Super-resolution without Explicit Subpixel Motion Estimation

Hiroyuki Takeda*, Peyman Milanfar§, Matan Protter†, Michael Elad‡

EDICS: TEC-ISR

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 spatio-temporal 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 superresolution 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.

*Corresponding author: Electrical Engineering Department, University of California, Santa Cruz CA. 95064 USA. Email: htakeda@soe.ucsc.edu, Phone:(831)-459-4141, Fax: (831)-459-4829

§Electrical Engineering Department, University of California, Santa Cruz CA. 95064 USA. Email: milanfar@soe.ucsc.edu, phone:(831)-459-4929, Fax: (831)-459-4829

†Department of Computer Science Technion, Israel Institute of Technology, Haifa 32000, Israel. Email: matanpr@cs.technion.ac.il

‡Department of Computer Science Technion, Israel Institute of Technology, Haifa 32000, Israel. Email: elad@cs.technion.ac.il

This work was supported in part by AFOSR Grant FA9550-07-1-0365 and the United States – Israel Binational Science Foundation Grant No. 2004199.

Index Terms

kernel function, non-parametric, kernel regression, local polynomial, spatially adaptive, denoising, scaling, interpolation, super-resolution, non-linear filter, bilateral filter, frame rate upconversion.

I. Introduction

The emergence of high definition displays in recent years (e.g. 720×1280 and 1080×1920 or higher spatial resolution, and up $240\texttt{Hz}$ 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 upscaling, denoising, and deblurring algorithms has become an urgent market priority, with correspondingly interesting challenges for the academic community. Although the existing literature on enhancement and upscaling (sometimes called super-resolution) of video is vast and rapidly growing [1], [2], [3], [4], [5], [6], [7], [8], [9], and many new algorithms for this problem have been proposed recently, one of the most fundamental roadblocks has not been overcome. In particular, in order to be effective, all the existing super-resolution approaches must perform (sub-pixel) accurate motion estimation [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. 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.

In this paper, 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 [11], [12]. Classical kernel regression is a well-studied, non-parametric point estimation procedure. In our earlier work [12], 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 [13]. Other related non-parametric techniques for multidimensional signal processing have emerged in recent years as well. In particular, the concept of normalized convolution [14], and the introduction of support vector machines [15] are notable examples.

In the present work, the steering techniques in [12] 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 [16], we have generalized the non-local means (NLM) framework [17] 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 the need for explicit motion estimation and gave relatively larger weights to more similar patches used in the computation of the high resolution estimate. The objectives of the present work and our NLM-based approach 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, spatio-temporal 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 (spatio-temporal) gradients in the window of interest are used to calculate a covariance matrix, sometimes referred to as the "local structure tensor" [18]. 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 [12] for two-dimensional 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 [19], 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. [20].

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 III-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 III-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, we briefly describe the fundamental concepts behind the SKR framework in 2-D. In Section III, we present the extension of the SKR framework to 3-D including discussions of how our method captures local complex motions and performs rough motion

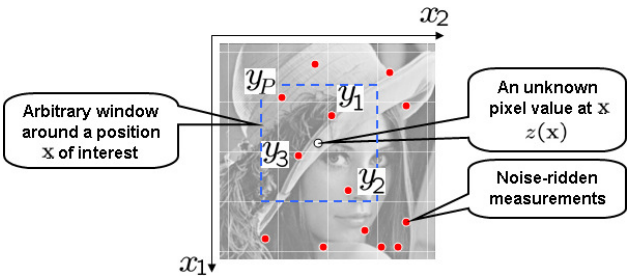


Fig. 1. The data model for the kernel regression framework.

compensation, and explicitly describe its iterative implementation. In Section IV, we provide some experimental results with both synthetic and real video sequences, and we conclude this paper in Section V.

II. Steering Kernel Regression in 2-D

In this section, we first review the fundamental framework of *kernel regression* [13] and its extension, the steering kernel regression (SKR) [12], in 2-D.

A. Kernel Regression

The KR framework defines its data model as

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

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 N-th order Taylor series

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

where ∇ and $\mathcal{H}$ are the gradient (2×1) and Hessian (2×2) operators, respectively, and $\operatorname{vech}(\cdot)$ 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

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

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 [12] which we will review in Section II-B.

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:

$$\widehat{\mathbf{b}} = \arg\min_{\mathbf{b}} \Big[(\mathbf{y} - \mathbf{A}_{\mathbf{x}}\mathbf{b})^T \mathbf{K}_{\mathbf{x}} (\mathbf{y} - \mathbf{A}_{\mathbf{x}}\mathbf{b}) \Big], \tag{6}$$

where

$$\mathbf{y} = [y_1,\ y_2,\ \cdots,\ y_P]^T, \quad \mathbf{b} = \left[\beta_0,\ \boldsymbol{\beta}_2^T,\ \cdots,\ \boldsymbol{\beta}_N^T\right]^T, \tag{7}$$

$$\mathbf{K}_{\mathbf{x}} = \operatorname{diag}\Big[K_{\mathbf{H}_1}(\mathbf{x}_1 - \mathbf{x}),\ K_{\mathbf{H}_2}(\mathbf{x}_2 - \mathbf{x}), \cdots,\ K_{\mathbf{H}_P}(\mathbf{x}_P - \mathbf{x})\Big], \tag{8}$$

and

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

with "diag" defining a diagonal matrix. Using the notation above, the optimization (4) provides the weighted least square estimator

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

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:

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

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. [12] or [13] for more detail). For example, for zero-th 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) [21].

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.

B. 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-A, 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

$$\mathbf{H}_i^{\mathrm{S}} = h\mathbf{C}_i^{-\frac{1}{2}}, \tag{13}$$

which we call the *steering* matrix and where, for each given sample y_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

$$\widehat{\mathbf{C}}_i^{\mathrm{naive}} = \mathbf{J}_i^T\mathbf{J}_i, \tag{14}$$

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 $\mathbf{J}_i$:

$$\mathbf{J}_i = \mathbf{U}_i\mathbf{S}_i\mathbf{V}_i^T, \tag{16}$$

where $\mathbf{S}_i = \mathrm{diag}[s_1, s_2]$, and $\mathbf{V}_i = [\mathbf{v}_1, \mathbf{v}_2]$. The singular vectors contain direct information about the local orientation structure, and the corresponding singular values represent the energy (strength) in these respective orientation directions. Using the singular vectors and values, we compute a more stable estimate of our covariance matrix as:

$$\widehat{\mathbf{C}}_i = \gamma_i \sum_{q=1}^{2} \varrho_q \mathbf{v}_q \mathbf{v}_q^T, \tag{17}$$

where

$$\varrho_1 = \frac{s_1 + \lambda'}{s_2 + \lambda'}, \quad \varrho_2 = \varrho_1^{-1}, \quad \gamma_i = \left(\frac{s_1 s_2 + \lambda''}{P}\right)^{\alpha}. \tag{18}$$

The parameters ϱ_q and γ_i are the *elongation* and *scaling* parameter, respectively, and λ' and λ'' are "regularization" parameters, respectively, which dampen the effect of the noise and restrict γ_i and the denominator of ϱ_q from becoming zero. The parameter α is called the *structure sensitivity*. More details about the effectiveness and the choice of the parameters can be found in Section II-C and in our earlier work [12].

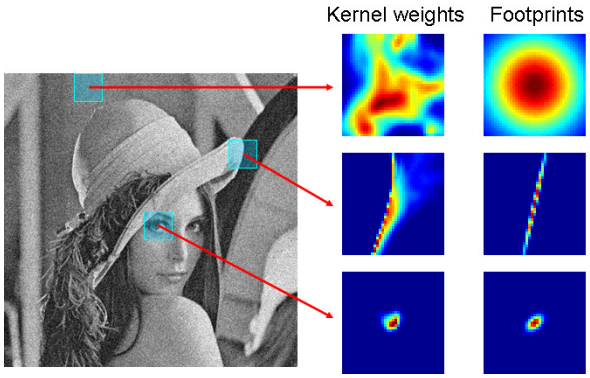


Fig. 2. Steering kernel function and its footprints for a low SNR case at flat, edge, and texture areas. We added white Gaussian noise with standard deviation 25 (the corresponding PSNR is 20.16[dB]).

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

$$K_{\mathbf{H}_i^s}(\mathbf{x}_i - \mathbf{x}) = \frac{\sqrt{\det(\mathbf{C}_i)}}{2\pi h^2} \exp\left\{-\frac{1}{2h^2}\left\|\mathbf{C}_i^{\frac{1}{2}}(\mathbf{x}_i - \mathbf{x})\right\|_2^2\right\}. \tag{19}$$

Fig. 2 shows visualizations of the 2-D steering kernel function for a low PSNR[1] case (we added white Gaussian noise with standard deviation 25, the corresponding PSNR being 20.16[dB]). Note that the *footprints* illustrate the steering kernels $K_{\mathbf{H}_i^s}(\mathbf{x}_i - \mathbf{x})$ as a function of $\mathbf{x}$ when $\mathbf{x}_i$ and $\mathbf{H}_i^s$ are fixed at the center of each window. As explained above and shown in Fig. 2, the steering kernel footprints capture the local image "edge" structure (large in flat areas, elongated in edge areas, and compact in texture areas). Meanwhile, the steering kernel weights (which are the normalized $K_{\mathbf{H}_i^s}(\mathbf{x}_i - \mathbf{x})$ 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 (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 III-D.

C. The Choice of the Regression Parameters

The parameters 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 (18). It is generally known

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

that the parameters N and h control the balance between the variance and bias of the estimator [22]. The larger N and the smaller h, the higher the variance becomes and the lower the bias.

The structure sensitivity α ($0 \leq \alpha \leq 0.5$) controls how strongly the size of the kernel footprints is affected by the local structure. For the denoising problem, in a region where there is a strong edge (a large range in radiometric values), we wish to have smaller kernel footprints in order to preserve the edge. On the other hand, in a flat region, more uniform kernel weights are suitable to remove noise. Since the product of the singular values is an indicator of the strength of local signal and it tends to be large in an edge area and small in a radiometrically "flat" region, with a larger α (e.g. $\alpha = 0.5$), we have the desired kernels and can strengthen the denoising effect. However, for upscaling problems, it is more important to preserve and enhance all the edge structures in the reconstructed higher resolution image (or video.) Therefore, as the denoising effect is less critical, we tend to use smaller values of α. To summarize, we use a larger value of α for denoising, and a relatively smaller value when upscaling.

Ideally, although one would like to automatically set these regression parameters using a method such as cross-validation [23], [24] or SURE (Stein's unbiased risk estimator) [25], 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 pick the values of the parameters within these ranges to yield the best results, as discussed in Section IV.

III. Space-Time Steering Kernel Regression

So far, we presented SKR in 2-D, i.e. for image processing and reconstruction purposes. 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.

A. Steering Kernel Regression in 3-D

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

$$y_i = z(\mathbf{x}_i) + \varepsilon_i, \quad i = 1, \cdots, P, \quad \mathbf{x}_i = [x_{1i}, x_{2i}, t_i]^T, \tag{20}$$

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 $\boldsymbol{\nabla}$ and $\boldsymbol{\mathcal{H}}$ 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:

$$\hat{z}(\mathbf{x}) = \widehat{\boldsymbol{\beta}}_0 = \sum_{i=1}^{P} W_i(K, \mathbf{H}_i^{\mathrm{s}}, N, \mathbf{x}_i - \mathbf{x})\, y_i. \tag{21}$$

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

$$\mathbf{H}_i^{\mathrm{s}} = h\mathbf{C}_i^{-\frac{1}{2}}. \tag{22}$$

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

$$\mathbf{J}_i = \begin{bmatrix} z_{x_1}(\mathbf{x}_1) & z_{x_2}(\mathbf{x}_1) & z_t(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ z_{x_1}(\mathbf{x}_P) & z_{x_2}(\mathbf{x}_P) & z_t(\mathbf{x}_P) \end{bmatrix}, \tag{23}$$

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-B, we compute a more stable estimate of $\mathbf{C}_i$ by invoking the SVD of $\mathbf{J}_i$ with regularization as:

$$\widehat{\mathbf{C}}_i = \gamma_i \sum_{q=1}^{3} \varrho_q \mathbf{v}_q \mathbf{v}_q^T, \tag{24}$$

with

$$\varrho_1 = \frac{s_1 + \lambda'}{\sqrt{s_2 s_3} + \lambda'}, \quad \varrho_2 = \frac{s_2 + \lambda'}{\sqrt{s_1 s_3} + \lambda'},$$
$$\varrho_3 = \frac{s_3 + \lambda'}{\sqrt{s_1 s_2} + \lambda'}, \quad \gamma_i = \left(\frac{s_1 s_2 s_3 + \lambda''}{P}\right)^{\alpha}, \tag{25}$$

where 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. 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 $\mathbf{J}_i$:

$$\mathbf{J}_i = \mathbf{U}_i\mathbf{S}_i\mathbf{V}_i^T = \mathbf{U}_i\ \mathrm{diag}[s_1, s_2, s_3]\ [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]^T. \tag{26}$$

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

$$K_{\mathbf{H}_i^{\mathrm{s}}}(\mathbf{x}_i - \mathbf{x}) = \sqrt{\frac{\det(\mathbf{C}_i)}{(2\pi h^2)^3}}\ \exp\left\{-\frac{1}{2h^2}\left\|\mathbf{C}_i^{\frac{1}{2}}(\mathbf{x}_i - \mathbf{x})\right\|_2^2\right\}, \quad \mathbf{x} = [x_1, x_2, t]^T. \tag{27}$$

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

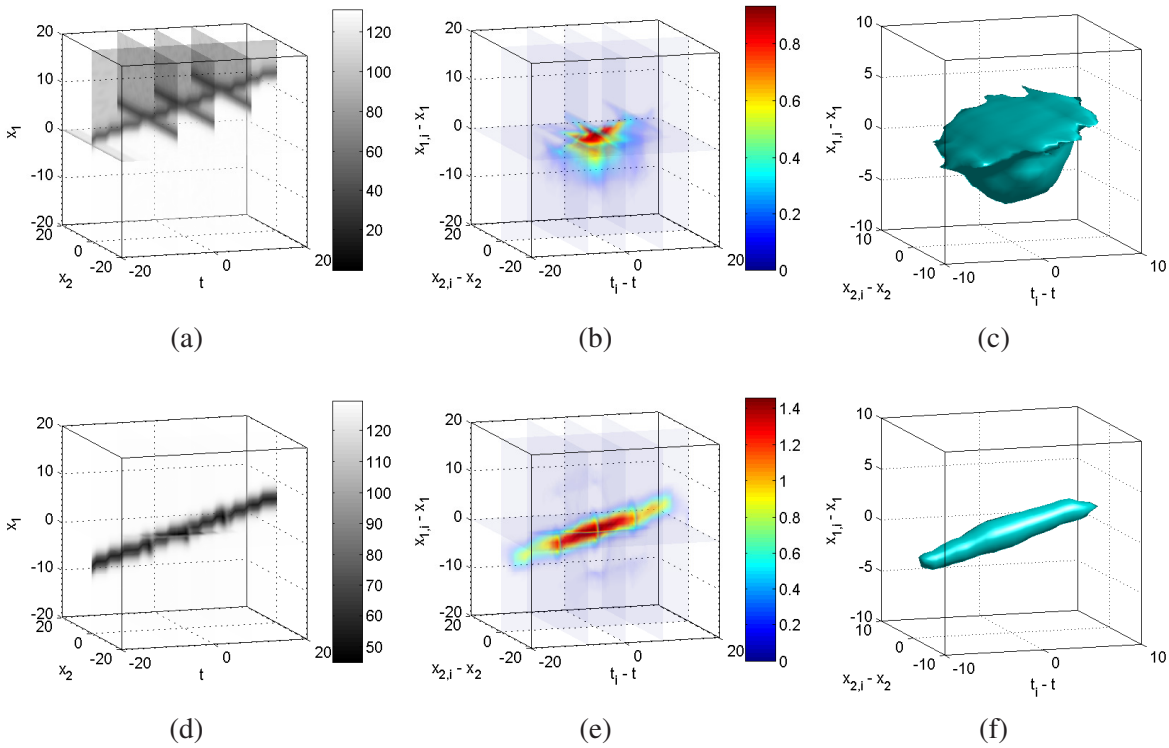


Fig. 3. Steering kernel visualization examples for (a) the case one horizontal edge moving up (this creates a tilted plane in a local cubicle) and (d) the case one small dot moving up (this creates a thin tube in a local cubicle). (a) and (d) show some cross-sections of the 3-D data, and (b)-(c) show the 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)-(f) are the cross-sections and the isosurface of the steering kernel weights for denoising the center sample of the data cube of (d).

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 Figs. 3(a) and (d), we have the steering kernel weights illustrated in Figs. 3(b)(c) and (e)(f). Figs. 3(b)(e) and (c)(f) show 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.

B. Implicit Motion Estimation

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 [26], [27], [28] to describe the underlying

idea. Defining the 3-D motion vector as $\widetilde{\mathbf{m}}_i = [m_1, m_2, 1]^T = [\mathbf{m}_i^T, 1]^T$ 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 (23) to write the BCE as

$$\mathbf{J}_i \widetilde{\mathbf{m}}_i = \mathbf{J}_i \begin{bmatrix} \mathbf{m}_i \\ 1 \end{bmatrix} = \underline{\mathbf{0}}. \tag{28}$$

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

$$\mathbf{J}_i^T \mathbf{J}_i \widetilde{\mathbf{m}}_i = \widehat{\mathbf{C}}_i^{\text{naive}} \widetilde{\mathbf{m}}_i \approx \underline{\mathbf{0}} \tag{29}$$

Now invoking the decomposition of $\widehat{\mathbf{C}}_i$ in (24) we can write

$$\sum_{q=1}^{3} \varrho_q \mathbf{v}_q \left(\mathbf{v}_q^T \widetilde{\mathbf{m}}_i \right) \approx \underline{\mathbf{0}}. \tag{30}$$

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 $\frac{\widetilde{\mathbf{m}}_i}{\|\widetilde{\mathbf{m}}_i\|}$ [29], [30]. 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 (25). 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[2]. In Figs. 4(a) and (b), we illustrate the

[2]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.

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. 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 ($\mathbf{m}_i$):

$$\mathbf{m}_i^{\text{true}} = \mathbf{m}_i^{\text{large}} + \mathbf{m}_i, \tag{31}$$

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 $y(\widetilde{\mathbf{x}}_i)$, as follows:

$$\widetilde{\mathbf{x}}_i = \mathbf{x}_i + \begin{bmatrix} \mathbf{m}_i^{\text{large}} \\ 0 \end{bmatrix} (t_i - t), \tag{32}$$

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 $y(\widetilde{\mathbf{x}}_i)$ does not require any

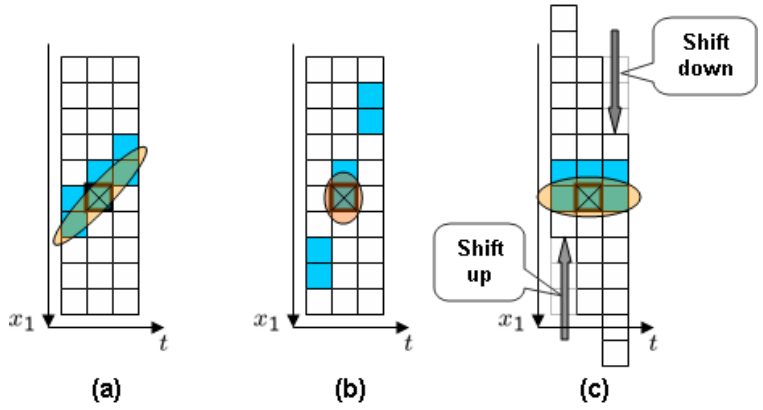


Fig. 4. Steering kernel footprints for (a) a video with small motions, (b) a video with large motions, (c) a motion neutralized video.

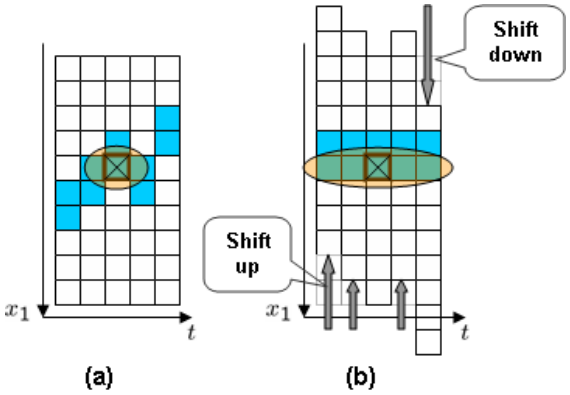


Fig. 5. Steering kernel footprints for (a) a video with a complex motion trajectory, and (b) a motion neutralized video.

interpolation, and therefore no artifacts are introduced. Rewriting the 3-D SKR problem for the new sequence $y(\widetilde{\mathbf{x}}_i)$, we have:

$$\min_{\{\boldsymbol{\beta}_n\}_{n=0}^{N}} \sum_{i=1}^{P} \left[y(\widetilde{\mathbf{x}}_i) - \beta_0 - \boldsymbol{\beta}_1^T(\widetilde{\mathbf{x}}_i - \mathbf{x}) - \boldsymbol{\beta}_2^T \operatorname{vech}\left\{(\widetilde{\mathbf{x}}_i - \mathbf{x})(\widetilde{\mathbf{x}}_i - \mathbf{x})^T\right\} - \cdots \right]^2 K_{\widetilde{\mathbf{H}}_i}(\widetilde{\mathbf{x}}_i - \mathbf{x}). \quad (33)$$

where $\widetilde{\mathbf{H}}_i$ is computed from the motion-compensated sequence $y(\widetilde{\mathbf{x}}_i)$.

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

D. Iterative Refinement and Super-Resolution

As we explained earlier, since the performance of the steering KR (SKR) depends strongly on the accuracy of the orientations, we adopt an iterative scheme which results in improved orientation estimates

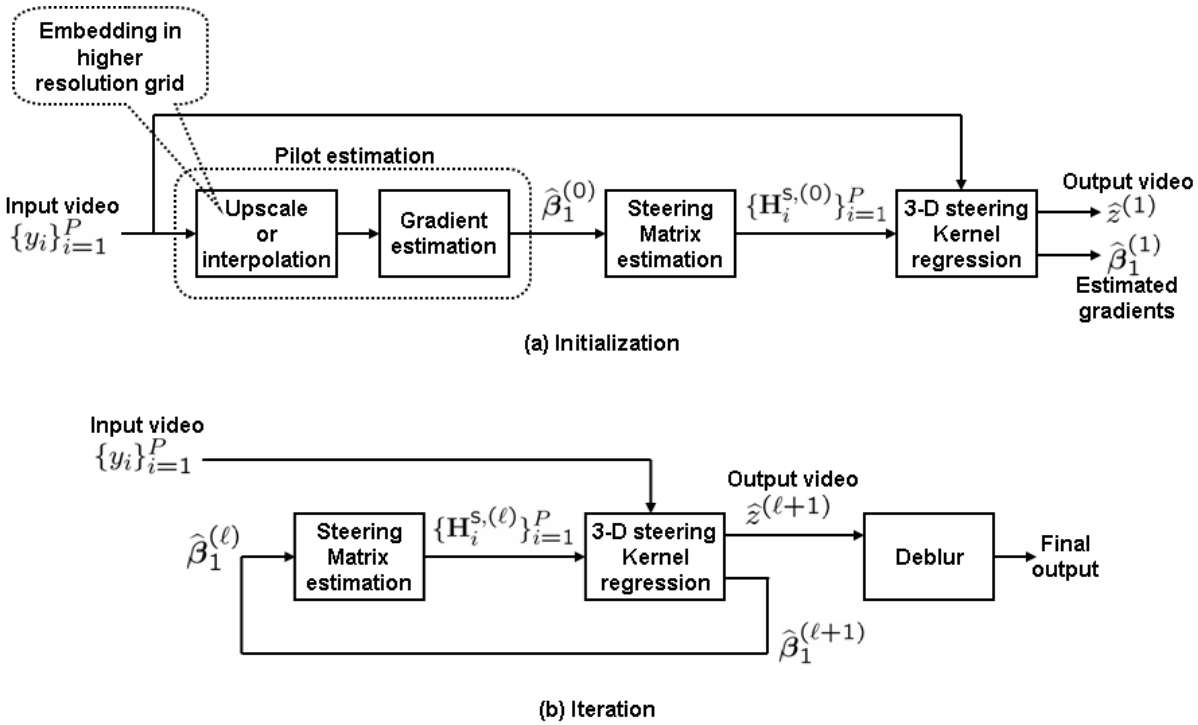


Fig. 6. Block diagram representation of the iterative 3-D steering kernel regression: (a) Initialization process, and (b) Iteration process.

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 gradients $\widehat{\boldsymbol{\beta}}_1^{(0)}$ from the pilot estimation (we use classic kernel regression with $N = 2$), and create steering matrices $\mathbf{H}_i^{s,(0)}$ for all the samples y_i as explained in Section III. Once $\mathbf{H}_i^{s,(0)}$ are available, we apply SKR to the input video embedded in a higher resolution grid, and estimate not only the output video $\hat{z}^{(1)}$ but also its gradients $\widehat{\boldsymbol{\beta}}_1^{(1)}$. This is the initialization process which is shown in Fig. 6(a). Next, using $\widehat{\boldsymbol{\beta}}_1^{(1)}$, we re-create the steering matrices $\mathbf{H}_i^{s,(1)}$. Since the estimated gradients $\widehat{\boldsymbol{\beta}}_1^{(1)}$ are also denoised and upscaled by SKR, the new steering matrices contain better orientation information. With $\mathbf{H}_i^{s,(1)}$, we apply SKR to the embedded input video again. We repeat this procedure several times. While we do not discuss the convergence properties of this approach here, it is worth mentioning that

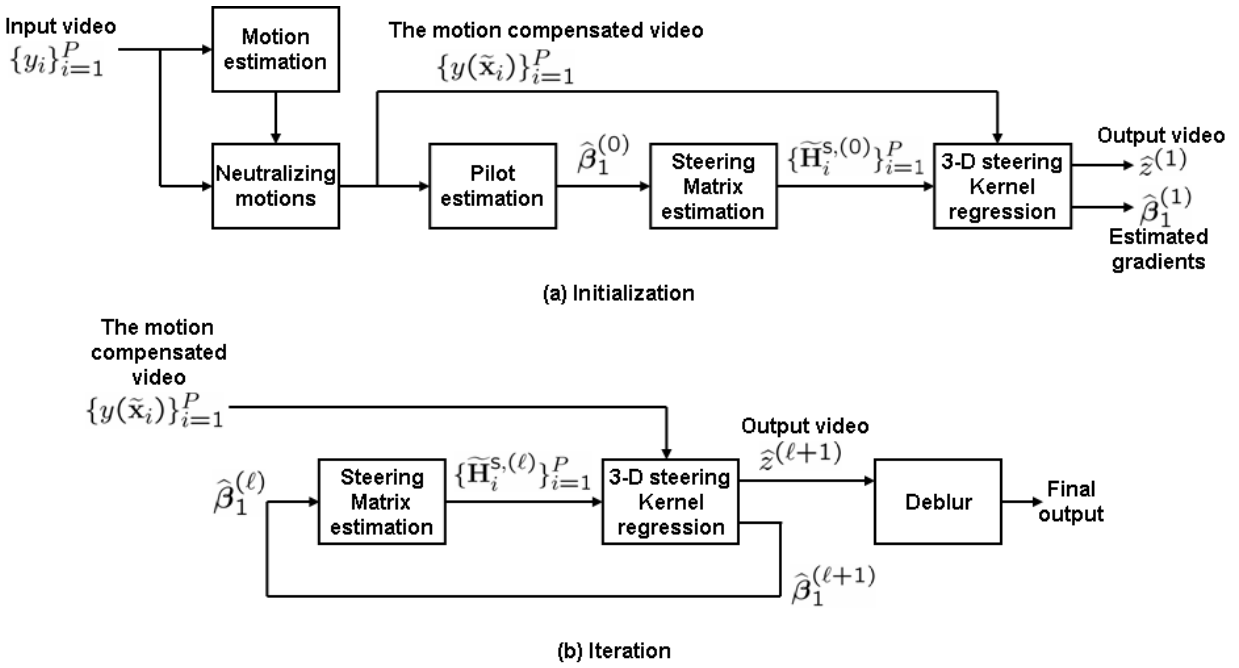


Fig. 7. Block diagram representation of the 3-D iterative steering kernel regression with motion compensation: (a) Initialization process, and (b) Iteration process.

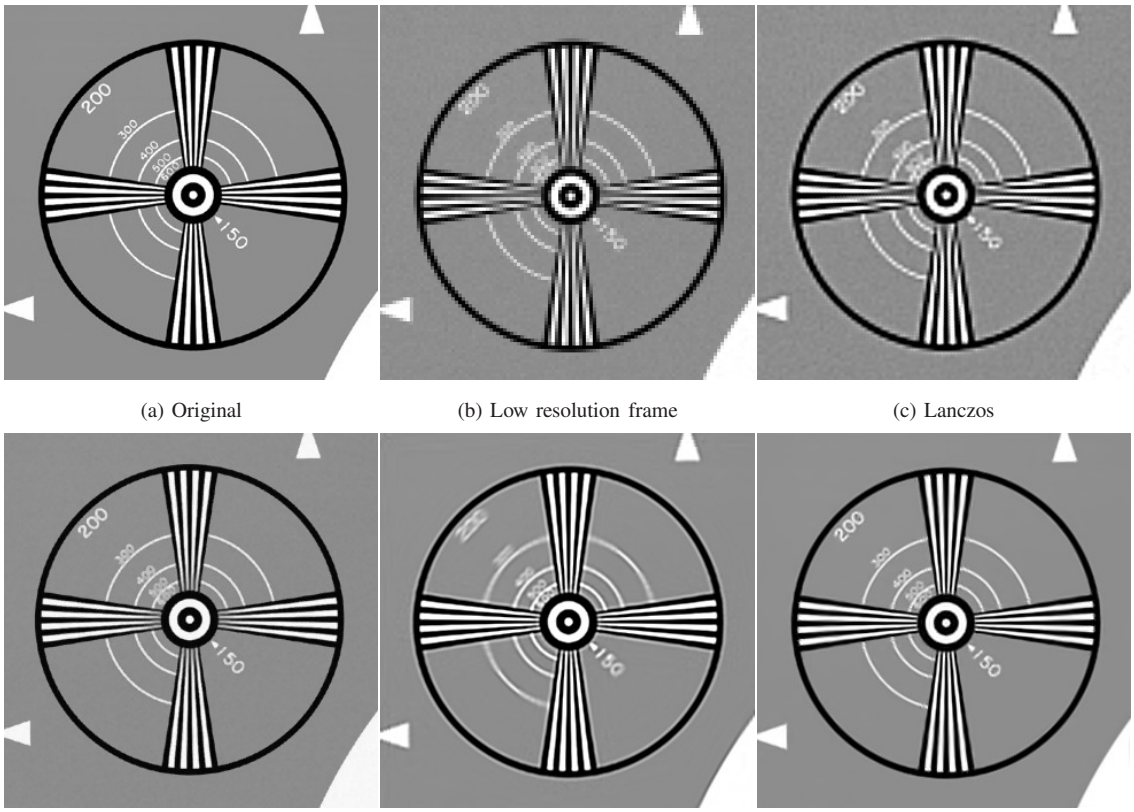
typically, no more than a few iterations are necessary to reach convergence[3].

Fig. 8 illustrates a simple super-resolution example. In this example, we created 9 of synthetic low resolution frames from the image shown in Fig. 8(a) by blurring with a 3×3 uniform PSF, shifting the blurred image by 0, 4, or 8 pixels[4] 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. 8(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 [2], non-local based super-resolution [16], and 3-D ISKR with rough motion compensation at time $t = 5$ are shown in Figs. 8(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.

[3]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) [31].

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



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

(d) Robust super-resolution [2] (e) Non-local based super-resolution [16] (f) 3-D ISKR with motion compensation

Fig. 8. A simple super-resolution example using 3-D ISKR with motion compensation: (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 [2], and (e) an upscaled image by non-local based super-resolution [16]. The corresponding PSNR values are (c)19.67, (d)30.21, (e)27.94, and (f)29.16[dB].

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. 8.

It is worth noting that while in the proposed algorithm in Fig. 7 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 [32]. Similarly, in our implementation, we used the optical flow [33] framework to compute the rough motion estimates. This step too can be replaced by other methods such as a block

matching algorithm [34].

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 post-processing step to improve the outputs by 3-D ISKR further. Defining the estimated frame at time t as $\widehat{\underline{\mathbf{Z}}}(t) = [\cdots, \hat{z}(x_{1j}, x_{2j}, t), \cdots]^T$ 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:

$$\widehat{\underline{\mathbf{U}}}(t) = \arg\min_{\underline{\mathbf{U}}(t)} \left\| \underline{\mathbf{U}}(t) - \mathbf{G}\widehat{\underline{\mathbf{Z}}}(t) \right\|_2^2 + \lambda C_R(\underline{\mathbf{U}}(t)), \tag{34}$$

where $\mathbf{G}$ is the blur matrix, λ (≥ 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 [2], where

$$C_R(\underline{\mathbf{U}}(t)) = \sum_{v_1=-w}^{w} \sum_{v_2=-w}^{w} \eta^{|v_1|+|v_2|} \left\| \underline{\mathbf{U}}(t) - \mathbf{S}_{x_1}^{v_1} \mathbf{S}_{x_2}^{v_2} \underline{\mathbf{U}}(t) \right\|_1 \tag{35}$$

where η is the smoothing parameter, w is the window size, and $\mathbf{S}_{x_1}^{v_1}$ is the shift matrix that shifts $\underline{\mathbf{U}}(t)$ v_1-pixels along 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 [35], we have introduced a different regularization function called *Adaptive Kernel Total Variation* (AKTV) [12]. 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.

IV. Experimental Results

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 IV-B, we illustrate additional examples with motion compensation.

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. 9(a) and 10(a), respectively. Then, we upscale and denoise the degraded videos by Lanczos interpolation (frame-by-frame upscaling), the NL-means based approach of [16], and 3-D ISKR, which includes deblurring[5] the upscaled video frames using the BTV approach [2]. 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. 9(b)-(d) and 10(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 [16]), 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.12 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 as shown in Fig. 11, 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 Figs 11 and 16, 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. 9 and 10, we iterated SKR 6 times, and the regularization parameters $\lambda' = 1.0$ and $\lambda'' = 0.1$ were used.

B. Spatiotemporal Upscaling Examples

In this section, we present two video upscaling examples by 3-D ISKR with rough motion compensation. Unlike the previous examples (Miss America, Salesman, 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 translational motions by the optical flow technique[6] [33], and apply 3-D ISKR to the roughly motion-compensated inputs.

The first example in Fig. 13 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

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

[6]We used L_2-norm for the optimization with no regularization.

(a) Degraded frames (b) Lanczos (c) NL-means based SR [16] (d) 3-D ISKR

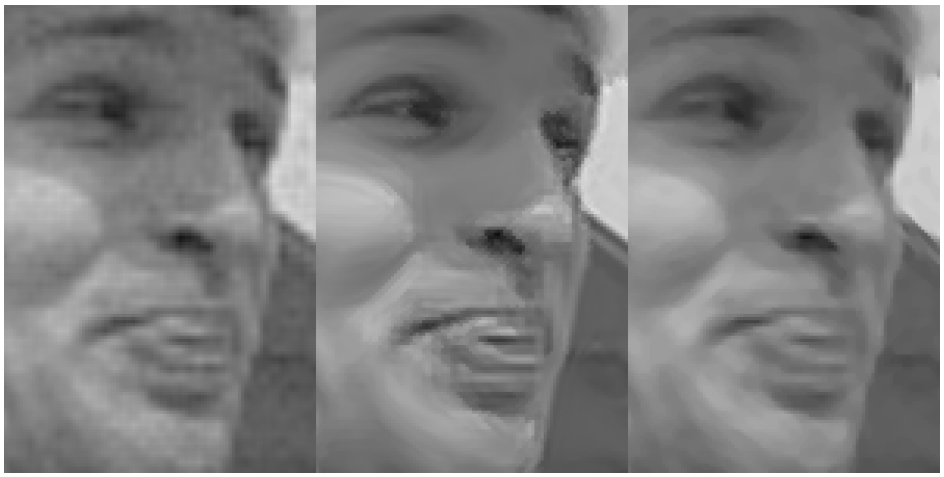
Fig. 9. A video upscale example using Miss America sequence: (a) the degraded frames at time $t = 8$ and 13, (b) the upscaled frames by Lanczos interpolation (PSNR: 34.28 (top) and 33.95 (bottom)), (c) the upscaled frames by NL-means based SR [16] (PSNR: 34.67 (top) and 35.34 (bottom)), and (d) the upscaled 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. 12(a).

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 ($h = 1.35$, $\alpha = 0.15$, $\lambda' = 0.1$ and $\lambda'' = 1.0$). Similar to the first example, we used the cropped "Stefan" sequence for the next video upscaling example. The results are shown in Fig. 14. The parameters $h = 1.35$, $\alpha = 0.15$, $\lambda' = 0.1$ and $\lambda'' = 1.0$ 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, the temporal upscaling is effected by producing a pilot estimate and improving the estimate iteratively just as in the

(a) Degraded frames (b) Lanczos (c) NL-means based SR [16] (d) 3-D ISKR

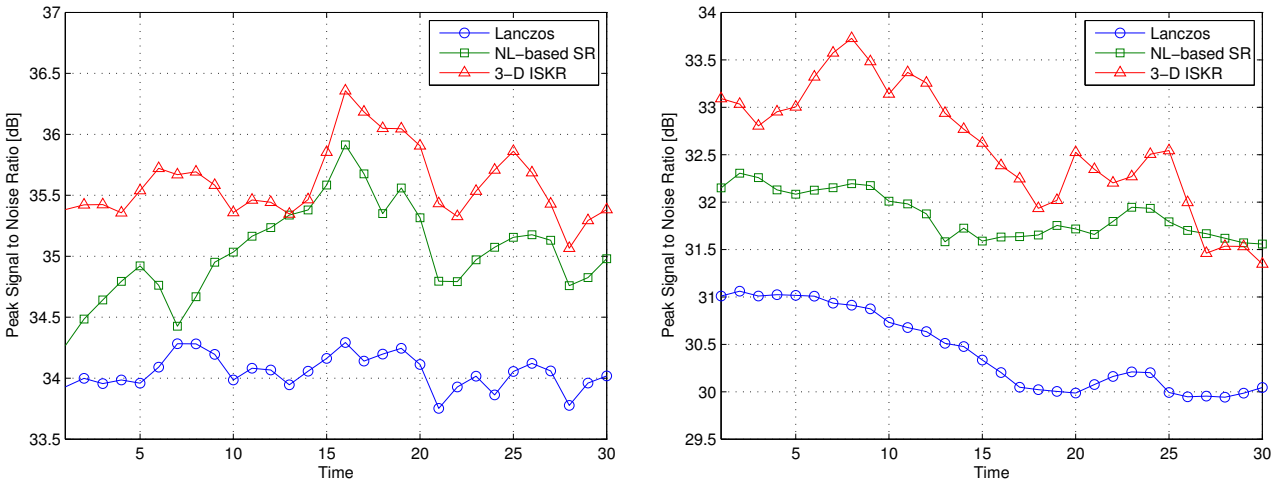
Fig. 10. A video upscaling example using Foreman sequence: (a) the degraded frames, (b) the upscaled frames by Lanczos interpolation (PSNR: 31.01 (top) and 30.21 (bottom)), (c) the upscaled frames by NL-means based SR [16] (PSNR: 32.13 (top) and 31.94 (bottom)), and (d) the upscaled frames by 3-D ISKR (PSNR: 33.02 (top) and 32.12 (bottom)). Also the PSNR values for all the frames are shown in Fig. 12(c)



(a) Lanczos (b) NL-means based SR [16] (c) 3-D ISKR

Fig. 11. Enlarged images of the cropped sections from the upscaled Foreman frame ($t = 6$) shown in Fig. 10.

spatial upscaling case illustrated in the block diagrams in Fig.7. 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 [16]. The examples in Figs. 13(d) and 14(d) show this application of 3-D ISKR, namely simultaneous space-time upscaling, using the same inputs of the Coastguard and Stefan sequences. Figs. 13(d) and 14(d) illustrate estimated intermediate frames by 3-D ISKR.



(a) Miss America (b) Foreman

Fig. 12. The PSNR values of each upscaled frame by Lanczos, NL-means based SR [16], and 3-D ISKR for (a) the results of Miss America shown in Fig. 9 and (b) the results of Foreman shown in Fig. 10.

The final example in Fig. 15 is a real experiment[7] of space-time upscaling with a native QCIF sequence, Carphone (144×176, 30 frames). Fig. 15 shows (a) the input frame at $t = 25$ to 27 and (b) the upscaled frames by NL-based method [16], and (c) the upscaled frames by 3-D , and (d) the estimated intermediate frames by 3-D ISKR. Also, in Fig. 16 shows the visual comparison small sections of the upscaled frames at $t = 27$ by (a) Lanczos, (b) NL-based method, and (c) 3-D ISKR, and we can see the visual differences more clearly.

V. 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 III-C, which yields excellent results in both spatial and temporal upscaling.

[7]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.

Performance analysis of super-resolution algorithm remains an interesting area of work, particularly with the new class of algorithms such the proposed and NL-based method [16] which can avoid subpixel motion estimation. Some results already exist which provide such bounds under certain simplifying conditions [36], but these results need to be expanded and studied further.

REFERENCES

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

[2] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multi-frame super-resolution," *IEEE Transactions on Image Processing*, vol. 13, no. 10, pp. 1327–1344, October 2004.

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

[4] B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau, "Multiframe resolution enhancement methods for compressed video," *IEEE Signal Processing Letter*, vol. 9, pp. 170–174, June 2002.

[5] M. Irani and S. Peleg, "Super resolution from image sequence," *Proceedings of 10th International Conference on Pattern Recognition (ICPR)*, vol. 2, pp. 115–120, 1990.

[6] M. M. J. Koo and N. Bose, "Constrained total least squares computations for high resolution image reconstruction with multisensors," *International Journal of Imaging Systems and Technology*, vol. 12, pp. 35–42, 2002.

[7] P. Vandewalle, L. Sbaiz, M. Vetterli, and S. Susstrunk, "Super-resolution from highly undersampled images," *Proceedings of International Conference on Image Processing (ICIP)*, pp. 889–892, September 2005, italy.

[8] N. A. Woods, N. P. Galatsanos, and A. K. Katsaggelos, "Stochastic methods for joint registration, restoration, and interpolation of multiple undersampled images," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 201–213.

[9] A. Zomet, A. Rav-Acha, and S. Peleg, "Robust super-resolution," *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, Hawaii.

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

[11] J. V. D. WEIJER and R. V. D. BOOMGAARD, "Least squares and robust estimation of local image structure," *Scale Space. International Conference*, vol. 2695, no. 4, pp. 237–254, 2003.

[12] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349–366, February 2007.

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

[14] H. Knutsson and C. F. Westin, "Normalized and differential convolution - methods for interpolation and filtering of incomplete and uncertain data," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Regocnition (CVPR)*, pp. 515–523, June 1993.

[15] K. S. Ni, S. Kumar, N. Vasconcelos, and T. Q. Nguyen, "Single image superresolution based on support vector regression," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, May 2006.

[16] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the non-local-means to super-resolution reconstruction," accepted for Publication in IEEE Transactions on Image Processing, March 2008.

[17] A. Buades, B. Coll, and J. M. Morel, "A review of image denosing algorithms, with a new one," *Multiscale Modeling and Simulation, Society for Industrial and Applied Mathematics (SIAM) Interdisciplinary Journal*, vol. 4, no. 2, pp. 490–530, 2005.

[18] H. Knutsson, "Representing local structure using tensors," *Proceedings of the 6th Scandinavian Conference on Image Analysis*, pp. 244–251, 1989.

[19] R. M. Haralick, "Edge and region analysis for digital image data," *Computer Graphic and Image Processing (CGIP)*, vol. 12, no. 1, pp. 60–73, January 1980.

[20] K. Q. Weinberger and G. Tesauro, "Metric learning for kernel regression," *Proceedings of the Eleventh International Workshop on Artificial Intelligence and Statistics*, pp. 608–615, (AISTATS-07), Puerto Rico.

[21] E. A. Nadaraya, "On estimating regression," *Theory of Probability and its Applications*, pp. 141–142, September 1964.

[22] D. Ruppert and M. P. Wand, "Multivariate locally weighted least squares regression," *The annals of statistics*, vol. 22, no. 3, pp. 1346–1370, September 1994.

[23] W. Hardle and P. Vieu, "Kernel regression smoothing of time series," *Journal of Time Series Analysis*, vol. 13, pp. 209–232, 1992.

[24] N. Nguyen, P. Milanfar, and G. H. Golub, "Efficient generalized cross-validation with applications to parametric image restoration and resolution enhancement," *IEEE Transactions on Image Processing*, vol. 10, no. 9, pp. 1299–1308, September 2001.

[25] F. Luisier, T. Blu, and M. Unser, "A new sure approach to image denoising: Inter-scale orthonormal wavelet thresholding," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 593–606, March 2007.

[26] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding*, vol. 63.

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

[28] D. N. Lee and H. Kalmus, "The optic flow field: the foundation of vision," *Philosophical Transactions of the Royal Society of London Series B-Biological Sciences*, vol. 290, no. 1038, pp. 169–179.

[29] J. Wright and R. Pless, "Analysis of persistent motion patterns using the 3d structure tensor," *Proceedings of the IEEE Workshop on Motion and Video Computing*, 2005.

[30] S. Chaudhuri and S. Chatterjee, "Performance analysis of total least squares methods in three-dimensional motion estimation," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 5, pp. 707–714, October 1991.

[31] H. Takeda, H. Seo, and P. Milanfar, "Statistical approaches to quality assessment for image restoration," *Proceedings of the International Conference on Consumer Electronics*, January 2008, Las Vegas, NV, Invited paper.

[32] G. Farnebäck, "Polynomial expansion for orientation and motion estimation," Ph.D. dissertation, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 2002, dissertation No 790, ISBN 91-7373-475-6.

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

[34] S. Zhu and K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, February.

[35] H. Takeda, S. Farsiu, and P. Milanfar, "Deblurring using regularized locally-adaptive kernel regression," *IEEE Transactions on Image Processing*, vol. 17, no. 4, pp. 550–563, April 2008.

[36] D. Robinson and P. Milanfar, "Statistical performance analysis of super-resolution," *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1413–1428, June 2006.

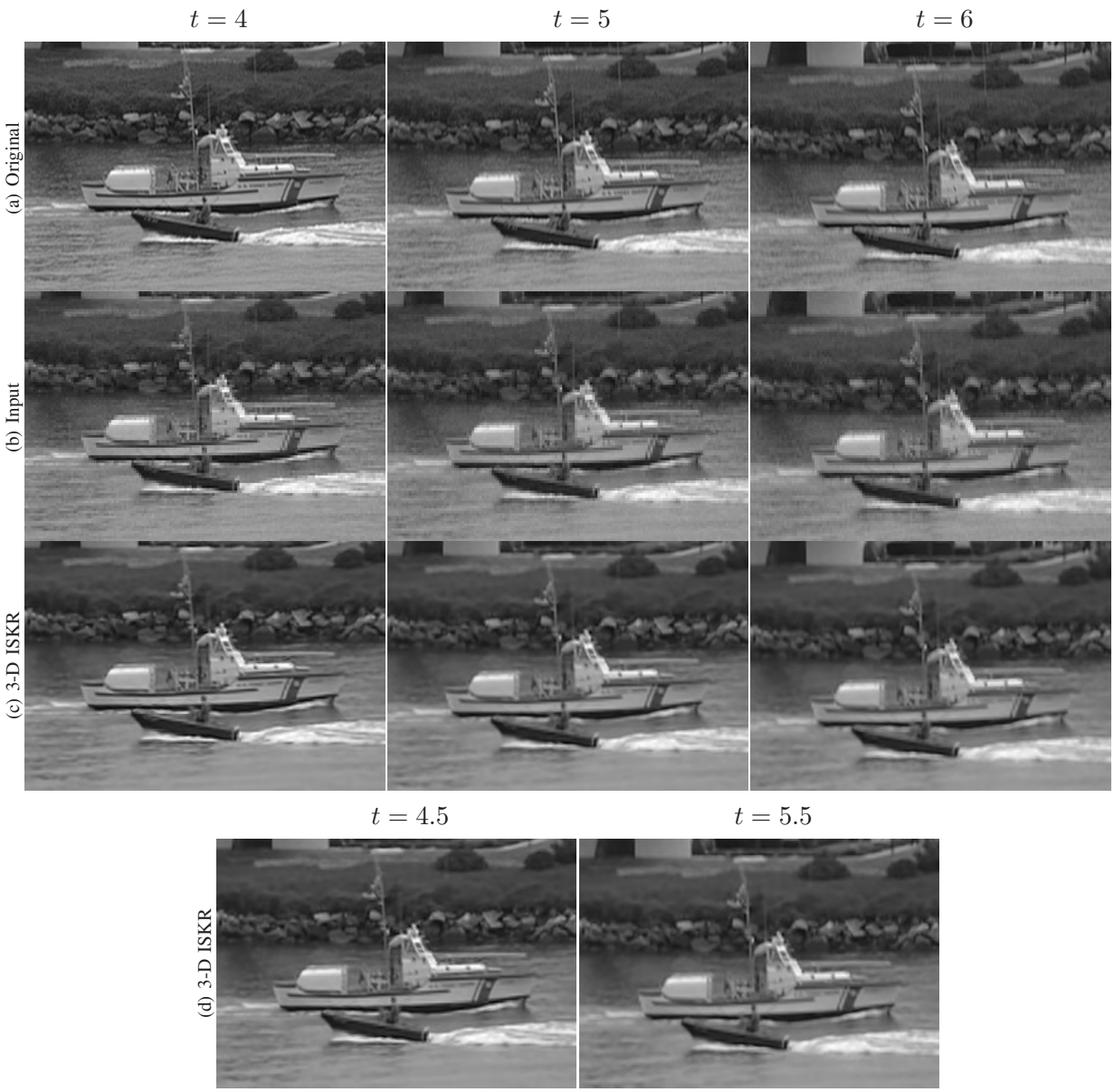


Fig. 13. A Coastguard example of spatiotemporal upscaling: from the top row to the bottom, (a) original video frames at time $t = 4$ to 6 (b) the input videos generated by blurred with a 2×2 uniform PSF, adding white Gaussian noise with standard deviation $\sigma = 2$, and spatially downsampling with 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].

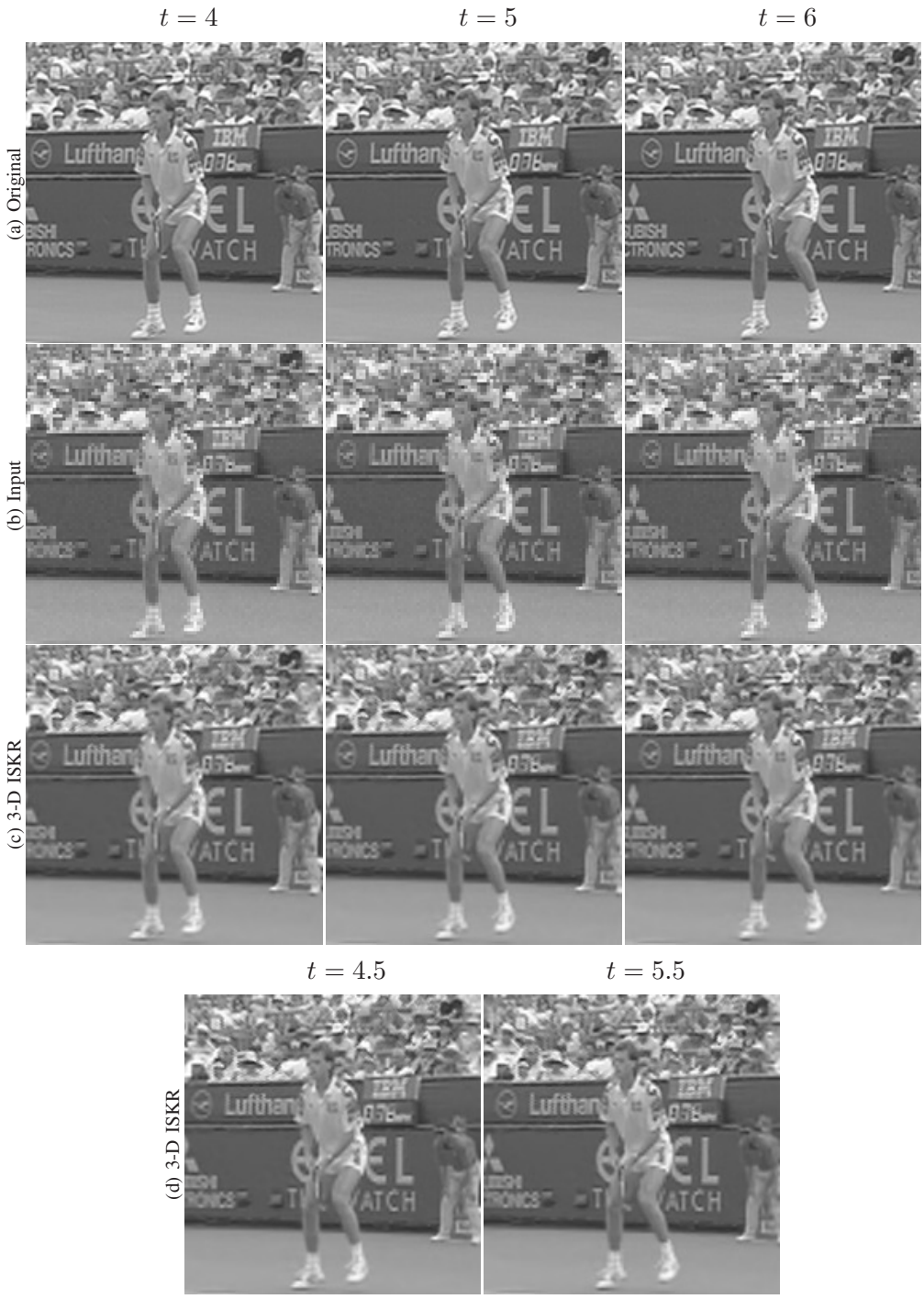


Fig. 14. A Stefan example of video upscaling: from the top row to the bottom, (a) original video frames at time $t = 4$ to 6, (b) the input videos generated by blurred with a 2×2 uniform PSF, adding white Gaussian noise with standard deviation $\sigma = 2$, and spatially downsampling with the factor 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].

Fig. 15. A 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.

(a) Lanczos (b) NL-means based SR [16] (c) 3-D ISKR

Fig. 16. Enlarged images of the cropped sections from the upscaled Carphone frame ($t = 27$) shown in Fig. 15.