsilon_i, \quad i = 1, \ldots, P, \quad \mathbf{x}_i = [x_{1i}, x_{2i}, t_i]^T \tag{17}$$

[2]Peak Signal to Noise Ratio $= 10\log_{10}(255^2/\text{Mean Square Error})$ [dB].

[3]For the sake of completeness, we note that the steering kernel (16) is different from the steerable filter proposed by Freeman and Adelson [27]. One fundamental difference between them is that the contours of the steerable Freeman-Adelson filter are always symmetric as it takes into account the orientation angle at the center position of interest only. On the other hand, in the steering kernel approach, we take the steering matrices of all the nearby samples into account, which results in steering kernels that can become quite asymmetric and capture the local image structures more effectively, as illustrated in Fig. 2.

$$\mathbf{A} = \begin{bmatrix} 1, & (\mathbf{x}_1 - \mathbf{x})^T, & \mathrm{vech}^T\left\{(\mathbf{x}_1 - \mathbf{x})(\mathbf{x}_1 - \mathbf{x})^T\right\}, & \cdots \\ 1, & (\mathbf{x}_2 - \mathbf{x})^T, & \mathrm{vech}^T\left\{(\mathbf{x}_2 - \mathbf{x})(\mathbf{x}_2 - \mathbf{x})^T\right\}, & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ 1, & (\mathbf{x}_P - \mathbf{x})^T, & \mathrm{vech}^T\left\{(\mathbf{x}_P - \mathbf{x})(\mathbf{x}_P - \mathbf{x})^T\right\}, & \cdots \end{bmatrix} \tag{9}$$



(a) Lena image (no noise) (b) Steering kernel weights (c) Noisy Lena image ($\sigma = 25$)

Fig. 2. 2-D steering kernel weights for Lena image without/with noise (white Gaussian noise with standard deviation $\sigma = 25$) at flat, edge, and texture areas.

where y_i is a noisy sample at $\mathbf{x}_i$, x_{1i} and x_{2i} are spatial coordinates, t_i $(= x_{3i})$ is the temporal coordinate, $z(\cdot)$ is the regression function to be estimated, ε_i is an i.i.d. zero-mean noise process, and P is the total number of nearby samples in a 3-D neighborhood of interest, which we will henceforth call a "cubicle." As in (2), we also locally approximate $z(\cdot)$ by a Taylor series in 3-D, where ∇ and $\mathcal{H}$ are now the gradient (3×1) and Hessian (3×3) operators, respectively. With a (3×3) steering matrix $(\mathbf{H}_i^{\mathrm{s}})$, the estimator takes the familiar form

$$\hat{z}(\mathbf{x}) = \hat{\beta}_0 = \sum_{i=1}^{P} W_i(K, \mathbf{H}_i^{\mathrm{s}}, N, \mathbf{x}_i - \mathbf{x})\, y_i. \tag{18}$$

The derivation for the adaptive steering kernel is quite similar to the 2-D case. Indeed, we again define $\mathbf{H}_i^{\mathrm{s}}$ as

$$\mathbf{H}_i^{\mathrm{s}} = h\mathbf{C}_i^{-1/2} \tag{19}$$

where the covariance matrix $\mathbf{C}_i$ can be naively estimated as $\hat{\mathbf{C}}_i = \mathbf{J}_i^T\mathbf{J}_i$ with

$$\mathbf{J}_i = \begin{bmatrix} z_{x_1}(\mathbf{x}_1) & z_{x_2}(\mathbf{x}_1) & z_t(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ z_{x_1}(\mathbf{x}_P) & z_{x_2}(\mathbf{x}_P) & z_t(\mathbf{x}_P) \end{bmatrix} \tag{20}$$

where $z_{x_1}(\cdot)$, $z_{x_2}(\cdot)$, and $z_t(\cdot)$ are the first derivatives along x_1-, x_2-, and t-axes, and P is the total number of samples in a local analysis cubicle around a sample position at $\mathbf{x}_i$. Once again for the sake of robustness, as explained in Section II-A2, we compute a more stable estimate of $\mathbf{C}_i$ by invoking the SVD of $\mathbf{J}_i$ with regularization as

$$\hat{\mathbf{C}}_i = \gamma_i \sum_{q=1}^{3} \varrho_q \mathbf{v}_q \mathbf{v}_q^T, \tag{21}$$

with

$$\varrho_1 = \frac{s_1 + \lambda'}{s_2 s_3 + \lambda'}, \quad \varrho_2 = \frac{s_2 + \lambda'}{s_1 s_3 + \lambda'}$$
$$\varrho_3 = \frac{s_3 + \lambda'}{s_1 s_2 + \lambda'}, \quad \gamma_i = \left(\frac{s_1 s_2 s_3 + \lambda''}{P}\right)^{\alpha} \tag{22}$$

where ϱ_q and γ_i are the *elongation* and *scaling* parameters, respectively, λ' and λ'' are regularization parameters that dampen the noise effect and restrict γ_i and the denominators of ϱ_q's from being zero (q.v. Appendix A for the derivations). We fix $\lambda' = 1$ and $\lambda'' = 0.1$ throughout this paper. The singular values (s_1, s_2, and s_3) and the singular vectors ($\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$) are given by the (compact) SVD of $\mathbf{J}_i$

$$\mathbf{J}_i = \mathbf{U}_i\mathbf{S}_i\mathbf{V}_i^T = \mathbf{U}_i \operatorname{diag}\{s_1, s_2, s_3\} [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]^T. \tag{23}$$

Similar to the 2-D case, the steering kernel function in 3-D is defined as

$$K_{\mathbf{H}_i^{\mathrm{s}}}(\mathbf{x}_i - \mathbf{x}) = \sqrt{\frac{\det(\mathbf{C}_i)}{(2\pi h^2)^3}} \exp\left\{-\frac{1}{2h^2}\left\|\mathbf{C}_i^{1/2}(\mathbf{x}_i - \mathbf{x})\right\|_2^2\right\}$$
$$\mathbf{x} = [x_1, x_2, t]^T. \tag{24}$$

The main tuning parameters are the global smoothing parameter (h) in (19) and the structure sensitivity (α) in (22). The specific choices of these parameters are indicated in Section III, and Appendix B gives more details about h and α.

Fig. 3 shows visualizations of the 3-D weights given by the steering kernel functions for two cases: (a) a horizontal edge moving vertically over time (creating a tilted plane in the local cubicle), and (d) a small circular dot also moving vertically over time (creating a thin tube in the local cubicle). Considering the case of denoising for the sample located at the center of each data cube of Fig. 3(a) and (d), we have the steering kernel weights illustrated in Fig. 3(b) and (c) and Fig. 3(e) and (f). Fig. 3(b) and (e) and Fig. 3(c) and (f) shows the cross sections and the iso-surfaces of the weights, respectively. As seen in these figures, the weights faithfully reflect the local signal structure in space-time.

As illustrated in Fig. 3, the weights provided by the steering kernel function capture the local signal structures which include both spatial and temporal edges. Here we give a brief description of how orientation information thus captured in 3-D contains the motion information implicitly. It is convenient in this respect to use the (gradient-based) optical flow framework [28]–[30] to describe the underlying idea. Defining the 3-D motion vector as $\tilde{\mathbf{m}}_i = [m_1, m_2, 1]^T = [\mathbf{m}_i^T, 1]^T$ and invoking the brightness constancy equation (BCE) in a local "cubicle" centered at $\mathbf{x}_i$, we can use the matrix of gradients $\mathbf{J}_i$ in (20) to write the BCE as

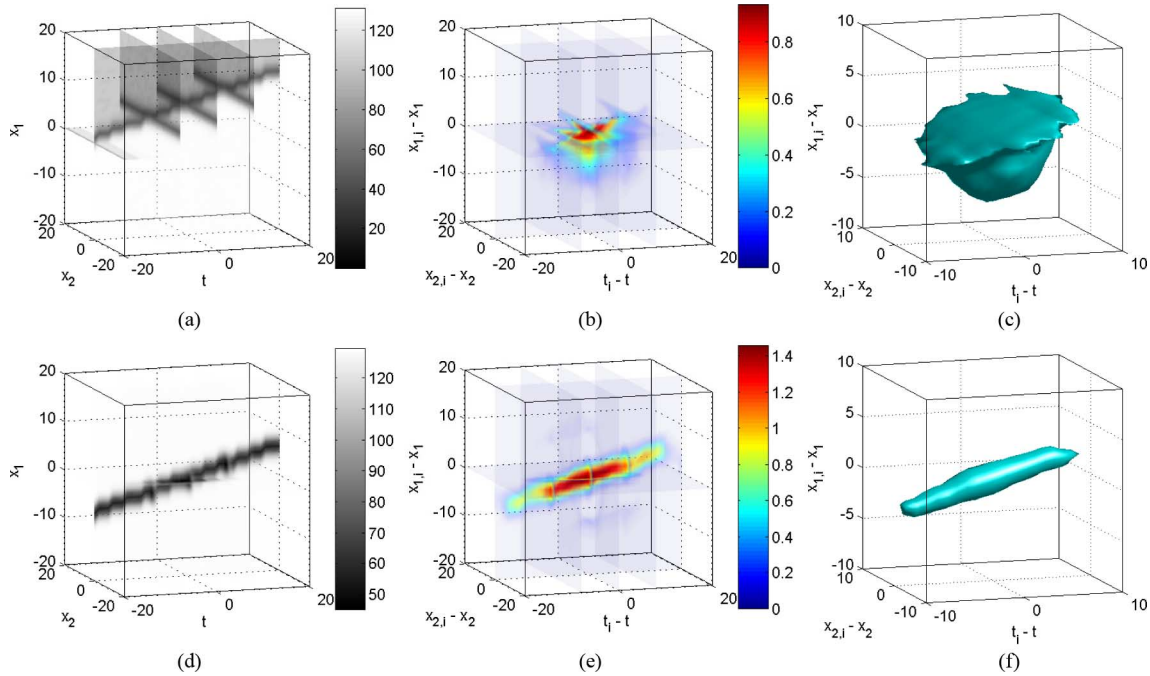


Fig. 3. Steering kernel visualization examples for (a) the case one horizontal edge moving up (this creates a tilted plane in a local cubicle) and (d) the case one small dot moving up (this creates a thin tube in a local cubicle). (a), (d) Cross sections of the 3-D data and (b), (c) cross sections and the isosurface of the weights given by the steering kernel function when we denoise the sample located at the center of the data cube of (a). Similarly, (e) and (f) are the cross sections and the isosurface of the steering kernel weights for denoising the center sample of the data cube of (d).

$$\mathbf{J}_i\tilde{\mathbf{m}}_i = \mathbf{J}_i\begin{bmatrix}\mathbf{m}_i\\1\end{bmatrix} = \underline{\mathbf{0}}. \tag{25}$$

Multiplying both sides of the BCE above by $\mathbf{J}_i^T$, we have

$$\mathbf{J}_i^T\mathbf{J}_i\tilde{\mathbf{m}}_i = \hat{\mathbf{C}}_i\tilde{\mathbf{m}}_i \approx \underline{\mathbf{0}}. \tag{26}$$

Now invoking the decomposition of $\hat{\mathbf{C}}_i$ in (21), we can write

$$\sum_{q=1}^{3}\varrho_q\mathbf{v}_q\left(\mathbf{v}_q^T\tilde{\mathbf{m}}_i\right) \approx \underline{\mathbf{0}}. \tag{27}$$

The above decomposition shows explicitly the relationship between the motion vector and the principal orientation directions computed within the SKR framework. The most generic scenario in a small cubicle is one where the local texture and features move with approximate uniformity. In this generic case, we have $\varrho_1,\ \varrho_2 \gg \varrho_3$, and it can be shown that the singular vector $\mathbf{v}_3$ (which we do not directly use) corresponding to the smallest singular value ϱ_3 can be approximately interpreted as the total least squares estimate of the homogeneous optical flow vector $\tilde{\mathbf{m}}_i/||\tilde{\mathbf{m}}_i||$ [31], [32]. As such, the steering kernel footprint will, therefore, spread along this direction, and consequently assign significantly higher weights to pixels along this implicitly given motion direction. In this sense, compensation for small local motions is taken care of implicitly by the assignment of the kernel weights. It is worth noting that a significant strength of using the proposed implicit framework (as opposed to the direct use of estimated motion vectors for compensation) is the flexibility it provides in terms of smoothly and adaptively changing the elongation parameters defined by the singular values in (22). This flexibility allows the accommodation of even complex motions, so long as their magnitudes are not excessively large. When the magnitude of the motions is large (relative to the support of the steering kernels, specifically,) a basic form of coarse but explicit motion compensation will become necessary. We describe this scenario next.

C. Kernel Regression With Rough Motion Compensation

Before formulating the 3-D SKR with motion compensation, first, let us discuss how the steering kernel behaves in the presence of relatively large motions.[4] In Fig. 4(a) and (b), we illustrate the contours of steering kernels for the pixel of interest marked "×." For the small motion case illustrated in Fig. 4(a), the steering kernel ideally spreads across neighboring frames, taking advantage of information contained in the the space-time neighborhood. Consequently, we can expect to see the effects of resolution enhancement and strong denoising. On the other hand, in the presence of large displacements as illustrated in Fig. 4(b), similar pixels, though close in the time dimension, are found far away in space. As a result, the estimated kernels will tend not to spread across time. That is to say, the net result is that the 3-D SKR estimates in effect default to the 2-D case. However, if we can roughly estimate the relatively large motion of the block and compensate (or "neutralize") for it, as illustrated in Fig. 4(c), and then compute the 3-D steering kernel, we find that it will again spread across neighboring frames and we regain the interpolation/denoising performance of 3-D SKR.

[4]It is important to note here that by large motions we mean speeds (in units of pixels/frame) which are larger than the typical support of the local steering kernel window, or the moving object's width along the motion trajectory. In the latter case, even when the motion speed is slow, we are likely to see temporal aliasing locally.

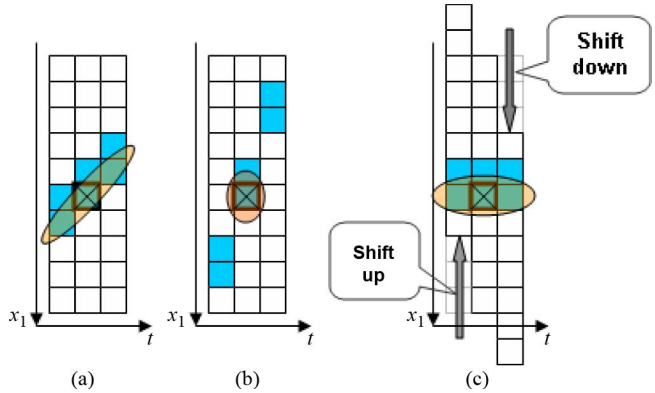


Fig. 4. Steering kernel footprints for (a) a video with small motions, (b) a video with large motions, and (c) alarge-motion neutralized video.

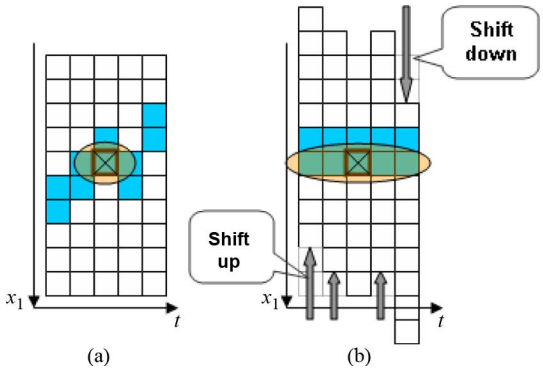


Fig. 5. Steering kernel footprints for (a) a video with a complex motion trajectory and (b) the large-motion neutralized video.

The above approach can be useful even in the absence of aliasing when the motions are small but complex in nature. As illustrated in Fig. 5(b), if we cancel out these displacements, and make the motion trajectory smooth, the estimated steering kernel will again spread across neighboring frames and result in good performance.

In any event, it is quite important to note that the above compensation is done for the sole purpose of computing the more effective steering kernel weights. More specifically, (i) this large motion "neutralization" is *not* an explicit motion compensation in the classical sense invoked in coding or video processing, (ii) it requires absolutely no interpolation and, therefore, introduces no artifacts, and (iii) it requires accuracy no better than a whole pixel.

To be more explicit, 3-D SKR with motion compensation can be regarded as a two-tiered approach to handle a wide variety of transitions in video. Complicated transitions can be split into two different motion components: large whole-pixel motions $(\mathbf{m}_i^{\text{large}})$ and small but complex motion $(\mathbf{m}_i)$

$$\mathbf{m}_i^{\text{true}} = \mathbf{m}_i^{\text{large}} + \mathbf{m}_i \tag{28}$$

where $\mathbf{m}_i^{\text{large}}$ is easily estimated by, for instance, optical flow or block matching algorithms, but, $\mathbf{m}_i$ is much more difficult to estimate precisely.

Suppose a motion vector $\mathbf{m}_i^{\text{large}} = [m_1^{\text{large}}, m_2^{\text{large}}]^T$ is computed for each pixel in the video. We neutralize the motions of the given video data y_i by $\mathbf{m}_i$, to produce a new sequence of data $y(\tilde{\mathbf{x}}_i)$, as follows:

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i + \begin{bmatrix} \mathbf{m}_i^{\text{large}} \\ 0 \end{bmatrix} (t_i - t) \tag{29}$$

where t is the time coordinate of interest. It is important to reiterate that since the motion estimates are rough (accurate to at best a single pixel) the formation of the sequence $y(\tilde{\mathbf{x}}_i)$ does not require any interpolation, and, therefore, no artifacts are introduced. Rewriting the 3-D SKR problem for the new sequence $y(\tilde{\mathbf{x}}_i)$, we have

$$\min_{\{\boldsymbol{\beta}_n\}_{n=0}^{N}} \sum_{i=1}^{P} \Big[y(\tilde{\mathbf{x}}_i) - \beta_0 - \boldsymbol{\beta}_1^T(\tilde{\mathbf{x}}_i - \mathbf{x}) - \boldsymbol{\beta}_2^T \text{vech}\left\{(\tilde{\mathbf{x}}_i - \mathbf{x})(\tilde{\mathbf{x}}_i - \mathbf{x})^T\right\} - \cdots \Big]^2 K_{\tilde{\mathbf{H}}_i}(\tilde{\mathbf{x}}_i - \mathbf{x}). \tag{30}$$

where $\tilde{\mathbf{H}}_i$ is computed from the motion-compensated sequence $y(\tilde{\mathbf{x}}_i)$. Similar to the estimator in 2-D (11), the above minimization yields the following pixel estimator at the position of interest $(\mathbf{x})$ as

$$\begin{aligned} \hat{z}(\mathbf{x}) = \hat{\beta}_0 &= \mathbf{e}_1^T \left(\mathbf{A}^T \mathbf{K} \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{K}\, \tilde{\mathbf{y}} \\ &= \sum_{i=i}^{P} W_i(K, \mathbf{H}_i^s, N, \tilde{\mathbf{x}}_i - \mathbf{x})\, y(\tilde{\mathbf{x}}_i) \end{aligned} \tag{31}$$

where $\tilde{\mathbf{y}}$ is a column-stacked vector of the given pixels $(y(\tilde{\mathbf{x}}_i))$, and $\mathbf{A}$ and $\mathbf{K}$ are the basis matrix and the kernel weight matrix constructed with the motion-compensated coordinates $(\tilde{\mathbf{x}}_i)$; that is to say [see (32)–(34), shown at the bottom of the page]. In the

$$\tilde{\mathbf{y}} = [y(\tilde{\mathbf{x}}_1),\ y(\tilde{\mathbf{x}}_2),\ \ldots,\ y(\tilde{\mathbf{x}}_P)]^T \tag{32}$$

$$\mathbf{K} = \text{diag}\left[K_{\tilde{\mathbf{H}}_1}(\tilde{\mathbf{x}}_1 - \mathbf{x}),\ K_{\tilde{\mathbf{H}}_2}(\tilde{\mathbf{x}}_2 - \mathbf{x}), \ldots,\ K_{\tilde{\mathbf{H}}_P}(\tilde{\mathbf{x}}_P - \mathbf{x})\right] \tag{33}$$

$$\mathbf{A} = \begin{bmatrix} 1, & (\tilde{\mathbf{x}}_1 - \mathbf{x})^T, & \text{vech}^T\left\{(\tilde{\mathbf{x}}_1 - \mathbf{x})(\tilde{\mathbf{x}}_1 - \mathbf{x})^T\right\}, & \cdots \\ 1, & (\tilde{\mathbf{x}}_2 - \mathbf{x})^T, & \text{vech}^T\left\{(\tilde{\mathbf{x}}_2 - \mathbf{x})(\tilde{\mathbf{x}}_2 - \mathbf{x})^T\right\}, & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ 1, & (\tilde{\mathbf{x}}_P - \mathbf{x})^T, & \text{vech}^T\left\{(\tilde{\mathbf{x}}_P - \mathbf{x})(\tilde{\mathbf{x}}_P - \mathbf{x})^T\right\}, & \cdots \end{bmatrix} \tag{34}$$

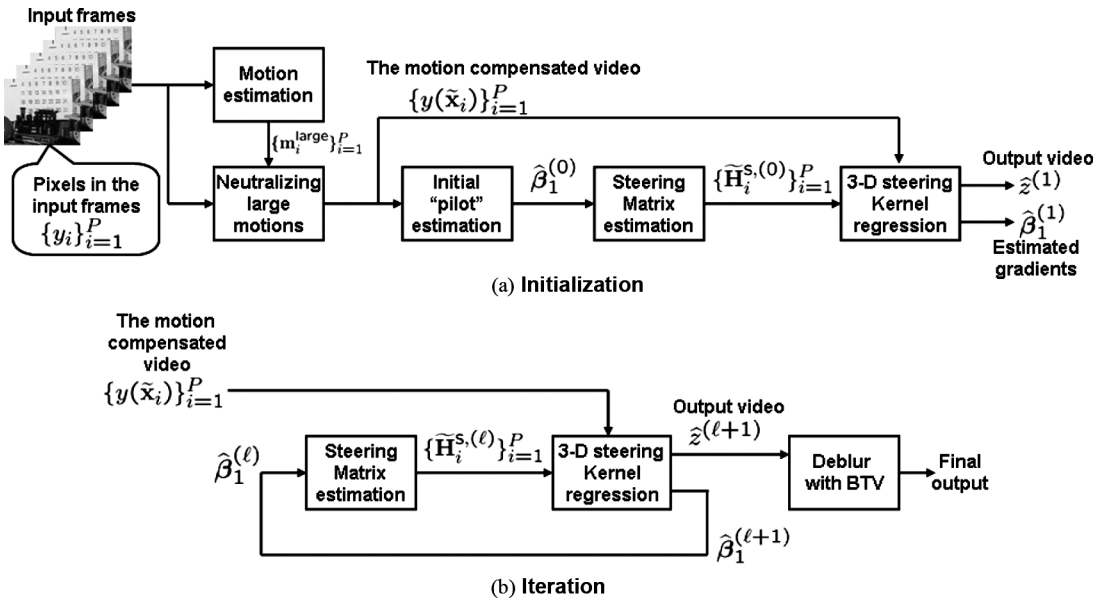


Fig. 6. Block diagram representation of the 3-D iterative steering kernel regression with motion compensation: (a) initialization process and (b) iteration process.

following section, we further elaborate on the implementation of the 3-D SKR for enhancement and super-resolution, including its iterative application.

D. Implementation and Iterative Refinement

As we explained earlier, since the performance of the steering KR (SKR) depends strongly on the accuracy of the orientations, we adopt an scheme (we call this ISKR) which results in improved orientation estimates and, therefore, a better final denoising and upscaling result. The extension for upscaling is done by first interpolating or upscaling using some reasonably effective low-complexity method (say the "classic" KR method) to yield what we call a *pilot* initial estimate. The orientation information is then estimated from this initial estimate and the SKR method is then applied to the input video data y_i which we embed in a higher resolution grid. To be more precise, the basic procedure, as shown in Fig. 6, is as follows.

First we estimate the large motions $(\mathbf{m}_i^{\text{large}})$ of the given input sequence $(\{y_i\}_{i=1}^P)$. Then, using $\mathbf{m}_i^{\text{large}}$, we neutralize the large motions and generate a motion compensated video sequence $(\{y(\tilde{x}_i)\}_{i=1}^P)$. Next, we compute the gradients $(\hat{\boldsymbol{\beta}}_1^{(0)} = [\hat{z}_{x_1}(\cdot), \hat{z}_{x_2}(\cdot), \hat{z}_t(\cdot)]^T)$ at the positions $\{\tilde{x}_i\}_{i=1}^P$ of the motion-compensated data. This process is indicated as the "pilot estimate" in the block diagram. After that, we create steering matrices $(\tilde{\mathbf{H}}_i^{s,(0)})$ for all the samples $y(\tilde{x}_i)$ by (19) and (21). Once $\hat{\mathbf{H}}_i^{s,(0)}$ are available, we plug them into (33) and estimate not only an unknown pixel value $(z(\mathbf{x}))$ at a position of interest $(\mathbf{x})$ by (31) but also its gradients $(\hat{\boldsymbol{\beta}}_1^{(1)})$. This is the initialization process which is shown in Fig. 6(a). Next, using $\hat{\boldsymbol{\beta}}_1^{(1)}$, we re-create the steering matrices $\tilde{\mathbf{H}}_i^{s,(1)}$. Since the estimated gradients $\hat{\boldsymbol{\beta}}_1^{(1)}$ are also denoised and upscaled by SKR, the new steering matrices contain better orientation information. With $\hat{\mathbf{H}}_i^{s,(1)}$, we apply SKR to the embedded input video again. We repeat this procedure several times as shown in Fig. 6(b). While we do not discuss the convergence properties of this approach here, it is worth mentioning that typically, no more than a few iterations are necessary to reach convergence.[5]

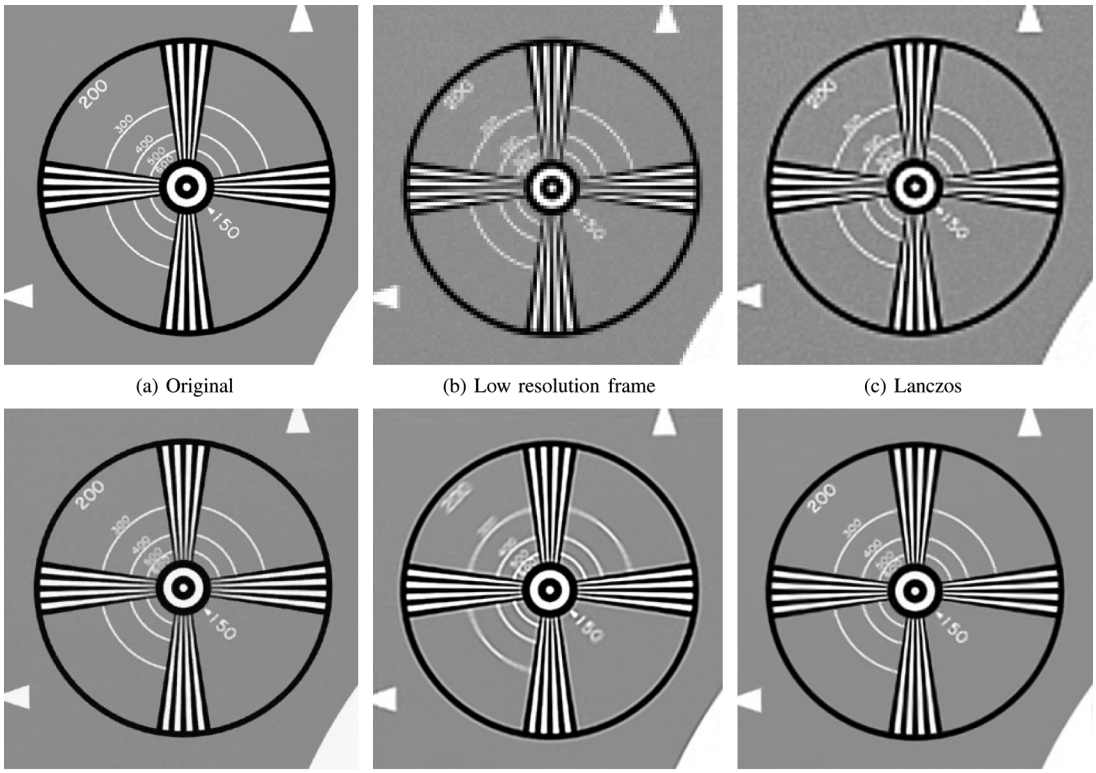
Fig. 7 illustrates a simple super-resolution example. In this example, we created 9 synthetic low resolution frames from the image shown in Fig. 7(a) by blurring with a 3×3 uniform PSF, shifting the blurred image by 0, 4, or 8 pixels[6] along the x_1- and x_2-axes, spatially downsampling with a factor 3:1, and adding White Gaussian noise with standard deviation $\sigma = 2$. One of the low resolution frames is shown in Fig. 7(b). Then, we created a synthetic input video by putting those low resolution images together in *random order*. Thus, the motion trajectory of the input video is not smooth and the 3-D steering kernel weights cannot spread effectively along time as illustrated in Fig. 5(a). The upscaled frames by Lanczos, robust super-resolution [4], nonlocal based super-resolution [19], and 3-D ISKR with rough motion compensation at time $t = 5$ are shown in Fig. 7(c)–(f).

With the presence of severe aliasing arising from large motions, the task of accurate motion estimation becomes significantly harder. However, rough motion estimation and compensation is still possible. Indeed, once this compensation has taken place, the level of aliasing artifacts within the new data cubicle becomes mild, and as a result, the orientation estimation step is able to capture the true space-time orientation (and, therefore, implicitly the motion) quite well. This estimate then leads to the recovery of the missing pixel at the center of the cubicle, from the neighboring compensated pixels, resulting in true super-resolution reconstruction as shown in Fig. 7.

It is worth noting that while in the proposed algorithm in Fig. 6 we employ an SVD-based method for computing the 3-D orientations, other methods can also be employed such as that proposed by Farnebäck *et al.* using local tensors in [34].

[5]It is worth noting that the application of the iterative procedure results in a tradeoff of bias and variance in the resulting final estimate. As for an appropriate number of iterations, a relatively simple stopping criterion can be developed based on the behavior of the residuals (the difference images between the given noisy sequence and the estimated sequence) [33].

[6]Note: this amount of shift creates severe temporal aliasing.



(a) Original (b) Low resolution frame (c) Lanczos

(d) Robust super-resolution [4] (e) Non-local based super-resolution [19] (f) 3-D ISKR

Fig. 7. Simple super-resolution example using 3-D ISKR: (a) the original image, (b) one of 9 low resolution images generated by blurring with a 3×3 uniform PSF, spatially downsampling with a factor of 3:1, and adding white Gaussian noise with standard deviation $\sigma = 2$, (c) an upscaled image by Lancsoz (single frame upsclae), (d) an upscaled image by robust super-resolution [4], and (e) an upscaled image by nonlocal based super-resolution [19]. The corresponding PSNR values are (c) 19.67, (d) 30.21, (e) 27.94, and (f) 29.16 [dB].

Similarly, in our implementation, we used the optical flow [35] framework to compute the rough motion estimates. This step too can be replaced by other methods such as a block matching algorithm [36].

E. Deblurring

Since we did not include the effect of sensor blur in the data model of the KR framework, deblurring is necessary as a postprocessing step to improve the outputs by 3-D ISKR further. Defining the estimated frame at time t as $\underline{\hat{\mathbf{Z}}}(t) = [\ldots, \hat{z}(x_{1j}, x_{2j}, t), \ldots]^T$ where j is the index of the spatial pixel array and $\mathbf{U}(t)$ as the unknown image of interest, we deblur the frame $\underline{\mathbf{Z}}(t)$ by a regularization approach

$$\underline{\hat{\mathbf{U}}}(t) = \arg\min_{\underline{\mathbf{U}}(t)} \left\| \underline{\mathbf{U}}(t) - \mathbf{G}\underline{\hat{\mathbf{Z}}}(t) \right\|_2^2 + \lambda C_R(\underline{\mathbf{U}}(t)) \qquad (35)$$

where $\mathbf{G}$ is the blur matrix, $\lambda\ (\geq 0)$ is the regularization parameter, and $C_R(\cdot)$ is the regularization function. More specifically, we rely on our earlier work and employ the Bilateral Total Variation framework

$$C_R(\underline{\mathbf{U}}(t)) = \sum_{v_1=-w}^{w} \sum_{v_2=-w}^{w} \eta^{|v_1|+|v_2|} \times \left\| \underline{\mathbf{U}}(t) - \mathbf{S}_{x_1}^{v_1} \mathbf{S}_{x_2}^{v_2} \underline{\mathbf{U}}(t) \right\|_1 \qquad (36)$$

where η is the smoothing parameter, w is the window size, and $\mathbf{S}_{x_1}^{v_1}$ is the shift matrix that shifts $\underline{\mathbf{U}}(t)$ by v_1-pixels along the x_1-axis.

In the present work, we use the above BTV regularization framework to deblur the upscaled sequences frame-by-frame, which is admittedly suboptimal. In our very recent work [37], we have introduced a different regularization function called *Adaptive Kernel Total Variation* (AKTV) [15]. This framework can be extended to derive an algorithm which can simultaneously interpolate and deblur in one integrated step. This promising approach is part of our ongoing work and is outside the scope of the this paper.

III. Experimental Results

The utility and novelty of our algorithm lies in the fact that it is capable of both spatial and temporal (and, therefore, spatiotemporal) upscaling and super-resolution. Therefore, in this section, we study the performance of our method in both spatial and spatiotemporal cases.

A. Spatial Upscaling Examples

In this section, we present some denoising/upscaling examples. The sequences in this section contain motions of relatively modest size due to the effect of severe spatial downsampling (we downsampled original videos with the downsampling factor 3:1) and, therefore, motion compensation as we described earlier was not necessary. In Section III-B, we illustrate additional examples of spatiotemporal video upscaling.

Fig. 8. Video upscale example using Miss America sequence: (a) the degraded frames at time $t = 8$ and 13, (b) the upscaled frames by Lanczos interpolation [PSNR: 34.28 (top) and 33.95 (bottom)], (c) the upscaled frames by NL-means based SR [19] [PSNR: 34.67 (top) and 35.34 (bottom)], and (d) the upscaled frames by 3-D ISKR [PSNR: 35.53 (top) and 35.15 (bottom)]. Also, the PSNR values for all the frames are shown in Fig. 10(a).

Fig. 9. Video upscaling example using Foreman sequence: (a) The degraded frames, (b) the upscaled frames by Lanczos interpolation [PSNR: 31.01 (top) and 30.21 (bottom)], (c) the upscaled frames by NL-means based SR [19] [PSNR: 32.13 (top) and 31.94 (bottom)], and (d) the upscaled frames by 3-D ISKR [PSNR: 33.02 (top) and 32.12 (bottom)]. Also, the PSNR values for all the frames are shown in Fig. 10(b).

First, we degrade two videos (Miss America and Foreman sequences), using the first 30 frames of each sequence, blurring with a 3×3 uniform point spread function (PSF), spatially downsampling the videos by a factor of 3:1 in the horizontal and vertical directions, and then adding white Gaussian noise with standard deviation $\sigma = 2$. Two of the selected degraded frames at time $t = 8$ and 13 for Miss America and $t = 6$ and $t = 23$ for Foreman are shown in Figs. 8(a) and 9(a), respectively. Then, we upscale and denoise the degraded videos by Lanczos interpolation (frame-by-frame upscaling), the NL-means based approach of [19], and 3-D ISKR, which includes deblurring[7]

[7]Note that the 3×3 uniform PSF is no longer suitable for the deblurring since the kernel regression gives its own blurring effects.

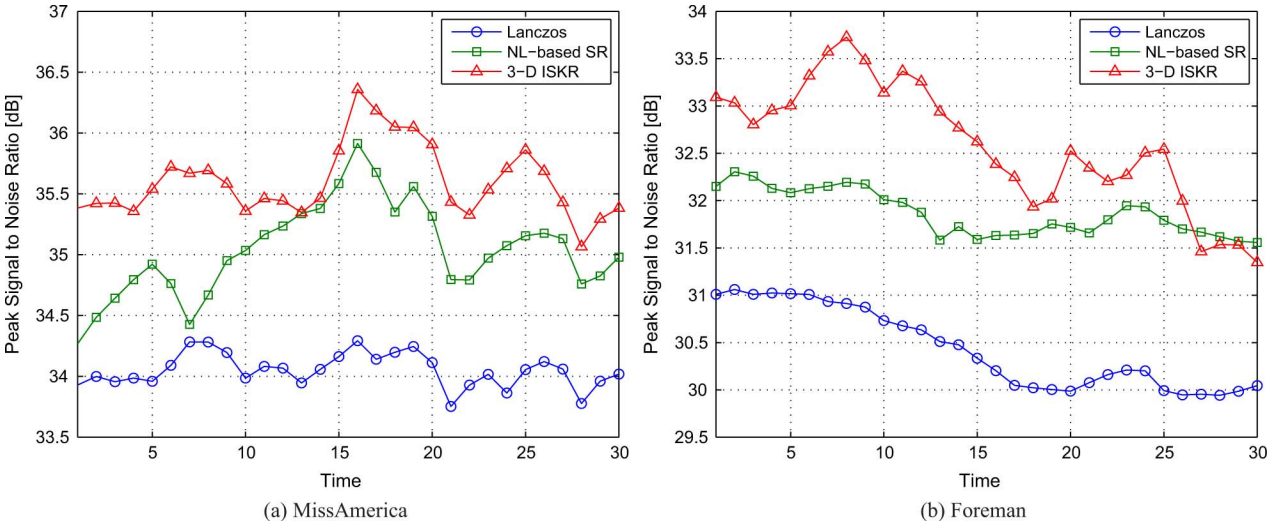


Fig. 10. PSNR values of each upscaled frame by Lanczos, NL-means based SR [19], and 3-D ISKR for (a) the results of Miss America shown in Fig. 8 and (b) the results of Foreman shown in Fig. 9.

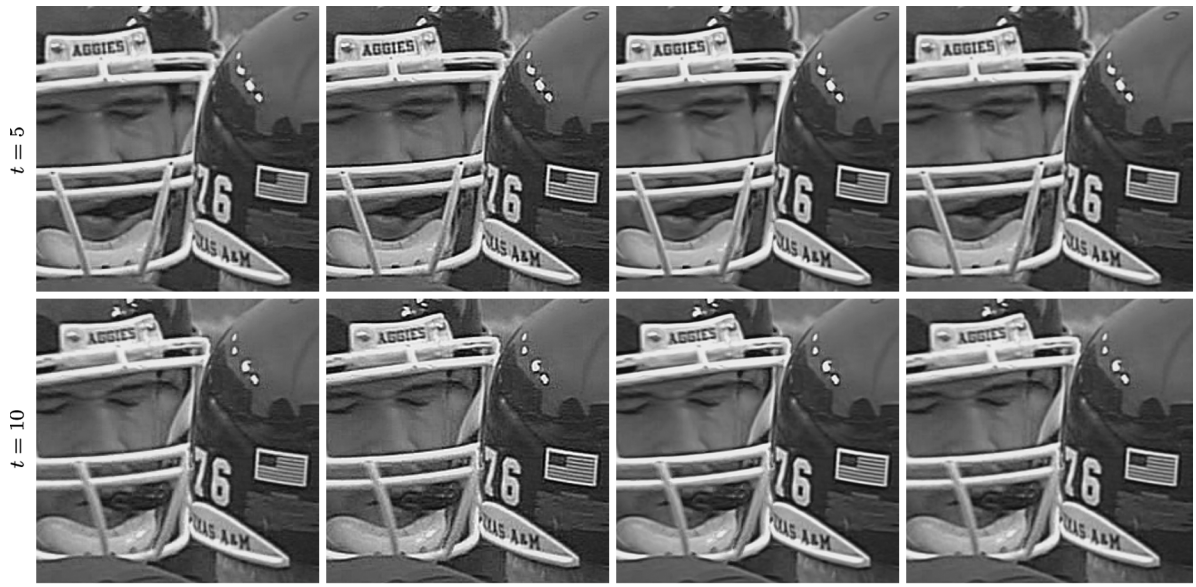
Fig. 11. Vdeo upscaling example using Foreman sequence: (a) the upscaled frames by NL-means based SR [19] [PSNR: 32.77 (top) and 32.06 (bottom)], (b) the upscaled frames by V-BM3D [21] [PSNR: 33.45 (top) and 32.87 (bottom)], and (c) the upscaled frames by 3-D ISKR [PSNR: 33.30 (top) and 33.06 (bottom)].

the upscaled video frames using the BTV approach [4]. Hence, we used a radially symmetric Gaussian PSF which reflects an "average" PSF induced by the kernel function used in the reconstruction process. The final upscaled results are shown in Figs. 8(b)–(d) and 9(b)–(d), respectively. The corresponding average PSNR values across all the frames for the Miss America example are 34.05 [dB] (Lanczos), 35.04 [dB] (NL-means based SR [19]), and 35.60 [dB] (3-D ISKR) and the average PSNR values for Foreman are 30.43 [dB] (Lanczos), 31.87 [dB] (NL-means based SR), and 32.60 [dB] (3-D ISKR), respectively. The graphs in Fig. 10 illustrate the PSNR values frame by frame. It is interesting to note that while the NL-means method appears to produce more crisp results in this case, the corresponding PSNR values for this method are surprisingly lower than that for the proposed 3-D ISKR method. We believe, as partly indicated in Fig. 17, that this may be in part due to some leftover high frequency artifacts and possibly lesser denoising capability of the NL-means method.

As for the parameters of our algorithm, we applied SKR with the global smoothing parameter $h = 1.5$, the local structure sensitivity $\alpha = 0.1$ and a $5 \times 5 \times 5$ local cubicle and used an 11×11 Gaussian PSF with a standard deviation of 1.3 for the deblurring of Miss America and Foreman sequences. For the experiments shown in Figs. 8 and 9, we iterated SKR 6 times.

(a) NL-means based SR [19] (b) Video-BM3D [21] (c) 3-D ISKR

Fig. 12. Enlarged images of the cropped sections from the upscaled Foreman frames ($t = 22$) shown in Fig. 11.



(a) Input (b) Lanczos (c) NL-means based SR [19] (d) 3-D ISKR

Fig. 13. Spatial upscaling example of a real video: (a) Texas football sequence in luminance channel, and (b)–(d) the upscaled frames by Lanczos interpolation, NL-based SR [19] and 3-D ISKR, respectively. The input sequence has 24 frames in total and it is a real HD-TV content which carries compression artifacts, namely block artifacts. We upscale the video with the spatial upscaling factor of 1:3.

The next example involves spatial upscaling of a different segment of the Foreman sequence. As suggested by one of the reviewers, we carried out this example in order to illustrate a comparison to the Video-BM3D method of [21]. Since the code for that method is not currently available publicly, only a limited comparison was possible. Namely, we compared the performance of our algorithm to exactly the same results reported in [21], which were derived as follows: A simulated input sequence (without the addition of noise) was created by blurring with a 3×3 uniform PSF and downsampling with a factor of 3:1. Two upscaled frames by nonlocal means-based approach [19], video-BM3D [21], and the proposed method are shown in Figs. 11 and 12, respectively. Although the results are visually different, they are numerically close as indicated in the figure caption.

The third example is a spatial upscaling example using a section of a real HDTV video sequence (300×300 pixels, 24 frames) without any additional simulated degradation which is shown in Fig. 13. As seen in the input frames, the video has real compression artifacts (i.e., blocking). In this example, we show the deblocking capability of the propose method, and the upscaled results by Lanczos interpolation, NL-based SR [19] and 3-D ISKR with a factor of 1:3 (i.e., the output resolution is 900×900 pixels) are shown in Fig. 13(b)–(d), respectively.

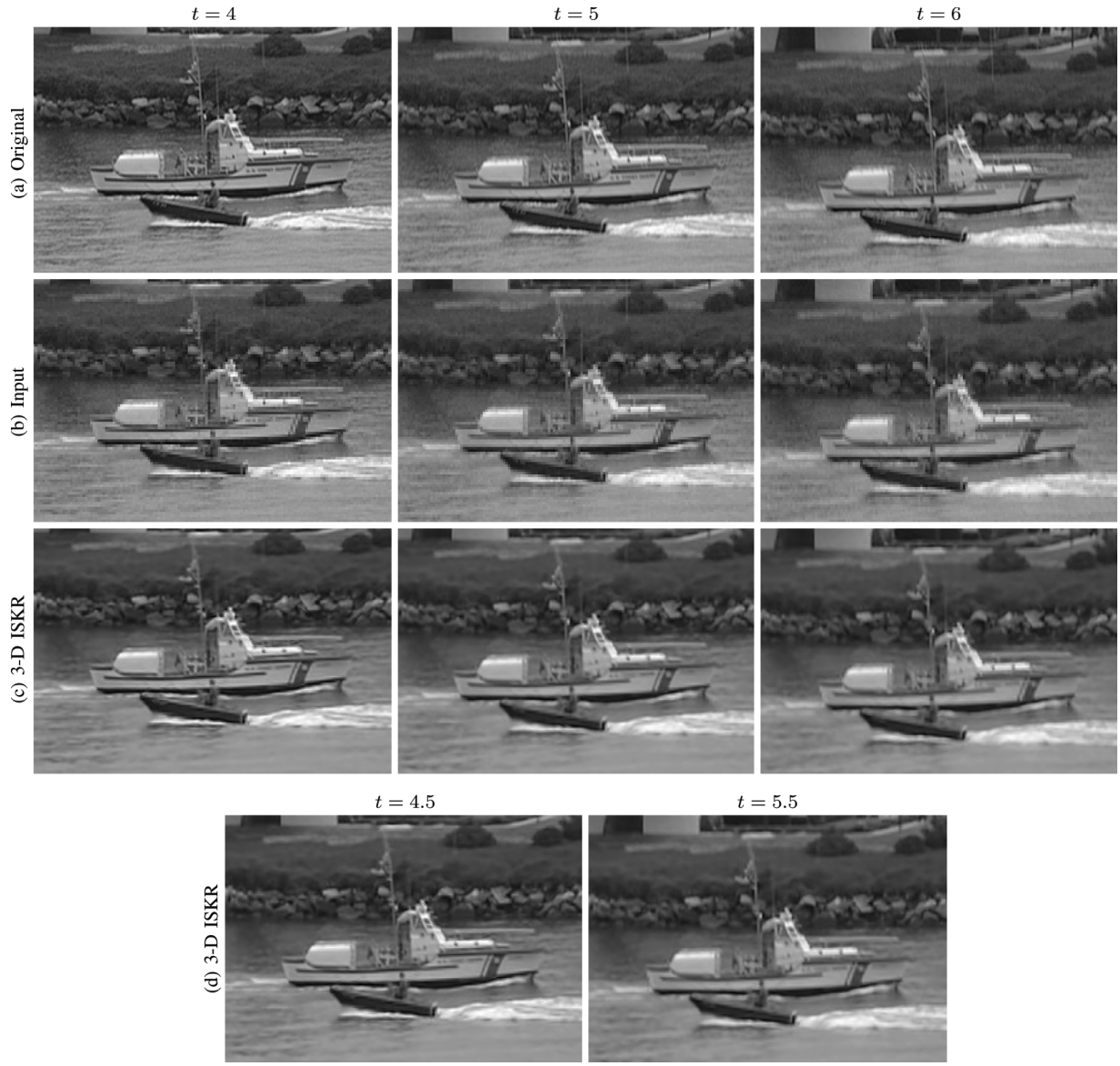


Fig. 14. Coastguard example of spatiotemporal upscaling: from the top row to the bottom, (a) original video frames at time $t = 4$ to 6 (b) the input videos generated by blurred with a 2×2 uniform PSF, adding white Gaussian noise with standard deviation $\sigma = 2$, and spatially downsampling with the factor of $2:1$, (c) upscaled and deblurred frames by 3-D ISKR, and (d) estimated intermediate frames at $t = 4.5$ (left) and $t = 5.5$ (right). The corresponding average PSNR value across all the upscaled frames, except the intermediate frames, by 3-D ISKR with is 29.77 [dB].

Without modification, the proposed method[8] is able to remove the blocking artifacts effectively as well as to upscale the video.

B. Spatiotemporal Upscaling Examples

In this section, we present two video upscaling examples by 3-D ISKR. Unlike the previous examples (Miss America and Foreman), in the next examples, the input videos have relatively large and more complex displacements between frames. In order to have better estimations of steering kernel weights, we estimate patchwise (4×4 block) translational motions by the optical flow technique [35], and apply 3-D ISKR to the roughly motion-compensated inputs.

The first example in Fig. 14 shows (a) cropped original frames from the Coastguard sequence (CIF format, 8 frames), (b) the input video generated by blurring with a 2×2 uniform PSF, spatially downsampling the cropped sequence by a factor of 2:1, and then adding white Gaussian noise with standard deviation $\sigma = 2$, and (c) upscaled and deblurred frames by 3-D ISKR with motion compensation ($h = 1.35$, $\alpha = 0.15$). Similar to the first example, we used the cropped "Stefan" sequence for the next video upscaling example. The results are shown in Fig. 15. The parameters $h = 1.35$ and $\alpha = 0.15$ were used for 3-D ISKR. The corresponding average PSNR value for across the upscaled frames by 3-D ISKR with motion compensation the Coastguard example is 29.77 [dB], and the one for the Stefan example is 23.63 [dB], respectively.

Though we did not discuss temporal upscaling much explicitly in the text of this paper, the presented algorithm is capable of this functionality as well in a very straightforward way. Namely,

[8]We applied our method to the luminance channel only.

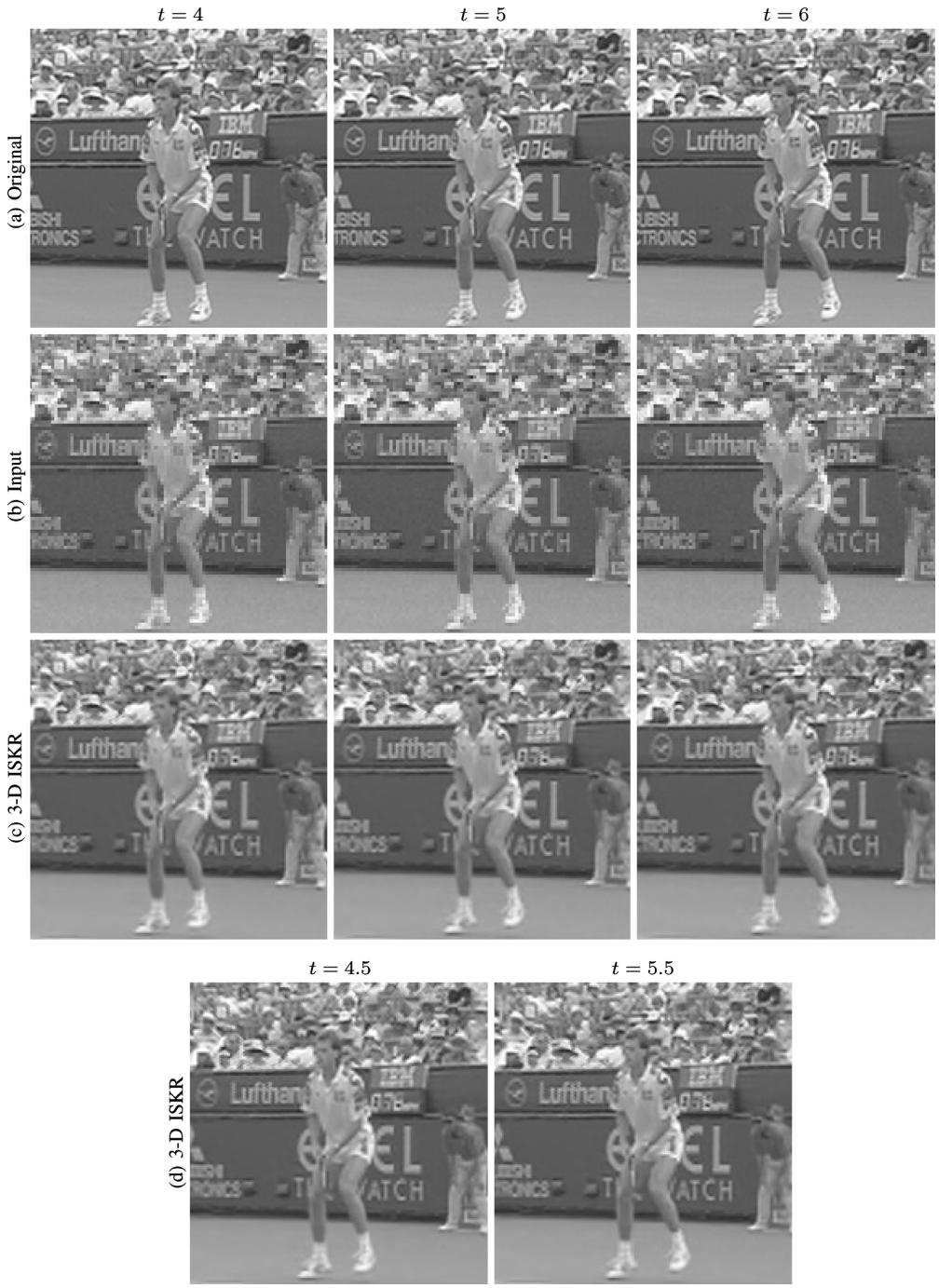


Fig. 15. Stefan example of video upscaling: from the top row to the bottom, (a) original video frames at time $t = 4$ to 6, (b) the input videos generated by blurred with a 2×2 uniform PSF, adding white Gaussian noise with standard deviation $\sigma = 2$, and spatially downsampling with the factor of $2 : 1$, (c) upscaled and deblurred frames by 3-D ISKR, and (d) estimated intermediate frames at $t = 4.5$ (left) and $t = 5.5$. The corresponding average PSNR values across all the upscaled frames, except the intermediate frames, by 3-D ISKR is 23.63 [dB].

the temporal upscaling is effected by producing a pilot estimate and improving the estimate iteratively just as in the spatial upscaling case illustrated in the block diagrams in Fig. 6. We note that this temporal upscaling capability, which essentially comes for free in our present framework, was not possible in the NL-means based algorithm [19]. The examples in Figs. 14(d) and 15(d) show this application of 3-D ISKR, namely simultaneous space-time upscaling, using the same inputs of the Coast-guard and Stefan sequences. Figs. 14(d) and 15(d) illustrate estimated intermediate frames by 3-D ISKR.

The final example in Fig. 16 is a real experiment[9] of space-time upscaling with a native QCIF sequence, Carphone (144×176, 30 frames). Fig. 16 shows (a) the input frame at $t = 25$ to 27 and (b) the upscaled frames by NL-based method

[9]That is to say, the input to the algorithm was the native resolution video, which was subsequently upscaled in space and time directly. In other words, the input video is *not* simulated by downsampling a higher resolution sequence.

Fig. 16. Carphone example of video upscaling: from the top row to the bottom, (a) input video frames at time $t = 25$ to 27 (144×176, 30 frames) and (b) upscaled frames by Lanczos interpolation.

[19], and (c) the upscaled frames by 3-D , and (d) the estimated intermediate frames by 3-D ISKR. Also, Fig. 17 shows the visual comparison between small sections of the upscaled frames at $t = 27$ by (a) Lanczos, (b) NL-based method, and (c) 3-D ISKR, where we can see the visual differences more clearly.

(a) Lanczos (b) NL-means based SR [19] (c) 3-D ISKR

Fig. 17. Enlarged images of the cropped sections from the upscaled Carphone frame ($t = 27$) shown in Fig. 16.

IV. Conclusion

Traditionally, super-resolution reconstruction of image sequences has relied strongly on the availability of highly accurate motion estimates between the frames. As is well-known, subpixel motion estimation is quite difficult, particularly in situations where the motions are complex in nature. As such, this has limited the applicability of many existing upscaling algorithms to simple scenarios. In this paper, we extended the 2-D steering KR method to an iterative 3-D framework, which works well for both (spatiotemporal) video upscaling and denoising applications. Significantly, we illustrated that the need for explicit subpixel motion estimation can be avoided by the two-tiered approach presented in Section II-C, which yields excellent results in both spatial and temporal upscaling.

Performance analysis of super-resolution algorithm remains an interesting area of work, particularly with the new class of algorithms such as the proposed and NL-based method [19] which can avoid subpixel motion estimation. Some results already exist which provide such bounds under certain simplifying conditions [38], but these results need to be expanded and studied further.

Reducing the computational complexity of 3-D ISKR is of great interest, and we are in the process of developing a fast algorithm. Most of the computational load is due to (in order of severity): (i) the computations of steering (covariance) matrices $(\mathbf{C}_i)$ in (19), (ii) the generation of the equivalent kernel coefficients (W_i) in (31) from the steering kernel function with higher order (i.e., $N = 2$), and (iii) iterations. For (i), to speed up the estimation of $\mathbf{C}_i$, instead of application of SVD, which is computationally heavy, we can create a lookup table containing a discrete set of representative steering matrices (using, say, vector quantization), and choose an appropriate matrix from the table given local data. For (ii), computation of the second order $(N = 2)$ filter coefficients (W_i) from the steering kernel weights (24) maybe sped up by using an approximation using the lower order (e.g., zeroth order, $N = 0$) kernels. This idea was originally proposed by Haralick in [24] and may be directly applicable to our case as well. For (iii), we iterate the process of steering kernel regression in order to obtain better estimates of orientations. If the quantization mentioned above gives us fairly reasonable estimates of orientations, we may not need to iterate.

Appendix

A) Steering Kernel Parameters: Using the (compact) SVD (23) of the local gradient vector $\mathbf{J}_i$ (20), we can express the naive estimate of steering matrix as

$$\begin{aligned}\hat{\mathbf{C}}_i^{\text{naive}} &= \mathbf{J}_i^T\mathbf{J}_i = \mathbf{V}_i\mathbf{S}_i^T\mathbf{S}_i\mathbf{V}_i^T \\ &= \mathbf{V}_i \operatorname{diag}\{s_1^2, s_2^2, s_3^2\}\,\mathbf{V}_i^T \\ &= s_1s_2s_3\mathbf{V}_i \operatorname{diag}\left\{\frac{s_1}{s_2s_3}, \frac{s_2}{s_1s_3}, \frac{s_3}{s_1s_2}\right\}\mathbf{V}_i^T \\ &= P\gamma_i[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] \operatorname{diag}\{\varrho_1, \varrho_2, \varrho_3\}[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]^T \\ &= P\,\gamma_i \sum_{q=1}^{3} \varrho_q \mathbf{v}_q\mathbf{v}_q^T \end{aligned} \tag{37}$$

where

$$\begin{aligned}\varrho_1 &= \frac{s_1}{s_2s_3}, \quad \varrho_2 = \frac{s_2}{s_1s_3} \\ \varrho_3 &= \frac{s_3}{s_1s_2}, \quad \gamma_i = \frac{s_1s_2s_3}{P}\end{aligned} \tag{38}$$

and P is the number of rows in $\mathbf{J}_i$. Since the singular values (s_1, s_2, s_3) may become zero, we regularize the elongation parameters (ϱ_q) and the scaling parameter (γ_i) as shown in (22) in order to restrict the parameters from being zero.

B) The Choice of the Regression Parameters: The parameter which have critical roles in steering kernel regression are the regression order (N), the global smoothing parameter (h) in (19) and the structure sensitivity (α) in (22). It is generally

known that the parameters N and h control the balance between the variance and bias of the estimator [39]. The larger N and the smaller h, the higher the variance becomes and the lower the bias. In this paper, we fix the regression order $N = 2$.

The structure sensitivity α (typically $0 \leq \alpha \leq 0.5$) controls how strongly the size of the kernel footprints is affected by the local structure. The product of the singular values (s_1, s_2, and s_3) indicates the amount of the energy of the local signal structure: the larger the product, the stronger and the more complex the local structure is. A large α is preferable when the given signal carries severe noise. In this paper, we focus on the cases where the given video sequences have a moderate amount of noise and fix $\alpha = 0.1$.

Ideally, although one would like to automatically set these regression parameters using a method such as cross validation [40], [41] or SURE (Stein's unbiased risk estimator) [42], this would add significant computational complexity to the already heavy load of the proposed method. So for the examples presented in the paper, we make use of our extensive earlier experience to note that only certain ranges of values for the said parameters tend to give reasonable results. We fix the values of the parameters within these ranges to yield the best results, as discussed in Section III.

ACKNOWLEDGMENT

The authors would like to thank A. Foi for providing video upscaling results from [21].

REFERENCES

[1] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super resolution," *IEEE Comput. Graph.*, vol. 22, no. 2, pp. 56–65, Mar./Apr. 2002.

[2] S. Kanumuri, O. G. Culeryuz, and M. R. Civanlar, "Fast super-resolution reconstructions of mobile video using warped transforms and adaptive thresholding," in *Proc. SPIE*, 2007, vol. 6696, p. 66960T.

[3] M. Elad and Y. Hel-Or, "A fast super-resolution reconstruction algorithm for pure translational motion and common space-invariant blur," *IEEE Trans. Image Process.*, vol. 10, no. 8, pp. 1187–1193, Aug. 2001.

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

[5] H. Fu and J. Barlow, "A regularized structured total least squares algorithm for high-resolution image reconstruction," *Linear Algebra Appl.*, vol. 391, pp. 75–98, Nov. 2004.

[6] B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau, "Multiframe resolution enhancement methods for compressed video," *IEEE Signal Process. Lett.*, vol. 9, no. 2, pp. 170–174, Jun. 2002.

[7] M. Irani and S. Peleg, "Super resolution from image sequence," in *Proc. 10th Int. Conf. Pattern Recognition*, 1990, vol. 2, pp. 115–120.

[8] M. M. J. Koo and N. Bose, "Constrained total least squares computations for high resolution image reconstruction with multisensors," *Int. J. Imag. Syst. Technol.*, vol. 12, pp. 35–42, 2002.

[9] P. Vandewalle, L. Sbaiz, M. Vetterli, and S. Susstrunk, "Super-resolution from highly undersampled images," in *Proc. Int. Conf. Image Processing*, Genova, Italy, Sep. 2005, pp. 889–892.

[10] N. A. Woods, N. P. Galatsanos, and A. K. Katsaggelos, "Stochastic methods for joint registration, restoration, and interpolation of multiple undersampled images," *IEEE Trans. Image Process.*, vol. 15, no. 1, pp. 201–213, Jan. 2006.

[11] A. Zomet, A. Rav-Acha, and S. Peleg, "Robust super-resolution," presented at the Int. Conf. Computer Vision and Pattern Recognition (CVPR), 2001.

[12] S. Park, M. Park, and M. Kang, "Super-resolution image reconstruction: A technical overview," *IEEE Signal Process. Mag.*, vol. 20, no. 3, pp. 21–36, May 2003.

[13] L. Baboulaz and P. L. Dragotti, "Exact feature extraction using finite rate of innovation principles with an application to image super-resolution," *IEEE Trans. Image Process.*, vol. 18, no. 2, pp. 281–298, Feb. 2009.

[14] J. van de Weijer and R. van de Boomgaard, "Least squares and robust estimation of local image structure," in *Proc. Scale Space. Int. Conf.*, 2003, vol. 2695, pp. 237–254.

[15] 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.

[16] M. P. Wand and M. C. Jones, *Kernel Smoothing, ser. Monographs on Statistics and Applied Probability*. London, U.K.: Chapman & Hall, 1995.

[17] 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 Regocnition*, Jun. 1993, pp. 515–523.

[18] K. S. Ni, S. Kumar, N. Vasconcelos, and T. Q. Nguyen, "Single image superresolution based on support vector regression," presented at the IEEE Int. Conf. Acoustics, Speech and Signal Processing, May 2006.

[19] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the non-local-means to super-resolution reconstruction," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 36–51, Jan. 2009.

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

[21] A. Danielyan, A. Foi, V. Katkovnik, and K. Egiazarian, "Image and video super-resolution via spatially adaptive block-matching filtering," presented at the Int. Workshop on Local and Non-Local Approximation in Image Processing, Lausanne, Switzerland, Aug. 2008.

[22] K. Dabov, A. Foi, V. Katkovnic, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[23] H. Knutsson, "Representing local structure using tensors," in *Proc. 6th Scand. Conf. Image Analysis*, 1989, pp. 244–251.

[24] R. M. Haralick, "Edge and region analysis for digital image data," *Comput. Graph. Image Process.*, vol. 12, no. 1, pp. 60–73, Jan. 1980.

[25] K. Q. Weinberger and G. Tesauro, "Metric learning for kernel regression," in *Proc. 11th Int. Workshop on Artificial Intelligence and Statistics*, 2007, pp. 608–615.

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

[27] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 9, pp. 891–906, Sep. 1991.

[28] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Comput. Vis. Image Understand.*, vol. 63, no. 1, pp. 75–104, Jan. 1996.

[29] J. J. Gibson, *The Perception of the Visual World*. Boston, MA: Houghton Mifflin, 1950.

[30] D. N. Lee and H. Kalmus, "The optic flow field: The foundation of vision," *Philosoph. Trans. Roy. Soc. Lond. B, Biol. Sci.*, vol. 290, no. 1038, pp. 169–179, 1980.

[31] J. Wright and R. Pless, "Analysis of persistent motion patterns using the 3d structure tensor," presented at the IEEE Workshop Motion and Video Computing, 2005.

[32] S. Chaudhuri and S. Chatterjee, "Performance analysis of total least squares methods in three-dimensional motion estimation," *IEEE Trans. Robot. Autom.*, vol. 7, pp. 707–714, Oct. 1991.

[33] H. Takeda, H. Seo, and P. Milanfar, "Statistical approaches to quality assessment for image restoration," presented at the Int. Conf. Consumer Electronics, Las Vegas, NV, Jan. 2008.

[34] G. Farnebäck, "Polynomial Expansion for Orientation and Motion Estimation," Ph.D. dissertation, Linköping Univ., Linköping, Sweden, 2002.

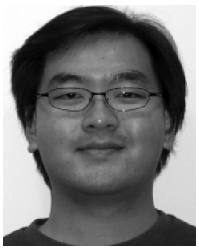
[35] B. Lucas and T. Kanade, "An iterative image registration technique with an application to sterio vision," in *Proc. DARPA Image Understanding Workshop*, 1981, pp. 121–130.

[36] S. Zhu and K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[37] H. Takeda, S. Farsiu, and P. Milanfar, "Deblurring using regularized locally-adaptive kernel regression," *IEEE Trans. Image Process.*, vol. 17, no. 4, pp. 550–563, Apr. 2008.

[38] D. Robinson and P. Milanfar, "Statistical performance analysis of super-resolution," *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1413–1428, Jun. 2006.

[39] D. Ruppert and M. P. Wand, "Multivariate locally weighted least squares regression," *Ann. Statist.*, vol. 22, no. 3, pp. 1346–1370, Sep. 1994.
[40] W. Hardle and P. Vieu, "Kernel regression smoothing of time series," *J. Time Ser. Anal.*, vol. 13, pp. 209–232, 1992.
[41] N. Nguyen, P. Milanfar, and G. H. Golub, "Efficient generalized cross-validation with applications to parametric image restoration and resolution enhancement," *IEEE Trans. Image Process.*, vol. 10, no. 9, pp. 1299–1308, Sep. 2001.
[42] F. Luisier, T. Blu, and M. Unser, "A new sure approach to image denoising: Inter-scale orthonormal wavelet thresholding," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 593–606, Mar. 2007.



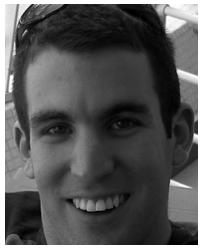
Hiroyuki Takeda (S'05) received the B.S. degree in electronics from Kinki University, Osaka, Japan, and the M.S. degree in electrical engineering from the University of California, Santa Cruz (UCSC), in 2001 and 2006, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at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

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

Peyman Milanfar (SM'98) received the B.S. degree in electrical engineering/mathematics from the University of California, Berkeley, in 1988, and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 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 there in 2002. His technical interests are in statistical signal and image 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 Signal Processing Society's Image, Video, and Multidimensional Signal Processing (IVMSP) Technical Committee.



Matan Protter received the B.Sc. degree in mathematics, physics, and computer sciences from the Hebrew university of Jerusalem, Israel, in 2003. He is currently pursuing the PhD. degree in computer sciences from the Computer Sciences Department, The Technion—Israel Institute of Technology, Haifa.

Since 2003, he has concurrently served in the Israeli Air Force, as a senior R&D engineer. Matan's research interests are in the area of image processing, focusing on example-based image models and their application to various inverse problems.

Michael Elad (SM'08) received the B.Sc. (1986), M.Sc. (supervision by Prof. D. Malah, 1988), and D.Sc. (supervision by Prof. A. Feuer 1997) degrees from the Department of Electrical engineering, The Technion—Israel Institute of Technology, Haifa.

From 1988 to 1993, he served in the Israeli Air Force. From 1997 to 2000, he was with Hewlett-Packard Laboratories as an R&D engineer. From 2000 to 2001, he headed the research division at Jigami corporation, Israel. From 2001 to 2003, he spent a postdoctorate period as a research associate with the Computer Science Department, Stanford University (SCCM Program), Stanford, CA. In September 2003, he returned to The Technion, assuming a tenure-track assistant professorship position in the Department of Computer Science. In May 2007, he was tenured to an associate professorship. He currently works in the field of signal and image processing, specializing, in particular, in inverse problems, sparse representations, and overcomplete transforms.

Prof. Elad received The Technion's best lecturer award five times (1999, 2000, 2004, 2005, and 2006), he is the recipient of the Solomon Simon Mani award for excellence in teaching (2007), and he is also the recipient of the Henri Taub Prize for academic excellence (2008). He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING.