Generalizing the Nonlocal-Means to Super-Resolution Reconstruction

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

***Abstract*—Super-resolution reconstruction proposes a fusion of several low-quality images into one higher quality result with better optical resolution. Classic super-resolution techniques strongly rely on the availability of accurate motion estimation for this fusion task. When the motion is estimated inaccurately, as often happens for nonglobal motion fields, annoying artifacts appear in the super-resolved outcome. Encouraged by recent developments on the video denoising problem, where state-of-the-art algorithms are formed with no explicit motion estimation, we seek a super-resolution algorithm of similar nature that will allow processing sequences with general motion patterns. In this paper, we base our solution on the Nonlocal-Means (NLM) algorithm. We show how this denoising method is generalized to become a relatively simple super-resolution algorithm with no explicit motion estimation. Results on several test movies show that the proposed method is very successful in providing super-resolution on general sequences.**

***Index Terms*—Nonlocal-means, probabilistic motion estimation, super-resolution.**

I. Introduction

SUPER-RESOLUTION reconstruction proposes a fusion of several low quality images into one higher quality result with better optical resolution. This is an *Inverse Problem* that combines denoising, deblurring, and scaling-up tasks, aiming to recover a high quality signal from degraded versions of it. Fig. 1 presents the process that explains how a low-resolution image sequence $\mathbf{y}_t$ is related to an original higher resolution movie $\mathbf{x}_t$. During the imaging, the scene may become blurred due to atmospheric, lens, or sensors' effects. The blur is denoted by $\mathbf{H}$, assumed for simplicity to be linear space and time invariant. Similarly, the loss of spatial resolution due to the sensor array sampling is modeled by the fixed decimation operator $\mathbf{D}$, representing the resolution factor s between the original sequence, and the measured one. White Gaussian iid noise is assumed to be added to the measurements, both in order to refer to actual noise in imaging systems, as well as for accommodating model mismatches.

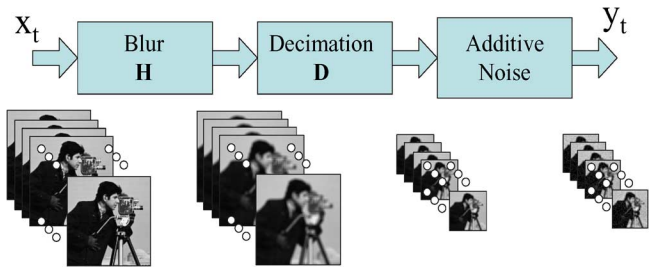


Fig. 1. Imaging process to be reversed by super resolution.

The super resolution goal is the recovery of $\mathbf{x}_t$ from the input set of images $\mathbf{y}_t$, reversing the above process. Such reconstruction relies on motion in the scene to recover details that are finer than the sampling grid. Fig. 2 demonstrates how small details can be recovered when the motion between the images in the sequence is known with a high degree of accuracy. The top row in the figure is the input sequence. The middle row is the up-scaled version of each image (unknown values are set to a background color), shifted by the known translation between the current image and the first (reference) image. The bottom row shows the construction of the super resolution image, from left to right. Initially, the first image is placed on the grid. Then, every new image in the sequence is placed on the same grid, with a displacement reflecting the motion it underwent. The merger of all images represents the outcome of the super resolution algorithm. We note that this description of the mechanics of super-resolution is somewhat simplistic; In most cases, one cannot assume the translations to be exact multiples of the high-resolution pixel sizes. This makes the estimation of accurate motion parameters and the merger of all images much more complex than described here.

While the above-described method is somewhat simplistic, it is a faithful description of the foundations for all classic super resolution algorithms. The first step of such algorithms is an estimation of the motion in the sequence, followed by a fusion of the inputs according to these motion vectors. A wide variety of super resolution algorithms have been developed in the past two decades; we refer to [1]–[26] as representatives of this vast literature.

In the currently available super-resolution algorithms, only global motion estimation (e.g., translation or affine global warp) is accurate enough to lead to a successful reconstruction of a super-resolved image. This is very limiting, as most actual

Manuscript received December 23, 2007; revised April 23, 2008. First published December 2, 2008; current version published December 12, 2008. This work was supported in part by the United States-Israel Binational Science Foundation Grant 2004199 and in part by the U.S. Air Force Office of Scientific Research Grant FA9550-07-1-0365. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Pier Luigi Dragotti.

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

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

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

Digital Object Identifier 10.1109/TIP.2008.2008067

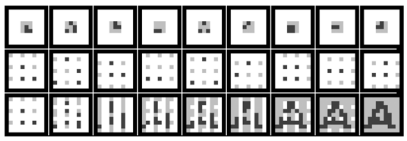


Fig. 2. Super resolving an image using low resolution inputs with known translations. Reconstruction proceeds from left to right. Top: input images; middle: corresponding up-scaled images, shifted using known translations; bottom: accumulated reconstruction by adding the current low resolution image to the output canvas.

scenes contain motion that is local in its nature (e.g., a person talking). Obtaining highly accurate local motion estimation, known as optical flow, is a very difficult task, and particularly so in the presence of aliasing and noise. When inaccurately estimated motion is used within one of the existing reconstruction algorithms, it often leads to disturbing artifacts that cause the output to be inferior, even when compared to the given measurements. This discussion leads to the commonly agreed unavoidable conclusion that general content movies are not likely to be handled well by classical super-resolution techniques.

This severe restriction leads us to seek a different approach to super-resolution. Can such an algorithm be proposed with no explicit motion estimation? Our starting point for this quest (after a super-resolution algorithm that is able to process sequences with a general motion pattern) is the video denoising application, where several recent contributions demonstrate state-of-the-art results with algorithms that avoid motion estimation [27]–[30]. Among these, we choose to take a closer look at the Nonlocal Means (NLM) algorithm, with the aim to generalize it to perform super-resolution reconstruction.

The NLM is the weakest among the recent motion-estimation-free video denoising algorithms, and yet, it is also the simplest. As such, it stands as a good candidate for generalization. The NLM is posed originally in [31] as a single image denoising method, generalizing the well-known bilateral filter [32], [33]. Denoising is obtained by replacing every pixel with a weighted average of its neighborhood. The weights for this computation are evaluated by using block-matching fit between image patches centered around the center pixel to be filtered, and the neighbor pixels to be averaged. Recent work has shown how this method can be used for video denoising by extending the very same technique to 3-D neighborhoods [27]. An improvement of this technique, considering varying size neighborhoods is suggested in [28], so as to trade bias versus variance in an attempt to get the best mean-squared-error (MSE).

The NLM was proposed intuitively in [27] and [31], and, thus, it is natural to try to extend it to perform super-resolution using a similar intuition. This intuition leads to independent up-scaling of each image in the sequence using a smart interpolation method, followed by NLM processing. However, extensive experiments indicate that this intuitive method does not provide super-resolution results. For this reason, a more profound understanding of the NLM filter is required for its successful generalization to super-resolution.

In order to gain a better understanding of the NLM, we propose redefining it as an energy minimization task. We show that the novel penalty term we propose indeed leads when minimized to the NLM. We then carefully extend the penalty function to the super-resolution problem. We show how a tractable algorithm emerges from the minimization of this penalty function, leading to a local, patch-based, super-resolution process with no explicit motion estimation. Empirical tests of the derived algorithm on actual sequences with general motion patterns are then presented, thus demonstrating the capabilities of the derived algorithm.

The structure of the paper is as follows. Section II describes the NLM denoising filter, as posed in [31]. This section can be skipped by readers who are familiar with the NLM. Section III introduces an energy function to be minimized for getting a denoising effect for a single image; We show that this minimization leads to a family of image denoising algorithms, NLM included as a special case. We also provide a simpler penalty function addressing the same goal, which will be effectively used in the later part of the paper. Section IV proposes a generalization of the introduced energy function to cope with resolution changes, thereby enabling super-resolution reconstruction. In this section we also derive the eventual super-resolution algorithm we propose, and discuss its numerical structure. Section V shows results on sequences with general motion, demonstrating the successful recovery of high frequencies. We conclude in Section VI, outlining the key contribution of this work, and describing several directions for further research.

II. Bilateral and the NLM Denoising Filters

We begin our journey with a description of the bilateral and the NLM filters, as the development that follows relies on their structure. The description given in this section is faithful to the one found in [31] and [32]. The bilateral and the NLM filters are two very successful image denoising filters. While not the very best in denoising performance, these methods are very simple to understand and implement, and this makes them a good starting point for our needs.

Both the bilateral and the NLM filters are based on the assumption that image content is likely to repeat itself within some neighborhood. Therefore, denoising each pixel is done by averaging all pixels in its neighborhood. This averaging is not done in a blind and uniform way, however. Instead, each of the pixels in the relevant neighborhood is assigned a weight, that reflects the probability that this pixel and the pixel to be denoised had the same value, prior to the additive noise degradation. A formula describing these filters looks like[1]

$$\hat{\mathbf{x}}[k,l] = \frac{\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{y}[i,j]}{\sum_{(i,j)\in N(k,l)} w[k,l,i,j]} \tag{1}$$

where $N(k,l)$ stands for the neighborhood of the pixel (k,l), and the term $w[k,l,i,j]$ is the weight for the (i,j)-th neighbor pixel. The input pixels are $\mathbf{y}[k,l]$, and the output result in that location is $\mathbf{x}[k,l]$.

The two filters differ in the method by which the weights are computed. The weights for the bilateral filter are computed

[1]As we shall see next, in this framework the coefficients $w[k,l,i,j]$ are all restricted to be positive. This is a shortcoming, which can be overcome by extending the framework to higher order—see [34].

based both on radiometric (gray-level) proximity and geometric proximity between the pixels, namely

$$w_{BL}[k,l,i,j] = \exp\left\{-\frac{(\mathbf{y}[k,l]-\mathbf{y}[i,j])^2}{2\sigma_r^2}\right\} \cdot f\left(\sqrt{(k-i)^2+(l-j)^2}\right). \quad (2)$$

The function f takes the geometric distance into account, and as such, it is monotonically nonincreasing. It may take many forms, such as a Gaussian, a box function, a constant, and more. The parameter σ_r controls the effect of the grey-level difference between the two pixels. This way, when the two pixels that are markedly different, the weight is very small, implying that this neighbor is not to be trusted in the averaging.

The radiometric part in the weights of the NLM is computed slightly differently, by computing the Euclidean distance between two image patches centered around these two involved pixels. Defining $\hat{\boldsymbol{R}}_{k,l}$ as an operator that extracts a patch of a fixed and predetermined size (say $q \times q$ pixels) from an image, the expression $\hat{\boldsymbol{R}}_{k,l}\mathbf{y}$ ($\mathbf{y}$ is represented as a vector by lexicographic ordering) results with a vector of length q^2 being the extracted patch. Thus, the NLM weights are given by

$$w_{NLM}[k,l,i,j] = \exp\left\{-\frac{\|\hat{\mathbf{R}}_{k,l}\mathbf{y}-\hat{\mathbf{R}}_{i,j}\mathbf{y}\|_2^2}{2\sigma_r^2}\right\} \cdot f\left(\sqrt{(k-i)^2+(l-j)^2}\right). \quad (3)$$

Obviously, setting $\hat{\boldsymbol{R}}_{k,l}$ to extract only a single pixel, the bilateral filter emerges as a special case of the NLM algorithm.

We note that there are various other ways to choose the weights in (1), and the above separable choice of the weights (product of radiometric and Euclidean distance terms) is only one choice. For example, the steering kernel may provide an interesting alternative, taking into account the correlation between the pixel positions and their value [34]. Nevertheless, in this paper we shall restrict our choice of weights to those used by the NLM.

III. NLM via Energy Minimization

Both the bilateral and the NLM filters described-above were presented intuitively as algorithmic formulas, as in (1). We claim that both these filters can be derived by minimizing a properly defined penalty function. Following the rationale and steps taken in [33] and [35], we present such a penalty function, and show how these algorithms emerge from it. This will prove valuable when taking the next step of generalizing these methods to a super-resolution reconstruction algorithm, as will be shown in Section IV. Sections III-A and III-C present two possible and novel penalty functions for denoising, and derive from both the NLM algorithm and some variations of it. The readers interested in the super-resolution portion of this work can start their reading in Section III-C.

A. Penalty Function

The penalty function we start with reflects two forces: i) We desire a proximity between the reconstructed and the input images—this is the classic likelihood term; and ii) We would like each patch in the resulting image to resemble other patches in its vicinity. However, we do not expect such a fit for every pair, and, thus, we introduce weights to designate which of these pairs are to behave alike. Putting these two forces together with proper weighting,[2] we propose a maximum *a posteriori* probability (MAP) penalty of the form

$$\epsilon^2(\mathbf{x}) = \frac{\lambda}{2}\cdot\|\mathbf{x}-\mathbf{y}\|_2^2 + \frac{1}{4}\cdot\sum_{(k,l)\in\Omega}\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\cdot\|\mathbf{R}_{k,l}\mathbf{x}-\mathbf{R}_{i,j}\mathbf{x}\|_2^2. \quad (4)$$

The first term is the log-likelihood function for a white and Gaussian noise. The second term stands for a prior, representing the (minus) log of the probability of an image $\mathbf{x}$ to exist. The weights in the above expression, $w[k,l,i,j]$, are assigning a confidence that the patches around $\mathbf{x}[k,l]$ and $\mathbf{x}[i,j]$ are to be close to each other. Computing these weights can be done in a number of ways, one of which is using (3) and using $\mathbf{y}$ instead of the unknown $\mathbf{x}$. In order to keep the discussion simple, from this point on we shall assume that the weights $w[k,l,i,j]$ are the NLM ones. It is important to note that the patch extraction operator $\hat{\boldsymbol{R}}_{kl}$ used for computing the weights (as in (3)) and the operator $\mathbf{R}_{kl}$ used in the penalty term are generally of different sizes.

The notation Ω stands for the support of the entire image. Thus, the second term sweeps through each and every pixel (k,l) in the image, and for each we require a proximity to surrounding patches in its neighborhood.

B. Derivation of the NLM Filter

Assuming the weights are predetermined and considered as constants, we can minimize this penalty term with respect to $\mathbf{x}$ by zeroing its derivative

$$\frac{d\epsilon^2(\mathbf{x})}{d\mathbf{x}} = \lambda(\mathbf{x}-\mathbf{y}) + \frac{1}{2}\sum_{(k,l)\in\Omega}\sum_{(i,j)\in N(k,l)} w[k,l,i,j] \times (\mathbf{R}_{k,l}-\mathbf{R}_{i,j})^T(\mathbf{R}_{k,l}-\mathbf{R}_{i,j})\mathbf{x} = 0. \quad (5)$$

In order to simplify this equation, we open the brackets

$$\frac{d\epsilon^2(\mathbf{x})}{d\mathbf{x}} = 0 = \lambda(\mathbf{x}-\mathbf{y}) + \frac{1}{2}\sum_{(k,l)\in\Omega}\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\mathbf{x} - \frac{1}{2}\sum_{(k,l)\in\Omega}\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\mathbf{x}$$

[2]The reason for the factor 1/4 in the second term will be made clear shortly.

$$- \frac{1}{2} \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{i,j}^T\mathbf{R}_{k,l}\mathbf{x} + \frac{1}{2} \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{i,j}^T\mathbf{R}_{i,j}\mathbf{x}. \quad (6)$$

We proceed by invoking two assumptions: i) The neighborhood is symmetric, i.e., if $(i,j) \in N(k,l)$, then $(k,l) \in N(i,j)$ is also true; and ii) The weights are symmetric, i.e., $w[k,l,i,j] = w[i,j,k,l]$. Both these assumptions are natural—typical neighborhood definitions satisfy the first condition, and the weights of the NLM satisfy the second one. Using these two assumptions, we get the following two equalities:

$$\sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{i,j}^T\mathbf{R}_{k,l}\mathbf{x} = \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\mathbf{x}$$

$$\sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\mathbf{x} = \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{i,j}^T\mathbf{R}_{i,j}\mathbf{x}.$$

A formal proof (Theorem 1) for these equalities is given in Appendix A. Using these, (6) simplifies and becomes

$$0 = \lambda(\mathbf{x}-\mathbf{y}) + \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\mathbf{x} - \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\mathbf{x}. \quad (7)$$

As can be seen, the choice 1/4 in the original definition led to a simpler final outcome.

While solving the above equation directly is possible in principle, it requires an inversion of a very large matrix. Instead, we adopt an iterative approach based on the fixed-point strategy [36], [37]. Denoting $\mathbf{x}^{n-1}$ the outcome of the previous iteration, and $\mathbf{x}^n$ the desired outcome of the current iteration, we rewrite (7) with assignments of iteration stage per each instance of the unknown $\mathbf{x}$. The equation we propose is

$$0 = \lambda(\mathbf{x}^n-\mathbf{y}) + \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\mathbf{x}^n - \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\mathbf{x}^{n-1} \quad (8)$$

which leads to the relation

$$\left(\lambda\mathbf{I} + \sum_{(k,l)\in\Omega} \left[\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\right] \mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\right)\mathbf{x}^n = \lambda\mathbf{y} + \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\mathbf{x}^{n-1}. \quad (9)$$

Notice that the term $\sum_{(i,j)\in N(k,l)} w[k,l,i,j]$ generates a scalar, being a function of (k,l)—we shall denote this as $\bar{w}[k,l]$.

In the obtained equation, the right-hand-side (RHS) creates an image by manipulating image patches: for each location (k,l) in the image, we copy surrounding neighboring patches in locations (i,j) to the center position (k,l), multiplied by the weights $w[k,l,i,j]$. Once built, this image is added to $\mathbf{y}$ with a proper weight λ.

The matrix multiplying $\mathbf{x}^n$ on the left-hand-side is a diagonal positive definite matrix (see Appendix A). This matrix's only task is normalization of the weighted average that took place on the RHS. As this matrix is invertible, the new solution is obtained by

$$\mathbf{x}^n = \left(\lambda\mathbf{I} + \sum_{(k,l)\in\Omega} \bar{w}[k,l]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\right)^{-1} \times \left(\lambda\mathbf{y} + \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\mathbf{x}^{n-1}\right). \quad (10)$$

When using the fixed-point method, as we did above, every appearance of the unknown in the equation is assigned with an iteration number. Among the many possible assignments, one should seek one that satisfies two important conditions: i) The computation of $\mathbf{x}^n$ from $\mathbf{x}^{n-1}$ should be easy; and ii) The obtained iterative formula should lead to convergence. As for the first requirement, we indeed have an assignment that leads to a simple iterative step. Convergence of the above algorithm is guaranteed if the overall operator multiplying $\mathbf{x}^{n-1}$ is convergent, i.e.,

$$\rho\left\{\left(\lambda\mathbf{I} + \sum_{(k,l)\in\Omega} \bar{w}[k,l]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\right)^{-1} \times \left(\sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\right)\right\} < 1 \quad (11)$$

where $\rho\{\mathbf{A}\}$ is the spectral radius of $\mathbf{A}$. It is easily seen that for sufficiently large λ, this condition is met. Nevertheless, we do not worry about convergence, as we will be using the above for one iteration only, with the initialization of $\mathbf{x}^0 = \mathbf{y}$. This means that the output for the denoising process is $\hat{\mathbf{x}} = \mathbf{x}^1$, obtained by

$$\hat{\mathbf{x}} = \left(\lambda\mathbf{I} + \sum_{(k,l)\in\Omega} \bar{w}[k,l]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\right)^{-1} \times \left(\lambda\mathbf{I} + \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\right)\mathbf{y}. \quad (12)$$

The above computation is done just as described above, with the obvious substitution of $\mathbf{x}^0 = \mathbf{y}$. This process is quite reminiscent of the way NLM and the bilateral filters operate, and yet it is different. The obtained algorithm is a more general and more powerful denoising algorithm than NLM.

In order to see how the NLM emerges from this formulation as a special case, we shall assume further that the patch extraction operation we use, $\mathbf{R}_{i,j}$, extracts a single pixel in location (i,j). This change means that $\mathbf{R}_{i,j}\mathbf{y}$ is in fact the single pixel $\mathbf{y}[i,j]$. Thus, in this case we have

$$\hat{\mathbf{x}} = \left(\lambda\mathbf{I} + \sum_{(k,l)\in\Omega} \bar{w}[k,l]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\right)^{-1}$$

$$\times \left(\lambda \mathbf{y} + \sum_{(k,l)\in\Omega} \mathbf{R}_{k,l}^T \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{y}[i,j] \right). \quad (13)$$

The term $\mathbf{R}_{k,l}^T x$ implies that a zero image is generated, and the value x is inserted to location (k,l). Using this property in (12), we obtain a pixel-wise denoising formula

$$\hat{\mathbf{x}}[k,l] = \frac{\lambda \mathbf{y}[k,l] + \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{y}[i,j]}{\lambda + \sum_{(i,j)\in N(k,l)} w[k,l,i,j]}. \quad (14)$$

This formula seems familiar, as it is effectively the same as the one in (4), with a slight modification due to the additional λ. This means that under the simplifying assumption we made about $\mathbf{R}_{i,j}$, the first iteration of the developed algorithm reduces to NLM. A natural question to ask is: Does the above formula still stands for a patch-based algorithm? The answer is positive, as the weights may be computed using patches, as the NLM does. Thus, the next natural question to ask is: Why have we used a patch extraction operation in the definition of the penalty in (4)? The answer to this is the generalization that follows next for the super-resolution case.

C. Bayesian Versus Proximity—An Alternative Path

The penalty term in (4) was formed like a MAP estimator, having a likelihood term that ties the measurements to the unknown, and a regularization term, posing a prior on the desired outcome

$$\epsilon^2(\mathbf{x}) = \frac{\lambda}{2} \cdot \|\mathbf{x}-\mathbf{y}\|_2^2 + \frac{1}{4} \cdot \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j] \cdot \|\mathbf{R}_{k,l}\mathbf{x} - \mathbf{R}_{i,j}\mathbf{x}\|_2^2.$$

In the previous section, we developed a general denoising scheme based on the above penalty, showing that the NLM and the bilateral filters arise from it as special cases. Recall that one of the last steps in the derivation was the use of a single iteration and initialization with $\mathbf{x}^0 = \mathbf{y}$. This basically means that the prior term requires pixels in the output image $\mathbf{x}^1$ to have similar grey values to pixels in the corresponding neighborhood in $\mathbf{y}$, provided they have similar surrounding. This idea can be put directly into the penalty term, giving

$$\eta^2(\mathbf{x}) = \frac{1}{2} \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j] \cdot \|\mathbf{R}_{k,l}\mathbf{x} - \mathbf{R}_{i,j}\mathbf{y}\|_2^2. \quad (15)$$

Targeting now the minimization of this penalty term, we null the derivative of this function with respect to $\mathbf{x}$, getting the equation

$$\frac{d\eta^2(\mathbf{x})}{d\mathbf{x}} = 0 = \sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T \times (\mathbf{R}_{k,l}\mathbf{x} - \mathbf{R}_{i,j}\mathbf{y}). \quad (16)$$

This time we do not need the fixed-point strategy, as a simple solution is easily derived, leading to

$$\hat{\mathbf{x}} = \left(\sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l} \right)^{-1} \times \left(\sum_{(k,l)\in\Omega} \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\mathbf{y} \right)$$
$$= \left(\sum_{(k,l)\in\Omega} \bar{w}[k,l]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l} \right)^{-1} \times \left(\sum_{(k,l)\in\Omega} \mathbf{R}_{k,l}^T \sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{i,j}\mathbf{y} \right). \quad (17)$$

Notice the resemblance between this formula and the one obtained by a different route in (12). In fact, for $\lambda = 0$, the two formulas are the same. Furthermore, following the same assumptions and steps that led us from (12) to (14), it is clear that pixel-wise, the above aligns perfectly with the NLM filter, as written in (1), namely

$$\hat{\mathbf{x}}[k,l] = \frac{\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{y}[i,j]}{\sum_{(i,j)\in N(k,l)} w[k,l,i,j]}.$$

D. Introducing the Temporal Domain

We shall use hereafter the penalty function in (15) to derive further algorithms, due to its simplicity, and the fact that it leads directly to the NLM. Before we turn to super-resolution, we must introduce the temporal axis into the penalty function, so as to process a sequence of images and not just a single one, as done so far. The following small changes in (15) lead to such a treatment

$$\eta_T^2(\mathbf{x}) = \frac{1}{2} \sum_{(k,l)\in\Omega} \sum_{t\in[1,\ldots,T]} \sum_{(i,j)\in N(k,l)} w[k,l,i,j,t] \cdot \|\mathbf{R}_{k,l}\mathbf{x} - \mathbf{R}_{i,j}\mathbf{y}_t\|_2^2. \quad (18)$$

The above expression makes use of the input sequence $\{\mathbf{y}_t\}_{t=1}^T$, summing over all these images. The term $\mathbf{R}_{i,j}\mathbf{y}_t$ remains a 2-D patch, but one that is extracted at location (i,j) from the image at time t. The image $\mathbf{x}$ remains a single image, representing the desired output image to be created. Let us assume that this image aims to become a denoised version of $\mathbf{y}_{t_0}$. This fact is used only within the computation of the weights $w[k,l,i,j,t]$, which matches a reference patch $\mathbf{R}_{k,l}\mathbf{y}_{t_0}$ with the candidate ones $\mathbf{R}_{i,j}\mathbf{y}_t$, both 2-D. The temporal distance, $t - t_0$ can also influence the weight w, just as spatial distances $k-i$ and $l-j$ do in (3).

Assuming that the patches extracted by the operator $\mathbf{R}$ are of size of one pixel,[3] using the same algebraic steps as shown in the previous sub-section we get a closed-form formula for the computation of $\mathbf{x}$

$$\hat{\mathbf{x}}[k,l] = \frac{\sum_{t\in[1,\ldots,T]} \sum_{(i,j)\in N(k,l)} w[k,l,i,j,t]\mathbf{y}_t[i,j]}{\sum_{t\in[1,\ldots,T]} \sum_{(i,j)\in N(k,l)} w[k,l,i,j,t]}. \quad (19)$$

This is a generalization of the NLM to handle video sequences, and was shown to be an effective algorithm in [27].

Let us explain this formula in words, as a similar structure will serve us in the next Section quite similarly. Each output

[3]We remind the reader that the patch size related to the weights-computation is different than the one in the penalty function itself in general.

pixel is computed as a weighted average of pixels in its 3-D neighborhood in the input sequence $\mathbf{y}$. We would like pixels that originally had the same grey level to have high weights, and the weights are computed in a way that reflects this.

By taking a slightly different perspective, we can regard the weight $w[k,l,i,j,t]$ as reflecting the (relative) probability that the pixel (k,l) in the image to be denoised has gone to (or come from) the pixel (i,j) in the image $\mathbf{y}_t$. This is a very basic motion estimation approach for each pixel. However, since for each (k,l), there may be several nonzero values, this implies that each pixel may be assigned with *several* motion vectors, and not just one, as in classical motion estimation methods. Indeed, the above expression suggests that all motion vectors are allowed, and considered according to the probability that they actually took place based on the patch matching they exhibit. This way, one could consider the above as a fuzzy motion estimation process, rather than an explicit one. Fig. 3 presents the concept of fuzzy motion estimation. A specific pixel on the scene at time t (the left image), has several equally probable destinations in the surrounding images (including the same image at time t). This can be done for any pixel, some having many probable destinations, and some only a few.

An important benefit to the above fuzziness is the ability to get a stronger denoising effect. Whereas exact motion estimation leads to a correspondence between the reference patch and a single match in every other image, the NLM approach may find several good matches due to spatial (and not just temporal) redundancy. As seen in Fig. 3, several very good matches can be found and, thus, when averaged, lead to a more effective noise suppression due to the independence between the noise in each of those patches.

Generalization of the NLM approach to video denoising was shown to be very effective [27]–[29], leading to state-of-the-art results, while leaning strongly on the fuzziness of the estimated motion. These methods' successes encourage us to seek ways to exercise the same fuzzy motion estimation concept in other video processing tasks, where it is commonly assumed that explicit motion estimation is a mandatory step. One such task is super-resolution. In the rest of this paper we focus on a super-resolution scheme that relies on fuzzy, rather than explicit, motion estimation, generalizing the NLM as presented above.

IV. Super Resolution With No Explicit Motion Estimation

A. Super-Resolution Penalty Function

The main advantage of the fuzzy motion approach is its flexibility, allowing it to handle complex scenes. Whereas classic super resolution methods make many limiting assumptions regarding the motion in the scene, this is not the case with fuzzy motion estimation, which can handle local motions, occlusions, and so on. Our goal in this section is to exploit this flexibility to perform super-resolution on sequences with arbitrary motion patterns, thus avoiding the global motion limitation.

Developing a super resolution algorithm will again begin with writing a proper penalty term and minimizing it. The input to this algorithm is the set of low resolution images $\mathbf{y}_t$. For clarity

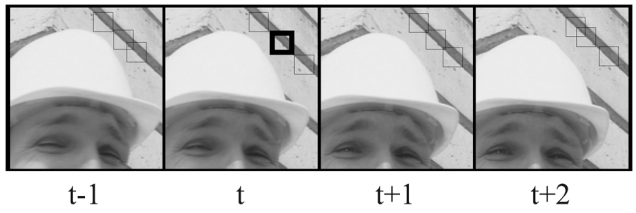


Fig. 3. NLM's fuzzy motion estimation: A patch in the reference image at time t (marked with a thick line) has several probable locations in the other images, and also within the same image itself (marked with narrow lines).

of description, we target one high resolution image $\mathbf{X}$ at a time (instead of working on the entire sequence at once). The usage of a capital letter to denote the output image, and lower-case letters to denote the input sequence, serves to remind us that they are indeed of different scales.

Since we want to exploit the insight gained in previous sections, we aim at defining a penalty function that is similar to that written for the video denoising problem in (18). However, the target image $\mathbf{X}$ and the input images $\mathbf{y}_t$ are not of the same scale, forcing a change to the penalty term. This change should reflect the fact that $\mathbf{X}$ undergoes blurring and decimation steps, in order to account for the different scales. This leads us to the following preliminary (and not final!) proposal:

$$\eta_{SR}(\mathbf{X}) = \frac{1}{2}\sum_{(k,l)\in\Omega}\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N(k,l)} w[k,l,i,j,t] \times\|\mathbf{R}_{k,l}\mathbf{DHX}-\mathbf{R}_{i,j}\mathbf{y}_t\|_2^2+\lambda TV(\mathbf{X}). \quad (20)$$

Introduced into the penalty term are two operators we have met in the introduction: $\mathbf{H}$—the blurring operator, and $\mathbf{D}$—the decimation operator. Applying these operators on $\mathbf{X}$ simulates the imaging process, bringing $\mathbf{X}$ to the same resolution as the input sequence $\mathbf{y}_t$. The additional TV (Total Variation) expression comes to regularize the deblurring that should take place in this expression [38], forcing piece-wise smoothness of the desired image, by accumulating the norms of the gradients with L_1 norm.

Taking a close look at the penalty term we have just written reveals that it cannot provide super resolution reconstruction. By denoting $\mathbf{x}=\mathbf{DHX}$, it is clearly seen the above is identical to the video denoising penalty term in (18) for $\mathbf{x}$. This means that minimizing this penalty term effectively divides the problem into two parts—the first performs denoising of the input sequence, and the second that interpolates and de-blurs the resulting image.

The reason this penalty term is not able to perform super resolution is the fact that the operator $\mathbf{R}_{k,l}$ "sees" only $1/s^2$ of the pixels in the image (s is the decimation factor) because of the decimation that precedes it. In order to solve this problem, we reverse the order of the patch extraction and decimation, so that a patch is first extracted from $\mathbf{HX}$, and then decimated to the same resolution level as $\mathbf{y}_t$ to be compared to a patch from it. This change requires a change in the operator $\mathbf{D}$ as well, as we no longer use the constant decimation grid of the image. Instead, we introduce the notation $\mathbf{D}_p$—a patch decimation operation that ensures that the center pixel of the patch is on the decimation grid.

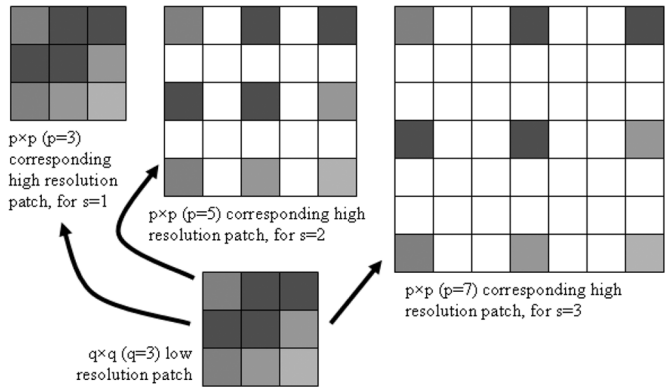


Fig. 4. Relation between the patch sizes q and $p = (q-1)s+1$ demonstrated on specific examples: $[q, s]$ = [3,1], [3,2], and [3,3].

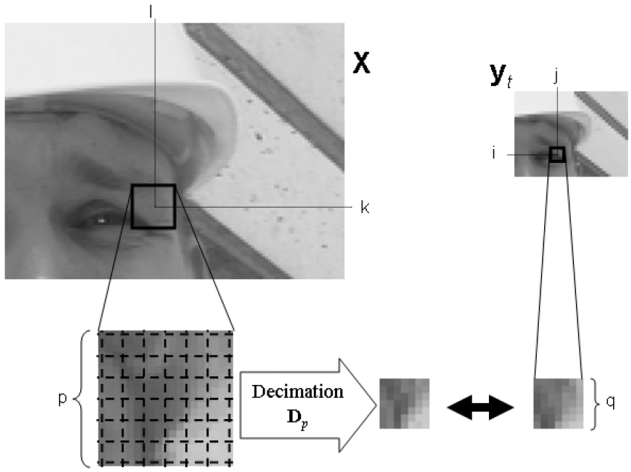


Fig. 5. Description of the expression $\mathbf{D}_p\mathbf{R}_{k,l}^H\mathbf{H}\mathbf{X} - \mathbf{R}_{i,j}^L\mathbf{y}_t$ in (21).

Since there are two different scales (and sizes) of patches used, we also must differentiate between the patch extraction operators. To this end, we denote by $\mathbf{R}_{k,l}^H$ and $\mathbf{R}_{i,j}^L$ the high- and low-resolution patch extraction operators respectively. Their sizes are linked through the value of the scaling parameter s: whereas $\mathbf{R}_{i,j}^L$ extracts a patch of size $q \times q$ pixels (arbitrary), $\mathbf{R}_{k,l}^H$ extracts a patch of size $p \times p$ pixels, where $p = s(q-1)+1$. The relation between these sizes is further explained in Fig. 4. This transforms the penalty term in (20) into

$$\eta_{SR}(\mathbf{X}) = \sum_{(k,l)\in\Omega}\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t] \times \left\|\mathbf{D}_p\mathbf{R}_{k,l}^H\mathbf{H}\mathbf{X} - \mathbf{R}_{i,j}^L\mathbf{y}_t\right\|_2^2 + \lambda TV(\mathbf{X}). \quad (21)$$

Fig. 5 presents a block diagram to explain this penalty. As can be seen, different size patches are extracted from $\mathbf{x}$ and $\mathbf{y}_t$, and brought to the same grounds by a decimation operation.

Note: The term $(i,j) \in N(k,l)$ is no longer applicable as before, since now (i,j) and (k,l) are pixels on different resolution grids. Therefore, we introduce a new notation N^L referring to the equivalent low-resolution neighborhood of (k,l), i.e., $(i,j) \in N^L(k,l)$ is short for the full notation (i,j) s.t. $(si, sj) \in N(k,l)$.

The penalty term we have defined above is the one we shall focus on hereafter. Next, we show how to use this penalty term as the basis for a practical and simple super resolution algorithm. The first step we take is a separation of the deblurring from the fusion of the images. Using the substitution $\mathbf{Z} = \mathbf{H}\mathbf{X}$, the first problem we have is the estimation of $\mathbf{Z}$ from the measurements $\{\mathbf{y}_t\}_{t=1}^T$. This is done by minimizing the energy function

$$\eta_{SR}^A(\mathbf{Z}) = \sum_{(k,l)\in\Omega}\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t] \times \left\|\mathbf{D}_p\mathbf{R}_{k,l}^H\mathbf{Z} - \mathbf{R}_{i,j}^L\mathbf{y}_t\right\|_2^2. \quad (22)$$

Once found, we use the found $\hat{\mathbf{Z}}$ to estimate $\mathbf{X}$ by minimizing

$$\eta_{SR}^B(\mathbf{X}) = \|\hat{\mathbf{Z}} - \mathbf{H}\mathbf{X}\|_2^2 + \lambda TV(\mathbf{X}). \quad (23)$$

The second part is a classic simple deblurring problem. This is of course an under-determined problem (since $\mathbf{H}$ is usually singular), and needs regularization. We will not discuss how to solve it here, see [39]–[44] for a background view of this field, along with several state-of-the-art deblurring methods.

The idea of breaking the super-resolution task into two parts—fusing the inputs and then deblurring—has been suggested previously in [13], [18], [20], and [21]. In the case of pure translational motion, and when both problems are treated as maximum-likelihood (which is not done here), such a separation is equivalent to the joint solution. It is important to note that in the derivation proposed above, the separation is not optimal, and leads to inferior results. However, the separation allows for a much simpler algorithm (both conceptually and implementation-wise), so the sub-optimality is the price to be paid for this simplicity.

The fusion stage in (22) seems to lack regularization, and as such, one may question its stability. Stability is gained through the weights—those should be computed in such a way that ensures that every pixel has at least a few "quality" (assigned a meaningful weight) destinations. Furthermore, by also assigning a nonzero weight to the original value of the pixel, further stability is obtained, guaranteeing that in case no appropriate patches are found, the result will default to the initial estimate.

We shall leave the deblurring aside, and focus hereafter on the fusion of the measurements. This will allow us to simplify the description of the algorithm. The next step is deriving the penalty function in (21) with respect to $\mathbf{Z}$, and finding the zero intersection

$$\frac{d\eta_{SR}^A(\mathbf{Z})}{d\mathbf{Z}} = 2\sum_{(k,l)\in\Omega}\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t] \times \left(\mathbf{D}_p\mathbf{R}_{k,l}^H\right)^T\left(\mathbf{D}_p\mathbf{R}_{k,l}^H\mathbf{Z} - \mathbf{R}_{i,j}^L\mathbf{y}_t\right) = 0. \quad (24)$$

This leads to the solution

$$\hat{\mathbf{Z}} = \left[\sum_{(k,l)\in\Omega}\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t]\right.$$

$$\times \left(\mathbf{D}_p\mathbf{R}_{k,l}^H\right)^T\left(\mathbf{D}_p\mathbf{R}_{k,l}^H\right)\Big]^{-1} \cdot \left[\sum_{(k,l)\in\Omega}\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t] \times \left(\mathbf{D}_p\mathbf{R}_{k,l}^H\right)^T\mathbf{R}_{i,j}^L\mathbf{y}_t\right] \quad (25)$$

which can be further simplified, by defining

$$\bar{w}[k,l] = \sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t] \quad (26)$$

leading to

$$\hat{\mathbf{Z}} = \left[\sum_{(k,l)\in\Omega}\bar{w}[k,l]\left(\mathbf{D}_p\mathbf{R}_{k,l}^H\right)^T\left(\mathbf{D}_p\mathbf{R}_{k,l}^H\right)\right]^{-1} \cdot \left[\sum_{(k,l)\in\Omega}\left(\mathbf{D}_p\mathbf{R}_{k,l}^H\right)^T \times\left(\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t]\mathbf{R}_{i,j}^L\mathbf{y}_t\right)\right]. \quad (27)$$

The right term that involves the measurements $\{\mathbf{y}_t\}_{t=1}^T$ performs a series of simple operations that include: (i) Extraction of patches from the measurements; (ii) Zero-padding interpolation of these patches (done by $\mathbf{D}_p^T$); (iii) An accumulation of the resulting patch with a proper weight at the destination location.

Following the same rationale as the one practiced for (9), the matrix to be inverted in the above expression is a diagonal one and positive semi-definite, normalizing the accumulation in each pixel. The term $(\mathbf{D}_p\mathbf{R}_{k,l}^H)^T(\mathbf{D}_p\mathbf{R}_{k,l}^H)$ extracts a patch from location (k,l), scales it down and then up again by zero padding, and finally puts it back into the same original location. Thus, this is a diagonal matrix with 1-es for the pixels surviving this path, and zeros elsewhere. The matrix to be inverted is a weighted sum of such diagonal matrices, and, thus, it is diagonal as well. If every pixel (k,l) gets some accumulation, this matrix is strictly positive definite, and its inversion is permitted.

B. Simplified Numerical Algorithm

We again follow the path of Section III, and propose a special and simplified case where the patch extraction operator $\mathbf{R}_{i,j}^L$ extracts only one pixel. This means that $\mathbf{R}_{k,l}^H$ extracts a patch of size $s\times s$ pixels, to become one pixel after the decimation operation $\mathbf{D}_p$.[4] This simplifies the penalty function in (22), since $\mathbf{D}_p\mathbf{R}_{k,l}^H\mathbf{Z} = \mathbf{Z}[k,l]$ and $\mathbf{R}_{i,j}^L\mathbf{y}_t = \mathbf{y}_t[i,j]$, leading to

$$\eta_{SR}^A(\mathbf{Z}) = \sum_{(k,l)\in\Omega}\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t] \times \left(\mathbf{Z}[k,l]-\mathbf{y}_t[i,j]\right)^2. \quad (28)$$

This functional is separable, handling every pixel in the target image $\mathbf{Z}$ separately. This implies that an independent penalty can be written for every destination pixel

$$\eta_{SR}^A\left(\mathbf{Z}[k,l]\right) = \sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t] \times \left(\mathbf{Z}[k,l]-\mathbf{y}_t[i,j]\right)^2 \quad (29)$$

leading to a closed-form solution

$$\hat{\mathbf{Z}}[k,l] = \frac{\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t]\mathbf{y}_t[i,j]}{\sum_{t\in[1,\ldots,T]}\sum_{(i,j)\in N^L(k,l)} w[k,l,i,j,t]}. \quad (30)$$

In order for this solution to exist, it is enough that for each pixel (k,l) there exists at least one none-zero weight $w[k,l,i,j,t]$.

Notice that the above formula looks exactly the same as the one used for video denoising, posed in (19). Are the two equivalent? The answer is negative, due to two important reasons. First, there is a gap between the definitions of the neighborhoods $N(k,l)$ and $N^L(k,l)$. Whereas for the video denoising this neighborhood is defined simply as the set of all neighbors for the central location (k,l), the neighborhood referred to in (30) considers locations (i,j) that after scaling up by s are neighbors of (k,l).

The second difference between (19) and (30) refers to the weights to be used. As described earlier, we want the weight $w[k,l,i,j,t]$ to reflect the probability that the pixel $\mathbf{Z}[k,l]$ and the pixel $\mathbf{y}_t[i,j]$ originated from the same place. This computation will be based on the similarity of the area around both pixels, in the same manner the weights for the NLM are computed (since they serve a similar purpose), as described in (3). The function f that takes into account the geometric distance between the patches is set to be constant, thereby giving no preference to nearby patches over distant ones. This allows more robustness to various motion patterns, including patterns that contain large motions. However, in some cases, it may be beneficial to assign higher weights to patches closer temporally or geometrically.

However, for matching these two areas, we need to address the fact that the neighborhoods around the pixels in $\mathbf{Z}$ and in $\mathbf{y}_t$ are not of the same scale. We can address this in one of two ways: (i) Down-sample the neighborhood in the high-resolution image and bring it down to the low resolution grid; or (ii) Up-sample the low-resolution neighborhood to match the high-resolution one. In our tests we have found that these two options perform similarly well, with a small advantage to the second (and more complex) one.

As in the denoising case, the computation of the weights requires the use of the unknown $\mathbf{Z}$. Instead, the weights are computed by using an estimated version of $\mathbf{Z}$ being a scaled-up version of the reference frame we aim to super-resolve. This scale-up is done using a conventional image interpolation algorithm such as bilinear, bicubic or the Lanczos method [45], [46].

The interpolated images are only crude estimates of the desired outcome, and, therefore, the weights computed by relying on these estimated are also somewhat crude. Since after one iteration of the algorithm we obtain a better estimate, it is possible to use these new estimates for re-computing the weights, and computing a better still estimate of the desired outcome. This

[4]Note that this change of patch-sizes applies only to the penalty term itself, and does not effect the patch size for the weights computation.

TABLE I
SUMMARY OF THE CORE OF THE SUPER RESOLUTION ALGORITHM—PROCESSING ONE IMAGE USING A GIVEN INITIAL ESTIMATE FOR THE SUPER-RESOLVED SEQUENCE

Objective: Estimate $\hat{\mathbf{X}}_{t_0}$ – a super-resolved image at time t_0.
Inputs: • $\mathbf{y}_t$ - input set of low resolution and noisy images.
• s - the desired scaling factor (any integer).
• q - the size of the low resolution patch ($\mathbf{R}^L$).
• p - size of the high resolution patch ($\mathbf{R}^H$): $p = s(q-1)+1$.
• $\{\mathbf{Y}_t\}_{t=1}^T$ - An initial estimate of the super-resolved sequence.
Initialization:
- Set $\mathbf{Z}_{t_0} = \mathbf{Y}_{t_0}$.
- Set $\mathbf{V}$ and $\mathbf{W}$ to be zero images of the same size as $\mathbf{Z}_{t_0}$.

Fusion: For each $(k,l) \in \Omega$ and each (i,j,t) such that $(si, sj, t) \in N(k,l,t_0)$
1) *Compute Weights*: $w[k,l,i,j,t] = \exp\{-\|\hat{R}_{k,l}\mathbf{Z}_{t_0} - \hat{R}_{si,sj}\mathbf{Y}_t\|_2^2/2\sigma^2\}$.
2) *Accumulate Inputs*:
   - Extract the low-resolution patch $\mathbf{R}_{i,j}^L\mathbf{y}_t$,
   - upscale it by zero-filling,
   - accumulate it in its proper location

$$\mathbf{V} = \mathbf{V} + w[k,l,i,j,t]\left(\mathbf{R}_{k,l}^H\right)^T \mathbf{D}_p^T \mathbf{R}_{i,j}^L \mathbf{y}_t.$$

3) *Accumulate Weights*: For each patch accumulated above, apply the following update of the weight image

$$\mathbf{W} += w[k,l,i,j,t]\left(\mathbf{R}_{k,l}^H\right)^T \mathbf{D}_p^T \mathbf{R}_{i,j}^L \mathbf{1}.$$

Normalization: Set $\mathbf{Z}_{t_0}[k,l] = \mathbf{V}[k,l]/\mathbf{W}[k,l]$.
Deblurring: Minimize

$$\eta_{SR}^B(\mathbf{X}) = \|\mathbf{Z}_{t_0} - \mathbf{H}\mathbf{X}\|_2^2 + \lambda TV(\mathbf{X}) \quad w.r.t. \quad \mathbf{X},$$

and set the result to be $\hat{\mathbf{X}}_{t_0}$.
Result: The output is $\hat{\mathbf{X}}_{t_0}$.

process may be iterated several times, although in our simulations we use only two such iterations (i.e., one iteration that relies on the interpolated frames for computing the weights, and a second iteration that relies on the results of the first iteration). This means that all frames are first processed using the interpolated frames for the weights computation, and only then is the second iteration applied. If throughput constraints do not allow this, it is possible to compute the weights in the second iteration while still using the interpolated versions of all frames, and a new estimate is only used for the currently processed frame. This requires some minor modifications to the weights computation.

C. Proposed Algorithms—A Summary

We now summarize the two algorithms previously described. First, we describe in Tables I and II the general super-resolution algorithms that were developed in this section. The simpler version that uses low-resolution patches of one pixel is obtained

TABLE II
SUMMARY OF THE ENTIRE SUPER RESOLUTION ALGORITHM—USED FOR RESOLVING ALL THE FRAMES IN THE SEQUENCE

Objective: Estimate $\left\{\hat{\mathbf{X}}_t\right\}_1^T$ – the super-resolved sequence.
Inputs: • $\{\mathbf{y}_t\}_{t=1}^T$ - input set of low resolution and noisy images;
• s - the desired scaling factor (any integer).
Preprocessing: Compute $\{\mathbf{Y}_t\}_{t=1}^T$ – a Lanczos interpolation of the input sequence.
Iterating: Perform the following steps per each iteration
1) Super-resolve each image using the algorithm outlined in Table I, obtaining a new estimate for the super-resolved sequence $\left\{\hat{\mathbf{X}}_t\right\}_1^T$.
2) Update current estimation $\{\mathbf{Y}_t\}_{t=1}^T = \left\{\hat{\mathbf{X}}_t\right\}_1^T$, and use as initial estimate for the following iteration.

Result: The output is $\left\{\hat{\mathbf{X}}_t\right\}_1^T$.

by using the proper assignments for $\mathbf{R}_{k,l}^H$ and $\mathbf{R}_{i,j}^L$ (while not changing $\hat{\mathbf{R}}$).

The complexity of these two algorithms is essentially the same as that of their NLM counterparts, with the addition of a deblurring process, which can be considered negligible. The core of the algorithm, which also requires most of the computations, is computing the weights. Considering a nominal case with a search area of 31×31 low-resolution pixels in the spatial domain, and 15 images in the temporal axis, we have ≈14,000 pixels in this spatio-temporal window. For each pixel in the search area, the block difference is computed, with a block size of 13×13 (high-res.) pixels. Thus, there is a total of almost 2,400,000 operations per pixel. This amount is of course very large, and must be reduced in order to make the algorithm practical. We will now describe a few speedup methods for the proposed algorithm. Several of the methods to speedup the NLM algorithm were suggested originally in [47], and were adopted in our simulations.

1) Computing the weights can be done using block differences in the low-resolution images, instead of on the interpolated images. This saves a factor of $\approx s^2$.
2) Computing fast estimations for the similarity between blocks, such as the difference between the average grey level or the average direction of the gradient, can eliminate many nonprobable destinations from further processing. Such an approach was suggested in [47], and was found to be very effective for the original NLM algorithm.
3) If the patch used to compute the weights is rectangular with equal weights to all pixels, the computation of the weight can be sped up dramatically. Using the Integral Image ($II(x,y) = \sum_{i=1}^{x}\sum_{j=1}^{y} I(i,j)$), the block difference can be computed using a small constant number of calculations per pixel, regardless of block size. Using such a patch structure has only a slight effect on the quality of the outputs [48].

Fig. 6. Results for the synthetic text sequence. (a) Original (ground-truth) image. (b) Pixel replicated image, 13.47 dB. (c) Lanczos interpolation, 13.84 dB. (d) Deblurred Lanczos interpolation, 13.9 dB. (e) Result of shift-and-add algorithm [18], [20], 18.4 dB. (f) Result of proposed algorithm, 18.48 dB.

4) A coarse-to-fine approach reduces the effective search area for each pixel, thus reducing the number of required calculations.
5) The search area can be adapted to the temporal distance, making it small for nearby images and growing for images further away, also reducing the total effective search area.
6) Since weights are symmetrical, half the calculations can be saved, by applying computed weights to both participating patches.
7) Since most of the algorithm is local in nature, it lends itself easily to parallelization. As 4 and 8 processor configurations are currently widely available, this can be used for speeding up the algorithm by about one order of magnitude.

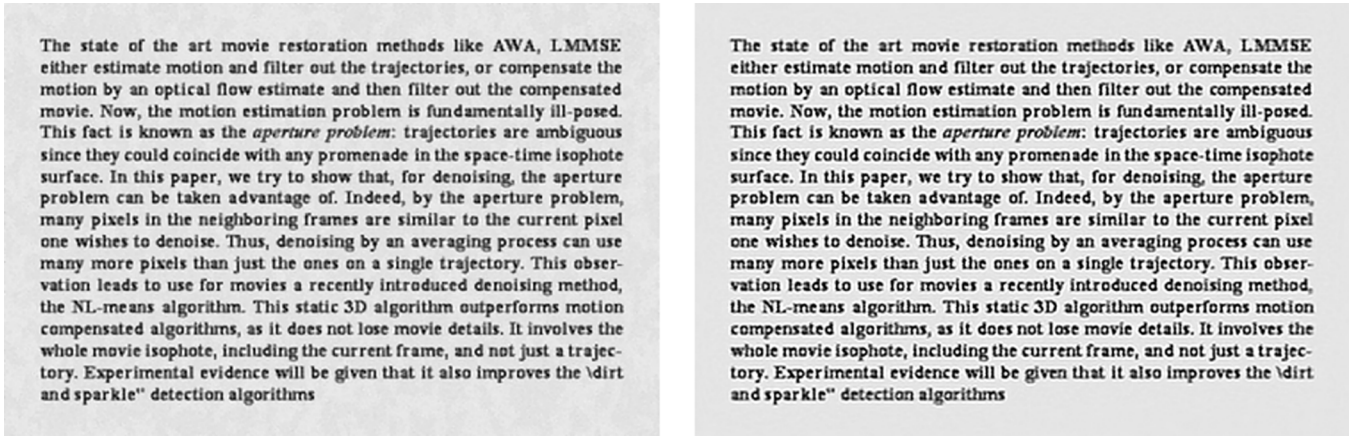


Fig. 7. Results for the synthetic text sequence, with one image missing. Left: result of shift-and-add algorithm [18], [20], 18.16 dB. Right: result of proposed algorithm, 17.7 dB.

These suggested speedup methods can reduce the complexity by at least 3–4 orders of magnitude without a noticeable drop in the quality of the outputs. This makes the proposed algorithm practical. As for the memory requirement, the proposed algorithm uses approximately as much memory as required to hold the entire processed sequence in the high resolution scale. This is usually not a limitation.

V. Experimental Results

A. Experiments

In this section, we validate the potential of the proposed algorithm (we use the simpler version discussed in Section IV-B) by presenting the obtained results of processing several image sequences. We start with one synthetic (text) sequence with global motion that comes to demonstrate the conceptual super-resolution capabilities of the proposed algorithms. Then we turn to three real sequences with a general motion pattern. As there are no published methods that perform super resolution on such general sequences, the comparison we provide is to a single image up-sampling using the Lanczos algorithm [45], [46] that effectively approximates the Sinc interpolation.

The Lanczos interpolation also serves as a benchmark for the synthetic case, even though super resolution algorithms that rely on global motion estimation can process this specific sequence quite successfully. This is because the benchmark for the proposed algorithm should also be able to handle all sequences, and not limited to the global motion case.

All the tests in this section are prepared in the following manner: An input sequence is blurred using a 3×3 uniform mask, decimated by a factor of 1:3 (in each axis), and then contaminated by an additive noise with $std = 2$. All images are in the input range [0,255].

The first test is a very simple synthetic test, that motion-estimated-based super-resolution algorithms are expected to resolve well, intended to show that the proposed algorithm indeed achieves super resolution. A text image is used to generate a 9-image input sequence, by applying integer displacements prior to blurring, decimation and the addition of noise. The displacements are chosen so that the entire decimation space is covered (i.e., $dx = \{0, 1, 2\}$ and $dy = \{0, 1, 2\}$). The result for this test is shown in Fig. 6, including a comparison to the results obtained by the regularized shift-and-add algorithm [18], [20], which is a conventional motion-estimation-based super-resolution algorithm . The block size used for computing the weights $(\hat{\mathbf{R}})$ was set to 31×31, since the motion in the sequence is limited to displacements, and a larger block allows capturing the true displacement better (for true sequences, this size will be greatly reduced, as explained later). The value of σ that moderates the weights was set to 7.5 (due to the large differences between white and black values in the scene). Two iterations were ran on the entire sequence, the first iteration used for computing the weights for the second iteration.

We also ran a similar test, displaying the behavior of the proposed algorithm and shift-and-add approach when one of the images from the set is omitted. The results for this test are displayed in Fig. 7.

Even though the proposed algorithm does not exploit the fact the motion in the sequence is only global translation, it still achieves good results. The text is almost completely readable in the result of the proposed algorithm. This shows that sub-pixel details (relative to the low resolution grid) are indeed recovered by the proposed algorithm. In terms of Peak Signal to Noise Ratio (PSNR),[5] a 3:1 pixel-replication in each axis leads to 13.47 dB, the Lanczos interpolation gives 13.84 dB, a deblurred Lanczos interpolation gives 13.9 dB, the regularized shift-and-add gives 18.4 dB, and the proposed algorithm gives 18.48 dB, slightly out-performing the classic approach. For the test with one image omitted, the shift-and-add gives 18.16 dB, while the proposed algorithm slightly under-performs with a result of 17.7 dB. The similarity in performance between the shift-and-add approach and the proposed approach attests to the power of the proposed method, as even though it does not rely directly on the global motion assumption, it achieves similar results to a successful method that does.

It is obvious that while the PSNR measures of the shift-and-add approach and the proposed algorithm are similar, the results of the shift-and-add are visually more pleasing. This is to be expected, since this approach is able to fully benefit

[5]Defined as $20 \log_{10}(255/\sqrt{MSE})$, where MSE is the mean-squared-error obtained per pixel.

Fig. 8. Results for the 3rd, 8th, 13th, 18th, 23rd, and the 28th frames from the "Miss America" sequence. From left to right: Low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.

Fig. 9. Results for the 3rd, 8th, 13th, 18th, 23rd, and the 28th frames from the "Foreman" sequence. From left to right: Low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.

from the global motion assumption, while the proposed algorithm does not. This is due to the proposed algorithm trading off quality for increased robustness to general motion patterns (which the shift-and-add approach can't cope with).

When observing the resulting images of the proposed algorithm, they seem somewhat over-sharpened. In fact, applying less deblurring iterations results in a more visually pleasing result, however, the objective PSNR measure yields a lower value for these images. It seems like a stronger deblurring mechanism, applied to the results of the fusion stage, can generate results that are both visually pleasing and relatively accurate reconstructions. Since the novelty of this paper lies in the fusion stage, we have not experimented with other such mechanisms.

We now turn to test the proposed algorithm on real sequences, containing general motion patterns, which represent the actual problem the proposed algorithm is designed to tackle. The 3 sequences we test on are usually used for testing compression algorithms, referred to as "Miss America," "Foreman, " and "Suzie." The super-resolution results for these sequences (scaling up by a factor of 3 to 1 in each axis) are shown in Figs. 8–10, respectively. All of these sequences (original, low-quality, Lanczos interpolation and processed sequences) appear in the first author's website, at http:\\www.cs.technion.ac.il\~matanpr\NLM-SR.

Each result shows several (3rd, 8th, 13th, 18th, 23rd, and the 28th) frames from each sequence (each row represents one frame). For each frame, the input low resolution image, the ground truth image, the result of the Lanczos interpolation, and the result of the proposed algorithm are presented from left to right. It is very evident that the results of the proposed algorithm are dramatically better when compared to the Lanczos interpolation—the output is sharper, contains more details, and is more visually pleasing. It is important to note, that we display

TABLE III
MEAN-PSNR RESULTS FOR THE THREE TEST SEQUENCES. THIS TABLES GIVES THE PSNR FOR THE PIXEL-REPLICATED SEQUENCES, THE LANCZOS INTERPOLATION, THE LANCZOS INTERPOLATION WITH CONSEQUENT DEBLURRING, AND THE RESULTS OF THE PROPOSED ALGORITHM (FIRST AND SECOND ITERATION)

Sequence	Pixel Replication	Lanczos Interpolation	Deblurred Lancozs Interpolation	Our Result (1st Iteration)	Our Result (2nd iteration)
Miss-America	31.43	34.08	34.26	35.91	34.97
Foreman	28.99	31.25	31.13	34.27	33.13
Suzie	30.02	31.4	31.59	33.74	33.32

Fig. 10. Results for the 3rd, 8th, 13th, 18th, 23rd, and the 28th frames from the "Suzie" sequence. From left to right: Low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.

the results of the Lanczos interpolation without any postdeblurring. Applying deblurring to the Lanczos interpolation results in a slightly sharper image, but one that also contains unwanted artifacts [as can be seen in Fig. 6(c) and (d)] and the PSNR measures are also equivalent.

In processing all of these sequences, all 30 frames participated in the reconstruction of each image. The similarity block size used for computing the weights ($\hat{\boldsymbol{R}}$) was 13×13, and did not vary between the different tests. The search area (i.e., the size of the neighborhood N) was determined manually, in order to reduce computation time:[6] a search area of 13×13 pixels in each image for the "Miss America" sequence, 21×21 for the "Foreman" sequence, and 31×31 for the "Suzie" sequence. The parameter σ was set to 2.2 for all sequences. Two iterations were again used, where the first provides the updated weights for the second iteration. Just to put things into perspective, we add that the entire simulation is done on Matlab, running on a regular Pentium 3 GHz (2-GB RAM) machine, requiring approximately 20 s per frame for the most demanding case—the "Suzie" sequence with high-resolution frame size of 210×250 pixels.

Table III presents the mean PSNR for each of the three test sequences, evaluating the quality of the pixel-replicated scaled-up sequence, the Lanczos interpolation, and the results we obtain after the first and the second iterations of the proposed algorithm. While the super-resolution results show higher PSNR, we see that i) the PSNR gain is mild, and does not reflect the visual quality of the sequences obtained; and ii) Even though the first iteration result is typically inferior in visual quality, its PSNR assessment tends to be higher.

Beyond the usual and common complaint on the inability of PSNR to grasp image quality in general, it seems that our algorithm in particular is very sensitive to this measure. One possible reason for this could be the fact that in avoiding explicit motion estimation, the result we obtain is slightly shifted (or warped) with respect to the reference it aims to recover. This, of-course, leads to a disastrous drop in PSNR. Another reason that may account for such behavior is the over-sharpening that the deblurring stage introduces. We intend to further explore this matter in our future work, as we hope to provide visually pleasing results that are also backed up by good PSNR behavior.

B. Discussion

The results obtained are very encouraging, being artifact-free and of high quality. Still, we believe that these results could be further improved and extended in various ways. The manual selection of the various parameters could be replaced by an adaptive setting of their values. One specific parameter of importance is the filtering parameter σ, effecting the weights computation. Selecting this parameter in a locally adaptive way seems like a natural step to take, allowing the algorithm to adapt better to both small and large details.

Similarly, an adaptive selection of the window size (both for the weights computation and in the algorithm itself), as done in [28] and [29], can improve the algorithm's performance. A large block can help estimate the motion more accurately. A small block can better adapt to general and varying motion. Adaptively selecting the window size, as in [28], is expected to deliver much better results than just settling for a global tradeoff.

A similar, yet somewhat different concept, is a better control over the search area. While the estimated motion pattern is fuzzy, reducing the number of candidate locations can help reduce complexity and improve the results. Relying on some coarse optical flow computation as a preceding step for the fuzzy motion estimation can bring these improvements. An alternative is computing a coarse fuzzy motion pattern in a coarser scale,

[6]Applying a bigger search area results in an only slightly different super-resolution outcome.

and proceeding to search in the actual scale only around the likely coarse destination.

A somewhat different direction for further research is adopting the fuzzy motion concept in other approaches to the super resolution problem. In the video denoising field, the very successful NLM was followed by even more successful contributions using fuzzy motion, such as the SW3D [29] and the method relying on sparse and redundant representations [30]. We believe that these approaches, and others, can also serve as the basis for more successful super resolution algorithms that do not rely on explicit motion estimation.

VI. Summary

This paper introduces a novel and successful super resolution algorithm, which does not rely on explicit motion estimation. Instead, a local and patch-based approach is combined with fuzzy motion estimation. This allows the algorithm to handle diverse motion fields, instead of the common global motion limitation that characterizes traditional super-resolution methods.

The algorithm developed here is an extension of the very successful video denoising Nonlocal-Means algorithm, reported in [31]. Its ability to denoise image sequences without explicit motion estimation, which was previously considered necessary in this field, led us to start our search for an explicit-motion-estimation-free super resolution algorithm there.

By analyzing the forces existing in the NLM algorithm, we were able to write an energy function, whose minimization leads to a powerful denoising algorithm, of which the NLM is a special case. Then, we extended this energy function to the super resolution problem, by making the necessary changes. Starting from this energy minimization, we developed a simple yet very effective super resolution algorithm.

Finally, we have processed some real sequences that contain complex motion patterns with the proposed algorithm. The obtained results are artifact free and of high quality, thus proving the ability of the proposed method to handle general sequences. Lastly, we have made several suggestions as to directions for further research.

Appendix A
Proofs

Theorem 1: Assuming that

1) $(i,j) \in N(k,l)$ implies $(k,l) \in N(i,j)$, and
2) $w[k,l,i,j] = w[i,j,k,l]$,

the following two equalities hold true:

$$\sum_{(k,l)\in\Omega}\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}\mathbf{x} = \sum_{(k,l)\in\Omega}\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{i,j}^T\mathbf{R}_{i,j}\mathbf{x} \quad \text{(A1)}$$

and

$$\sum_{(k,l)\in\Omega}\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{i,j}^T\mathbf{R}_{k,l}\mathbf{x} = \sum_{(k,l)\in\Omega}\sum_{(i,j)\in N(k,l)} w[k,l,i,j]\mathbf{R}_{k,l}^T\mathbf{R}_{i,j}\mathbf{x}. \quad \text{(A2)}$$

Proof: We start with (A1). Both sides of this equation describe processes where image patches are piled with proper weights. Consider an arbitrary location (m,n) in the resulting image, and lets see which patches are accumulated at this point as a center one. Recall that the operator $\mathbf{R}_{k,l}^T$ assigns a patch to location (k,l).

Looking at the Left-Hand-Side (LHS), the only patch to be positioned in location (m,n) (i.e., to be multiplied by $\mathbf{R}_{m,n}^T$) is $\mathbf{R}_{m,n}\mathbf{x}$, but this is done several times, accumulating a total weight of $\sum_{(i,j)\in N(m,n)} w[m,n,i,j]$.

The Right-Hand-Side (RHS) performs a slightly more involved accumulation. Among all (k,l) pixels in the image, only those that are neighbors to (m,n) are relevant, and each of those contributes one patch of the form $\mathbf{R}_{m,n}\mathbf{x}$ with a weight $w[k,l,m,n]$. Thus, again, we obtain that only one patch is used, but accumulated several times. The total weight of this accumulation is given by $\sum_{(k,l)\in N(m,n)} w[k,l,m,n]$. Due to the symmetry of the weights, we have

$$\sum_{(k,l)\in N(m,n)} w[k,l,m,n] = \sum_{(k,l)\in N(m,n)} w[m,n,k,l] \quad \text{(A3)}$$

which is the same as the LHS accumulated weight.

Turning to the equality in (A2), we adopt a similar analysis. Starting this time with the RHS, the overall patches to be accumulated in location (m,n) (i.e., those multiplied by $\mathbf{R}_{m,n}^T$) are given by

$$\sum_{(i,j)\in N(m,n)} w[m,n,i,j]\mathbf{R}_{i,j}\mathbf{x}.$$

In the LHS, only pixels (k,l) that have (m,n) in their neighborhood are relevant in the outer summation. For those, only one neighbor, the (m,n) one, is relevant, as this is the only one multiplied by $\mathbf{R}_{m,n}^T$, and, thus, positioned in the location we consider. Thus, the accumulation in this case becomes

$$\sum_{(k,l)\in N(m,n)} w[k,l,m,n]\mathbf{R}_{k,l}\mathbf{x} \quad \text{(A4)}$$

and due to the symmetry of the weights, this is the same as the RHS term. □

Theorem 2: The matrix $\mathbf{A} = \lambda\cdot\mathbf{I} + \sum_{(k,l)\in\Omega} \bar{w}[k,l]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}$ is diagonal and positive definite.

Proof: Consider first the term $\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}$ for an arbitrary (k,l) position. This matrix is positive semi-definite by definition, due to the multiplication of a matrix by its adjoint. Furthermore, as an operator that manipulates an image, the multiplication by $\mathbf{R}_{k,l}$ extracts a patch from location (k,l). The later multiplication by $\mathbf{R}_{k,l}^T$ creates a zero-filled image, with the same patch returned to the (k,l) location. Thus, this matrix is a diagonal matrix, with ones on the main diagonal for pixels belonging to the patch, and zeros elsewhere.

We assume that the weights $w[k,l,i,j]$ and, thus, also $\bar{w}[k,l]$, are non-negative by their definition, and, thus, the matrix $\sum_{(k,l)\in\Omega} \bar{w}[k,l]\mathbf{R}_{k,l}^T\mathbf{R}_{k,l}$ is a non-negative weighted average of positive semi-definite and diagonal matrices. Thus, the result is also diagonal and semi-definite. The addition of $\lambda\cdot\mathbf{I}$ preserves the diagonal structure, and lifts the smallest

eigenvalue of the overall matrix to $\lambda > 0$, thus leading to a strictly positive-definite matrix, as claimed. □

REFERENCES

[1] T. S. Huang and R. Y. Tsai, "Multi-frame image restoration and registration," *Adv. Comput. Vis. Image Process.*, vol. 1, pp. 317–339, 1984.

[2] S. P. Kim, N. K. Bose, and H. M. Valenzuela, "Recursive reconstruction of high resolution image from noisy undersampled multiframes," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 38, no. 6, pp. 1013–1027, Jun. 1990.

[3] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graph. Models Image Process.*, vol. 53, pp. 231–239, 1991.

[4] S. Peleg, D. Keren, and L. Schweitzer, "Improving image resolution using subpixel motion," *CVGIP: Graph. Models Image Process.*, vol. 54, pp. 181–186, Mar. 1992.

[5] H. Ur and D. Gross, "Improved resolution from subpixel shifted pictures," *CVGIP: Graph., Models, Image Process.*, vol. 54, pp. 181–186, March 1992.

[6] R. R. Schultz and R. L. Stevenson, "Extraction of high-resolution frames from video sequences," *IEEE Trans. Image Process.*, vol. 5, no. 6, pp. 996–1011, Jun. 1996.

[7] A. J. Patti, M. I. Sezan, and M. A. Tekalp, "Superresolution video reconstruction with arbitrary samplinglattices and nonzero aperture time," *IEEE Trans. Image Process.*, vol. 6, no. 8, pp. 1064–1076, Aug. 1997.

[8] R. C. Hardie, K. J. Barnard, and E. E. Armstrong, "Joint MAP registration and high-resolution image estimation using asequence of undersampled images," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp. 1621–1633, Dec. 1997.

[9] M. Elad and A. Feuer, "Restoration of single super-resolution image from several blurred, noisy and down-sampled measured images," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp. 1646–1658, Dec. 1997.

[10] N. R. Shah and A. Zakhor, "Resolution enhancement of color video sequences," *IEEE Trans. Image Process.*, vol. 8, no. 6, pp. 879–885, Jun. 1999.

[11] A. Zomet and S. Peleg, "Efficient super-resolution and applications to mosaics," in *Proc. Int. Conf. Pattern Recognition*, Sep. 2000, pp. 579–583.

[12] N. Nguyen, P. Milanfar, and G. H. Golub, "A computationally efficient image superresolution algorithm," *IEEE Trans. Image Process.*, vol. 10, no. 4, pp. 573–583, Apr. 2001.

[13] 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.

[14] C. Tom and A. Katsaggelos, "Resolution enhancement of monochrome and color video using motion compensation," *IEEE Trans. Image Processing*, vol. 10, no. 2, pp. 278–287, Feb. 2001.

[15] Y. Altunbasak, A. J. Patti, and R. M. Mersereau, "Super-resolution still and video reconstruction from MPEG-coded video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 4, pp. 217–226, Apr. 2002.

[16] S. Baker and T. Kanade, "Limits on super-resolution and how to break them," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 9, pp. 1167–1183, Sep. 2002.

[17] S. Park, M. Park, and M. G. Kang, "Super-resolution image reconstruction, a technical overview," *IEEE Signal Process. Mag.*, vol. 20, no. 5, pp. 21–36, May 2003.

[18] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Robust shift and add approach to superresolution," in *Proc. SPIE Conf. Applications of Digital Signal and Image Processing*, Aug. 2003, pp. 121–130.

[19] Z. Lin and H.-Y. Shum, "Fundamental limits of reconstruction-based superresolution algorithms under local translation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 83–97, Jan. 2004.

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

[21] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Advances and challenges in superresolution," *Int. J. Imag. Syst. Technol.*, vol. 14, pp. 47–57, Aug. 2004.

[22] T. Gotoh and M. Okutomi, "Direct super-resolution and registration using raw CFA images," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, Jul. 2004, vol. 2, pp. 600–607.

[23] G. T. Clement, J. Huttunen, and K. Hynynen, "Superresolution ultrasound imaging using back-projected reconstruction," *J. Acoust. Soc. Amer.*, vol. 118, pp. 3953–3960, 2005.

[24] P. Vandewalle, S. Susstrunk, and M. Vetterli, "A frequency domain approach to registration of aliased images with application to superresolution," *EURASIP J. Appl. Signal Process.*, no. 71459, 2006.

[25] J. Chung, E. Haber, and J. Nagy, "Numerical methods for coupled super resolution," *Inv. Probl.*, vol. 22, pp. 1261–1272, 2006.

[26] H. F. Shen, L. P. Zhang, B. Huang, and P. X. Li, "A MAP approach for joint motion estimation, segmentation, and super resolution," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 479–490, Feb. 2007.

[27] A. Buades, B. Coll, and J. M. Morel, "Denoising image sequences does not require motion estimation," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, Sep. 2005, pp. 70–74.

[28] J. Boulanger, C. Kervrann, and P. Bouthemy, "Space-time adaptation for patch based image sequence restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 1096–1102, Jun. 2007.

[29] D. Rusanovskyy, K. Dabov, and K. Egiazarian, "Moving-window varying size 3D transform-based video denoising," presented at the Int. Workshop on Video Processing and Quality Metrics, Scottsdale, AZ, 2006.

[30] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 27–35, Jan. 2009.

[31] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms with a new one," *Multiscale Model. Simul.*, vol. 4, pp. 490–530, 2005.

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

[33] M. Elad, "On the bilateral filter and ways to improve it," *IEEE Trans. Image Process.*, vol. 11, no. 10, pp. 1141–1151, Oct. 2002.

[34] 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.

[35] G. Gilboa and S. Osher, "Nonlocal linear image regularization and supervised segmentation," *SIAM Multiscale Model. Simul.*, vol. 6, no. 2, pp. 595–630, 2007.

[36] D. Luenberger, *Optimization by Vector Space Methods*. New York: Wiley, 1969.

[37] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. New York: Athena Scientific, 1999.

[38] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, pp. 259–268, 1992.

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

[40] M. A. T. Figueiredo and R. D. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 906–916, Aug. 2003.

[41] R. C. Puetter, T. R. Gosnell, and A. Yahil, "Digital image reconstruction: Deblurring and denoising," *Annu. Rev. Astron. Astrophys.*, vol. 43, pp. 139–194, 2005.

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

[43] R. M. Pan and S. J. Reeves, "Efficient Huber-Markov edge-preserving image restoration," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3728–3735, Dec. 2006.

[44] J. M. Bioucas-Dias, "Bayesian wavelet-based image deconvolution: A GEM algorithm exploiting a class of heavy-tailed priors," *IEEE Trans. Image Process.*, vol. 15, no. 4, pp. 937–951, Apr. 2006.

[45] G. Wolberg, *Digital Image Warping*. Washington, DC: IEEE Computer Soc. Press, 1990.

[46] K. Turkowski, "Filters for common resampling tasks," in *Graphics Gems*. New York: Academic, 1990.

[47] M. Mahamoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *IEEE Signal Process. Lett.*, vol. 12, no. 12, pp. 839–842, Dec. 2005.

[48] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001, pp. 511–518.

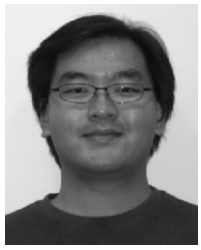
Matan Protter received the B.Sc. degree in mathematics, physics, and computer sciences from the Hebrew university of Jerusalem, Israel, in 2003. He is currently pursuing the PhD. degree in computer sciences from the Computer Sciences Department, The Technion—Israel Institute of Technology, Haifa.

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.



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 S.M., E.E., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1990, 1992, 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.

Prof. Milanfar won a National Science Foundation CAREER award in 2000. He is an Associate Editor of 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 and Multidimensional Signal Processing (IMDSP) Technical Committee.