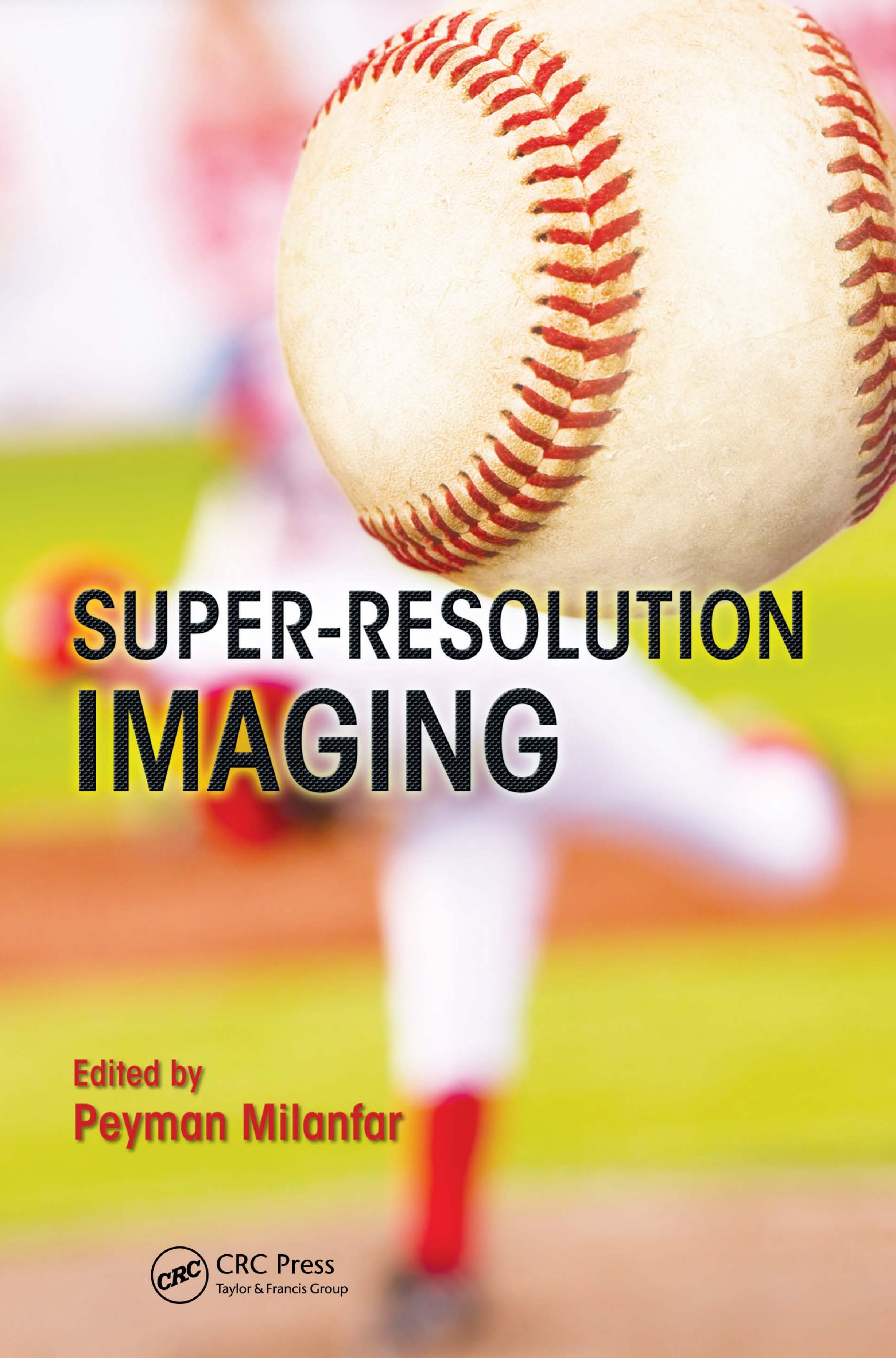


SUPER-RESOLUTION IMAGING

Edited by
Peyman Milanfar

CRC Press
Taylor & Francis Group

SUPER-RESOLUTION IMAGING

Edited by
Peyman Milanfar

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

4

Super-Resolution with Probabilistic Motion Estimation

Matan Protter

The Technion – Israel Institute of Technology

Michael Elad

The Technion – Israel Institute of Technology

CONTENTS

Classic super-resolution has long relied on very exact motion estimation for the recovery of sub-pixel details. As a highly accurate motion field is hard to obtain for general scenes, classic super-resolution has been known to be limited to specific cases, where the motion is of a global nature. In this chapter, we present a recently developed family of algorithms that shatters this barrier. These novel algorithms relax the requirement of a one-to-one motion field, and replace it with a simple, probabilistic motion estimation. The probabilistic motion field is integrated into the classic (and heavily investigated) SR framework, and ultimately results in a very simple family of algorithms. The obtained paradigm gets an algorithmic structure that resembles that of the nonlocal means, and as such, leads to a localized and easily parallelizable procedure. Despite their simplicity, the obtained algorithms are nevertheless very powerful in handling the most general scenes, with the probabilistic motion estimation enabling the handling of challenging motion patterns. The resulting image sequences are of high quality, and contain few artifacts. These novel algorithms open the door to a new era in super-resolution that bypasses the limiting traditional reliance on explicit motion estimation for super-resolution.

4.1 Introduction

Super-Resolution Reconstruction (SRR) proposes a fusion of several low quality images $\{y_t\}_{t=1}^T$ into one higher quality result x, which has better optical resolution than the input images. A wide variety of SRR algorithms have been developed in the past two decades – see [13] for a list of representatives of this vast literature. A popular model used for relating the measurements to the super-resolved image, assumes that $\{y_t\}_{t=1}^T$ are generated from x through a sequence of operations that includes (i) geometrical warps F_t, (ii) a linear space-invariant blur H, (iii) a decimation step represented by D, and finally (iv) an additive zero-mean white and Gaussian noise n_t that represents both measurements noise and model mismatch[1] [7]. All of these operators are linear, each represented by a matrix multiplying the image they operate on. We assume hereafter that H and D are identical for all images in the sequence. Mathematically, the relationship between the high-quality image x and the measurements $\{y_t\}_{t=1}^T$ is given by

$$y_t = DHF_t x + n_t \quad \text{for} \quad t = 1, 2, \ldots, T. \tag{4.1}$$

The recovery of x from $\{y_t\}_{t=1}^T$ is thus an inverse problem, combining denoising, deblurring, scaling-up operation, and fusion of the different images, all merged to one. We treat y_1 as our reference image, and aim to reconstruct x as its super-resolved version (this implies that $F_1 = I$).

SRR relies on the assumption that D, H, and 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 subpixel 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., [9, 15, 1, 7, 6]), or couple it with the recovery process, as a joint-estimation task [8, 19, 16].

Highly accurate general motion estimation, known as optical flow, is a severely under-determined problem. Various artifacts, and an output image that is even inferior to the given measurements, are often the result of using

[1]In [7], the model mismatches are represented as an iid Laplacian distribution, 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 3.

an inaccurately estimated motion within one of the existing SRR algorithms. In order to estimate the motion with enough accuracy to lead to a successful reconstruction of a super-resolved image, some simplifying assumptions as to the structure of the motion field must be made, such as global warps or rigid bodies. 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 [13, 17]. The method in [17] relies on extending the steerable kernel method to multiframe super-resolution. The method in [13] generalizes the very successful nonlocal means (NLM) [2] denoising method to performing super-resolution. The derivation of the SRR algorithm in [13], termed NLM-SR, 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 chapter we approach the explicit-motion-estimation-free SRR from a different perspective. Our starting point is the classic SRR, as in [7]. We then replace the bijective motion between pixels in each pair of images with a probabilistic motion field. This simple and alternative derivation is shown to lead to the same line of algorithms that are proposed in [13]. 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 chapter is as follows. Section 4.2 describes a classic SRR formulation, as used in [9, 15, 1, 7, 6], on which we build our eventual algorithm. Section 4.3 presents the use of probabilistic motion within the framework of classic SRR, and develops the proposed algorithm. The adaptation to other re-sampling tasks is also described in this section. Section 4.4 provides results for SRR and de-interlacing, demonstrating the abilities of the proposed method. The key contributions of this work are outlined in Section 4.5, with several directions of possible future work also suggested. We note that a preliminary version of this chapter has appeared in [12].

4.2 Classic Super-Resolution: Background

Using the model in Equation (14.6), one can seek the most likely high resolution image, given the existing low-resolution images (and the known decimation, blur, and transformations). This image is called the Maximum-Likelihood

(ML) estimate of x, and is obtained by minimizing the penalty function

$$\epsilon^2_{ML}(x) = \frac{1}{2}\sum_{t=1}^{T} \|DHF_t x - y_t\|_2^2 \tag{4.2}$$

with respect to x. Minimization of (4.2) leads to

$$\frac{\partial \epsilon^2_{ML}(x)}{\partial x} = \sum_{t=1}^{T} F_t^T H^T D^T (DHF_t x - y_t) = 0. \tag{4.3}$$

Denoting $A = \sum_{t=1}^{T} F_t^T H^T D^T DHF_t$ and $b = \sum_{t=1}^{T} F_t^T H^T D^T y_t$, we face a linear system of equations $A\hat{x}_{ML} = b$.

In many cases the measurements are not sufficient for recovering x. In such cases, the constraints matrix A is singular or possibly ill-conditioned, and regularization is required. The Maximum A-posteriori Probability (MAP) estimation proposes a penalty of the form

$$\epsilon^2_{MAP}(x) = \epsilon^2_{ML}(x) + \lambda \cdot R(x), \tag{4.4}$$

where the functional R is a regularization term that adds an algebraic stability to the inversion of A. Beyond the gained stability, R is also a way of incorporating prior knowledge about the sought x, such as spatial smoothness, sparsity of its wavelet representation, minimum entropy, etc.. In this work we force spatial smoothness, by choosing the Total Variation (TV) prior, that accumulates the gradients norms with ℓ^1 [14]. Thus, the MAP estimate in our case becomes the minimizer of

$$\epsilon^2_{MAP}(x) = \frac{1}{2}\sum_{t=1}^{T} \|DHF_t x - y_t\|_2^2 + \lambda \cdot TV(x), \tag{4.5}$$

which is typically obtained by an iterative algorithm [9, 15, 8, 1, 7, 6, 19, 16]. This is the core technique we build upon.

The operators D, H, and F_t are assumed to be known in all of the above discussions. The decimation 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 $s \geq 1$ in both axes. The blur H refers to the camera PSF in most cases, and therefore it is also accessible. Even if it is not, the blur is typically dependent on a small number of parameters, and those, in the worst case, can be manually set.

While D and H are relatively easy to obtain, this is not the case of F_t. The warp operators depend on the scene and require highly accurate motion estimation for their construction. Since 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. Such constraints stabilize the motion estimation, as they substantially reduce the number of parameters to be estimated, allowing greater accuracy in the estimation (if indeed the motion field obeys these assumptions). Attempts to embed the motion estimation (without assuming a specific structure) within the SRR process have been made, with little success [8, 19, 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 y_1. This fact motivated a quest for bypassing explicit motion estimation, as indeed practiced in [13, 17].

4.3 The Proposed Algorithm

4.3.1 The New Formulation

We now aim to integrate the notion of probabilistic motion estimation into the classic SRR formulation introduced in the previous section. Before we dive into the formulation, we note that when the motion is of a global nature, and therefore lends itself to an accurate estimation, motion-estimation-based techniques are likely to obtain better results than the proposed algorithm in many cases. In other cases, the usage of the proposed algorithm makes of intra-image redundancy may bring better results even compared to the motion-compensated algorithms. As such sequences comprise only a small subset of the sequences to be super-resolved, we don't continue this discussion further, and rather tackle general motion sequences by using the probabilistic motion estimation technique, which we now describe.

The starting point is the observation that the warp operator F_t considers a bijective (one-to-one) correspondence between pixels in the reference and the t-th 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.

Can this become useful for super-resolution for handling general motion patterns? We now offer one possible way that illustrates that it can. We start by analyzing the operator F_t, which 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. Equivalently, the motion field can be described by listing a single 2D translation vector for each pixel, independently of other pixels. Therefore, the entire motion field is represented as a collection of various displacement vectors, one for each pixel.

If the size of the maximal translation is at most D pixels, then the set of all the possible displacements are covered by a set of $M = (2D + 1)^2$ displacements. By defining $\{F_m\}_{m=1}^{M}$ to be this set of global translations,[2] we

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

can write the following equation

$$F_t x = \sum_{m=1}^{M} Q_{m,t} F_m x, \tag{4.6}$$

which describes the action of warping the image x based on the operator F_t. The matrices $\{Q_{m,t}\}_1^M$ are diagonal weighting ones, containing ones along the main diagonal for pixels whose motion is the displacement F_m, namely $[dx(m), dy(m)]$, and zeros for the rest of the pixels. Using such a decomposition, even the most complicated of motion fields can be represented by a linear combination of global translations.

While we have replaced the single warping operator with a linear combination of global translation (representing the same general motion field), a one-to-one relationship between pixels in both images is still implied by this notation. The next natural step for introducing a probabilistic motion field is to relax the definition of $Q_{m,t}$, where varying confidences per pixel and per motion trajectory are reflected by continuous values. This leads to a newly defined super-resolution penalty that replaces the use of F_t by their decompositions as in (4.6).

While this seems like a worthy path to consider, we slightly divert from this approach, seeking yet a simpler algorithm. We modify the ML formulation posed in Equation (4.2) by proposing the following probabilistic ML (PML) penalty[3]

$$\epsilon_{PML}^2(x) = \frac{1}{2} \sum_{m=1}^{M} \sum_{t=1}^{T} \|DHF_m x - y_t\|_{W_{m,t}}^2 . \tag{4.7}$$

The same intuition, although applied differently, is used in proposing this penalty. Rather than accumulate the various global translations to form the effect of F_t as in Equation (4.6), we accumulate the least-squares errors that result from such global displacements,[4] and assign a weight matrix $W_{m,t}$ to each. Notice that the weights used in Equation (4.7) are different from those introduced in (4.6). Whereas $Q_{m,t}$ are defined for each pixel in the high resolution image, $W_{m,t}$ are also diagonal matrices, but defined over the low-resolution grid. We shall proceed with the assumption that $W_{m,t}$ are known, and revisit their computation in Section 4.3.5.

Even though this formulation contains only global translations, it should be noted that using the same rational that has led to Equation (4.6), it can represent any complex motion field. A known motion field can be re-created by properly assigning the values of $W_{m,t}$ to be 1s for those pixels whose motion is F_m and zeros for all others.

One particular interpretation of the above expression is a marginalization of the least-squared error term with respect to the motion probability density

[3]We use the notation $\|a\|_W^2 = a^T W a$.

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

function, in a way that resembles the concept proposed in [11]. However, the authors of [11] perform such a marginalization in order to avoid inaccuracies in the motion estimation, and their integration is only 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 viewpoint contributes both to a better handling of the estimated motion inaccuracies and also 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. Suppose one of the images in the low-resolution set is in fact an outlier, and does not belong in the sequence. Since this outlier image does not match the rest of the images, the pixels in it will be assigned zero weights. This can be understood qualitively from the weights reflecting the matching of the patch. In Section 4.3.5 the computation of the weights is discussed, demonstrating how outliers are indeed assigned zero (or negligible) weights. Effectively, since all pixels in an outlier image are ignored, and are not considered in the minimization – they are indeed treated as outliers. The same logic can be applied to local outliers, such as transmission errors, graphics, boundaries, and more.

4.3.2 Separating the Blur Treatment

Our task is the minimization of a functional that has two terms: $\epsilon^2_{PML}(x)$ and a regularization (e.g., TV). Rather than handling this problem directly, we decompose it, following the methods developed in [4, 7, 6]. Since both H and 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 $z = Hx$, we separate the estimation into two stages, first concentrating on estimating the "blurry" high resolution image z by minimizing

$$\epsilon^2_{PML}(z) = \frac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T}\|DF_m z - y_t\|^2_{W_{m,t}}, \tag{4.8}$$

which is the *fusion step*. The second step is applying a conventional *deblurring step*, that minimizes

$$\epsilon^2_{DB}(x) = \|Hx - z\|_2^2 + \lambda \cdot TV(x). \tag{4.9}$$

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.

4.3.3 The Algorithm: A Matrix-Vector Version

We now focus on the fusion step; the minimization of Equation (4.8). The derivative of this functional is given by

$$\frac{\partial \epsilon^2_{PML}(z)}{\partial z} = \sum_{m=1}^{M} \sum_{t=1}^{T} F_m^T D^T W_{m,t}(DF_m z - y_t), \tag{4.10}$$

which leads to a linear system of equations. In order to simplify the obtained expressions, we introduce the following new notations:

$$\widetilde{W}_m = \sum_{t=1}^{T} W_{m,t} \text{ and } \widetilde{y}_m = \sum_{t=1}^{T} W_{m,t} y_t. \tag{4.11}$$

The matrix $\widetilde{W}_m$ is s sum of diagonal matrices, and therefore diagonal in itself. By rearranging and substituting $\widetilde{W}_m$ and $\widetilde{y}_m$, we obtain

$$\left[\sum_{m=1}^{M} F_m^T D^T \widetilde{W}_m D F_m\right] z = \sum_{m=1}^{M} F_m^T D^T \widetilde{y}_m. \tag{4.12}$$

While this linear system of equations seems complicated, we show next that it can be rewritten for each pixel in z in a closed form, revealing a simple structure that leads to a stable solution.

4.3.4 The Algorithm: A Pixel-Wise Version

The Right-Hand-Side (RHS) in Equation (4.12) is an image of the same size as z. Furthermore, as we are about to show, the matrix multiplying 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.

Since the RHS is an image of the same size as z, we start by looking at how a specific pixel at location $[i, j]$ in the RHS is constructed. A specific F_m shifts by $[dx(m), dy(m)]$. Therefore, the term $F_m^T v$ positions the $[i + dx(m), j + dy(m)]$-th element from the image v in the destination $[i, j]$ (since the transpose has the effect of an inverse displacement). The image $u = D^T \widetilde{y}_m$ is a scale-up version of the low-resolution image $\widetilde{y}_m$ by zero-filling. Combining the two 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 $\widetilde{y}_m[k, l]$, where $[k, l] = [i + dx(m), j + dy(m)]/s$. Accounting for all the displacements in the set and for all input images, we get that at location $[i, j]$

$$\text{RHS}[i, j] = \sum_{[k,l] \in N(i,j)} \widetilde{y}_m[k, l], \tag{4.13}$$

where we have defined the neighborhood set

$$N(i, j) = \{[k, l] \mid \forall\, m \in [1, M],\ s \cdot k = i + dx(m),\ s \cdot l = j + dy(m)\} \tag{4.14}$$

Plugging the definition of $\widetilde{y}_m$ from Equation (4.11) yields

$$\text{RHS}[i,j] = \sum_{[k,l]\in N(i,j)} \sum_{t=1}^{T} W_{m,t}[k,l] y_t[k,l]. \tag{4.15}$$

In this expression, $W_{m,t}[k,l]$ refers to the entry on the main diagonal in $W_{m,t}$ that multiplies the $[k,l]$ entry in y_t. This formula indicates that each pixel in the RHS is a weighted sum of pixels, in a neighborhood centered around its equivalent location in the low-resolution image.

We now turn to discuss the Left-Hand-Side (LHS) in (4.12). The operator $D^T \widetilde{W}_m D$ within this expression is a diagonal matrix that decimates an image by a factor of s in each axis, weights each pixel by the diagonal weight matrix $\widetilde{W}_m$, and then up-scales back the image using the same factor by zero-filling. When this operator is applied to an image v, a pixel in location $[i,j]$ is nulled if $[i,j]/s$ is a non-integer (since it is one of the pixels to be zero-filled by D^T), and is simply weighted otherwise, i.e., it becomes $\widetilde{W}_m[i,j] \cdot v[i,j]$.

When the full operator $F_m^T D^T \widetilde{W}_m D F_m$ is applied to the $[i,j]$-th pixel in 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, $[i,j]$. Evidently, the operator $F_m^T D^T \widetilde{W}_m D F_m$ returns every pixel to its original location. Since every output pixel depends only on the value of the input pixel in the same location, this matrix is diagonal. Therefore, each pixel in the LHS is the pixel in z multiplied by a pixel-specific scalar, and can be computed by

$$\text{LHS}[i,j] = \sum_{[k,l]\in N(i,j)} \widetilde{W}_m[k,l] z[i,j] = \sum_{[k,l]\in N(i,j)} \sum_{t=1}^{T} W_{m,t}[k,l] z[i,j], \tag{4.16}$$

where we have substituted the definition of $\widetilde{W}_m$ in Equation (4.11). This expression is similar to Equation (4.15), summing only the weights and serving as a normalization term. Assuming that this sum is positive (i.e., at least one weight is non-zero), combining Equations (4.15) and (4.16) leads to a closed form expression for the $[i,j]$-th pixel in the estimated z,

$$\hat{z}[i,j] = \frac{\sum_{[k,l]\in N(i,j)} \sum_{t=1}^{T} W_{m,t}[k,l] y_t[k,l]}{\sum_{[k,l]\in N(i,j)} \sum_{t=1}^{T} W_{m,t}[k,l]}, \tag{4.17}$$

where m is related to $[k,l]$ through $[i,j] + [dx(m), dy(m)] = [k,l]$. The resemblance to the fusion algorithm in NLM-SR is evident (see Equation (30) in [13]). 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 averaging. The proposed algorithm differs considerably from an interpolation followed by application of NLM. A visual comparison between the two in the experimental section will demonstrate the difference.

4.3.5 Computing the Weights

In the development of the closed-form formula for z, we assumed that the weighting matrices $W_{m,t}[i,j]$ are known. We now turn to explain how $W_{m,t}[i,j]$ are computed, in order to complete the description of the algorithm. Observing Equation (4.8), these weights are supposed to encompass the fit, per pixel, of the desired high resolution image z after being transformed by F_m and decimated by D, with the input image y_t. Thus, the weights could be related to the error $DF_mz - y_t$. Since the pixel value in itself is not enough to properly estimate the fit, we propose to use some spatial support for each pixel instead of computing the difference on a single pixel. Defining $R_{i,j}$ 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

$$\begin{aligned} W_{m,t}[i,j] &= \exp\left\{-\frac{\|R_{i,j}\left(DF_mz - y_t\right)\|_2^2}{2\sigma^2}\right\} \\ &\cdot \quad f\left(\sqrt{\left(dx(m)\right)^2 + \left(dy(m)\right)^2 + (t-1)^2}\right). \end{aligned} \tag{4.18}$$

This formula is composed of the two independent parts. The first yields a value that is inversely proportional to the Euclidean distance between the transformed image DF_mz and the input image y_t, computed over some support around each pixel. This term reflects the per-pixel fit of the displacement (after decimation). The second term reflects a decreasing confidence in large spatial and temporal displacements, and 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 relies on the knowledge of the unknown z. Instead, at the beginning, the weights are computed by using an estimated version of z, such as a scaled-up version of the reference frame y_1. This scale-up is done using a conventional image interpolation algorithm such as bilinear, bicubic, or the Lanczos method. As this is only a crude version of the desired outcome, the process can be iterated, using the newly estimated image $\hat{z}$ to obtain more accurate weights that contribute to an improved outcome. 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 both approaches, block-matching is used to crudely estimate the probability of each trajectory. However, in classic block-matching based SR, only the most likely of those trajectories is selected, while all other trajectories are ignored. In the proposed algorithm, all trajectories are considered together, in a probabilistic framework, reflecting the varying confidences of the trajectories. This difference is what enables the proposed algorithm to handle complex scenarios where highly accurate motion estimation is not currently possible.

As said earlier, outliers are to be assigned zero weights. Outliers are characterized by very different patches. Therefore, the block distance in Equation 4.18 is very large, which through the inverse exponent is translated to a negligible weight. Thus, the outliers are indeed assigned practically zero weight, and are effectively ignored.

4.3.6 Other Resampling Tasks

In this section we describe how the proposed framework can be adapted to other re-sampling tasks, such as de-interlacing, inpainting and more, and start by explaining this extension intuitively. Re-sampling tasks 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 in 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) y_t, it can be linked to the original (unknown) image Y_t using a masking operator $M_t : y_t = M_t Y_t$. Simply put, M_t discards all unsampled pixels. It is a binary matrix, with as many rows as the number of pixels in y_t and as many columns as pixels in Y_t, with entries of ones indicating which pixels are to be kept. Note that y_t contains only sampled pixels. In the in-painting case, it contains only the unmasked pixels.

In line with the idea of the probabilistic motion estimation, Y_t can be constructed as a (pixel-wise) weighted average of different transformations of the target image x. The image x that we seek should be as similar as possible to each 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 Equation (4.7) arises, where the decimation operator is replaced by M_t,

$$\epsilon^2_{PML}(x) = \frac{1}{2} \sum_{m=1}^{M} \sum_{t=1}^{T} \|M_t H F_m x - y_t\|^2_{W_{m,t}} . \tag{4.19}$$

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 z is given by Equation (4.17). The difference is in the 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 Equation (4.18). However, these tasks can benefit from computing the weights in high-resolution scale. Thus, if we consider that $W_{m,t}$ is for the coarse scale, we denote $W_{m,t} = M_t \tilde{W}_{m,t}$, with $\tilde{W}_{m,t}$ being the same size as Y_t. The formula for each entry of $\tilde{W}_{m,t}$ (when arranged as an image) is therefore the same as in Equation (4.18), but with $F_m z - Y_t$ replacing $DF_m z - y_t$.

In these weights, Y_t is an interpolated version of 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 $W_{m,t} = M_t\tilde{W}_{m,t}$.

4.4 Experimental Validation

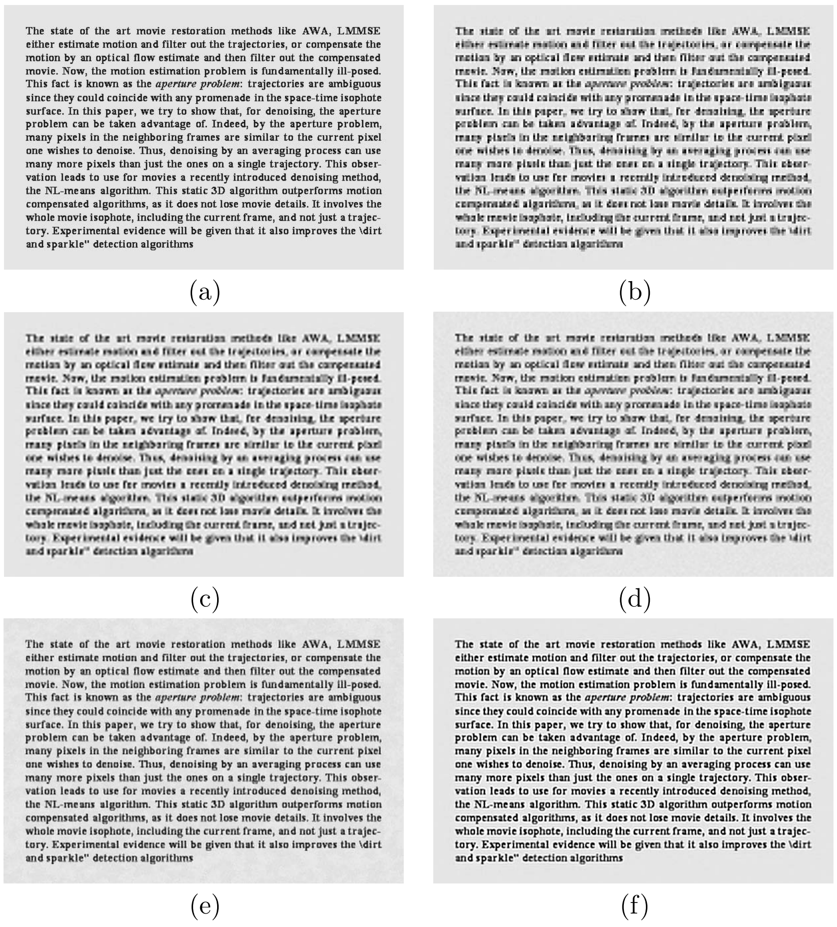
4.4.1 Experimental Results

In this section we demonstrate the abilities of the proposed algorithm in super-resolving general content 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 several real-world sequences with a general motion pattern. The comparison we provide in most sequences is to a single image upsampling using the Lanczos algorithm [21, 18], that effectively approximates the Sinc interpolation. Finally, we demonstrate the adaptation of the algorithm to the de-interlacing problem.

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 (in the input range $[0, 255]$) is used to generate a 9-image input sequence, by applying integer displacements prior to blurring (using a 3×3 uniform mask), decimation (by a factor of $1:3$ in each axis), and the addition of noise (with $std = 2$). 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 Figure 4.1, including a comparison to both Lanczos interpolation and the regularized shift-and-add algorithm [5, 7], which is a conventional motion-estimation-based super-algorithm resolution.

The block size used for computing the weights ($\hat{R}$) 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 real-world 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.

The similarity between the quality of the classic SR result and the proposed algorithm is evident. This similarity stems from the large block size used in the proposed algorithm. This large block size, together with the exiting global translation, makes the proposed algorithm converge to classic motion-estimation-based format, as such large blocks basically identify the correct motion vector for each pixel. We note that such large blocks cannot be used



The state of the art movie restoration methods like AWA, LMMSE either estimate motion and filter out the trajectories, or compensate the motion by an optical flow estimate and then filter out the compensated movie. Now, the motion estimation problem is fundamentally ill-posed. This fact is known as the *aperture problem*: trajectories are ambiguous since they could coincide with any promenade in the space-time isophote surface. In this paper, we try to show that, for denoising, the aperture problem can be taken advantage of. Indeed, by the aperture problem, many pixels in the neighboring frames are similar to the current pixel one wishes to denoise. Thus, denoising by an averaging process can use many more pixels than just the ones on a single trajectory. This observation leads to use for movies a recently introduced denoising method, the NL-means algorithm. This static 3D algorithm outperforms motion compensated algorithms, as it does not lose movie details. It involves the whole movie isophote, including the current frame, and not just a trajectory. Experimental evidence will be given that it also improves the \dirt and sparkle" detection algorithms

(a) (b) (c) (d) (e) (f)

FIGURE 4.1: Results for the synthetic text sequence. (a) Original (ground-truth) image. (b) Pixel replicated image, 13.47dB. (c) Lanczos interpolation, 13.84dB. (d) Deblurred Lanczos interpolation, 13.9dB. (e) Result of shift-and-add algorithm [5, 7], 18.4dB. (f) Result of proposed algorithm, 18.48dB.

in real-world sequences (shown next), as they do not allow enough adaptation to the various motion patterns, and therefore much smaller block sizes will be used.

We now turn to demonstrate the potential of the proposed SRR algorithm by presenting the results for image sequences with a general motion pattern. These sequences are also in the input range $[0, 255]$. Each Low Resolution (LR) frame is generated from one High Resolution (HR) frame. The HR frame is 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 $STD = 2$. It is important to note the while the LR images are synthetically generated from the HR images (using a known blur kernel and decimation operator) the motion in the sequence is real, and is not the result of synthetic manipulations.

The degraded sequence is then input to the proposed SRR algorithm. The results for three such degraded sequences: "Miss-America," "Foreman" and "Suzie" appear in Figures 4.2, 4.3, and 4.4 respectively, for the 3rd, 8th, 13th, 18th, 23rd, and the 28th frames of each sequence.[5] The window size used for computing these weights is set to 13×13, to allow handling complex and local motion patterns (unlike the text example, in which the motion was global). The search area was manually adapted for each sequence to ensure that the real motion is within the search area.

Another example along the same lines appears in Figure 4.5. In this test, a color High-Definition (HD) sequence was blurred by a 2×2 uniform mask, and downsampled by a factor of 2 (in each axis). The figure shows a portion of one HD frame and the same portion of the result of the proposed algorithm. Since this is a color sequence, the images are converted into the YUV colorspace, and only the Y channel is processed by the proposed algorithm. The U and V channels are interpolated, and the three components are then converted back to RGB colorspace to create the final SR result. This example shows that while the result is not identical to the input, they are of comparable quality.

In order to demonstrate the proposed algorithm on a directly captured sequence, we provide another experiment on the sequence "Trevor," the results of which are displayed in Figure 4.6. In this case, there is no ground-truth image available to compare to. Therefore, to demonstrate that a super-resolution effect is indeed 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 an 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 (see Equation 4.17). Clearly, a far better image is obtained with the proposed algorithm.

In order to demonstrate the generalized algorithm, we apply it to an interlaced sequence. We used the Foreman sequence and composed each interlaced frame from a pair of original frames 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.

The result of processing this sequence with the framework suggested in

[5]The sequences appearing in this section (input and output) and others from [13], along with the various parameters used to generate them, can be found at http://www.cs.technion.ac.il/~matanpr/NLM-SR.

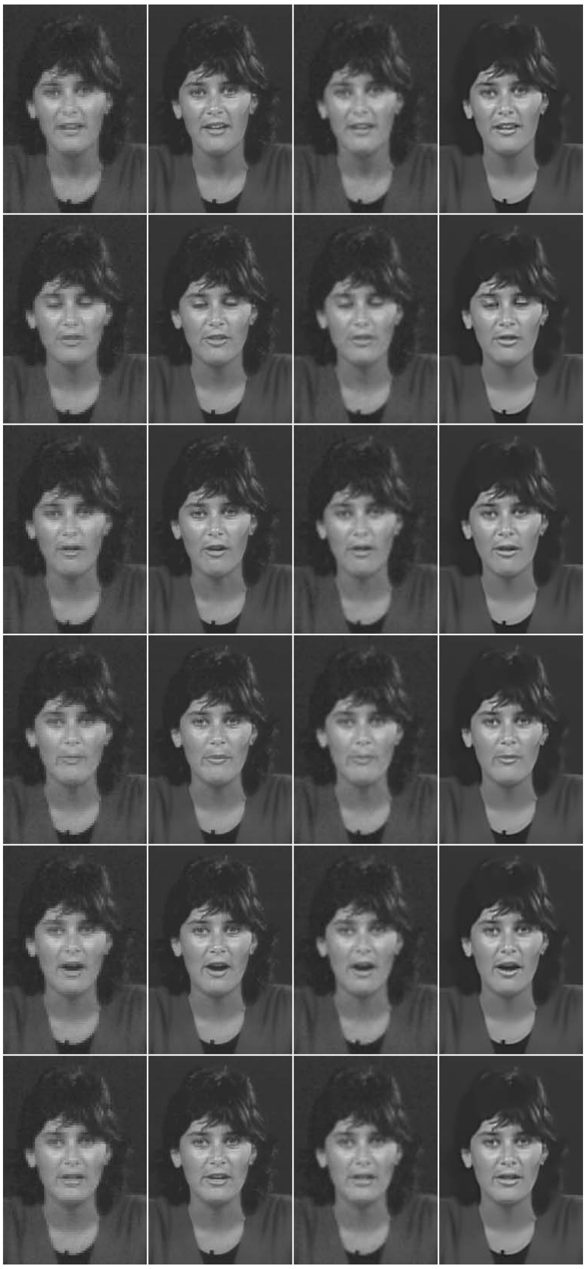


FIGURE 4.2: Results for the 3rd, 8th, 13th, 18th, 23rd, and the 28th frames from the "Miss America" sequence. From left to right: pixel-replicated low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.

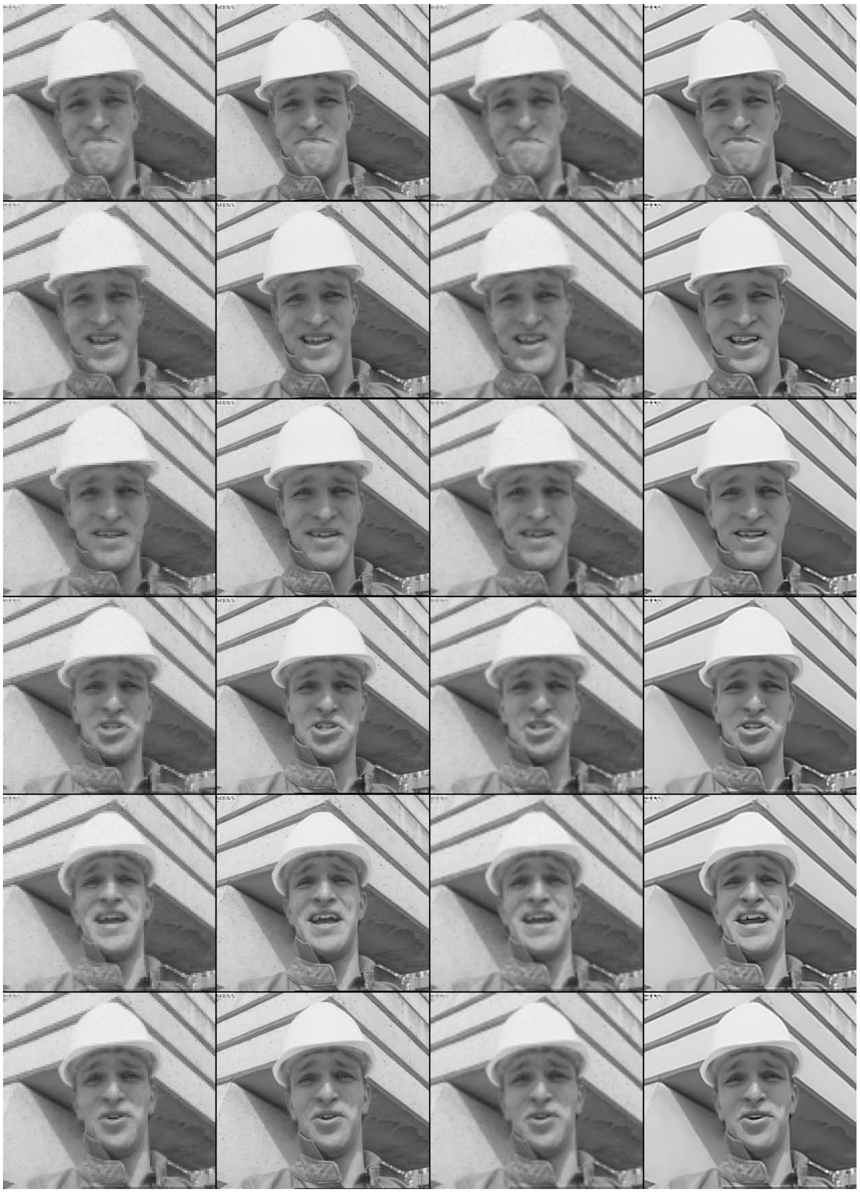


FIGURE 4.3: Results for the 3rd, 8th, 13th, 18th, 23rd, and the 28th frames from the "Foreman" sequence. From left to right: pixel-replicated low-resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.

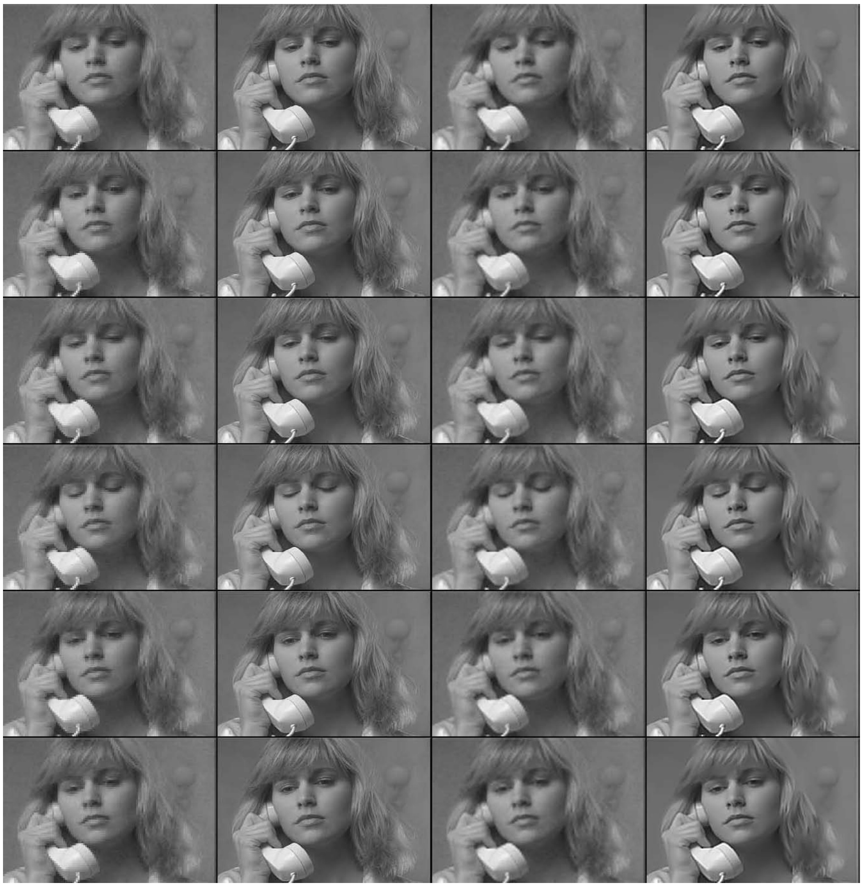


FIGURE 4.4: Results for the 3rd, 8th, 13th, 18th, 23rd, and the 28th frames from the "Suzie" sequence. From left to right: pixel-replicated low resolution image; original image (ground truth); Lanczos interpolation; result of the proposed algorithm.

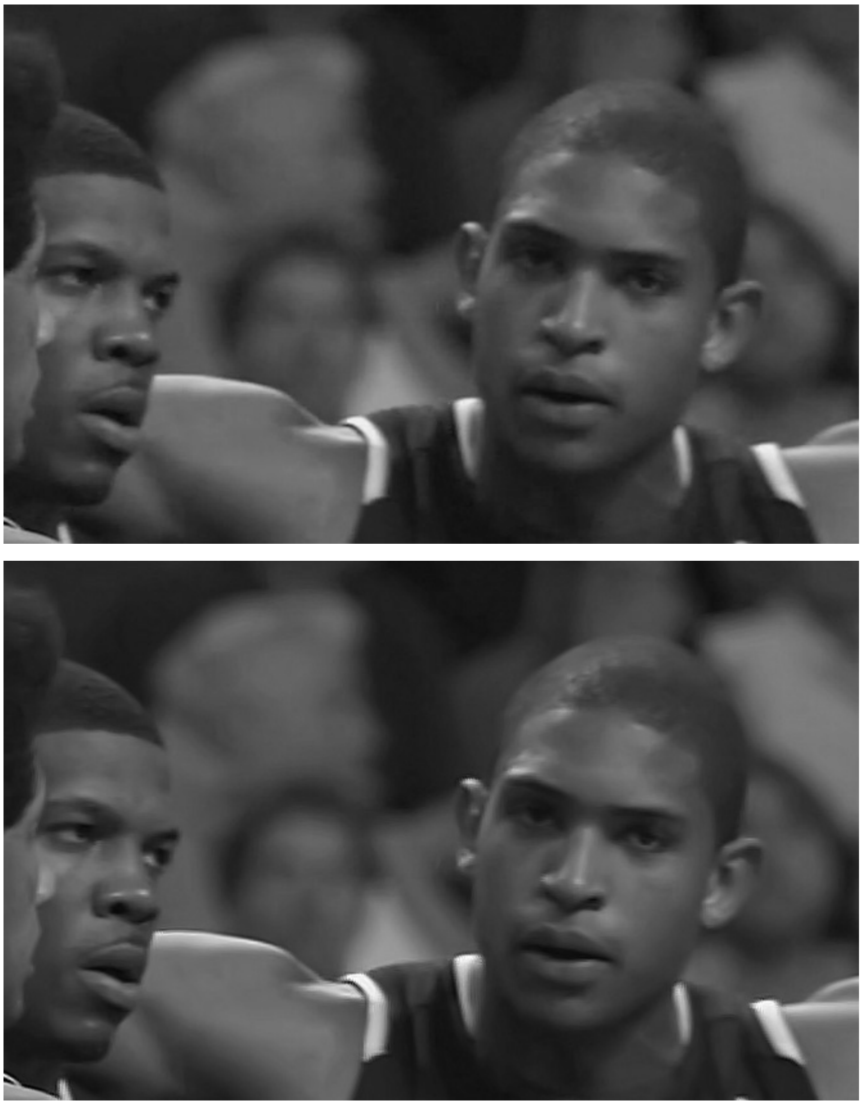


FIGURE 4.5: A High-Definition sequence. Top: Portion of original HD image. Bottom: Same portion of SR result (from an input downscaled by 2 in each axis).

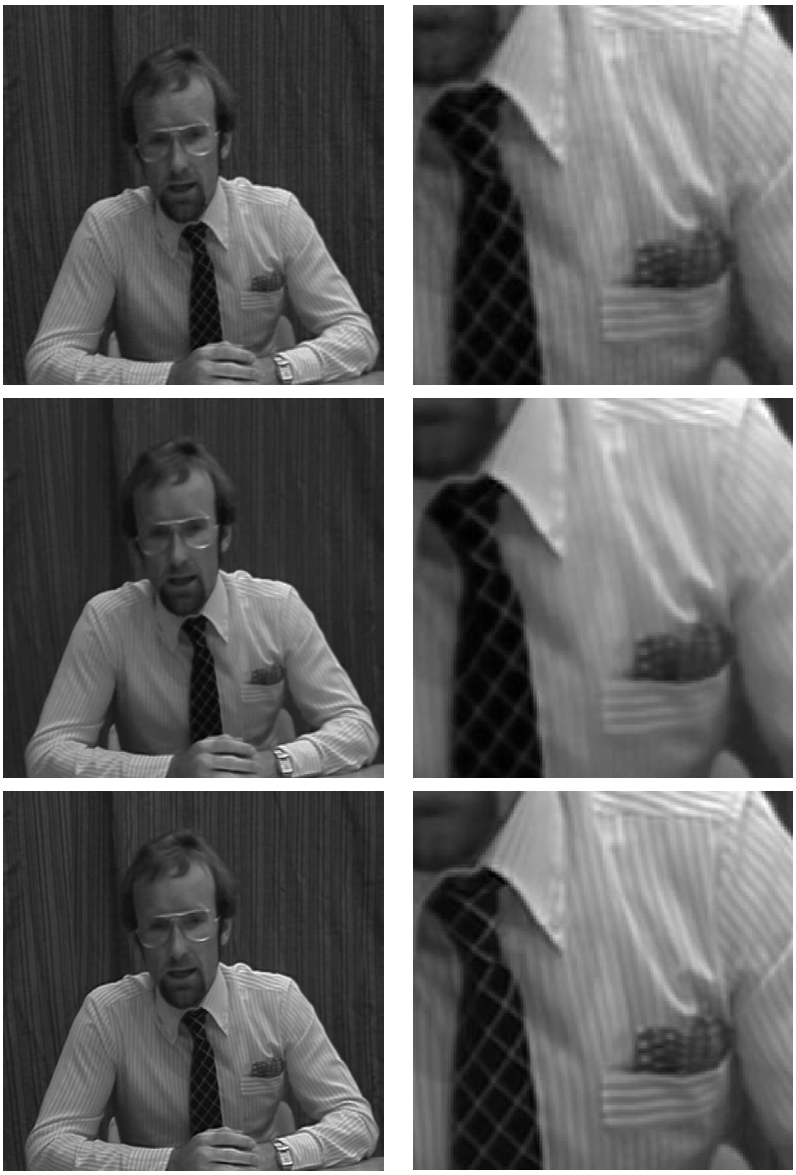


FIGURE 4.6: Original "Trevor" sequence. Top: Interpolated image. Middle: Interpolation, followed by NLM processing and deblurring. Bottom: Proposed algorithm. The right column offers a close-up of a portion of the images.

Section 4.3.6 appears in Figure 4.7. 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 M_t were designed to discard the even rows in the odd-numbered images, and the odd rows in the even-numbered images. 5 interlaced frames (10 fields) were used for processing, and the search area consisted of 10 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.

4.4.2 Computational Complexity

The complexity of the algorithm is essentially the same as that of the NLM algorithm, with the addition of a deblurring process, which is negligible compared to the fusion stage. The core of the algorithm, which also requires most of the computations, is computing the weights. In a nominal case in which a search area of 31×31 low-resolution pixels in the spatial domain, and 15 images in the temporal axis, we have $\approx 14,000$ pixels in this spatiotemporal window. For each pixel in the search area, the block difference is computed, with a block size of 13×13 (high-resolution) pixels. Thus, there is a total of almost $2,400,000$ operations per pixel. Performing this amount of calculations for every pixel makes the algorithm irrelevant for practical implementations, and therefore the computational load must be reduced. In this section we describe a few possible speed-up options for the proposed algorithm. Several of the methods to speed-up the NLM algorithm were suggested originally in [10], 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 $\approx s^2$. This can be applied for the first iteration, resulting in only a small loss of quality.

2. Computing fast estimations for the similarity between blocks, such as the difference between the average gray level or the average direction of the gradient, can eliminate many nonprobable destinations from further processing. Such an approach was suggested in [10], and was found to be very effective for the original NLM algorithm.

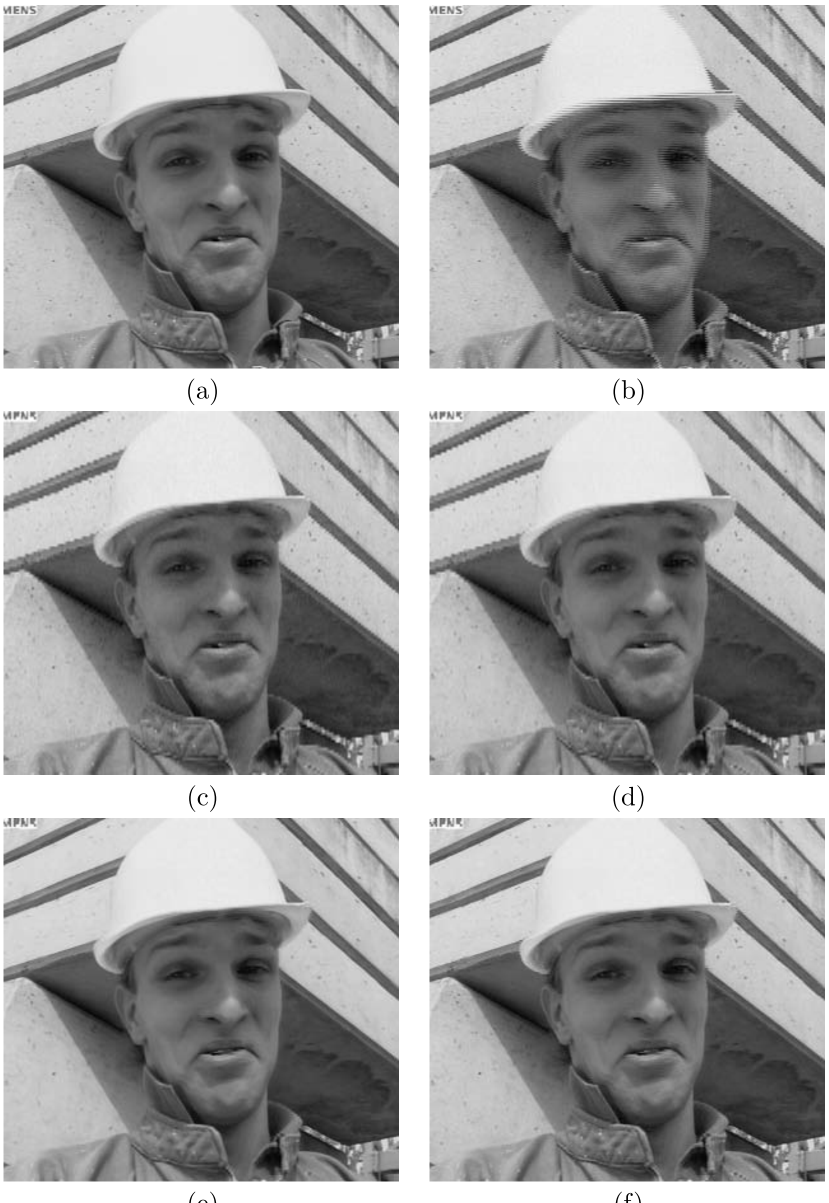


FIGURE 4.7: De-Interlacing Results. (a) Original (ground-truth) image. (b) Interlaced image. (c) Row Averaging, 29.87dB. (d) Row Averaging followed by NLM processing, 29.93dB. (e) Proposed algorithm first iteration, 30.69dB. (f) Proposed algorithm second iteration, 30.71dB.

3. If the patch used to compute the weights is rectangular and with uniform weights, the special structure of the patch can be used to dramatically speed up the computation of the weights. The Integral Image [20]. ($II(x,y) = \sum_{i=1}^{x} \sum_{j=1}^{y} I(i,j)$) can be used to compute the block differences using only a small constant number of calculations per pixel, regardless of block size. Fortunately, there is only a slight effect on the quality of the outputs of using such a patch structure.

4. A coarse-to-fine approach, transferring only high likelihood destinations from the coarse level, reduces the effective search area for each pixel, thus reducing the number of required calculations.

5. Since it is more likely that large spatial displacements will appear when the temporal distance is large, using a small search area in nearby frames and enlarging it as the temporal distance grows, can reduce the effective search area and thus the total number of calculations.

6. 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. Furthermore, as parallel hardware such as Graphical Processing Units (GPUs) are very common and powerful, with programming tools making implementations on such hardware easier than before, the parallelistic nature of this algorithm might allow a great speed-up by an implementation on such processing units.

These suggested speed-up methods can reduce the complexity by at least 3 to 4 orders of magnitude without a noticeable drop in the quality of the outputs. This makes the proposed algorithm practical.

As for the memory requirements, the proposed algorithm uses approximately as much memory as required to hold the entire processed sequence in the high-resolution scale, and is usually not a limitation. However, some of the speed-up methods suggested do require more memory, so a trade-off between memory requirements and run-time may be needed.

4.5 Summary

In NLM-SR [13], an earlier work, an explicit-motion-estimation-free SRR algorithm was developed by extending the NLM filter to SRR reconstruction. This chapter approaches the same task from a different perspective, basing it on a probabilistic and crude motion estimation instead. Interestingly, this approach (under some assumptions) leads to the same algorithm as in NLM-SR. However, since the formulation described here relies on the classic super-resolution framework and on the imaging model, we believe it is more intuitive. We give several examples of the abilities of the proposed algorithm in super-resolving various sequences.

Another benefit of this formulation is that it allows for different extensions than those proposed in [13]. In this chapter, we have shown that this framework can in fact be adapted to any re-sampling task. We have given one example of de-interlacing, showing 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.

While the results of the proposed algorithm are encouraging, we believe further research is needed in order to extract the full potential of this family of algorithms. We note that in developing the algorithm, we have made several choices such as relying on integer displacements, only in order to simplify the development of the algorithm, and to arrive at an algorithm that is relatively simple to understand and implement. Avoiding such compromises will result in a more complex algorithm, but one that might also bring about better results.

Bibliography

[1] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):1167–1183, September 2002.

[2] A. Buades, B. Coll, and J.M. Morel. Denoising image sequences does not require motion estimation. In *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance September (AVSS)*, pages 70–74, 2005.

[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 Transactions. on Consumer Electronics*, 54(3):1430–1438, August 2008.

[4] M. Elad and Y. Hel-Or. A fast super-resolution reconstruction algorithm for pure translational motion and common space-invariant blur. *IEEE Transactions on Image Processing*, 10(8):1187–1193, August 2001.

[5] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Robust shift and add approach to superresolution. In *SPIE Conference on Applications of Digital Signal and Image Processing*, pages 121–130, August 2003.

[6] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Advances and challenges in superresolution. *International Journal of Imaging Systems and Technology*, 14(2):47–57, August 2004.

[7] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe superresolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, October 2004.

[8] R.C. Hardie, K.J. Barnard, and E.E. Armstrong. Joint map registration and high-resolution image estimation using a sequence of undersampled images. *IEEE Transactions on Image Processing*, 6(12):1621–1633, December 1997.

[9] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP: Graphical Models and Image Processing*, 53(3):231–239, May 1991.

[10] M. Mahamoudi and G. Sapiro. Fast image and video denoising via nonlocal means of similar neighbourhoods. *IEEE Signal Processing Letters*, 12(12):839–842, December 2005.

[11] L.C. Pickup, D.P. Capel, S.J. Roberts, and A. Zisserman. Overcoming registration uncertainty in image super-resolution: Maximize or marginalize? *EURASIP Journal on Advances In Signal Processing*, 2007:Article ID 23565, 14 pages, 2007.

[12] M. Protter and M. Elad. Super-resolution with probabilistic motion estimation. *IEEE Transactions on Image Processing*, 18(8):1899–1904, August 2009.

[13] M. Protter, M. Elad, H. Takeda, and P. Milanfar. Generalizing the nonlocal-means to super-resolution reconstruction. *IEEE Transactions on Image Processing*, 18(1):36–51, January 2009.

[14] L.I. Rudin, S.J. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[15] R.R. Schultz and R.L. Stevenson. Extraction of high-resolution frames from video sequences. *IEEE Transactions on Image Processing*, 5(6):996–1011, June 1996.

[16] H. Shen, L. Zhang, B. Huang, and P. Li. A map approach for joint motion estimation, segmentation, and super resolution. *IEEE Transactions on Image Processing*, 16(2):479–490, February 2007.

[17] H. Takeda, P. Milanfar, M. Protter, and M. Elad. Super-resolution without explicit subpixel motion estimation. *IEEE Transactions on Image Processing*, 18(9):1958–1975, September 2009.

[18] K. Turkowski. Filters for common resampling tasks. In *Graphical Gems*, pages 147–165. Academic Press Professional Inc., San Diego, CA, USA, 1990.

[19] P. van de Walle, S. Susstrunk, and M. Vetterli. A frequency domain approach to registration of aliased images with application to super-resolution. *EURASIP Journal On Applied Signal Processing*, 2006:1–14, March 2006.

[20] P.A. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I: 511–518, December 2001.

[21] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.