Super Resolution With Probabilistic Motion Estimation

Matan Protter and Michael Elad

***Abstract*—Super-resolution reconstruction (SRR) has long been relying on very accurate motion estimation between the frames for a successful process. However, recent works propose SRR that bypasses the need for an explicit motion estimation [11], [15]. In this correspondence, we present a new framework that ultimately leads to the same algorithm as in our prior work [11]. The contribution of this paper is two-fold. First, the suggested approach is much simpler and more intuitive, relying on the classic SRR formulation, and using a probabilistic and crude motion estimation. Second, the new approach offers various extensions not covered in our previous work, such as more general re-sampling tasks (e.g., de-interlacing).**

***Index Terms*—Deinterlacing, probabilistic motion estimation, super resolution.**

I. Introduction

Super-resolution reconstruction (SRR) proposes a fusion of several low quality images $\{\mathbf{y}_t\}_{t=1}^T$ into one higher quality result $\mathbf{x}$ with better optical resolution. A wide variety of SRR algorithms have been developed in the past two decades—see [11] for a list of representatives of this vast literature. A popular model used for relating the measurements to the super-resolved image assumes that $\{\mathbf{y}_t\}_{t=1}^T$ are generated from $\mathbf{x}$ through a sequence of operations that includes (i) geometrical warps $\mathbf{F}_t$, (ii) a linear space-invariant blur $\mathbf{H}$, (iii) a decimation step represented by $\mathbf{D}$, and finally (iv) an additive zero-mean white and Gaussian noise $\mathbf{n}_t$ that represents both measurements noise and model mismatch[1] [6]. These are all linear operators, represented by a matrix multiplying the image they operate on. We assume hereafter that $\mathbf{H}$ and $\mathbf{D}$ are identical for all images in the sequence. This model leads to the following set of equations:

$$\mathbf{y}_t = \mathbf{DHF}_t\mathbf{x} + \mathbf{n}_t \text{ for } t = 1, 2, \ldots, T. \tag{1}$$

The recovery of $\mathbf{x}$ from $\{\mathbf{y}_t\}_{t=1}^T$ is, thus, an inverse problem, combining denoising, deblurring, scaling-up operation, and fusion of the different images, all merged to one. By setting $\mathbf{F}_1 = \mathbf{I}$, we refer to $\mathbf{y}_1$ as the reference image, and aim to construct $\mathbf{x}$ as its super-resolved version.

SRR relies on the assumption that $\mathbf{D}$, $\mathbf{H}$, and $\mathbf{F}_t$ are known, or can be reliably estimated from the given data. In particular, such reconstruction relies on the ability to estimate the motion in the scene with a sub-pixel accuracy, so as to enable the merger of the different image sampling grids properly. Many SRR algorithms start with such an estimating of the motion in the sequence (e.g., [1], [6], [7], [9], and [13]), or couple it with the recovery process, as a joint-estimation task [8], [14], [16].

Highly accurate general motion estimation, known as optical flow, is a severely under-determined problem. When inaccurately estimated motion is used within one of the existing SRR algorithms, it often leads to disturbing artifacts that cause the output to be inferior even when compared to the given measurements. For this reason, some simplifying assumptions as to the structure of the motion are made, such as global warps or rigid bodies. Only under these assumptions is the motion estimation in currently available SRR algorithms accurate enough to lead to a successful reconstruction of a super-resolved image. This had led to the commonly agreed and unavoidable conclusion that general content movies are not likely to be handled well by classical SRR techniques.

Recently, several papers have tried to circumvent this problem by avoiding explicit motion estimation altogether [11], [15]. The method in [15] relies on extending the steerable kernel method to multiframe super-resolution. The method in [11] generalizes the very successful nonlocal-means (NLM) [2] denoising method to performing super-resolution. The derivation of the SRR algorithm in [11] is done by defining an energy functional that explains the NLM, and then modifying it to serve the SRR task. Both methods do not explicitly estimate the motion, and both are shown to be able to handle general content video sequences quite successfully.

In this correspondence, we approach the explicit-motion-estimation-free SRR from a different perspective. Our starting point is the classic SRR, as in [6], and the bijective motion between pixels in each pair of images is replaced with a probabilistic motion field. This simple and alternative derivation is shown to lead to the same line of algorithms that are proposed in [11]. Furthermore, the framework proposed here allows different extensions, such as a treatment of spatio-temporal re-sampling problems. We show this adaptation in general, and demonstrate its applicability on the de-interlacing problem.

The structure of the paper is as follows. Section II describes a classic SRR formulation, as used in [1], [6], [7], [9], and [13], on which we build our eventual algorithm. Section III presents the use of probabilistic motion with the classic SRR, and develops the proposed algorithm. The adaptation to other re-sampling tasks is also described. Section IV provides results for SRR and de-interlacing, demonstrating the abilities of the proposed method. We conclude in Section V, outlining the key contributions of this work.

II. Classic Super-Resolution: Background

Using the model in (1), the maximum-likelihood (ML) estimate of $\mathbf{x}$ is obtained by minimizing the penalty function

$$\epsilon_{ML}^2(\mathbf{x}) = \frac{1}{2}\sum_{t=1}^{T}\|\mathbf{DHF}_t\mathbf{x} - \mathbf{y}_t\|_2^2 \tag{2}$$

with respect to $\mathbf{x}$. This expression suggests that we should seek an image $\mathbf{x}$ that explains best the set of measurements given. Minimization of (2) leads to

$$\frac{\partial\epsilon_{ML}^2(\mathbf{x})}{\partial\mathbf{x}} = \sum_{t=1}^{T}\mathbf{F}_t^T\mathbf{H}^T\mathbf{D}^T(\mathbf{DHF}_t\mathbf{x} - \mathbf{y}_t) = 0. \tag{3}$$

Denoting $\mathbf{A} = \sum_{t=1}^{T}\mathbf{F}_t^T\mathbf{H}^T\mathbf{D}^T\mathbf{DHF}_t$ and $\mathbf{b} = \sum_{t=1}^{T}\mathbf{F}_t^T\mathbf{H}^T\mathbf{D}^T\mathbf{y}_t$, we face a linear system of equations $\mathbf{A}\hat{\mathbf{x}}_{ML} = \mathbf{b}$.

In many cases, the measurements are not sufficient for recovering $\mathbf{x}$. This is manifested in a singular or possibly ill-conditioned matrix $\mathbf{A}$.

Manuscript received July 22, 2008; revised March 26, 2009. First published May 12, 2009; current version published July 10, 2009. This work was supported by The Israel Science Foundation under Grant 599/08. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Sabine Susstrunk.

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

Digital Object Identifier 10.1109/TIP.2009.2022440

[1]In [6], the model mismatches are modeled as Laplacian, with L_1 penalization as to obtain robustness to outliers. In our work, we choose a Gaussian model, which simplifies the algorithmic development. Nevertheless, a robustness to outliers is obtained by the probabilistic approach, as will be discussed later, in Section III.

In such cases, a regularization is required. The maximum *a posteriori* probability (MAP) estimation proposes a penalty of the form

$$\epsilon^2_{MAP}(\mathbf{x}) = \epsilon^2_{ML}(\mathbf{x}) + \lambda \cdot R(\mathbf{x}) \tag{4}$$

where the functional R is a regularization term that adds an algebraic stability to the inversion of $\mathbf{A}$. Beyond the gained stability, R also introduces the means to incorporate prior knowledge about the sought $\mathbf{x}$, such as spatial smoothness, sparsity of its wavelet representation, minimum entropy, etc. In this correspondence, we shall use the total variation choice $R(\mathbf{x}) = TV(\mathbf{x})$ that accumulates the gradients norms with ℓ^1, forcing (piece-wise) smoothness [12]. Thus, the MAP estimate in our case becomes the minimizer of

$$\epsilon^2_{MAP}(\mathbf{x}) = \frac{1}{2}\sum_{t=1}^{T} \|\mathbf{DHF}_t\mathbf{x} - \mathbf{y}_t\|_2^2 + \lambda \cdot TV(\mathbf{x}) \tag{5}$$

which is typically obtained by an iterative algorithm [1], [6]–[9], [13], [14], [16]. This is the core technique we build upon.

In all of the above discussion, we assume that the operators $\mathbf{D}$, $\mathbf{H}$, and $\mathbf{F}_t$ are known. The decimation $\mathbf{D}$ is dependent on the resolution scale-factor we aim to achieve, and as such, it is easily fixed. In this work, we shall assume that this resolution factor is an integer $s \geq 1$ on both axes. In most cases, the blur $\mathbf{H}$ refers to the camera PSF, and, therefore, it is also accessible. Even if this is not the case, the blur is typically dependent on few parameters, and those, in the worst case, can be manually set.

As opposed to these operators, the matrices $\mathbf{F}_t$ are harder to obtain. They depend on the scene and require highly accurate motion estimation for their construction. As such accuracy is hard to obtain in general, classical SRR algorithms often assume a simple motion pattern, such as pure translation or global affine warp. Attempts to embed the motion estimation within the SRR process have been made, with little success [8], [14], [16]. As already mentioned, inaccurately estimated motion within SRR often leads to disturbing artifacts that cause the output to be inferior even when compared to a simple interpolated version of $\mathbf{y}_1$. This fact motivated a quest for bypassing explicit motion estimation, as indeed practiced in [11] and [15].

III. Proposed Algorithm

A. New Formulation

We aim to introduce the notion of probabilistic motion estimation to the above classic SRR formulation. Note that the warp operator $\mathbf{F}_t$ considers a bijective (one-to-one) correspondence between pixels in the reference and the tth image, and as such, it introduces sensitivity to errors. We replace this motion field with a probabilistic one that assigns each pixel in the reference image with *many* possible correspondences in all the images in the sequence (including itself), each with an assigned probability of being correct.

How could this become useful for super-resolution for handling general motion patterns? Here we offer one possible way that illustrates that such ideas could fit in SRR. The operator F_t represents the motion field between the first image and image t, by indicating for each pixel in the first image its destination in image t. This is equivalent to independently listing a single 2-D translation vector for each pixel (where each pixel is assigned a translation, independent of other pixels). Therefore, the entire motion field is represented as a collection of various displacement vectors.

If the size of the maximal translation is, at most, D pixels, then a set of $M = (2D+1)^2$ displacements covers all the possible ones to be encountered. By defining $\{\mathbf{F}_m\}_{m=1}^{M}$ to be this set of global translations,[2] we can write the following equation:

$$\mathbf{F}_t\mathbf{x} = \sum_{m=1}^{M} \mathbf{Q}_{m,t}\mathbf{F}_m\mathbf{x} \tag{6}$$

which describes the action of warping the image $\mathbf{x}$ based on the operator $\mathbf{F}_t$. The matrices $\{\mathbf{Q}_{m,t}\}_1^M$ are diagonal weighting ones, containing 1-es along the main diagonal for pixels whose motion is the displacement $\mathbf{F}_m$, namely $[dx(m), dy(m)]$, and zeros otherwise. In such a way, it is possible to represent the most complex of motion fields by a linear combination of global translations.

In this formulation, we have replaced the single warping operator with a linear combination of global translation representing the same general motion field. Still, this notation implies a one-to-one relationship between pixels in both images. The next natural step for introducing a probabilistic motion field is to relax the definition of $\mathbf{Q}_{m,t}$, enabling continuous values to reflect varying confidences per pixel and per motion trajectory. This leads to a newly defined super-resolution penalty that replaces the use of $\mathbf{F}_t$ by their decompositions as in (6).

While this seems like a worthy path to consider, in this work we slightly divert from this approach, in a quest for a yet simpler algorithm. We modify the ML formulation posed in (2) by proposing the following probabilistic ML (PML) penalty:[3]

$$\epsilon^2_{PML}(\mathbf{x}) = \frac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T} \|\mathbf{DHF}_m\mathbf{x} - \mathbf{y}_t\|^2_{\mathbf{W}_{m,t}}. \tag{7}$$

We rely on the same intuition as described above, but in a slightly different way. Rather than accumulate the various global translations to form the effect of $\mathbf{F}_t$ as in (6), we accumulate the squared errors that result from such global displacements,[4] and assign a weight matrix $\mathbf{W}_{m,t}$ to each. Notice that the weights used in (7) are different from those introduced in (6). Whereas $\mathbf{Q}_{m,t}$ are defined for each pixel in the high resolution image, $\mathbf{W}_{m,t}$ are also diagonal matrices, but defined over the low-resolution grid. We shall proceed with the assumption that $\mathbf{W}_{m,t}$ are known, and revisit their computation in Section III-E.

It is important to note that although this formulation contains only global translations, it is still able to process any complex motion field, using the same rational that has led to (6). If the motion field is known, it can be re-created by properly assigning the values of $\mathbf{W}_{m,t}$ to be 1-es for those pixels whose motion is F_m and zeros for all others.

One could interpret the above expression as a marginalization of the squared error term with respect to the motion probability density function, in a way that resembles the concept proposed in [10]. The authors of [10] perform such a marginalization in order to avoid inaccuracies in the motion estimation, but their integration is performed over the parameters of a global motion model. In our case, very similar to the video denoising scenario, we handle local motion, and the probabilistic view-point contributes both to a better handling of the estimated motion inaccuracies and to the noise reduction.

As a final point in this section, we return to the matter of robustness. The usage of the above PML has another distinct advantage of robustifying the algorithm to outliers. One such example could be a scenario in which one of the images in the set is an outlier. In such a case, the weights assigned to the pixels in this image will be zeros, since the images do not match. Therefore, those pixels will effectively not be

[2]For simplicity, we shall use a set of integer displacements only.

[3]We use the notation $\|\mathbf{a}\|^2_{\mathbf{W}} = \mathbf{a}^T\mathbf{W}\mathbf{a}$.

[4]It is possible to use other sets of warps, such as ones that allow rotations, as well.

considered in the minimization, or in other words, treated as outliers, as required.

B. Separating the Blur Treatment

Our task is the minimization of a functional that has two terms in it: $\epsilon^2_{PML}(\mathbf{x})$ and a regularization (e.g., TV). Rather than handling this problem directly, we decompose it, following the methods developed in [5]–[7]. Since both $\mathbf{H}$ and $\mathbf{F}_m$ are space-invariant operators, they can be assumed to have a block-circulant structure (assuming a cyclic boundary treatment), and as such, they commute. Thus, defining $\mathbf{z} = \mathbf{Hx}$, we concentrate first on estimating the "blurry" high resolution image $\mathbf{z}$ by minimizing

$$\epsilon^2_{PML}(\mathbf{z}) = \frac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T}\|\mathbf{DF}_m\mathbf{z} - \mathbf{y}_t\|^2_{\mathbf{W}_{m,t}} \tag{8}$$

which will be the *fusion step*. Then we apply a conventional *deblurring step*, by minimizing

$$\epsilon^2_{DB}(\mathbf{x}) = \|\mathbf{Hx} - \mathbf{z}\|_2^2 + \lambda \cdot TV(\mathbf{x}). \tag{9}$$

This two-step process is sub-optimal to the joint treatment, but nevertheless leads to a simplified algorithm. As the second step is conventional and well-known, we focus hereafter on the fusion step. Note that the deblurring mechanism chosen here is relatively simple and could be replaced by more advanced techniques, thereby leading to better results.

C. Algorithm: A Matrix-Vector Version

We focus now on the minimization of (8). The derivative of this functional is given by

$$\frac{\partial \epsilon^2_{PML}(\mathbf{z})}{\partial \mathbf{z}} = \sum_{m=1}^{M}\sum_{t=1}^{T}\mathbf{F}_m^T\mathbf{D}^T\mathbf{W}_{m,t}(\mathbf{DF}_m\mathbf{z} - \mathbf{y}_t) \tag{10}$$

which leads to a linear system of equations. We introduce the following new notations, in order to simplify the obtained expressions

$$\widetilde{\mathbf{W}}_m = \sum_{t=1}^{T}\mathbf{W}_{m,t} \text{ and } \widetilde{\mathbf{y}}_m = \sum_{t=1}^{T}\mathbf{W}_{m,t}\mathbf{y}_t. \tag{11}$$

The matrix $\widetilde{\mathbf{W}}_m$ is a diagonal matrix, as it is the sum of diagonal matrices. We obtain

$$\left[\sum_{m=1}^{M}\mathbf{F}_m^T\mathbf{D}^T\widetilde{\mathbf{W}}_m\mathbf{DF}_m\right]\mathbf{z} = \sum_{m=1}^{M}\mathbf{F}_m^T\mathbf{D}^T\widetilde{\mathbf{y}}_m. \tag{12}$$

This linear system of equations seems complicated. As we show next, it can be rewritten for each pixel in $\mathbf{z}$ in a closed form, revealing a simple structure that leads to a stable solution.

D. Algorithm: A Pixel-Wise Version

The right-hand-side (RHS) in (12) is an image of the same size as $\mathbf{z}$. Furthermore, as we are about to show, the matrix multiplying $\mathbf{z}$ on the left-hand-side (LHS) is a diagonal positive definite matrix. Thus, we can turn the above vector-matrix formulation into a pixel-wise one. Let us consider a specific pixel at location $[i,j]$ in $\mathbf{z}$, and see its construction. As this pixel is dependent only on the $[i,j]$th pixel in the RHS image (up to a scalar being the diagonal element in the matrix on the LHS), we start by constructing this element.

For a specific $\mathbf{F}_m$ that shifts by $[dx(m), dy(m)]$, the term $\mathbf{F}_m^T\mathbf{v}$ positions the $[i+dx(m), j+dy(m)]$th element from the image $\mathbf{v}$ in the destination $[i,j]$ (the transpose has the effect of an inverse displacement). The image $\mathbf{u} = \mathbf{D}^T\widetilde{\mathbf{y}}_m$ is a scale-up version of the low-resolution image $\widetilde{\mathbf{y}}_m$ by zero-filling. This implies that if the location $[i+dx(m), j+dy(m)]$ is not an integer multiple of s (the resolution ratio), this location has a zero entry. Otherwise, the entry is simply $\widetilde{\mathbf{y}}_m[k,l]$, where $[k,l] = [i+dx(m), j+dy(m)]/s$. Thus, at location $[i,j]$, we get

$$\mathrm{RHS}[i,j] = \sum_{[k,l]\in N(i,j)}\widetilde{\mathbf{y}}_m[k,l] \tag{13}$$

where we have defined the neighborhood set

$$N(i,j) = \{[k,l] |\ \forall\, m \in [1,M], \quad s\cdot k = i + dx(m),\ s\cdot l = j + dy(m)\}. \tag{14}$$

Plugging the definition of $\widetilde{\mathbf{y}}_m$ from (11) yields

$$\mathrm{RHS}[i,j] = \sum_{[k,l]\in N(i,j)}\sum_{t=1}^{T}\mathbf{W}_{m,t}[k,l]\mathbf{y}_t[k,l]. \tag{15}$$

In this expression, $\mathbf{W}_{m,t}[k,l]$ refers to the entry on the main diagonal in $\mathbf{W}_{m,t}$ that multiplies the $[k,l]$ entry in $\mathbf{y}_t$.

We now discuss the left-hand-side (LHS) in (12). The operator $\mathbf{D}^T\widetilde{\mathbf{W}}_m\mathbf{D}$ within this expression is a diagonal matrix that decimates an image by a factor s in each axis, weights each pixel by the diagonal weight matrix $\widetilde{\mathbf{W}}_m$, and then up-scales back the image using the same factor by zero-filling. This means that when operating on an image $\mathbf{v}$, a pixel in location $[i,j]$ is nulled if $[i,j]/s$ is a noninteger, and otherwise it is simply weighted, i.e., it becomes $\widetilde{\mathbf{W}}_m[i,j]\cdot\mathbf{v}[i,j]$.

When the operator $\mathbf{F}_m^T\mathbf{D}^T\widetilde{\mathbf{W}}_m\mathbf{DF}_m$ is applied to the $[i,j]$th pixel in $\mathbf{z}$, it shifts it to the $[i+dx(m), j+dy(m)]$th location, nulls it or weights it, based on whether $[i+dx(m), j+dy(m)]/s$ is an integer, and finally shifts the outcome back by $[-dx(m), -dy(m)]$ to its original place, $[i,j]$. The fact that the operator $\mathbf{F}_m^T\mathbf{D}^T\widetilde{\mathbf{W}}_m\mathbf{DF}_m$ returns every pixel to its original location means that this matrix is diagonal, as every output pixel depends only on the value of the input pixel in the same location. Thus, the scalar that multiplies the $[i,j]$th pixel in $\mathbf{z}$ is

$$\begin{aligned}\mathrm{LHS}[i,j] &= \sum_{[k,l]\in N(i,j)}\widetilde{\mathbf{W}}_m[k,l]\mathbf{z}[i,j] \\ &= \sum_{[k,l]\in N(i,j)}\sum_{t=1}^{T}\mathbf{W}_{m,t}[k,l]\mathbf{z}[i,j]\end{aligned} \tag{16}$$

where we have made use of the definition of $\widetilde{\mathbf{W}}_m$ in (11). This expression sums all the weights in (15), serving as a normalization term. Assuming that this sum is positive (i.e., at least one weight is nonzero), combining (15) and (16) leads to a closed form expression for the $[i,j]$th pixel in the estimated $\mathbf{z}$

$$\hat{\mathbf{z}}[i,j] = \frac{\sum_{[k,l]\in N(i,j)}\sum_{t=1}^{T}\mathbf{W}_{m,t}[k,l]\mathbf{y}_t[k,l]}{\sum_{[k,l]\in N(i,j)}\sum_{t=1}^{T}\mathbf{W}_{m,t}[k,l]} \tag{17}$$

and the resemblance to the fusion algorithm in our prior work is evident (see [11, Equation (30)]). Just as explained there, the similarity of the final algorithm to the NLM stands out, but there is a subtle difference between the two, related to the domain of the averaging. The proposed algorithm differs considerably from an interpolation followed by application of NLM—we show a visual comparison between the two in Section III-E.

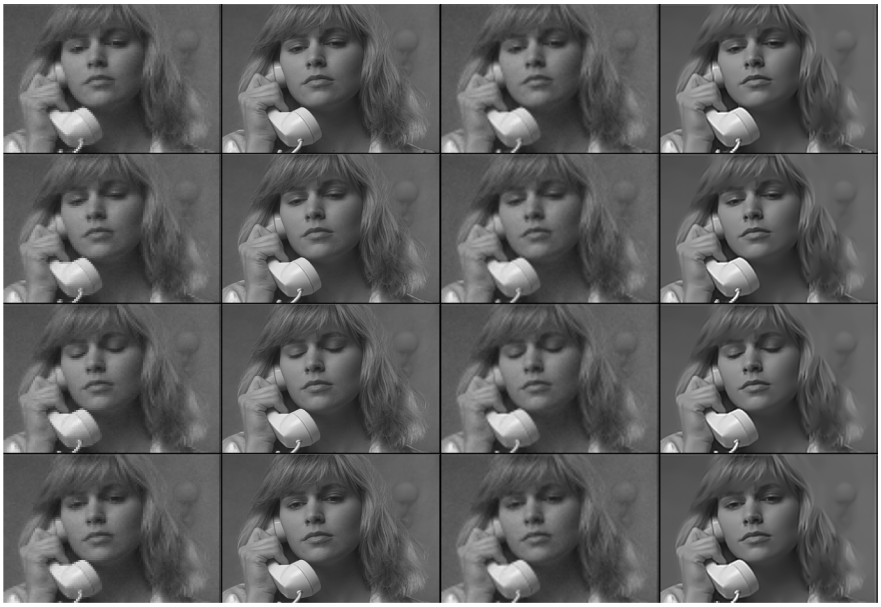

Fig. 1. Results for the 8th, 13th, 18th, and the 23rd frames from the "Suzie" sequence. From left to right: pixel-replicated low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.

E. Computing the Weights

In order to complete the description of the algorithm, we must explain how $\mathbf{W}_{m,t}[i,j]$ are computed. Based on (8), these weights are supposed to encompass the fit, per pixel, of the desired high resolution image $\mathbf{z}$ after being transformed by $\mathbf{F}_m$ and decimated by $\mathbf{D}$, with the input image $\mathbf{y}_t$. Thus, the weights could be related to the error $\mathbf{DF}_m\mathbf{z}-\mathbf{y}_t$. In order to better estimate the fit, we propose to use some spatial support for each pixel instead of computing the plain difference. Defining $\mathbf{R}_{i,j}$ as an operator that extracts a patch of a fixed and predetermined size (say $q \times q$ pixels) from an image, the weights are computed by

$$\mathbf{W}_{m,t}[i,j] = \exp\left\{-\frac{\|\mathbf{R}_{i,j}\left(\mathbf{DF}_m\mathbf{z}-\mathbf{y}_t\right)\|_2^2}{2\sigma^2}\right\} \cdot f\left(\sqrt{\left(dx(m)\right)^2+\left(dy(m)\right)^2+(t-1)^2}\right). \quad (18)$$

The first part in the above formula gives a value that is inversely proportional to the Euclidean distance between the transformed image $\mathbf{DF}_m\mathbf{z}$ and the input image $\mathbf{y}_t$, computed over some support around each pixel. The second part adds a decaying weight as a function of the displacement and time shift magnitudes versus the reference frame. The function f can be chosen as any monotonically non increasing function (e.g., box function or Gaussian bell).

The computation of the weights requires the use of the unknown $\mathbf{z}$. Instead, the weights are computed at the beginning by using an estimated version of $\mathbf{z}$, being a scaled-up version of the reference frame $\mathbf{y}_1$. This scale-up is done using a conventional image interpolation algorithm such as bilinear, bicubic, or the Lanczos method. As this is a crude version of the desired outcome, the process can be iterated, using the newly estimated image $\hat{\mathbf{z}}$. In our tests, we employ two such iterations only.

The method in which the weights are computed is reminiscent of classic block-matching based SR algorithms (e.g., [3]). However, there is a key difference between these algorithms and the one proposed here. In the classic block-matching based SR, block-matching is used to determine a single trajectory for each pixel in the current image (that is being processed) to the others, and as such, estimate the motion. In contrast, in the proposed algorithm a method similar to block-matching is used to estimate the probability of each trajectory. Once computed, all these trajectories are considered together, according to their probabilities, as opposed to selecting only the single most likely one. This difference is what enables the proposed algorithm to handle complex scenarios where highly accurate motion estimation is not currently possible.

F. Other Resampling Tasks

We wish to adapt the proposed framework to other re-sampling tasks, such as de-interlacing, inpainting and more. We start by explaining this extension intuitively. The re-sampling task can be considered as computing pixel values for only some of the pixels in each image ("missing pixels"). For example, the de-interlacing task may be viewed as providing pixel values only for the even rows for the odd numbered fields, as well as for the odd rows in the even numbered fields. Formulating this idea, given each input image (or field) $\mathbf{y}_t$, it can be linked to the original (unknown) image $\mathbf{Y}_t$ using a masking operator $\mathbf{M}_t : \mathbf{y}_t = \mathbf{M}_t\mathbf{Y}_t$. Simply put, $\mathbf{M}_t$ discards all un-sampled pixels. It is a binary matrix, with as many rows as the number of pixels in $\mathbf{y}_t$ and as many columns as pixels in $\mathbf{Y}_t$, with entries of ones indicating which pixels are to be kept. Note that $\mathbf{y}_t$ contains only sampled pixels. In the in-painting case, it contains only the un-masked pixels.

In line with the idea of the probabilistic motion estimation, $\mathbf{Y}_t$ can be constructed as a (pixel-wise) weighted average of different transformations of the target image $\mathbf{x}$. The image $\mathbf{x}$ that we seek should be as similar as possible to each $\mathbf{y}_t$, after undergoing each of the transformations and the relevant masking. This required similarity is weighted on a pixel-wise basis, according to the (local) probability of the specific transformation having taken place. Put into the maximum likelihood formulation, a penalty function very similar to (7) arises, where the decimation operator is replaced by $\mathbf{M}_t$

$$\epsilon_{PML}^2(\mathbf{x}) = \frac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T}\|\mathbf{M}_t\mathbf{HF}_m\mathbf{x}-\mathbf{y}_t\|_{\mathbf{W}_{m,t}}^2. \quad (19)$$

Minimizing this functional proceeds very similarly to the steps described before. The treatment of the blur is separated, and a pixel-wise formula for the values of $\mathbf{z}$ is given by (17). The difference is in the

Fig. 2. Original "Trevor" sequence. Left: Interpolated image. Middle: Interpolation, followed by NLM processing and deblurring. Right: Proposed algorithm. Bottom row offers a close-up of a portion of the images.

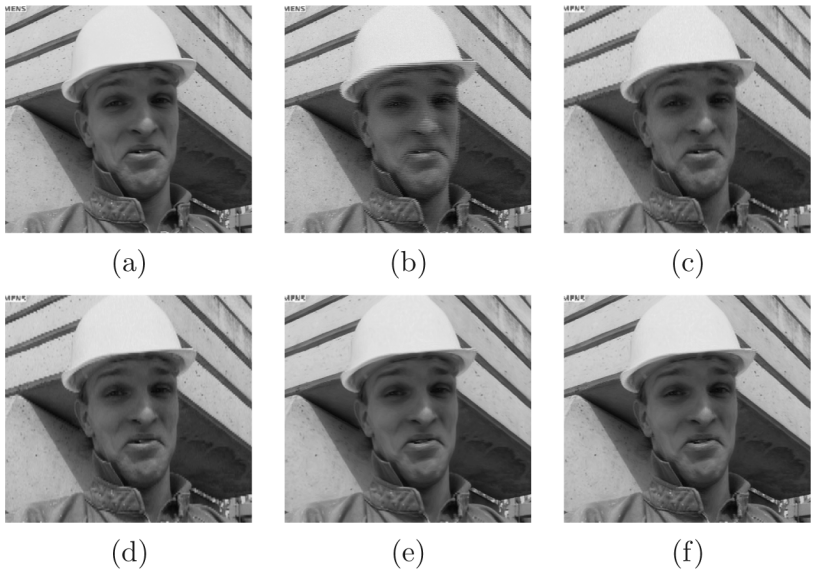

Fig. 3. De-Interlacing Results. (a) Original (ground-truth) image. (b) Interlaced image. (c) Row averaging, 29.87 dB. (d) Row averaging followed by NLM processing, 29.93 dB. (e) Proposed algorithm—first iteration, 30.69 dB. (f) Proposed algorithm—second iteration, 30.71 dB.

order of summation, as the neighborhood $N(i,j)$ of a pixel is now time (and spatial) dependent. This is because the masking may be different for every image in the sequence.

The weights for this formula are computed very similarly to the SRR case, described in (18). However, these tasks can benefit from computing the weights in high resolution scale. Thus, if we consider that $\mathbf{W}_{m,t}$ is for the coarse scale, we denote $\mathbf{W}_{m,t} = \mathbf{M}_t\tilde{\mathbf{W}}_{m,t}$, with $\tilde{\mathbf{W}}_{m,t}$ being the same size as $\mathbf{Y}_t$. The formula for each entry of $\tilde{\mathbf{W}}_{m,t}$ (when arranged as an image) is, therefore, the same as in (18), but with $\mathbf{F}_m\mathbf{z} - \mathbf{Y}_t$ replacing $\mathbf{DF}_m\mathbf{z} - \mathbf{y}_t$. In these weights, $\mathbf{Y}_t$ is an interpolated version of $\mathbf{y}_t$ (with the interpolation method depending on the specific task). Of course, these weights should be computed only for pixels that are kept after the masking $\mathbf{W}_{m,t} = \mathbf{M}_t\tilde{\mathbf{W}}_{m,t}$.

IV. Experimental Results

We now turn to demonstrate the potential of the proposed SRR algorithm by presenting the results for image sequences with a general motion pattern. Since the algorithm tested here is the very same one as in [11], we concentrate on one such example. The original sequence "Suzie" has been blurred using a 3×3 uniform mask, decimated by a factor of 1:3 (in each axis), and then contaminated by additive white zero-mean Gaussian noise with $STD = 2$. The degraded sequence

was the input to the proposed SRR algorithm, and Fig. 1 presents the obtained results for the 8th, 13th, 18th, and the 23rd frames.[5]

We also compare the results using the average PSNR, an objective quality measure ($\mathrm{PSNR} = 10\log_{10}\left(255^2 \cdot p/\|\hat{\mathbf{X}} - \mathbf{X}\|_2^2\right)$ [dB], where $\hat{\mathbf{X}}$ and $\mathbf{X}$ are the original and reconstructed images, respectively, and p the number of pixels in the image). For the above sequence, the PSNR for the pixel replicated low-quality sequence, the Lanczos results and the proposed algorithm are 30, 31.4, and 33.74 dB, respectively.

The above presented sequence, and the others in [11], are all synthetic, in the sense that the blur kernel and decimation are known. Note, however, that the motion in the sequences is real, and not synthetically generated. In order to demonstrate the proposed algorithm on a directly captured sequence, we provide a second experiment "Trevor," whose results are displayed in Fig. 2. In this case, there is no ground-truth image available to compare to. Therefore, to demonstrate that a super-resolution effect is achieved, a comparison is made to an interpolated sequence. This interpolation is obtained by a Lanczos interpolation, followed by NLM filtering for denoising, and then deblurring. This comparison serves two goals: (1) It indeed verifies that the proposed algorithm obtains SR effect; and (2) it demonstrates the difference between simply running NLM and deblurring after up-scaling, compared to running the proposed algorithm. This comparison is important, as the two schemes are confusingly similar. Clearly, a far better image is obtained with the proposed algorithm.

We have also tested the proposed generalized algorithm on an interlaced sequence. We used the Foreman sequence and composed each interlaced frame by taking the odd numbered rows from one frame, and the even numbered rows from the next, resulting in a sequence with half as many frames. This sequence was also contaminated by additive white zero-mean Gaussian noise with $STD = 2$. This generated sequence can be considered a true interlaced sequence, as no manipulation (e.g., simulated blurring) of the pixels has been made other than half the pixels being discarded.

This sequence has been processed by the framework suggested in Section III, with the result appearing in Fig. 3. The initial interlaced sequence was split into fields, and each field was expanded by a factor of two in the vertical axis only. The missing rows were interpolated by averaging the rows immediately above and below each missing row. The masks $\mathbf{M}_t$ were designed to discard the even rows in the odd numbered images, and the odd rows in the even numbered images. Five interlaced frames (ten fields) were used for processing, and the search area consisted of ten pixels in every direction. We display the results for two iterations (where the first is used for computing the weights for the second), although the differences are much less dramatic than in the SRR case. As done above, we also show the results of directly filtering the re-scaled sequence with the NLM filter, to highlight the difference of the proposed approach. Note how the staircase effect (on the wall) is much decayed by the proposed algorithm. It should be noted that the purpose of this test is only to demonstrate the applicability of the proposed framework to other re-sampling tasks, without claiming that it out-performs other de-interlacing methods. Further work is required to compare the proposed technique to existing de-interlacing algorithms.

Before concluding the results section, we address the computational complexity of the algorithm presented here. As already explained in [11], the overall algorithm is very heavy—the weights' computation stage is the most demanding. For a nominal case, in which the search area is 31×31 pixels in the low-resolution, 15 images in the sequence, and a patch size of 13×13 pixels for computing the weights, there are about 2,400,000 operations per pixel. While this value may seem prohibitive, there are various methods in which this computational burden can be substantially reduced. We refer the reader to [11, Section 4.3] for an elaborate discussion.

[5]The sequences appearing in this section (input and output) and others from [11], along with the various parameters used to generate them, can be found at http://www.cs.technion.ac.il/~matanpr/NLM-SR.

V. Summary

In our earlier work, we developed an explicit-motion-estimation-free SRR algorithm by extending the NLM [11]. In this paper we approach the same task from a different perspective, basing it on a probabilistic and crude motion estimation. Interestingly, this approach (under some assumptions) leads to the same algorithm described in [11]. However, the formulation described here is more intuitive, as it relies on the classic super-resolution framework and on the imaging model. Furthermore, this formulation allows for different extensions than those proposed in [11]. We also show how the framework can in fact be adapted to any re-sampling task. An example of de-interlacing is given, to show the validity of this adaptation. This example shows than even sequences with large, highly nonrigid motion patterns can be successfully de-interlaced by the proposed framework.

References

[1] 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.

[2] 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 September (AVSS)*, 2005, pp. 70–74.

[3] G. M. Callicó, S. López, O. Sosa, J. F. Lopez, and R. Sarmiento, "Analysis of fast block matching motion estimation algorithms for video super-resolution systems," *IEEE Trans. Consum. Electron.*, vol. 54, no. 3, Aug. 2008.

[4] J. Chen and C. K. Tang, "Spatio-temporal markov random field for video denoising," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, Minneapolis, MN, Jun. 2007, pp. 1–8.

[5] 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. 1187–93, Aug. 2001.

[6] 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.

[7] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Advances and challenges in superresolution," *Int. J. Imag. Syst. Technol.*, vol. 14, no. 8, pp. 47–57, Aug. 2004.

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

[10] L. C. Pickup, D. P. Capel, S. J. Roberts, and A. Zisserman, "Overcoming registration uncertainty in image super-resolution: Maximize or marginalize?," *EURASIP J. Adv. Signal Process.*, no. 23565, 2007.

[11] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the non-local-means to super-resolution reconstruction," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 36–51, Jan. 2009.

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

[13] 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.

[14] 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.

[15] H. Takeda, P. Milanfar, M. Protter, and M. Elad, "Super-resolution without explicit subpixel motion estimation," *IEEE Trans. Image Process*, to be published.

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