A Novel De-Interlacing Method Based on Locally-Adaptive Nonlocal-Means

Roozbeh Dehghannasiri and Shahram Shirani
Department of Electrical and Computer Engineering, McMaster University, 1280 Main Street West, Hamilton, Canada L8S 4K1, dehghr@mcmaster.ca, shirani@mcmaster.ca

Abstract—This paper presents an efficient method for video de-interlacing based on Nonlocal-means (NL-means). In the proposed scheme, every interpolated pixel is set to a weighted average of its neighboring pixels. Weights of the pixels are calculated according to the radiometric distance between the surrounding areas of them and the pixel being interrogated. To calculate the weights, we need an estimate of the progressive frames. Therefore, we initially de-interlace frames using a simple and reliable edge-based de-interlacing method. We use steering kernel in NL-means to adapt it locally to the features of the image. Experimental results show the effectiveness of our method.

Index Terms—De-interlacing, nonlocal-means, image interpolation, kernel

I. INTRODUCTION

In the interlaced format, the even and odd fields of the frame are transmitted alternatively. Interlacing can halve the required bandwidth, but because of the visual quality, most of the devices support progressive format. By using de-interlacing, we can convert an interlaced frame to a progressive frame by doubling the number of lines per frame. The objective in de-interlacing is achieving the best image quality with the lowest computational cost. During recent years, many de-interlacing methods have been introduced. We can divide these methods into spatial and motion-compensated de-interlacing methods.

Spatial de-interlacing techniques use the correlation between vertically neighboring pixels in a field. We can use these methods where we have motion or low vertical details. Line repetition, line averaging and edge-based line average (ELA) [1] are some examples of simple spatial schemes. Also, newEDI [2] is an edge dependant de-interlacing method based on a horizontal pattern. The main benefit of these methods is their low implementation cost but their performance decreases when frame has high vertical details.

Motion-compensated (MC) de-interlacing methods interpolate missing pixels along the motion trajectory. Although motion-compensated de-interlacing schemes are computationally expensive, they can result in higher quality outputs compared to the other de-interlacing methods especially in high motion regions. Many motion-compensated de-interlacing algorithms have been proposed. In [3], a de-interlacing method which uses both directional interpolation and motion compensation has been proposed. In [4], a de-interlacing method which is both motion-adaptive and motion-compensated has been introduced. Another motion-compensated de-interlacing is 4FLMC [5] which is an adaptive 4-field global/local motion-compensated de-interlacing method. The performance in these methods is highly dependant on the accuracy of the motion estimation. Motion vectors (MV) used in de-interlacing have to be true motion vectors which means that they must have low estimation errors. However, finding true motion vectors is very difficult specially in the cases such as de-interlacing that we do not have the whole frame to estimate motion with high accuracy.

In this paper, we have proposed a de-interlacing method based on Nonlocal-means (NL-means). In NL-means, every missing pixel is set to a weighted linear combination of pixels in a search region. The pixels that we use in the NL-means to interpolate the missing pixel must have their original values. Therefore, we use half of the pixels in the search region to estimate the value of the missing pixel. We calculate the weight of each pixel in the search region according to the radiometric distance between the surrounding area of it and that of the pixel being interpolated. We refer to these surrounding areas as similarity windows. In order to make similarity window adaptive to the local information of the image, we use a steering kernel. To compute the distance between similarity windows, we need an initial estimate of the progressive frame before applying NL-means. Therefore, we perform a simple de-interlacing to find an estimate of the progressive frame. We use the frames obtained from this initial de-interlacing to calculate the weights of the pixels. The main advantage of using NL-means for de-interlacing is that no explicit motion compensation is needed. This is very interesting for de-interlacing because we will not need to obtain true motion vectors to have a good de-interlaced frame.

The remainder of this paper is organized as follows. Section II describes the NL-means algorithm. Section III introduces the proposed NL-means based de-interlacing algorithm in detail. In section IV, we present the experimental results and we compare the results of this algorithm with other algorithms. Finally, we conclude this paper in section V.

II. NL-MEANS FILTER

NL-means [6] and Bilateral filtering [7] are two image and video reconstruction methods that are very similar to each other. In the Bilateral filtering, the output of the filter depends on the values of the pixels in a small search region around it:

$$f(n) = \sum_{m \in T} w(m,n)c(f(m), f(n))f(m) \tag{1}$$
where \( \mathbf{n} \) and \( \mathbf{m} \) are pixel positions, \( \hat{f}(\mathbf{n}) \) is the pixel estimate, \( w(\mathbf{m}, \mathbf{n}) \) is the weight that depends on the geometric distance, \( c(g(\mathbf{m}), g(\mathbf{n})) \) is a distance function that quantifies the intensity closeness, and \( f(\mathbf{m}) \) is a pixel in the search region \( T \).

The NL-means algorithm improves the Bilateral filtering by considering the similarity between blocks (neighborhoods of the pixels) instead of single pixels. In fact, the proximity between pixels \( \mathbf{n} \) and \( \mathbf{m} \) is based on the proximity between intensities of two square neighborhoods around these two pixels. We call these blocks similarity window. The intuition behind NL-means is that natural images contain a significant amount of repeating structures (self-similarity). It is beneficial to use these self-similarities to estimate the value of pixels. The basic equation of the NL-means filter is given by:

\[
\hat{f}(\mathbf{n}) = \sum_{\mathbf{m} \in T} w(c(\mathbf{N}_m, \mathbf{N}_n)) f(\mathbf{m})
\]

(2)

where \( \mathbf{N}_m \) and \( \mathbf{N}_n \) are two \( L \times L \) similarity windows around pixels \( \mathbf{m} \) and \( \mathbf{n} \). \( c(\mathbf{N}_m, \mathbf{N}_n) \) is the function calculating distance between similarity windows, and \( w(c(\mathbf{N}_m, \mathbf{N}_n)) \) is the weight assigned to the pixel \( \mathbf{m} \) in the search region \( T \). The search region can be extended to the other frames.

We use kernel-based Euclidean distance for measuring intensity closeness between similarity windows. Steering kernel [8] is found to be very useful in the distance calculation. In the similarity window, we want to assign higher weights to the pixels that are near to the center of the window. We can use a Gaussian kernel for this purpose. Moreover, we use an elongated, rotated and scaled Gaussian kernel instead of a simple symmetric one. The intuition is that if a pixel in similarity window is located on an edge or near to an edge, it must have greater influence on the distance than other pixels. Also, by using steering kernel, we can change the size of similarity window implicitly and make it smaller for textured regions and larger for smooth regions. To this end, we have to find the dominant orientation angle \( \theta \), the elongation parameter \( \sigma \), and the scaling parameter \( \gamma \) for the similarity window corresponding to the pixel that is being estimated. To find these parameters, we have to compute the local gradient matrix for the similarity window of the pixel being interpolated. If \( d_x(.) \) and \( d_y(.) \) are the first derivatives along \( x \) and \( y \) directions, then \( \mathbf{G} \) (local gradient matrix) is a \( L^2 \times 2 \) matrix and have a form like this:

\[
\mathbf{G} = \begin{bmatrix}
  d_x(i, j) & d_y(i, j) \\
  \vdots & \vdots \\
  d_x(i, j) & d_y(i, j)
\end{bmatrix} = \mathbf{USV}^T, \quad (i, j) \in \mathbf{N}_n
\]

(3)

where \( \mathbf{N}_n \) is the similarity window around the pixel being interpolated and \( \mathbf{USV}^T \) is the singular value decomposition of \( \mathbf{G} \). \( \mathbf{U} \) is a \( L^2 \times L^2 \) orthogonal matrix, \( \mathbf{S} \) is a \( L^2 \times 2 \) rectangular diagonal matrix, and \( \mathbf{V} \) is a \( 2 \times 2 \) orthogonal matrix. If the second column of \( \mathbf{V} \) is \( \mathbf{v}_2 = [v_1, v_2]^T \), then we can find the dominant orientation angle \( \theta \) using the following formula:

\[
\theta = \arctan\left(\frac{v_1}{v_2}\right)
\]

(4)

The elongation parameter \( \sigma \) can be computed according to the energy of the dominant gradient direction as follows:

\[
\sigma = \frac{s_1 + 1}{s_2 + 1}
\]

(5)

where \( s_1 \) and \( s_2 \) are diagonal elements of \( \mathbf{S} \). We define the scaling factor as follows:

\[
\gamma = \frac{l^2}{\sqrt{s_1s_2 + 0.01}}
\]

(6)

where the size of the similarity window is \( L = 2 \times l + 1 \). The intuition behind this equation is that we want the scaling factor to be high when both singular values are small (smooth region), be small when both singular values are large(textured region) and has a medium value when one of the singular values is high and the other is small (sharp edge). Then we find the Gaussian kernel matrix using the following equations:

\[
\mathbf{K} = k(i, j)
\]

(7)

\[
k(i, j) = \exp \left( -\frac{(i - f)^2}{2 \gamma^2} + \left(\frac{j - f}{\gamma}\right)^2 \right)
\]

(8)

\[
i' = (i - f - 1) \cos \theta + (j - f - 1) \sin \theta
\]

(9)

\[
j' = (j - f - 1) \cos \theta - (i - f - 1) \sin \theta.
\]

(10)

Some examples of resulting kernels for different image patches are shown in Fig. 1. After finding the kernel, we use it in the distance function:

\[
c(\mathbf{N}_m, \mathbf{N}_n) = \sum_{i=1}^{L} \sum_{j=1}^{L} k(i, j)(\mathbf{N}_m(i, j) - \mathbf{N}_n(i, j))^2
\]

(11)

where \( \mathbf{N}_m(i, j) \) is the pixel at row \( i \) and column \( j \) in \( \mathbf{N}_m \). The weights of the pixels in the NL-means equation can be computed using following equation:

\[
w(c(\mathbf{N}_m, \mathbf{N}_n)) = \frac{1}{Z(\mathbf{n})} e^{-\frac{c(\mathbf{N}_m, \mathbf{N}_n)}{\gamma}}.
\]

(12)

Fig. 1: Different image patches and their corresponding kernel matrices. (a) Textured region. (b) Smooth region. (c) Vertical edge. (d) Horizontal edge. (e) -45 Degree. (f) 45 Degree.
In the above formulation $Z(n)$ is the normalizing constant and defined as:

$$Z(n) = \sum_{m \in T} e^{-\frac{(N_{m_n} N_{m_b})}{h}}. \quad (13)$$

The parameter $h$ is the degree of filtering and sets the decay rate of the exponential function.

### III. The Proposed Method

In order to compute the distance between neighborhoods, we need to have an estimation of the progressive frame. For this purpose, we perform an initial de-interlacing step. For every interlaced pixel, we consider five spatial and three temporal directions as depicted in Fig. 3. Direction 1 is the vertical direction, directions 2 and 3 are near-vertical spatial directions, directions 4 and 5 are near-horizontal spatial directions, and directions 6, 7, and 8 are temporal directions.

![Fig. 3: The eight directions that we consider for the initial de-interlacing.](image)

First, we choose the minimum variation direction (the direction whose pixels difference is the smallest), then we look at pixels above and below the pixel being interpolated and assume that we want to de-interlace them and find the best direction (the direction whose pixels’ average is closest to the value of the pixel which is assumed to be interlaced) for them. In order to be able to estimate the best direction for the above and below pixels we perform a line averaging on the current and next frames. Based on the best direction for the above and below pixels and the minimum variation direction, we can estimate the best direction for the de-interlaced pixel using the flowchart in Fig. 2. The intuition behind looking at the above and below pixels is that the minimum variation direction is not the best direction in many cases. For example, when we have a narrow vertical edge on a completely black background, the result of the minimum variation direction is a black pixel but we know that the pixel is not black therefore we do not choose the minimum variation direction. How can we know that the pixel is not black? By looking at the above and below pixels. We have a tougher condition to choose a near-horizontal spatial direction because it is very unlikely that, for a de-interlaced pixel, a near-horizontal direction be the best interpolation direction. The interlaced pixel is set to the average value along the final estimated direction. We perform this initial de-interlacing for the current and the next frames.

Then we apply the NL-means filter to the resulting frame from the initial step. We consider a 3-dimensional search region consisting of the previous, current, and following frames.

![Fig. 2: Flowchart for the initial de-interlacing.](image)
In Fig. 4, we have shown an example for the search region and similarity window. As you can see, while we use all pixels in the similarity window, we only use original ones in the search region. The interesting point about our method is that it can implicitly adapt itself to the motion. In case of motion, the part of the search region in the current frame can be very useful, while in the vertically detailed areas the search region in the previous and following frames can be very effective.

In order to prevent blurring, for every pixel being interpolated, we ignore the pixels whose similarity window distance is very large compared to the smallest one. In other words, for every pixel being interpolated, first we find all distances using (11) and then we ignore those pixels which have large distance compared to the smallest distance. Then we compute the weights \( w(c(N_m, N_n)) \) for the remaining pixels based on (12) and (13) and find the estimation of the de-interlaced pixel by using (2).

### IV. SIMULATION RESULTS

In our simulations, we used eight 352 × 288 video test sequences in the CIF format: Mother, Foreman, Stefan, Flower, Hall, Silent, Coastguard, and Akiyo. For the search region, based on our experiments a 9 × 9 search area in the previous, current, and following frames yields suitable results, while it maintains the low computational cost condition. We also set the parameter \( h \) to 10. We found that two times the smallest distance is a suitable threshold for considering distances to compute the weights. Finally, for the similarity window we obtained the best results when it was 21 × 21.

In Table 1, we compared the PSNR results of our proposed method with seven other methods. We have also presented the results of the proposed ELA method that we used as initial de-interlacing to show its improvement with respect to the conventional ELA [1]. This table shows the better performance of our proposed method compared to the other methods. Interestingly, our results are even better than motion-compensated methods such as 4FLMC [5] and MOMA [4], while it does not perform any kind of explicit motion compensation.

In Fig. 5, the visual results of the different de-interlacing methods for the Stefan sequence have been presented. We have compared the visual result of our proposed method with four other algorithms: LA, ELA, MOMA, and 4FLMC de-interlacing methods. Stefan is a very complex sequence.
Fig. 5: Visual evaluation of the different de-interlacing methods for the 12th frame of the Stefan sequence. (a) Original frame. (b) De-interlaced by the line averaging (LA) method. (c) De-interlaced by the edge-based line averaging (ELA) method. (d) De-interlaced by the MOMA algorithm [4]. (e) De-interlaced by the 4FLMC method in [5]. (f) De-interlaced by our proposed method.

Many artifacts are visible in the frames reconstructed by other methods. However, these artifacts have been reduced when we use our proposed NL-means based de-interlacing method. The main reason of these improvements is using the spatial and temporal information simultaneously and considering the neighborhood around pixels in computing the weights of the pixels to interpolate the missing pixels.

Fig. 6 shows the original and the de-interlaced frames obtained by the LA, ELA, MNN [9], and our proposed NL-means based method for the Silent sequence. It is clearly evident that the performance of our proposed method is dramatically better compared to the other methods.

V. Conclusion

In this paper, we proposed a new de-interlacing method based on NL-means. The main difference of this method is that it uses the similarity between neighborhoods instead of pixels. To perform NL-means, we first apply an initial de-interlacing step to find an estimate of the progressive frame. The interesting point of the proposed method is that it does not require explicit motion estimation. The experiments show that our method outperforms other methods both subjectively and objectively.

REFERENCES


