One-step dejittering of digital video images

Mila Nikolova

CMLA, ENS Cachan, CNRS, PRES UniverSud, 61 Av. Président Wilson, F-94230 Cachan, France
(33/0) 1 47 50 59 08 nikolova@cmla.ens-cachan.fr

Key words: Error measures, Fast algorithms, Image modeling, Image restoration, Jitter, Non-smooth functions, Regularization, Video

Abstract

We propose several fast (quasi real-time) methods to dejitter digital video images in one step. They are based on an essential disproportion of the magnitude of the second-order vertical differences along the columns of a real-world image and all its jittered versions. These methods need one iteration to dejitter the image, which involves a number of steps equal to the number of its rows. They are designed for gray-value and color images (when the jitter is the same for all color channels), as well as to slightly noisy images. A reasonable version of these methods can be considered as parameter-free. We propose a two-phase extension enabling to restore jittered images corrupted with strong white noise (10 dB SNR or less). We consider specific error measures adapted to assess the success of dejittering. The results we obtain outperforms by far the existing methods: first in quality, second in speed (the ours need no more than 1 second for a 512 × 512 image on Matlab). Our methods provide a crucial step towards real-time dejittering of digital video sequences.

1 Intrinsic dejittering

Image jittering consists in a random horizontal displacement of each line of the image. Jittering occurs e.g. in video images when the synchronization signals—carrying the information about the proper location of the lines relative to each other—are corrupted by noise or degradation of the storage medium. Line jittering can also appear in wireless video transmission, due to some electromagnetic interference. The visual effect is quite disturbing since each line is randomly displaced within a range ±1 to ±5 pixels or more. Then shapes appear to be jagged in the vertical direction—a first example is shown in Fig. 1 (c). Another form—structured (e.g. sinusoidal jitter)—can be provoked by acoustic, electrical or other interferences [10]. The rows of the image are displaced with a number of pixels corresponding to the frequency and the amplitude of the electrical perturbation; then vertical lines are transformed into sinusoids—an example can be seen in Fig. 10. Time base corrector machines process with some success the analogue video signal in order to recover the line synchronization information [9]. In many cases, such an operation is unsuccessful or impossible. The alternative approach—to restore the image frames directly from the observed jittered data—is often called intrinsic dejittering [10]. It naturally uses some prior assumptions on natural images. We focus on such an approach since it is more flexible and more widely applicable.

The very trivial solution to this problem—searching for the shift between consecutive lines that maximizes their correlation—is known to cause a bias towards vertical lines and fails in most of the cases [9]. Intrinsic dejittering was really inaugurated by Kokaram et al. in [8]. An more mature version of the method is nicely explained in chapter 5 of the textbook on film and video of Kokaram [9]. The algorithm presented there is based on a 2-D autoregressive model (2D AR) of the image. The unknown 2D AR coefficients and the erroneous horizontal displacements are considered by blocks. They are estimated jointly using an iterative algorithm. Drift compensation may be necessary to finalize the restoration. Laborelli proposes in [10] a totally different approach where the ℓ1 norm of the differences between consecutive shifted (two or three) lines is compared. The optimal shifts are recovered during a backward iteration in the context of a dynamic programming approach. Later on, Shen proposed a fully Bayesian method using a TV model on the underlying image [15] for joint dejittering and denoising. It requires to minimize a non-smooth function for each frame, which is time consuming. A more
flexible two-step method, called *Bake and Shake* is proposed by Kang and Shen in [6]. It relies on the idea that a good PDE-based image restoration method, such as the Perona-Malik diffusion, can help to find the best shifts to arrange the displaced lines. Recently, the same authors studied in [7] the mathematical properties of the slicing moments of images of bounded variation (BV) and derived an original variational method for dejittering, based upon Bayesian regularization of the slicing moments (in the vertical direction). Its numerical coast is important since it needs the minimization of a functional over the whole image domain. We could remind that even though BV models characterize some important geometrical features of images, it was shown in [5] that *natural images do not fit the BV model*.

In all cases, intrinsic dejittering is based on priors relevant to the underlying image. In the case of jitter, we demonstrate that pertinent prior on the columns of natural images is sufficient to discriminate wrong line shifts. It is based on the magnitude of the second-order differences along the columns of the image. Using such a prior we construct simple *one-iteration* dejittering methods, which is much easier than the methods evoked above and can work in real time. Exhaustive experiments shows that our methods outperform by far—both in quality and speed—the existing methods. Our methods can be seen as a first step towards real-time dejittering of digital video sequences.

**Plan of the work.** We would suggest that readers start by having a look at the illustrations in sections 4, 5 and 6, and to notice that the computation time is less than one second using (non-optimized) Matlab codes. In section 2 we propose and discuss on several possible criterias among shifted consecutive lines. At a second time, consider how to assess the quality of dejittering. In section 3 we present the main algorithm to restore noise-free jittered images based on the magnitude of second-order differences between consecutive lines. We also propose simple ramifications to refine the algorithm (if needed). Section 4 provides a series of illustrations to dejitter some popular (and difficult) gray-value images. By way of justification of our approach, we conduct a large-scale experiment where the mean and the variance of the main error values are calculated for 1000 experiments on four images. In section 5 we present a simple and fast extension to color images (having the same jitter for all color channels). A series of illustrations are presented and commented as well. Section 6 proposes how to deal with noisy jittered images. In all cases, we consider both independent jitter and structures (e.g. sinusoidal) jitter. Conclusions and perspectives are outlined in section 7.

## 2 Preliminary notes

We denote by $f$ the real $r \times c$ image and by $g$ its version degraded by jitter, also of size $r \times c$. Their rows are systematically denoted by $f_i$ and $g_i$, for $1 \leq i \leq r$, and the components of the latter by $f_i(j)$ and $g_i(j)$, for $1 \leq j \leq c$, respectively. The usual model for the production of a jittered image $g$ reads [9]

\[
\forall j \in \{1, \ldots, c\}, \forall i \in \{1, \ldots, r\}, \quad g_i(j) = \begin{cases} 
  f_i(j + d_i), & \text{if } 1 \leq j + d_i \leq c \\
  \text{any, e.g. } = 0, & \text{otherwise}
\end{cases} \quad d_i \in \mathbb{Z}, \quad |d_i| \leq M, \quad (1)
\]

where $d_i$ is a bounded integer. In practice, $M$ is no larger than 5-6 pixels [9]. Different types of jitter are systematized in the literature [10]:

- **Independent line jitter** which is usually modeled using a (truncated) Gaussian or a Uniform distribution;
- **Structured jitter**—such as low frequency or high frequency sinusoidal jitter.

Independent jitter is considered in [9, 10, 15, 6]. Sinusoidal jitter is considered in [10]. We provide experiments with both types of jitter.
Figure 1: Zoom of a 50 × 50 subimage of Lena. (a) Original. (b) One column of the original against one column of the jittered image. (c) Jittered image (±6 pixel random displacements).

2.1 Local criteria based on consecutive lines

We focus on methods where the recovery of the starting position of each line is based on the previously restored line positions. Even in such a restricted context, a variety of criteria can be imagined, based on the vertical behavior of natural images. Let us consider the example in Fig. 1(b): the left side shows a column of a real image while its right side shows the “same” column of the jittered image.

Remark 1

Our first observation is that a column of a real image can be seen as piecewise polynomial, of degree 2 or 3, which is hard to claim for the jittered column.

The dejittered image and the corresponding recovered displacements are systematically denoted by \( \hat{f} \in \mathbb{R}^{r \times c} \) and \( \hat{d} \in \mathbb{R}^{r} \). The rows of \( \hat{f} \) read \( \hat{f}_{i} \), \( 1 \leq i \leq r \); the components of each row \( \hat{f}_{i} \) are denoted by \( \hat{f}_{i}(j) \). Suppose that \( \hat{f}_{1}, \ldots \hat{f}_{i−1} \) (and hence \( \hat{d}_{1}, \ldots \hat{d}_{i−1} \)) are already recovered. Based on Remark 1, we are going to estimate the next displacement \( \hat{d}_{i} \) by comparing several previously dejittered rows \( \hat{f}_{i−1}, \hat{f}_{i−2}, \ldots \) with all possible shifts of the next data row \( g_{i}(j + d_{i}) \) for \( d_{i} \in [-N, N] \) where \( N \geq M \) is an overestimate of \( M \). Let us now focus on the left-hand and the right-hand side boundaries of a jittered image (e.g. Fig. 3).

Remark 2

Each row of \( g \) has at one of its extremes a certain number of undetermined pixels (zero-valued in practice) due to the jitter. We know that their number is at most equal to \( N \). Involving such pixels in our criterion \( J \) cannot help but can distort its meaning. For this reason, we restrict our attention only to \( g_{i}(j) \) for \( j = N + 1, \ldots, c - N \) which contains only information on the true image.

Using Remarks 1 and 2, we initially suggest several criteria, given in equations (2), (3) and (4) below.

\[
\hat{d}_{i} = \arg \min_{|d_{i}| \leq N} J_{1, \alpha}(d_{i})
\]

\[
J_{1, \alpha}(d_{i}) = \sum_{j=N+1}^{c-N} \left| g_{i}(j + d_{i}) - \hat{f}_{i−1}(j) \right|^\alpha, \ \alpha \in (0, 1].
\]  

Let us remind that this criterion, for \( \alpha = 1 \), was involved in a dynamic programming scheme in [10]. For any \( \alpha \in (0, 1] \), this \( J_{1, \alpha} \) favors constant transitions between consecutive lines, i.e. vertical lines in the dejittered image (the mathematical explanation can be found in [12]). The illustration in Fig.1(b) shows that this can hold only exceptionally. Considering second-order finite differences between three consecutive lines suggests

\[
\hat{d}_{i} = \arg \min_{|d_{i}| \leq N} J_{2, \alpha}(d_{i})
\]

\[
J_{2, \alpha}(d_{i}) = \sum_{j=N+1}^{c-N} \left| g_{i}(j + d_{i}) - 2\hat{f}_{i−1}(j) + \hat{f}_{i−2}(j) \right|^\alpha, \ \alpha \in (0, 1].
\]
This criterion favors the reconstruction of columns where each three consecutive samples form a linear segment, i.e. \( \hat{d}_i \) is such that for a large number of pixels \( j \) of the \( i \)th row, \( g_i(j + \hat{d}_i) \approx 2\hat{f}_i-1(j) - \hat{f}_i-2(j) \). This nearly holds along each columns of the true image but is false for the arbitrary displacements in \( g \), as seen in Fig. 1(b). Let us remind that the \( \ell_1 \) norm of the second-order differences was used in [10] within a dynamical approach where the conditionally optimal shift for the middle line is predicted based on the conditionally optimal shifts of the previous and the next lines; we can remark that this does not completely prevent from favoring vertical lines during the backward iteration.

Yet another option that seems reasonable with respect to Fig. 1(b) is to consider third-order finite differences,

\[
\hat{d}_i = \arg \min_{|d_i| \leq N} J_{3,\alpha}(d_i)
\]

\[
J_{3,\alpha}(d_i) = \sum_{j=N+1}^{c-N} |g_i(j + d_i) - 3\hat{f}_i-1(j) + 3\hat{f}_i-2(j) - \hat{f}_i-3(j)|^\alpha, \quad \alpha \in (0, 1].
\]

Such a criterion will favor columns that are locally quadratic on four pixels.

Once we have \( \hat{d} \), the rows of the dejittered image \( \hat{f} \) read

\[
\hat{f}_i(j) = g_i(j - \hat{d}_i),
\]

in accordance with (1).

**Remark 3** The effect of choosing \( \alpha < 1 \) in these criteria reinforces the polynomial constraint and may reduce the bias toward edges in the true image—according to the theoretical results in [13, 14]. However, the function \(|.|^\alpha \) must be increasing enough; so we consider only \( \alpha \in [\frac{1}{2}, 1] \).

Fig. 2 presents a toy example (My Sun) which is a real challenge for dejittering—the original image in (a) has many zero-valued patches at the left and right boundaries whereas its size is small, 128 \( \times \) 128. The image in (b) is corrupted with integer-valued jitter which is uniform on \{\(-5, \ldots, 5\)\}, so the displacements are important with respect of the size of the image and the thinness of the details that it involves. We observe that \( J_{1,\alpha}, \alpha \in (0, 1] \) works badly since it tends to recover vertically constant pieces. The criteria \( J_{2,0.5} \) and \( J_{2,1} \) work equally and
they yield the original image with no error, i.e. \( \hat{d} = d \). The criteria based on 3-\text{rd} order differences—\( J_{3,0.5} \) and \( J_{3,1} \)—perform less well. One explanation is that piecewise quadratic polynomials on 4 points cannot discriminate well between the true image and its wrongly shifted versions. The numerous experiments we have done show that \( J_{3,\bullet} \) is systematically less efficient than \( J_{2,\bullet} \); moreover it involves more numerical calculations than the previous ones. We focus basically on criteria of the form \( J_{2,\bullet} \) as given in (3).

### 2.2 Error measures for dejittering

The standard tools to assess the quality of a dejittered image with respect to the original one—e.g. SNR, PSNR, MSE, etc.—cannot be applied directly since the restored image \( \hat{f} \) is shifted with respect to \( f \) and the extremities of its rows are unknown (zero-valued in practice), because of the jitter. A possible way out is described next. We shrink \( f \) to \( \hat{f}^s \) according to

\[
\hat{f}^s(j) = \hat{f}(j + N), \quad 1 \leq j \leq c - 2N, \quad \forall i \in \{1, \ldots, r\},
\]

so that \( \hat{f}^s \) contains only proper image information (and has no gaps at the ends of its rows). Then we choose a shrink by 2\( N \) columns version of the original \( f \), say \( f^s \), that matches \( \hat{f}^s \) the best, and then evaluate the error of \( f^s - \hat{f}^s \). Different criteria can be used to define an optimal matching for \( f^s \), as well as for the measure of \( f^s - \hat{f}^s \). Notice that any measure of the error in \( f^s - \hat{f}^s \) is sensitive to the efficiency of the matching criterion for the choice of \( f^s \). Having in mind that the \( \ell_1 \) norm (defined by \( \|u\|_1 = \sum_{i=1}^{m} |u_i|, \forall u \in \mathbb{R}^m \)) is well suited for image evaluation, we seek for an \( f^s \in \mathbb{R}^{c \times (c - 2N)} \) such that

\[
\|f^s - \hat{f}^s\|_1 = \min_{0 \leq k \leq 2N} \sum_{i=1}^{r} \sum_{j=1}^{c-2N} |\hat{f}(j + k) - \hat{f}^s(j)|.
\]

For the same reason, we choose the \( \ell_1 \)-based mean absolute error (MAE), defined by

\[
\text{MAE}(\hat{f}, f) = \frac{1}{r(c-2N)}\|f^s - \hat{f}^s\|_1.
\]

The dynamic range of \( (\hat{f}^s, f^s) \) reads \( \delta \overset{\text{def}}{=} \max \{ \max_{i,j} \{f^s(i,j)\}, \max_{i,j} \{\hat{f}^s(i,j)\} \} - \min \{ \min_{i,j} \{f^s(i,j)\}, \min_{i,j} \{\hat{f}^s(i,j)\} \} \). We will also consider the peak-signal-to-noise ratio (PSNR), defined by:

\[
\text{PSNR}(\hat{f}, f) = 10 \log_{10} \frac{\delta^2 r(c-2N)}{\|f^s - \hat{f}^s\|_2^2}.
\]

where \( \|u\|_2 \) denotes the \( \ell_2 \)-norm (defined by \( \|u\|_2 = (\sum_{i=1}^{m} u_i^2)^{1/2}, \forall u \in \mathbb{R}^m \)). Remind that \( \text{PSNR} = \infty \) if \( f^s = \hat{f}^s \).

The quality of dejittering can also be evaluated based on \( d - \hat{d} \) where we remind that \( d \in \mathbb{R}^r \) is the vector of the horizontal displacements that degrade the original image \( f \) in (1) and that \( \hat{d} \) is its estimate. We will use error measures for \( d - \hat{d} \), based on the \( \ell_\infty \)-norm, the \( \ell_1 \)-norm, or the \( \ell_0 \) “norm” (defined by \( \|u\|_0 = \# \{ i \in \{1, \ldots, r\} : u_i \neq 0 \} \), where the symbol \( \# \) denotes cardinality).

#### Remark 4

Dejittering an isolated video image inevitably produces a shifted version of the displacement, say \( \hat{\rho} = \hat{d} + C \). When \( d \) is known, we can estimate \( \hat{d} \) by selecting \( C \) so that \( \hat{d}_i = d_i \) for a maximum number of samples, i.e.

\[
C = \arg \min_{C \in \mathbb{Z}} \|\hat{\rho} - C - d\|_0.
\]

More details are given in section 3—Algorithm: Shift recovery.

The following \( \ell_1 \)-based error measure \( e_1 \)

\[
e_1(\hat{d}, d) = \frac{1}{r} \|d - \hat{d}\|_1,
\]
evaluates the average displacement of the pixels along each column. It is the same for the pixels along any column of the image. It is quite a relevant measure for the quality of dejittering.

A measure that can be “absolute” in some cases is just

\[ e_\infty(\hat{d}, d) = ||d - \hat{d}||_\infty \overset{\text{def}}{=} \max_{1 \leq i \leq r} |d_i - \hat{d}_i|. \]  

(8)

When \( e_\infty \) is very small—say 1 or 2 pixels—we are guaranteed that dejittering is completely satisfying, independently of any other error measure. For some images, \( e_\infty \leq 3 \) can be fine as well. The reason is that a horizontal displacement of 1-2 (or 3) pixels remains invisible in an image having several hundreds of columns. For instance, Fig. 7 is almost perfectly dejittered and \( e_\infty(d, \hat{d}) = 2 \) whereas \( \text{MAE} \) and \( e_1 \) are quite important, the PSNR is not very high. However, if \( e_\infty \) is larger, its value is meaningless. E.g. Fig. 12 is visually perfectly dejittered but we have \( e_\infty(d, \hat{d}) = 20 \); for this images, \( \text{MAE} \) and \( e_1 \) are reasonably small and the PSNR is very high. More comments are provided in subsection 5.3.

The \( \ell_0 \)-based error measure \( e_0 \) below

\[ e_0(\hat{d}, d) = \frac{100}{r} ||d - \hat{d}||_0 \% \]  

(9)

gives the percentage of displaced rows in the restored image with respect to the original one. It is relevant in some cases, but if the image involves large homogeneous regions, it is meaningless.

If \( \hat{d} \) is exact, \( \hat{d} - d = 0 \). Otherwise, we observe that \( \hat{d} - d \) is composed of a certain number of constant segments (see e.g. Fig. 12). The \( \ell_0 \)-based error measure \( e_0^\Delta \) given below

\[ e_0^\Delta(\hat{d}, d) = \frac{100}{r - 1} \# \{ (\hat{d}_i - d_i) - (\hat{d}_{i+1} - d_{i+1}) \neq 0, \ 1 \leq i \leq r - 1 \} \% \]  

(10)

gives the percentage of changes in \( d - \hat{d} \). It mainly assess up to what degree our model for the underlying image (involved in J) fits the image under consideration.

For jitter degradation, the visual convenience still remains of paramount importance. Let us notice that the error measures given above do not take into account the content of the image \( f \).

**Remark 5 (Conclusion on error measures)** If \( e_\infty \leq 2 \) (or 3 in some cases), we are guaranteed that dejittering is satisfying, independently of any other error measure. The other error measures assess only indirectly if the important features of the underlying image are accurately recovered.

We conjecture that better measures could be constructed based on the error between the coefficients of the decompositions of \( f^s \) and \( f^s \) into an appropriate frame system, e.g. curvelets or contourlets.

### 3 Algorithms for gray-value images

Based on the observed jittered image \( g \in \mathbb{R}^r \times c \), we restore the original displacements \( \hat{d} \) within a range \( \{-N, +N\} \) such that that \( N \geq M \). For stability reasons, it is better to take \( N > M \) (this is explained in Remark 6 in § 3.1). The algorithm we propose below is based on the criterion \( J_{2,\bullet} \) in (3). We systematically denote by \( \theta_n \) the zero-valued row-vector of length \( n \):

\[ \theta_n = [0, \ldots , 0]_n. \]  

(11)

We use the observation that the shift of the top line can be assigned arbitrarily (since we work with a single image). We construct a larger matrix \( \hat{f} \) of size \( r \times (c + 2N) \) whose rows are denoted \( \hat{f}_i \); in the middle of its first row \( \hat{f}_1 \) we insert \( g_1 \). The sought-after dejittered image \( \hat{f} \) is an \( r \times c \) inner sub-matrix of \( \hat{f} \).
3.1 The main algorithm

The Matlab-inspired notation \([a, b, c]\) means that we concatenate horizontally the row-vectors \(a\), \(b\) and \(c\). Furthermore, the notation \(a \leftarrow b\) is used to say that we replace \(a\) by \(b\). Remind that the rows of a matrix \(\gamma\) having \(r\) rows are denoted by \(\gamma_i\), \(1 \leq i \leq r\). Before to run the algorithm, several choices are to do.

- Fix \(N > M\), e.g., \(N = M + 1\).
- Fix \(\alpha \in (0, 1]\), e.g. \(\alpha = 1\) or \(\alpha = 0.5\)—if the image involves regions with well organized texture.

Algorithm 1. (Gray value images)

1. Put \(\tilde{f}_1 = [\theta_N, \ g_1, \ \theta_N]\).
2. Split \(g\) into 3 sub-matrices \(g = [g^L : \gamma : g^R]\) where\(^1\) \(g^L \in \mathbb{R}^{r \times N}\), \(\gamma \in \mathbb{R}^{r \times (c-2N)}\) and \(g^R \in \mathbb{R}^{r \times N}\).
3. Put \(\hat{p}_0 = \hat{p}_1 = N + 1\) and \(\phi_1 = \phi_2 = [\theta_N, \ \gamma_1, \ \theta_N]\).
4. For any \(i = 2, \ldots, r\) do the following:
   
   (a) for any \(k = 1, \ldots, 2N + 1\)
   
   i. Put \(h^k = [\theta_{k-1}, \ \gamma_i, \ \theta_{2N-k+1}]\);  
   ii. Find \(m = \min \{k, \hat{p}_{i-1}, \hat{p}_{i-2}\}\) and \(n = \max \{k, \hat{p}_{i-1}, \hat{p}_{i-2}\} + c - 1\);
   iii. calculate \(J(k) = \frac{1}{n - m + 1} \sum_{j=m}^{n} \left|h^k(j) - 2\phi_1(j) + \phi_2(j)\right|^\alpha\);
   
   (b) find \(\hat{p}_i = \arg \min_{1 \leq k \leq 2N + 1} J(k);\)
   
   (c) Substitute: \(\phi_2 \leftarrow \phi_1\) and \(\phi_1 \leftarrow h^{\hat{p}_i} = [\theta_{\hat{p}_{i-1}}, \ \gamma_i, \ \theta_{2N-\hat{p}_{i+1}}];\)
   
   (d) Put \(\tilde{f}_i = [\theta_{\hat{p}_{i-1}}, \ g_i, \ \theta_{2N-\hat{p}_{i+1}}]\);
5. Extract \(\tilde{f} \in \mathbb{R}^{r \times c}\) from \(\tilde{f} \in \mathbb{R}^{r \times (c+2N)}\) by eliminating \(2N\) columns at the extreme left and right ends that contain the largest number of zeros or jitter.

In this algorithm, \(\hat{p}_i\) is the estimated position for the first pixel of \(g_i\) in \(\tilde{f}_i\)—the \(i\)th row of the enlarged image \(\tilde{f}\). Notice that \(\phi_1, \ \phi_2\) and \(h^k\) are \(c\)-length row vectors such that at step \(i\), \(\phi_1\) corresponds to the estimate for row \((i-1)\), \(\phi_2\)—for row \((i-2)\) and \(\hat{p}_i\) in item 4c—for row \((i)\).

The range for \(k\) in step 4a comes from the fact that \#\{-\(N\), \ldots, 0, \ldots, \(N\)\} = \(2N + 1\). Thus \(h^k\) in step 4(ai), for \(k = 1, \ldots, 2N + 1\), contains all possible shifts for the line that is considered. The introduction of \(m\) and \(n\) in 4(aii) and their use in the sum \(J\) in 4(aiii) comes from Remark 2: we exclude from criterion \(J\) in 4(aiii) all samples that could be due to the jitter—zeros at the extremities of \(h^k, \ \phi_1\) and \(\phi_2\). The normalization factor there accounts for the fact that the number of terms in the summation varies. Notice that in step 4d we insert in \(\tilde{f}_i\) the whole observed \(g_i\) (and not only its restriction \(\gamma_i\), as we did in 4(ai) i).

The size of the image \(\tilde{f}\) obtained at the end of step 4 is \(r \times (c + 2N)\). Step 5 is aimed at obtaining an \(r \times c\) size restoration \(\tilde{f}\) as an inner submatrix of \(\tilde{f}\). Some preliminary illustrations of ALGORITHM 1 can be seen in Fig. 2(e)-(f) as well as in Fig. 3. In both cases, the jitter is very strong compared to the width of the image.

\(^1\)More precisely, \(\forall i \in \{1, \ldots, r\}\) we have

\[g^i(j) = g_i(j), \quad \forall j \in \{1, \ldots, N\},\]
\[\gamma^i(j) = g_i(j), \quad \forall j \in \{N+1, c-N\},\]
\[h^i(j) = g_i(j), \quad \forall j \in \{c-N+1, c\};\]
Algorithm 1, $\alpha = 1$ \Original image. Algorithm 1, $\alpha = 0.5$.

Figure 3: The original image (size $116 \times 200$) is contaminated with independent jitter uniformly distributed on $\{-6, \ldots, 6\}$.

As noticed in remark 4, the vector $\hat{p} \in \mathbb{R}^r$ is shifted with respect to the estimate of the displacement $\hat{d}$, namely

$$\hat{d}_i = \hat{p}_i - C, \ \forall i \in \{1, \ldots, r\}. \quad (12)$$

The constant $C$ satisfies

$$1 - N \leq C \leq 3N + 1. \quad (13)$$

Indeed, $\forall i \in \{1, \ldots, r\}$ we have $|d_i| \leq N$ and $1 \leq \hat{p}_i \leq 2N + 1$, hence $1 - N \leq \hat{p}_i - d_i \leq 3N + 1$. In order to compute the error measures defined in (7), (8) and (9), we have to find the shift $C$. Given the true displacement $d$, we find $C$ using the criterion in Remark 4. This can be done using the numerical scheme given below.

Algorithm: \textit{Shift Recovery.}

1. Define the set of integers $I = \{-N + 1, \ldots, 3N + 1\}$.
2. Let $H$ be the histogram of $\hat{p} - d$ on $I$—i.e. $H(n) = \#\{j \in I : \hat{p}(j) - d(j) = n\}$, for $n \in I$.
3. Obtain $C = \arg \max_{n \in I} H(n)$. Then $\hat{d}_i = \hat{p}_i - C, \ \forall i \in \{1, \ldots, r\}$.

Remark 6 (Choice of $N$) Suppose that we choose $N = M$ and that a $\hat{p}_i$ is wrongly estimated as $\hat{p}_i = 1$ or $\hat{p}_i = 2M + 1$. Then the possible shifts for the next line are restricted only to the right with respect to row $i$ if $\hat{p}_i = 1$, or only to the left with respect to the same row if $\hat{p}_i = 2M + 1$. It can happen that the true position of row $i + 1$ with respect to row $i$ is in the opposite side, in which case the error on $\hat{p}_{i+1}$ will increase. This is the reason why we strongly recommend to choose $N > M$. Overall, our algorithms work quite correctly and in all our experiments it was enough to use

$$N = M + 1. \quad (14)$$

Nevertheless, we observed that in some cases, an improvement can be reached for $N = M + 2$ or $N = M + 3$. Even though these results were slightly better, we displayed only those obtained using (14). Remind that taking a larger $N$ slightly decreases the speed of the algorithm.

Computation times. The computation time depends clearly on the size of the image, on the value of $N$, and it is higher if we choose $\alpha = 0.5$ instead of $\alpha = 1$. We did some comparisons using Matlab 7.2 on a PC with a Pentium 4 CPU 2.8GHz and 1GB RAM, running on Windows XP Professional service pack 2. For a $512 \times 512$ size gray-value image and $N = 7$ we got the solution in 0.62 second for $p = 1$ and in 1 second for $p = 0.5$. Let us mention that our Matlab implementation is not optimal.

3.2 Refinement strategies

3.2.1 Boundary condition for the first row

The boundary condition for the first row chosen in step 3 amounts to $\gamma_0 = \gamma_1$. It means that we restore the second row $\hat{f}_2$ (and its displacement $\hat{d}_2$) using the first-order criterion given in (2), which is not fully satisfying. In many cases, a mirror boundary condition have been seen to produce better results. Such a condition cannot be
applied directly to a jittered image: we have first to align in some way the first two rows of the image. We propose a modification of Algorithm 1 where we first align $\gamma_1$ and $\gamma_2$ (using a criterion of the form (2)) and take for $\phi_1$ the aligned version of $\gamma_2$, along with its estimated position for $\hat{p}_0$. Algorithm 1(a) below differs from Algorithm 1 only in the step 3.

**Algorithm 1(a).**

Steps 1 and 2 are the same as in Algorithm 1. Step 3 is replaced by

3 Set $\hat{p}_1 = N + 1$ and $\phi_1 = [\theta_N, \gamma_1, \theta_N]$;
   
   (a) for any $k = 1, \ldots, 2N + 1$, calculate:
   
   i. $h_k \triangleq [\theta_{k-1}, \gamma_2, \theta_{2N-k+1}]$;
   ii. $m = \min \{k, N + 1\}$ and $n = \max \{k, N + 1\} + c - 1$;
   iii. $J(k) = \frac{1}{n - m + 1} \sum_{j=m}^{n} |h_k(j) - \phi_1(j)|^\alpha$;

(b) set $\hat{p}_0 = \arg \min_{1 \leq k \leq 2N+1} J(k)$;
(c) set $\phi_2 = [\theta_{\hat{p}_0-1}, \gamma_2, \theta_{2N-\hat{p}_0+1}]$.

Then continue with the steps 4 and 5 of **ALGORITHM 1**.

The computation cost of Algorithm 1(a) is slightly higher than Algorithm 1. This boundary is not completely satisfying since it may favor constant vertical transition between the mirror boundary and the first row.

Yet another possibility is to fix $\hat{p}_1$ as in Algorithm 1 and then to find jointly $\hat{p}_2$ and $\hat{p}_3$ by considering all possible shifts for $\gamma_2$ and $\gamma_3$. Then we choose $\hat{p}_2$ such that $J_{2,\alpha}^\prime$ is minimal for $(\hat{p}_2, \hat{p}_3)$. (This boundary condition is can be related to the forward step of the 3-line algorithm in [10].) Only step 3 of Algorithm 1 is modified, as explained below.

**Algorithm 1(b).**

Steps 1 and 2 are the same as in Algorithm 1. Step 3 is replaced by

3 Set $\hat{p}_1 = N + 1$ and $\phi_2 = [\theta_N, \gamma_1, \theta_N]$;
   
   (a) for any $k = 1, \ldots, 2N + 1$
   
   i. $h_k^0 \triangleq [\theta_{k-1}, \gamma_2, \theta_{2N-k+1}]$;
   ii. for any $\ell = 1, \ldots, 2N + 1$, calculate
   
   A. $h^\ell \triangleq [\theta_{\ell-1}, \gamma_3, \theta_{2N-\ell+1}]$;
   B. $m = \min \{k, \ell, N + 1\}$ and $n = \max \{k, \ell, N + 1\} + c - 1$;
   C. $J(k, \ell) = \frac{1}{n - m + 1} \sum_{j=m}^{n} |h^\ell(j) - 2h_k^0(j) + \phi_2(j)|^\alpha$;

iii. find $\hat{J}(k) = \min_{1 \leq \ell \leq 2N+1} J(k, \ell)$;

(b) find $\hat{p}_2 = \arg \min_{1 \leq \ell \leq 2N+1} \hat{J}(k)$;
(c) set $\phi_1 = [\theta_{\hat{p}_2-1}, \gamma_2, \theta_{2N-\hat{p}_2+1}]$.

Then continue with steps 4—for $i = 3, \ldots, r$—and 5 of **ALGORITHM 1**.

The computational cost of Algorithm 1(b) is slightly higher than Algorithms 1 and 1(a).

Even though the experimental results (in the following sections) are highly encouraging, more research is needed to find in a fast way a pertinent boundary condition.
3.2.2 Using sequences of perfectly dejittered rows

A possible way to improve a result \( \hat{f} \) obtained using Algorithm 1 (or 1(a), or 1(b))—if really needed—is to detect a large set of consecutive lines where dejittering is exact: \( \hat{d}_i = d_i \) for \( i \in \{j_0, j_0 + 1, j_0 + 2, \ldots, j_1\} \). Then one can apply the method on the already restored \( \hat{f} \) backward from row \( j_0 \) towards row 1, and forward from row \( j_1 \) to row \( r \). If \( e_\infty(\hat{d}, d) \leq 2N + 1 \), we can use \( N' \) in place of \( N \) which is the smallest integer satisfying \( N' \geq \frac{e_\infty - 1}{2} \). Refinement is much faster than dejittering. In all our experiments, we have not applied any such refinement.

3.2.3 Compound models for images involving a noticeable vertical trend

In case when the image exhibits a dominant vertical structure—see e.g. Figs. 14, 15 and 17—better results can be obtained by using a compound criterion that mixes first and second-order differences.

Algorithm 1(c).

Do Algorithm 1 (or 1(a) or 1(b)) by replacing\(^2\) step 4(a)iii by the following:

\[
\mathcal{J}(k) = \frac{1}{n - m + 1} \sum_{j=m}^{n} \left( |h_k(j) - 2\phi_1(j) + \phi_2(j)| + \beta|h_k(j) - \phi_1(j)| \right)^\alpha.
\]

Let us notice that this kind of images are not very common. Using Algorithm 1(c) needs some training to fix the additional parameter \( \beta \). Our experiments have shown that the algorithm is quite stable for a wide range of values for \( \beta \).

4 Experimental justification

4.1 Illustrations

We present and comment the experimental results on several popular (and difficult) gray-value images. For all images, the dejittering algorithms are systematically applied for \( N = M + 1 \), as specified in Remark 6 and (14). We adopt the usual notation \( \mathcal{N}(0, \sigma^2) \) to address zero-mean Gaussian noise with standard deviation \( \sigma \).

The picture of Lena on the first row in Fig. 4, of size 512 × 512, is degraded with independent jitter obtained from \( \mathcal{N}(0, \sigma^2) \) for \( \sigma = 3 \), quantized on \( \mathbb{Z} \) and constrained to \( \{-6, \ldots, 6\} \). The second row of Fig. 4 addresses the picture of Lena of size 256 × 256 which is degraded with independent uniform jitter on \( \{-6, \ldots, 6\} \). Since the width of the image is twice smaller, the jitter is twice stronger than in the first row. In both cases, we applied successively Algorithms 1 and 1(a) for \( \alpha = 1 \) and \( \alpha = 0.5 \). In all cases, we obtained perfect reconstructions where all errors are null (and PSNR = +\( \infty \)). For both experiments, it was our first trial and we kept the results since they are perfect. Naturally, there are jitters that cannot be recovered perfectly, as seen in Tables 1, 2 and 3, attached to § 4.2.

The next Fig. 5—Peppers—was degraded with strong uniform independent jitter on \( \{-10, \ldots, 10\} \) (i.e. \( M = 10 \)). For \( \alpha = 1 \), Algorithms 1, 1(a) and 1(b) yields the same result with errors (as defined in § 2.2) \( \text{MAE} = 1.3834 \), \( \text{PSNR} = 31.2692 \), \( e_1 = 0.4336 \), \( e_\infty = 26 \), \( e_0 = 23.6328\% \) and \( e_\infty^2 = 2.3483\% \). It is not displayed since visually it is indistinguishable with the result for \( \alpha = 0.5 \). For \( \alpha = 0.5 \), Algorithms 1 and 1(a) give the same result—displayed in the figure—whose errors read \( \text{MAE} = 1.3528 \), \( \text{PSNR} = 31.5146 \), \( e_1 = 0.4043 \), \( e_\infty = 23 \), \( e_0 = 23.6328\% \) and \( e_\infty^2 = 1.9569\% \). Visually, one can hardly distinguish the dejittered image from the original image. Nevertheless, the image of the error—where the central \( c - 2N \) part of the restored and the original images are matched as explained in subsection 2.2—shows a slight displacement of several pixels. For comparison, one can check the

\( \mathcal{J}(k) = \frac{1}{n - m + 1} \sum_{j=m}^{n} \left( |h_k(j) - 2\phi_1(j) + \phi_2(j)|^\alpha + \beta|h_k(j) - \phi_1(j)|^\alpha \right) \) may seem more “natural”. Experiments have shown that (15) yields slightly better results while involving less demanding computations. We do not have a mathematical explanation. Let us mention that they amount to the same criterion if \( \alpha = 1 \).
Figure 4: First row: Lena’s picture, 512 × 512. The jittering vector \( d \) corresponds to \( \mathcal{N}(0, 3^2) \), quantized and constrained to \( \{-6, \ldots, 6\} \). Second row: Lena’s picture of size 256 × 256, corrupted with a jittering vector \( d \) uniform on \( \{-6, \ldots, 6\} \). In both cases, Algorithms 1 and 1(a) for both \( \alpha = 1 \) and \( \alpha = 0.5 \) yield the same restoration \( \hat{f} \) which is perfect since \( \text{MAE} = e_1 = e_\infty = 0 \).
quality of the results obtained in [10, 6, 7] for the same image, corrupted with a comparable jitter, using slower methods, especially in the last two references.

The picture of Barbara (512 × 512) is challenging for many image processing tasks since it contains a lot of small details with a strong local geometry. In Fig. 6, the picture is contaminated with independent jitter obtained from $\mathcal{N}(0, \sigma^2)$ for $\sigma = 3$, quantized on $\mathbb{Z}$ and constrained to $\{-6, \ldots, 6\}$. Here we compare the role of $\alpha$. For $\alpha = 1$, there are wrongly estimated displacements visible on the leg of the table. For $\alpha = 0.5$, the local polynomial constraint is reinforced which fits well the nature of the image. The result for $\alpha = 0.5$ is very satisfying and undistinguishable from the original; indeed $e_\infty(\hat{d}, d) = 2$ (see Remark 5). The next Fig.7 considers the same image contaminated with independent uniform jitter on $\{-6, \ldots, 6\}$. The dejittered image is impossible to distinguish visually from the original since $e_\infty(\hat{d}, d) = 2$. Observe that the errors $\text{mae}$, $\text{psnr}$ and $e_1$ seem unfavourable. The error image $\hat{f} - f$ in Fig. 8(left) shows that the dejittering is perfect in the central rows of the image but that it is slightly wrong somewhere in the middle of the face of Barbara, as well as in the bottom part, at the level of row 430. On the zoom of the face of Barbara, it is hard to find where is the wrong displacement since it is only one pixel. For a comparison with an alternative and much slower method, one can check Fig. 7 in [7].

The first image in Fig. 9 is degraded with strong uniform jitter on $\{-10, \ldots, 10\}$ (i.e. $M = 10$). Algorithms 1 and 1(a) for both $\alpha = 1$ and $\alpha = 0.5$ lead to the same dejittering shown on the right. The latter is completely satisfying since $e_\infty(\hat{d}, d) = 2$. The original Boat image can be seen on the right side of Fig. 10 where the restorations are exact (i.e. $\hat{f}^* = f^*$, as defined in § 2.2). For comparison, one may check the dejittering of the same image obtained in [6, 7] using much slower methods.

Fig. 10 illustrates the cases of structured jitter as mentioned in the Introduction—in particular low-frequency and high-frequency sinusoidal jitter. Restoration from both the low-frequency and the high-frequency sinusoidal jitter using Algorithm 1 for $\alpha = 1$ yields the original image.

4.2 Large-scale experiment

Even if our method has a strong theoretical background, we cannot justified it a fully theoretical way. Instead, we check its stability and its performance using a very large number of experiments. These help us to make relevant choices for the boundary conditions as well as for $\alpha$.

For each one of the following popular and “difficult” gray-value images—Lena, Peppers, Barbara and Boat—we realized 1000 independent experiments as described below. These four images present completely different features: Lena involves a lot of smooth textures along with regular homogeneous parts; Peppers is composed out of nicely homogeneous regions separated by regular contours; Barbara involves textures with a strong local geometry with lots of small edges as well as nicely homogeneous regions; Boats is very geometrical without important texture.

- **Uniform jitter.**
  The components of the jittering vector $d \in \mathbb{R}^r$ are independent and uniformly distributed on the set of integers $\{-M, \ldots, M\}$ for $M = 6$ (see Fig. 11). All results relevant to uniform jitter are indicated in Tables 1, 2 and 3 given in Appendix with the letter “U”.
  Each original image was degraded using such a jittering vector $d$, according to (1). The same jittered image was then restored successively using Algorithm 1 for $\alpha = 1$ and $\alpha = 0.5$, then Algorithm 1(a) for $\alpha = 1$ and $\alpha = 0.5$ and last Algorithm 1(b) for $\alpha = 1$ and $\alpha = 0.5$.

- **Truncated Gaussian jitter.**
  The components of the jittering vector $d \in \mathbb{R}^r$ are integers belonging to the set $\{-M, \ldots, M\}$ for $M = 6$. They are obtained as described next. We generate $2r$ or $3r$ independent reals $\tilde{d}_i \sim \mathcal{N}(0, \sigma^2)$ for $\sigma = 3$. Then
Figure 5: Peppers image $512 \times 512$. The jittering vector $d$ is uniform on $\{-10, \ldots, 10\}$. Dejittering is the same with Algorithms 1 and 1(a), and is realized using $N = M + 1 = 11$ and $\alpha = 0.5$. The error measures are $\text{MAE} = 1.3528$, $\text{PSNR}=31.5146$, $e_1 = 0.4043$ and $e_\infty = 23$. 
Figure 6: Barbara 512 × 512. The jittering vector \( d \) is obtained from \( \mathcal{N}(0, \sigma^2) \), \( \sigma = 3 \), truncated and quantized on \( \{-6, \ldots, 6\} \). The error measures for \( \alpha = 1 \) are: \( \text{MAE} = 10.5672 \), \( \text{PSNR} = 21.4076 \), \( e_1 = 1.9238 \) and \( e_\infty = 5 \). The error measures for \( \alpha = 0.5 \) are considerably better: \( \text{MAE} = 4.1599 \), \( \text{PSNR} = 25.5321 \), \( e_1 = 0.5234 \) and \( e_\infty = 2 \).
Figure 7: Barbara 512 × 512. The jitter is uniform on \{-6, \ldots, 6\}. The errors for the restoration read m\text{AE} = 4.1618, p\text{SNR} = 25.5258, e_1 = 0.52344 and e_\infty = 2.

Figure 8: Error and zoom for Fig.7.
Jittered image Alg. 1 and 1(a), $\alpha = 1$ and $\alpha = 0.5$.

Figure 9: Boat image, $400 \times 512$. The jitter is uniform on $\{-10, \ldots, 10\}$. The restored image using Algorithms 1 and 1(a) is the same for $\alpha = 1$ and $\alpha = 0.5$. The errors read: $\text{MAE}=0.1587$, $\text{PSNR}=42.8749$, $e_1=0.065$ and $e_{\infty}=2$.

Figure 10: The sinusoidal jitter is quantized on $\{-6, \ldots, 6\}$. The dejittered image is perfect—all errors are null.
we select \( r \) elements that belong to \([-M - 0.5 + \varepsilon, M + 0.5 - \varepsilon]\) with \( \varepsilon = 2 \times 10^{-16} \) and approximate them to the nearest integer. Thus we get a sequence of independent samples following a quantized and truncated centered Gaussian distribution, as shown in Fig. 11. The results corresponding to this kind of jitter are indicated in Tables 1, 2 and 3 with the letter “G”.

The original image \( f \) was jittered with the so obtained vector \( d \), following (1). The same jittered image was then restored as in the previous case: Algorithm 1 for \( \alpha = 1 \) and \( \alpha = 0.5 \), then Algorithm 1(a) for \( \alpha = 1 \) and \( \alpha = 0.5 \) and last Algorithm 1(b) for \( \alpha = 1 \) and \( \alpha = 0.5 \).

For each restoration, we calculated the errors \( \text{MAE}, \epsilon_1, \epsilon_\infty, \epsilon_0 \% \) and \( \epsilon_\Delta \% \), as defined in equations (5), (7), (8), (9) and (10) respectively. (We cannot find any mean value for the PSNR error measure (6); we already observed that in some cases dejittering is exact (\( \hat{f}^s = f^s \)) which leads to \( \text{PSNR} = \infty \).) For each image, each case of jitter, each algorithm and each parameter, we thus obtained 1000 measures of these errors. Table 1 given in Appendix summarizes their means, Table 2 gives the percentage of the cases when \( \epsilon\infty \leq 2 \) and \( \epsilon\infty \leq 3 \), while Table 3 gives the variances of \( \text{MAE} \) and \( \epsilon_1 \). These results should be considered with some distance—in the body of the paper we present several cases where visually better restorations do not match the minimal best of these errors, except when \( \epsilon\infty \in [0, 2] \). Nevertheless, in most of the cases, these error measures are quite relevant. The results shown in Tables 1, 2 and 3 help to sketch a (partial) assessment on the following points: (i) if our algorithms are good enough, (ii) which one to choose among 1, 1(a) and 1(b), (iii) which value for \( \alpha \) is better.

1. Algorithm 1 gives smaller mean errors in Table 1 for almost all cases, and in all cases it is satisfactory enough. Moreover, it is the faster Algorithms 1(a) and 1(b).

2. The values for \( \epsilon_\Delta \% \) are always small which justifies the prior we adopted on magnitudes of the second-order differences in the vertical direction of real-world images, and the sequential way to treat them.

3. All tables show that \( \alpha = 0.5 \) gives better results for textured images (Lena, Barbara), or images involving lots of regular curvatures (Peppers), while \( \alpha = 1 \) is better for Boat which has a simpler geometrical structure. In all cases, \( \alpha = 1 \) leads to acceptable results, \( \alpha = 0.5 \) leads to a higher quality in most of the cases.

4. From Table 2 we see that dejittering of Lenna and Boat gives rise to high percentages for \( \epsilon\infty \leq 2 \) (between half and 3/4) which corresponds to high-quality restorations. These percentages are smaller for Barbara (10-20\%) and and Peppers (10\% or less). Globally, these results are encouraging, if we take into account that having \( \epsilon\infty > 2 \) or \( \epsilon\infty > 3 \) does not provide information about the quality of the restoration.

5. The variances of \( \text{MAE} \) and \( \epsilon_1 \) in Table 3 are small, especially for \( \alpha = 0.5 \).

6. Except for Barbara under uniform jitter in Table 1, Algorithm 1(b) seems to perform less well than the others. The simplest form—Algorithm 1—seems being the most “universal”.

Remark 7 Whenever we fix \( \alpha \), our algorithms are parameter-free. For noise-free jittered images, it is safer to fix \( \alpha = 0.5 \). If a slight compromise in terms of quality can be tolerated, algorithms are faster for \( \alpha = 1 \) while the results range between very good and quite acceptable (especially when compared with the state of the art).
5 Color images

In this section we extend Algorithm 1 in order to deal with RGB color images where all color channels incur the same jitter. Let us remind that RGB discrete images are represented by vector-valued matrices \( f \) where each pixel \( f_i(j) \) has three components, say \( f_i(j; \kappa) \) for \( 1 \leq \kappa \leq 3 \). More precisely, the jittering model (1) reads

\[
\forall j \in \{1, \ldots, c\}, \forall i \in \{1, \ldots, r\}, \quad g_i(j; \kappa) = \begin{cases} \quad f_i(j + d_i; \kappa), & \text{if } 1 \leq j + d_i \leq c, \\ \quad \text{any, e.g. } = 0, & \text{otherwise,} \\ \end{cases} \quad 1 \leq \kappa \leq 3,
\]

where \( d_i \) is as in (1).

5.1 Algorithms

The central part of the algorithm proposed below is based again on \( J_2 \) as given in (3). Since the jitter is the same for all color channels, we obtain from \( g \) a gray-value image \( \gamma \) and estimate the row displacements \( d_i = \hat{p}_i - C \) on \( \gamma \) using the ideas of Algorithm 1. The dejittered color image \( \hat{f} \) is calculated by inserting the estimated displacement vector \( \hat{d} \) into the jittered color image \( g \).

The equivalent counterpart of the \( n \)-length zero-valued row vector defined in (11) is denoted by \( \theta_{n \times 3} \): each one of its components is a 3D zero-valued vector, that is

\[
\theta_{n \times 3}(i; \kappa) = \theta_n, \quad \text{for} \quad 1 \leq \kappa \leq 3.
\]

In the algorithm below we use both \( \theta_n \) and \( \theta_{n \times 3} \).

As in Algorithm 1, before to start, we fix the following values:

- \( N > M \), e.g., \( N = M + 1 \).
- \( \alpha \in (0, 1] \), e.g. \( \alpha = 1 \) or \( \alpha = 0.5 \)—if the image involves regions with complex structure.

Algorithm 2. (Color images)

1. Put \( \hat{f}_1 = [\theta_{N \times 3}, \ g_1, \ \theta_{N \times 3}] \).
2. Split \( g \) into 3 vector-valued sub-matrices \( g = \begin{bmatrix} g_L & \gamma & g_R \end{bmatrix} \) where\(^3\) \( g_L, \ g_R \in \mathbb{R}^{r \times N} \) and \( \gamma \in \mathbb{R}^{r \times (c-2N)} \).
3. Calculate \( \gamma_1(j) = |\gamma_{i}(j; 1)| + |\gamma_{i}(j; 2)| + |\gamma_{i}(j; 3)|, \quad 1 \leq j \leq c - 2N \).
4. Put \( \hat{p}_0 = \hat{p}_1 = N + 1 \) and \( \phi_1 = \phi_2 = [\theta_N, \ \gamma_1, \ \theta_N] \).
5. For any \( i = 2, \ldots, r \), do the following:
   i. Calculate the scalar-valued row vector \( \gamma_i \) by
   \[
   \gamma_i(j) = |\gamma_{i}(j; 1)| + |\gamma_{i}(j; 2)| + |\gamma_{i}(j; 3)|, \quad 1 \leq j \leq c - 2N;
   \]
   ii. Set \( h^k = [\theta_{k-1}, \ \gamma_i, \ \theta_{2N-k+1}] \);
   iii. Find \( m = \min \{k, \hat{p}_{k-1}, \hat{p}_{k-2}\} \) and \( n = \max \{k, \hat{p}_{k-1}, \hat{p}_{k-2}\} + c - 1 \);

\(^3\)More precisely, \( \forall i \in \{1, \ldots, r\} \) we have

\[
\begin{align*}
g^L_i(j; \kappa) &= g_i(j; \kappa), \quad \forall j \in \{1, \ldots, N\}, \quad 1 \leq \kappa \leq 3; \\
\gamma_i(j; \kappa) &= g_i(j; \kappa), \quad \forall j \in \{N + 1, c - N\}, \quad 1 \leq \kappa \leq 3; \\
g^R_i(j; \kappa) &= g_i(j; \kappa), \quad \forall j \in \{r - N + 1, r\}, \quad 1 \leq \kappa \leq 3;
\end{align*}
\]
iv. Calculate \( J(k) = \frac{1}{n-m+1} \sum_{j=m}^{n} |h^k(j) - 2\phi_1(j) + \phi_2(j)|^\alpha \);

(b) Find \( \hat{p}_i = \arg \min_{1 \leq k \leq 2N+1} J(k) \);

(c) Set \( \phi_2 \leftarrow \phi_1 \) and \( \phi_1 \leftarrow h\hat{p}_i = [\theta_\hat{p}_{i-1}, \gamma_i, \theta_{2N-\hat{p}_{i+1}}] \);

(d) \( \tilde{f}_i = [\theta_\hat{p}_{i-1} \times 3; \gamma_i, \theta_{(2N-\hat{p}_{i+1}) \times 3}] \);

6. Extract \( \hat{f} \in \mathbb{R}^{r \times c} \) from \( f \in \mathbb{R}^{r \times (c+2N)} \) by eliminating 2\( N \) columns at the extreme left and right ends that contain the largest number of zeros (jitter).

The absolute value \(|.|\) in step 5(a)i is written only for clarity; in practice it is unnecessary since the pixels of the RGB components of the image \( \geq 0 \). Notice that this step can be replaced by any operation that transforms a color image into a gray-value image. A popular way to do this transform is

\[
\gamma_i(j) = \sqrt{g_i(j;1)^2 + g_i(j;2)^2 + g_i(j;3)^2}, \quad 1 \leq j \leq c - 2N.
\]

However, the computational cost of the latter is much higher than the one used in step 5(a)i. Let us mention that it is important that the obtained gray-value image has a good contrast. In all experiments we realized, the calculation proposed in step 5(a)i gave rise to satisfying results. Some refinements by the weighting the different color channels in this step could improve the results in some cases; we did not explore such issues.

**Computation times.** The computational time for a color image is naturally higher than for a gray-value image. However, in our Algorithm 3, most of the calculation is done using a gray-value version of the color image. We did some comparisons under the same conditions described at the end of Section 3. For a \( 512 \times 512 \times 3 \) image and \( N = 7 \) we got the solution in 1 sec. for \( p = 1 \) and in 1.4 second for \( p = 0.5 \).

5.2 **Refinement strategies.**

The modifications relevant to the first row, described in § 3.2.1, namely Algorithms 1(a) and 1(b), are straightforward to extend to Algorithm 2, by using the gray-value transform \( \gamma \). We skip a detailed description of the relevant algorithms. Refinement on a dejittered image, as explained in § 3.2.2, can be applied in a similar way by using the gray-value transform \( \gamma \).

The modification for images involving important vertical features, discussed in § 3.2.3 and formalized in Algorithm 1(c), are easily extended to color images:

**Algorithm 2(c).**

Do Algorithm 2 (or 2(a), or 2(b)); the only change is to replace step 5(a)iv by the following:

\[
J(k) = \frac{1}{n-m+1} \sum_{j=m}^{n} \left( |h^k(j) - 2\phi_1(j) + \phi_2(j)| + \beta |h^k(j) - \phi_1(j)| \right)^\alpha.
\]

5.3 **Illustrations**

We present and discuss the results obtained on various color images. In all cases, we apply Algorithm 2 or its modifications (e.g. Algorithm 2(c)) with \( N = M + 1 \).

The Man image in Fig. 12 incurs a uniform (quantized) jitter on \( \{-8, \ldots, 8\} \) (shown in the left upper image). Dejittering is realized using Algorithm 2 for \( \alpha = 1 \). The displacement error \( \hat{d} - d \) is shown on the left lower
The image of a cheetah in Fig. 13 is of size $707 \times 579$ and has a very textured appearance. It is degraded with independent jitter obtained from $\mathcal{N}(0, \sigma^2)$ for $\sigma = 5$, constrained to the set $\{-10, \ldots, 10\}$. The restoration is the same for $\alpha = 1$ and $\alpha = 0.5$. It is almost perfect since $e_\infty(\hat{d}, d) = 1$.

The famous Baboon image ($512 \times 512$) in Fig. 14 is degraded with quite a strong jitter corresponding to $\mathcal{N}(0, \sigma^2)$ for $\sigma = 6$, truncated and quantized on $\{-12, \ldots, 12\}$. The restoration using Algorithm 2 for $\alpha = 1$ is acceptable. The restoration for $\alpha = 0.5$ is visually more satisfying, even though all errors are worse than for $\alpha = 1$ (they are given in the figure). Taking into account the strong geometry in the vertical direction, we apply also Algorithm 2(c) for $\alpha = 0.5$ and $\beta = 2$ or $\beta = 3$. Now the visual result is really satisfying whereas the errors remain much worse than for Algorithm 2, $\alpha = 1$. In Fig. 15 we consider the same image contaminated with low-frequency
Figure 13: Cheetah, 707 × 579. The jitter is Gaussian with standard deviation 5, truncated and quantized on \{-10, \ldots, 10\}. The dejittered image is obtained using Algorithm 2 for $\alpha = 1$ or $\alpha = 0.5$. The errors read $\text{mae} = 1.554$, $\text{psnr} = 33.8577$, $e_1 = 0.16266$ and $e_\infty(d, \hat{d}) = 1$. 
Jitter

Independent jitter—$\mathcal{N}(0, 6^2)$,
truncated, quantized, $\{-12, ..., 12\}$

Algorithm 2, $\alpha = 1$

Baboon $(512 \times 512)$

MAE= 8.95, PSNR=21.32, $e_1=0.98$

Zooms—from top to bottom:

Jittered, Original, Alg. 2(c)

Algorithm 2, $\alpha = 0.5$

Alg. 2(c), $\alpha = 0.5$, $\beta = 2$

MAE= 13.67, PSNR=19.5, $e_1=1.39$

MAE= 13.12, PSNR=20.04, $e_1=1.12$

Figure 14: Baboon image restored using different dejittering schemes.
and high-frequency sinusoidal jitter, quantized on the set \{-6, \ldots, 6\}. The results obtained using Algorithm 2 are acceptable (not displayed). Better results are obtained using Algorithm 2(c) for \(\alpha = 0.5\) and \(\beta = 3\). Let us mention that these are not very sensitive to the exact value of \(\beta\).

Fig. 16, size 542 \times 410, shows a store window composed of numerous golden jewelries which are very finely engraved. It was corrupted with independent jitter which is uniform on \{-8, \ldots, 8\}. Both restorations using Algorithm 2 for \(\alpha = 1\) and \(\alpha = 0.5\) are visually very satisfying and the PSNR is high. The restoration for \(\alpha = 0.5\) is nearly perfect since \(e_\infty = 1\).

In Fig. 17 we consider the restoration of fronts and walls of Parisian buildings, degraded with low-frequency and high-frequency sinusoidal jitter. In both cases, all details—such as the balcony, the front decorations and the brick wall—are well recovered, except for the chimney stacks which are biased in the direction of the jitter. This defect is explained by the fact that the background of chimney stacks is perfectly uniform.

6 Noisy jittered images: dejitter then denoise

Our approach is to first dejitter images using the ideas of Algorithms 1 or 2, and then—at a second stage—to apply a fast, standard denoising method such as shrinkage estimation—see e.g. [2, 11]. A variety of fast denoising methods could be envisaged at the second stage.

6.1 Moderate noise

In the presence moderate of noise (SNR equal to 15-20 dB or more), taking \(\alpha < 1\) can be harmful since it favors too strongly locally polynomial constraint along the columns of the image; the latter is less characteristic in the presence of noise or other impairments. The theoretical explanation can be found in [14]. Whenever ALGORITHMS 1 AND 2 can be applied, we should choose

\[\alpha = 1.\]

Our experiments have shown that ALGORITHM 1 works very correctly under Gaussian noise 15-20 dB or above. In particular Algorithm 1(c) (resp. 2(c)) has provided better results in several cases.

6.1.1 Experiments

The Boat image on the first row in Fig. 18 is of size 512 \times 512, while on the second row its size is 256 \times 256. Both images are corrupted with zero-mean white Gaussian noise (15 dB for the first one, 20 dB for the second one) and with independent uniform jitter on \{-6, \ldots, 6\}. Since the width of the image on the second row is twice smaller, the jitter is twice stronger. The dejittering step in both cases is realized using Algorithm 1 for \(\alpha = 1\)
Figure 16: Jewelry image, 542 × 410, with independent uniform jitter on \([-8, \ldots, 8]\). The restoration using Algorithm 2 for \(\alpha = 1\) yields errors $\text{MAE} = 0.1957$, $\text{PSNR} = 40.3447$, $e_1 = 0.059$ and $e_\infty = 7$. The restoration for \(\alpha = 0.5\) is nearly perfect, $\text{MAE} = 0.1387$, $\text{PSNR} = 45.152$, $e_1 = 0.0332$ and $e_\infty = 1$. 
Figure 17: Fronts and walls image. The jitter is a quantized sinusoid in the range $[-6, 6]$. The bottom right corner shows relevant zooms.
The results are close to the noisy non-jittered images, shown on the third row. Denoising of the latter images is performed using hard-thresholding of the coefficients of the 2D Daubechies wavelet transform with 2 vanishing moments with thresholds $T = 30$ for the first row, $T = 15$ for the second one. The whole restoration is fast and easy since the shrinkage restoration we apply is almost instantaneous. The thresholds are chosen in order to obtain a good denoised image. For comparison, the original $512 \times 512$ boat image can be seen on the right side in Fig. 10 while the original $256 \times 256$ boat is on the third row in the same figure. The restored images are pretty clean. The obtained PSNR with respect to the original image are 30.8976 for the first image and 25.9035 for the second image.

The picture of Lena in Fig. 19 ($512 \times 512$) is corrupted with white zero-mean Gaussian noise (15 dB SNR) and independent uniform jitter on $\{-6, \ldots, 6\}$. This image is more sophisticated than Boat and involves an important vertical column in the background. Algorithm 1 was not efficient enough for dejittering and the result is not displayed. Instead, Algorithm 1(c) yields better results for a large set of values for $\beta$; the dejitterend image in Fig. 19 corresponds to $\alpha = 1$ and $\beta = 3$. The denoising step is performed by hard thresholding of the 2D Daubechies wavelet transform with 4 vanishing moments for $T = 30$. Compared with the original (see Fig. 4), for the restored image we have PSNR=28.7854.

The peppers in Fig. 20 are corrupted in the same way as the previous two images. Taking into account the presence of important nearly-vertical features, we dejitter the noisy image using Algorithm 1(c) for $\beta = 3$. Here again, the robustness with respect to the choice of $\beta$ is good. The denoising step is realized in the same way, for $T = 30$ again. The result can be compared with the original image in Fig. 5.

6.2 Strong noise

When the noise is strong (e.g. with a SNR less than 15 dB), we suggest a slightly different scheme which keeps a comparable computational cost. Because of the strength of the noise, the prior used in the previous sections is no longer justified. The idea is to slightly denoise the rows of the image using a fast shrinkage estimator and to replace the function $|.|^\alpha$ in step 4-a(iii) of Algorithm 1 by a better adapted edge-preserving function $\varphi$. More explanations are given after the statement of the Algorithm.

Let $W : \mathbb{R}^{1 \times n} \rightarrow \mathbb{R}^{1 \times n}$ denote a 1D wavelet transform and $W^*$ its inverse. Let us also introduce the hard thresholding operator $\tau : \mathbb{R}^{1 \times n} \rightarrow \mathbb{R}^{1 \times n}$ by

$$\tau_T(\gamma)(j) = \begin{cases} 0 & \text{if } |\gamma(j)| \leq T \\ \gamma(j) & \text{otherwise} \end{cases} \quad 1 \leq j \leq n,$$

where $T$ is a threshold. It is well known that hard thresholding is asymptotically optimal in the minimax sense if the $\gamma$ is contaminated with white Gaussian noise of standard deviation $\sigma$ and $T = \sigma \sqrt{2 \ln n}$. In equation (17), $\gamma$ is a row-vector of length $n$ and $\tau_T(\gamma)(j)$ are the components of the $n$-length row-vector $\tau_T(\gamma)$. In practice, we are far from these asymptotical conditions and asymptotically optimal thresholding is known to oversmooth—horizontal edges in our case. This can be harmful for the dejittering stage, especially when it is done row by row. For this reason we prefer an under-optimal threshold $T$.

Before to run the algorithm, several details must be fixed in advance.

- Fix $N > M$, e.g., $N = M + 1$.
- Choose a 1D wavelet transform $W$ (e.g. Daubechies wavelets).
- Fix an under-optimal threshold $T$.
- Choose an edge-preserving potential function $\varphi : \mathbb{R}_+ \rightarrow \mathbb{R}_+$, e.g.

$$\varphi(t) = |t|^\alpha \text{ or } \varphi(t) = \begin{cases} t^2/2 & \text{if } |t| \leq \alpha \\ \alpha|t| - \alpha^2/2 & \text{if } |t| > \alpha \end{cases} \quad (18)$$

and choose $\alpha > 0$.
Figure 18: Boat image: first row—512 × 512, second row—256 × 256. Both images are corrupted with white centered Gaussian noise and independent uniform jitter on \{-6, \ldots, 6\}. The images are dejittered using Algorithm 1 for \( \alpha = 1 \). The errors for the dejittered images (with respect to the noisy jitter-free images) read: first row—\( \text{MAE} = 1.9849, \text{PSNR} = 33.1282, e_1 = 0.3457 \) and \( e_\infty = 10 \); second row—\( \text{MAE} = 4.7292, \text{PSNR} = 25.6344, e_1 = 1.0273 \) and \( e_\infty = 18 \). Denoising is realized by hard-thresholding of the coefficients of the 2D Daubechies wavelet transform.
Figure 19: Lena image, 512×512, corrupted with 15 dB SNR white centered Gaussian noise and with independent, uniform jitter on \{-6, \ldots , 6\}. Dejittering is done using Algorithm 1(c) for \(\alpha = 1\) and \(\beta = 3\). The errors with respect to the noisy non-jittered image read: \(\text{MAE}=3.1898\), \(\text{PSNR}=29.4251\), \(e_1 = 0.588\) and \(e_{\infty} = 13\). Denoising: hard thresholding of the coefficients of the 2D Daubechies wavelet transform, \(T = 30\).

Figure 20: Peppers image, 512×512, corrupted with 15 dB SNR white centered Gaussian noise and with independent, uniform jitter on \{-6, \ldots , 6\}. Dejittered is done using Algorithm 1(c) for \(\beta = 3\). The errors with respect to the noisy non-jittered image read: \(\text{MAE}=5.803\), \(\text{PSNR}=27.5876\), \(e_1 = 0.728\) and \(e_{\infty} = 17\). Denoising is done by hard thresholding, \(T = 30\), of the coefficients of the 2D Daubechies wavelet transform, \(\text{PSNR}=29.3403\).
• According to the nature of the image (see § 3.2.3), choose $\beta \geq 0$.

Algorithm 3. *(Quite noisy images)*

1. Put \[
\begin{align*}
\text{RGB:} & \quad \tilde{f}_1 = [\theta_{N \times 3}, g_1, \theta_{N \times 3}], \\
\text{Gray:} & \quad \tilde{f}_1 = [\theta_{N}, g_1, \theta_{N}].
\end{align*}
\]

2. Split $g$ into 3 sub-matrices $g = [g^L : \overline{g} : g^R]$ where $g^L \in \mathbb{R}^{r \times N}$, $\overline{g} \in \mathbb{R}^{r \times (c-2N)}$ and $g^R \in \mathbb{R}^{r \times N}$.

3. \[
\begin{align*}
\text{RGB:} & \quad \gamma_1(j) = |\overline{g}_1(j; 1)| + |\overline{g}_1(j; 2)| + |\overline{g}_1(j; 3)|, 1 \leq j \leq c - 2N. \\
\text{Gray:} & \quad \tau_1 = \overline{g}_1.
\end{align*}
\]

4. Compute $\tilde{\tau}_1 = W^*(\tau_T(W\gamma_1))$.

5. Set $\tilde{p}_0 = \tilde{p}_1 = N + 1$ and $\phi_1 = \phi_2 = [\theta_{N}, \tilde{\tau}_1, \theta_{N}]$.

6. For any $i = 2, \ldots, r$ do the following:
   
   (a) for any $k = 1, \ldots, 2N + 1$
   
      o. \[
      \begin{align*}
      \text{RGB:} & \quad \gamma_i(j) = |\overline{g}_i(j; 1)| + |\overline{g}_i(j; 2)| + |\overline{g}_i(j; 3)|, 1 \leq j \leq c - 2N; \\
      \text{Gray:} & \quad \gamma_i = \overline{g}_i;
      \end{align*}
      \]

     i. $\tilde{\tau}_i = W^*(\tau_T(W\gamma_i))$;

     ii. $h^k = [\theta_{k-1}, \tilde{\tau}_i, \theta_{2N-k+1}]$;

     iii. $m = \min \{k, \tilde{p}_{i-1}, \tilde{p}_{i-2}\}$ and $n = \max \{k, \tilde{p}_{i-1}, \tilde{p}_{i-2}\} + c - 1$;

     iv. $J(k) = \frac{1}{n-m+1} \sum_{j=m}^{n} \phi \left( |h^k(j) - 2\phi_1(j) + \phi_2(j)| + \beta |h^k(j) - \phi_1(j)| \right)$;

   (b) find $\hat{p}_i = \arg \min_{1 \leq k \leq 2N+1} J(k)$;

   (c) Set $\phi_2 \leftarrow \phi_1$ and $\phi_1 \leftarrow h^\hat{p}_i = [\theta_{\tilde{p}_{i-1}}, \tilde{\tau}_i, \theta_{2N+1-\tilde{p}_i}]$;

   (d) \[
   \begin{align*}
   \text{RGB:} & \quad \tilde{f}_i = [\theta_{(\tilde{p}_{i-1}+1) \times 3}, g_2, \theta_{(2N-\tilde{p}_{i+1}) \times 3}] \\
   \text{Gray:} & \quad \tilde{f}_i = [\theta_{\tilde{p}_{i-1}}, g_2, \theta_{2N-\tilde{p}_{i+1}}].
   \end{align*}
   \]

7. Find $\tilde{f} \in \mathbb{R}^{r \times e}$ from $\tilde{f} \in \mathbb{R}^{r \times (c+2N)}$ by eliminating $2N$ columns at the extreme left and right ends that contain the largest number of zeros.

Hard-thresholding is better than other shrinkage functions since it keeps unchanged the important coefficients bearing the most important information. *We do not recommend a preliminary denoising of the entire image before the dejittering step* since this can deform the information in the vertical direction. In our implementation, we used Daubechies 1D wavelets. This stage is of critical importance and additional research is needed to exhibit more accurate and fast row denoising methods.

**Remark 8** Step 2 of Algorithm 3 supposes that the length of $\gamma_i$, $1 \leq i \leq r$ is a power of 2, in order to apply a wavelet transform. Whenever this is not the case, some arrangements are to be done. If the image width is a power of 2 (e.g. 256 or 512), then we apply the row under-denoising on the entire row and split each row into three pieces of length $N$, $c-2N$ and $N$, so that vertical matching using $J$ is done only on rows that are not affected by the jitter, according to Remark 2. Otherwise, we split $\overline{g}$ as $g = [g^L : \overline{g} : g^R]$ so that the number of columns of $\overline{g}$ is a power of 2, no larger than $c-2N$. 

29
The function $\varphi$ involved in $J$ (step 6(a)iv) must be edge-preserving. If the row-denoising in steps 4 and 6(a)i is efficient enough, $\varphi$ can be non-smooth at zero (e.g. $\varphi(t) = |t|^\alpha$ for $\alpha = 1$ or $\alpha = 0.5$). Otherwise, it is safer to use a smooth-at-zero function $\varphi$ in order to relax the polynomial constraint in the vertical direction (see [12], section 3 for the theoretical explanation). In the latter case, several choices can be done, giving quite similar results, e.g. $\varphi(t) = \sqrt{\alpha + t^2}, \alpha > 0$, or $\varphi(t) = \log(\cosh(\alpha t)), \alpha > 0$, or Huber’s function as given in the right side of (18) which is fast to compute and quadratic near the origin.

In step 6d of Algorithm 3 we do not recommend to insert 1D denoised rows of $g_i$ in place of $g_i$: the obtained dejittered image risk to contain noise whose statistics can be very different from the original Gaussian white noise and make the denoising step really tricky. Moreover, there is no gain in terms of speed.

6.2.1 Experiments

The peppers image in Fig. 21 is degraded with white, zero-mean Gaussian noise with 10 dB SNR and uniform jitter on the set $\{-6, \ldots, 6\}$. Dejittering is realized using Algorithm 3 for $N = M + 1$, $\beta = 5$, Daubechies 1D wavelet thresholding for $T = 50$ and $\varphi(t) = |t|$. The obtained dejittered image is denoised using 2D Daubechies wavelets with 4 vanishing moments, whose coefficients are hard-thresholded using $T = 60$.

The picture of Lena in Fig. 22(a) is degraded by white Gaussian noise with 10 dB SNR and uniform jitter on $\{-6, \ldots, 6\}$; the noisy jitter-free image is shown in Fig. 22(d). Dejittering in Fig. 22(b) is realized using Algorithm 3 for $N = M + 1$, $\beta = 30$ and thresholding of the 1D Daubechies wavelet coefficients with 2 vanishing moments in step 6(a)i for $T = 30$. We observed that dejittering is quasi-insensitive to the value of $\beta$, provided it is $\geq 10$. This helps mainly to dejitter the clear column on the left side of the background. Dejittering in Fig. 22(e) is obtained as in (b) with the difference that $\varphi$ is the Huber function for $\alpha = 10$. These dejittered images are very similar. Denoising in Figs. 22(c) and (f) is done by hard-thresholding of the coefficients of the curvelet transform of the dejittered image along with cycle-spinning [1]. (More precisely, we used the enhanced-denoising program in the CurveLab 2.1.2 toolbox available at http://www.curvelet.org/.) In both cases, the results are quite similar and acceptable.

The boat image in Fig. 23 is corrupted with 10 dB white zero-mean Gaussian noise and strong jitter, uniform on $\{-8, \ldots, 8\}$ (i.e. $M = 8$). The underlying image contains a lot of sloping (non-vertical) fine lines. This suggests to run Algorithm 3 for $\beta = 0$ and $\varphi(t) = |t|^\alpha$ for $\alpha = 0.5$ in step 6(a)iv; furthermore, in step 6(a)i we use hard-thresholding of the Daubechies wavelet coefficients with 8 vanishing moments for $T = 30$. The obtained dejittered image is very satisfying. Denoising is performed exactly as in the case of the picture of Lenna, Fig. 22 (c) and (f).

In the last experiment in Fig. 24 we focus on restoring the same noisy version of the boat image (zero-mean
Figure 22: Lenna image, $512 \times 512$ contaminated with 10 dB white Gaussian noise and independent jitter, uniform on $\{-6, \ldots, 6\}$. Dejittering is realized using Algorithm 3(c). Denoising is based on thresholding of the curvelet coefficients of the dejittered image.

Figure 23: Boat image, $512 \times 512$ contaminated with 10 dB white, zero-mean Gaussian noise and independent jitter, uniform on $\{-M, \ldots, M\}$ for $M = 8$. Dejittering is obtained using Algorithm 3 for $T = 50$, $|t| = |t|^{0.5}$, $\beta = 0$ and $N = M + 1$. Denoising is obtained using the enhanced-denoising program in the CurveLab 2.1.2 toolbox.
Gaussian noise with 10 dB SNR, deformed by sinusoidal jitter. We consider both low frequency and high frequency jitter whose details are given in the caption. Dejittering is realized using Daubechies wavelets with 2 vanishing moments and threshold $T = 30$. In both cases we observe that the dejittering step yields a very correct usual noisy image. Let us mention that the choice of the number of vanishing moments in the Daubechies wavelet basis plays an important role. The ultimate denoising step is performed again using the enhanced-denoising program in the CurveLab 2.1.2 toolbox. The whole restoration has quite a natural appearance and is close to the original—visible on the right side of Fig. 10.

### 6.3 General comments on dejittering of noisy images

We would like to notice several points concerning the restoration of noisy jittered images.

- The 1D row under-denoising in step THR of Algorithm 3 is of critical importance; further improvements need to optimize the choice of a frame for shrinkage estimation or even choose a different approach.

- Denoising of the dejittered image can be done by various methods. The approach of [6] suggests that PDE-based denoising might remove some remaining artifacts. Much better quality can be obtained using hybrid methods, such as [4, 3] etc., but this kind of methods need a considerable computation time.

- Overall, the proposed methods work well but they involve a numerous parameters whose role need to be clarified in order to fix them in a more pragmatic way. Nevertheless, it is much easier to fix them compared to methods such as those proposed in [15, 6, 7].
• The presence of additional impairments (especially when they involve sets of pixels) are harmful for the proposed methods. Adapted ways to deal with such situations have to be envisaged.

• Considerable improvement and simplifications can be expected if we use the correlation between consecutive images in a video sequence.

7 Conclusion and perspectives

The proposed dejittering approach is very simple, the obtained results have a remarkable quality while the algorithms are very fast, nearly real-time. The crux of the approach is (1) to minimize the magnitude of the second-order differences in the vertical direction and (2) to exclude from the displacement estimation all pixels at the left-end and the right-end of the image that can be due to the jitter.

Further improvement of the quality can be expected if our approach is inserted in a dynamical programming scheme similar to [10]. However, the speed will considerably decrease.

The natural evolution of this work is to involve it in a full video sequence restoration and to take advantage of the correlation between consecutive frames. Much better results can be expected then.

The dejittering of images corrupted with a strong noise clearly needs further improvements. Alternative ways to deal with jitter in presence of various impairments should be envisaged.

Acknowledgement

• The author would like to thank Louis Laborelli (Institut National de l’Audiovisuel—France) for his permanent and clever discussion on numerous practical questions concerning jittered video.

• The author acknowledges the support of the French Agence Nationale de la Recherche (ANR), under grant FREEDOM (ANR07-JCJC-0048-01), "Films, REstauration Et DOmées Manquantes".
References


### Appendix - Tables

<table>
<thead>
<tr>
<th>Alg.</th>
<th>α</th>
<th>Mean</th>
<th>Lena</th>
<th>Peppers</th>
<th>Barbara</th>
<th>Boat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (a)</td>
<td>1</td>
<td>MAE</td>
<td>0.0910</td>
<td>0.3326</td>
<td>0.571</td>
<td>1.0667</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_1$</td>
<td>0.0412</td>
<td>0.1115</td>
<td>0.1942</td>
<td>0.3430</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_\infty$</td>
<td>2.465</td>
<td>1.749</td>
<td>10.018</td>
<td>10.196</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$ %</td>
<td>2.0596</td>
<td>6.4406</td>
<td>8.8374</td>
<td>14.712</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0^\Delta$ %</td>
<td>0.3069</td>
<td>0.2971</td>
<td>1.9884</td>
<td>2.0714</td>
</tr>
<tr>
<td>1 (a)</td>
<td>1</td>
<td>MAE</td>
<td>0.0851</td>
<td>0.3253</td>
<td>0.5594</td>
<td>1.0035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_1$</td>
<td>0.0388</td>
<td>0.1068</td>
<td>0.1804</td>
<td>0.3027</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_\infty$</td>
<td>2.423</td>
<td>1.682</td>
<td>8.563</td>
<td>8.046</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$ %</td>
<td>1.9656</td>
<td>6.4055</td>
<td>8.833</td>
<td>14.355</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0^\Delta$ %</td>
<td>0.2941</td>
<td>0.2810</td>
<td>1.672</td>
<td>1.6781</td>
</tr>
<tr>
<td>1 (b)</td>
<td>1</td>
<td>MAE</td>
<td>0.1206</td>
<td>0.4594</td>
<td>0.5676</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_1$</td>
<td>0.0653</td>
<td>0.1742</td>
<td>0.1845</td>
<td>0.3107</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_\infty$</td>
<td>2.423</td>
<td>1.682</td>
<td>8.338</td>
<td>7.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$ %</td>
<td>1.9656</td>
<td>6.4055</td>
<td>8.839</td>
<td>14.756</td>
</tr>
<tr>
<td>1 (b)</td>
<td>1</td>
<td>MAE</td>
<td>0.1196</td>
<td>0.4819</td>
<td>0.5869</td>
<td>1.0987</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_1$</td>
<td>0.0644</td>
<td>0.1750</td>
<td>0.2048</td>
<td>0.3529</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$ %</td>
<td>0.7575</td>
<td>0.9336</td>
<td>1.7411</td>
<td>1.8446</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0^\Delta$ %</td>
<td>0.7812</td>
<td>0.9859</td>
<td>2.0129</td>
<td>2.1076</td>
</tr>
</tbody>
</table>

Table 1: Means of the errors ($\text{MAE}$, $e_1$, $e_\infty$, $e_0$%, $e_0^\Delta$%) based on 1000 experiments (see § 4.2). The letter “U” reminds that the jitter is uniform on $\{-6, \ldots, 6\}$ while “G”—that it is truncated Gaussian, quantized on the same set. The best performance in terms of mean($\text{MAE}$) and mean($e_1$), for each image and each case of jitter is in bold. Based on Remark 5, the cases when mean($e_\infty$) $\leq$ 2 are in bold as well.
Table 2: Percentage of the cases when $e_\infty \leq 2$ and $e_\infty \leq 3$ based on the same set of 1000 experiments of Table 1. The values for $e_\infty \leq 3$ are in italic to facilitate the reading. The best percentages for $e_\infty \leq 2$ are in bold while the best ones for $e_\infty \leq 3$ are in serif. For the interpretation of these result, reminding Remark 5 is useful.

<table>
<thead>
<tr>
<th>Alg.</th>
<th>$\alpha$</th>
<th>$e_\infty \leq 2$</th>
<th>$e_\infty \leq 3$</th>
<th>Lena</th>
<th>Peppers</th>
<th>Barbara</th>
<th>Boat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>52.8</td>
<td>52.8</td>
<td>73.5</td>
<td>71.4</td>
<td>1.7</td>
<td>1.4</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>52.8</td>
<td>68.2</td>
<td>84.1</td>
<td>3.5</td>
<td>11.4</td>
<td>28.2</td>
</tr>
<tr>
<td>1(a)</td>
<td>1</td>
<td>74.8</td>
<td>52.8</td>
<td>73.5</td>
<td>68.2</td>
<td>10.5</td>
<td>11.5</td>
</tr>
<tr>
<td>1(a)</td>
<td>1/2</td>
<td>69.8</td>
<td>68.2</td>
<td>84.1</td>
<td>10.5</td>
<td>11.5</td>
<td>28.2</td>
</tr>
<tr>
<td>1(b)</td>
<td>1</td>
<td>74.8</td>
<td>52.8</td>
<td>73.5</td>
<td>68.2</td>
<td>10.5</td>
<td>11.5</td>
</tr>
<tr>
<td>1(b)</td>
<td>1/2</td>
<td>69.8</td>
<td>68.2</td>
<td>84.1</td>
<td>10.5</td>
<td>11.5</td>
<td>28.2</td>
</tr>
</tbody>
</table>

Table 3: Variances of the errors $\text{MAE}$ and $e_1$, based on the same set of 1000 experiments of Table 1.

<table>
<thead>
<tr>
<th>Alg.</th>
<th>$\alpha$</th>
<th>Var</th>
<th>Lena</th>
<th>Peppers</th>
<th>Barbara</th>
<th>Boat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>MAE</td>
<td>0.0148</td>
<td>0.0028</td>
<td>0.0018</td>
<td>0.0092</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$</td>
<td>0.3542</td>
<td>0.0296</td>
<td>0.0286</td>
<td>0.3155</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>MAE</td>
<td>0.0108</td>
<td>0.0022</td>
<td>0.0006</td>
<td>0.0092</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$</td>
<td>0.3527</td>
<td>0.0286</td>
<td>0.0291</td>
<td>0.3155</td>
</tr>
<tr>
<td>1(a)</td>
<td>1</td>
<td>MAE</td>
<td>0.0148</td>
<td>0.0028</td>
<td>0.0018</td>
<td>0.0092</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$</td>
<td>0.3542</td>
<td>0.0296</td>
<td>0.0286</td>
<td>0.3155</td>
</tr>
<tr>
<td>1(a)</td>
<td>1/2</td>
<td>MAE</td>
<td>0.0108</td>
<td>0.0022</td>
<td>0.0006</td>
<td>0.0092</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$</td>
<td>0.3527</td>
<td>0.0286</td>
<td>0.0291</td>
<td>0.3155</td>
</tr>
<tr>
<td>1(b)</td>
<td>1</td>
<td>MAE</td>
<td>0.0105</td>
<td>0.0025</td>
<td>0.0018</td>
<td>0.0092</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$</td>
<td>0.2832</td>
<td>0.0180</td>
<td>0.0217</td>
<td>0.0456</td>
</tr>
<tr>
<td>1(b)</td>
<td>1/2</td>
<td>MAE</td>
<td>0.0092</td>
<td>0.0025</td>
<td>0.0018</td>
<td>0.0092</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$e_0$</td>
<td>0.3125</td>
<td>0.0180</td>
<td>0.0217</td>
<td>0.0456</td>
</tr>
</tbody>
</table>