Iterative based image and video denoising by fractional block matching and transform domain filtering

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

21-01-2021 / 25-01-2021

CITATION

Mehdizadeh Hemat Abadi, Amir; Hosseiny Fatemi, Mohammad Reza (2021): Iterative based image and video denoising by fractional block matching and transform domain filtering. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.13623953.v1>

DOI

[10.36227/techrxiv.13623953.v1](https://doi.org/10.36227/techrxiv.13623953.v1)

Iterative based image and video denoising by fractional block matching and transform domain filtering

Amir Mehdizadeh Hemat Abadi^a and Mohammad Reza Hosseiny Fatemi^{b, c}

^aDepartment of Electrical Engineering, Faculty of Engineering, Islamic Azad University of Mashhad, Mashhad, Iran

^bDepartment of Electrical and Biomedical Engineering, Faculty of Engineering, International Imam Reza University, Mashhad, Iran

^cTechnical Support Department, Islamic Republic of Iran Broadcasting, Mashhad, Iran

Emails: amir.mehdizadeh@mshdiau.ac.ir, mrh.fatemi@imamreza.ac.ir

Abstract

This paper presents an iterative algorithm for image and video denoising which is based on fractional block-matching and transform domain filtering. We propose fractional motion estimation technique to find the most accurate similar blocks for each block of an image which improves sparsity enabling effective image denoising. By taking the advantage of blocks similarity and wavelet transform domain filtering along with weighted average function (WAF) in an iterative based manner, we achieve a higher level of sparsity and a better exploiting of blocks similarity redundancies of noisy images that increase the chance of preserving details and edges in the restored image. Since our algorithm is iterative, we can tradeoff between image denoising degree and computational complexity. In addition, we develop a video denoising algorithm based on the proposed image denoising algorithm. The simulation results of images and videos contaminated by additive white Gaussian noise demonstrate that our algorithm substantially achieves better denoising performance compared with previously published algorithms in terms of subjective and objective measures.

I. Introduction

Nowadays, image and video have many applications in our life such as medical imaging systems, video conference, multimedia entertainment, camera surveillance and etc. However, image and video that are captured by imaging systems are corrupted by noise. As a result, efficient image and video denoising algorithms are required to compensate such data corruption. Over the past 6 decades, numerous research work have investigated image denoising problem from many different points of view that resulted in several significant noise removal algorithms. Though, image and video denoising still is a major challenge and an important job because noise removal leads to blurring and visual quality degradation.

In general, noise reduction methods can be classified into two broad categories: spatial filtering and transform domain filtering. Spatial filtering refers to the approaches that change the value of each image pixel by a function of its neighborhood pixels. Spatial domain methods can be further divided into linear filters (e.g. Mean and Wiener filters) and nonlinear filters such as median, weighted median [1], rank [2] and non-local means (NLM) [3] filters. Generally, nonlinear filters have better performance compared with linear filters at the price of higher complexity.

In transform domain methods, the noisy image is first transferred into intended domain and its transform coefficients are obtained. Then, high-amplitude coefficients that include most of

signal energy are kept. Next step is removing of remained coefficients which are originated from noise. Finally, the denoised image is obtained from the reverse transform of high-amplitude coefficients.

Transformed domain image denoising algorithms can be categorized based on basis functions. The basics functions can be further divided into data adaptive methods (such as ICA [4], PCA [5, 6] and K-SVD [7] algorithms) and non-adaptive approaches like frequency domain and wavelet algorithms [8]. Compared with non-adaptive methods, adaptive algorithms require much more computational complexity since they need to noiseless training samples.

In 2007, K. Dabov et al. introduced blocks matching and 3-D filtering (BM3D) algorithm [9]. By combining the spatial domain and transform domain methods, they increased the efficiency of image denoising process and proposed a new method for image and video denoising. To aim this, group of similar patches in spatial domain (i.e. block matching process) and collaborative filtering as a transform domain method are combined together leading to using of similar patches in filtering process of other patches. Based on the similar idea of Dabov et al., later algorithms such as MSK-SVD [10], LPG-PCA [11], NLM [12] and enhanced ICA [13] were proposed which used both spatial domain filtering and transform domain filtering. However, their higher image denoising efficiency comes at price of higher computational cost.

In this paper, we introduce iterative based fractional block matching transform domain filtering (IFBM) algorithm for image and video denoising which outperforms previous state of art image denoising works. In the proposed algorithm and in the first iteration, each image is divided to $N \times N$ blocks and for each block (i.e. current block) fractional block matching is carried out. This process determines several similar blocks for each current block. Then, wavelet domain filtering is done for the current block which includes forward discrete wavelet transform, soft threshold filter and inverse discrete wavelet transform. The result of this step is a denoised current block which is used for replacement of the similar blocks. The above described procedures are done for all current blocks. Then, WAF is used to compute the weighted average of several similar denoised blocks which increases the preserving the image details and edges. To improve the denoising performance the second and afterward iterations are carried out in the similar way except that we use Wiener filter instead of soft threshold filter. However, higher denoising performance comes at the price of more iterations and computational cost. The simulation results reveal that even with two iterations, the IFBM algorithm leads to better denoising results compare with the previous published works including the BM3D, K-SVD and LPG-PCA algorithms. The key process which results in a better denoising performance is fractional block matching which highly increases the block matching accuracy of similar blocks. As discussed in [14], natural images contain regions which are similar to other regions at the same scale/orientation and across scales/orientations. It has shown that there exists significant self-similarity between square blocks of fixed size extracted from a single image which this term is closely related to nonlocal image modeling. So with no surprise, fractional motion estimation with higher accuracy can achieve much better denoising performance in comparison with integer block matching motion estimation.

The rest of this paper is as follows. In Section II, the details of our algorithm and the proposed techniques are illustrated. Section III describes the extension of the proposed algorithm for color images and video. Computational cost analysis, experimental results and performance comparison with the state of art algorithms are presented in Section IV. Finally, we conclude this paper in Section V.

II. Details of the proposed algorithm

This section describes the details of our algorithm. The IFBM algorithm carries out noise removal operation in similar successive stages based on an iterative method, as shown in Figure 1. The performance of our algorithm depends on the number of iterations. The higher the number of iterations is, the better the quality of the obtained denoised image is. This means that we can tradeoff between the computational complexity and the quality of the denoised image. The most parts of first and the other iterations steps are similar. As a result, we first give the details of the building blocks of the first iteration step. Then, we briefly illustrate the next iteration steps with more emphasis on the new blocks. The main blocks of each stage are fractional block-matching process with quarter-pixel accuracy, wavelet filtering (which includes forward 2D discrete wavelet transform, threshold filter, inverse 2D discrete wavelet transform), WAF, estimating of the denoised image and block-matching process.

Figure 1: Block diagram of the IFBM algorithm

a) First iteration step:

• Fractional block-matching process with quarter pixel accuracy

Block-matching motion estimation algorithm with integer-pixel accuracy is widely used for image and video restoration and video compression. In general, each block of size $N \times N$ (i.e. 8×8) is matched against all reference blocks within a search area with a range of $[-w, w-1]$, as shown in Figure 2.

Figure 2: Block-matching process

The matching criterion is Sum of Absolute Differences (SAD), as given in (1).

$$SAD(i, j) = \sum_{m=1}^N \sum_{n=1}^N |C(m, n) - R(m + i, n + j)| \quad ; \quad -w \leq i \& j < w. \quad (1)$$

In (1), (i, j) is the current position (motion vector), $C(m, n)$ is a pixel value of the current block, and $R(m+i, n+j)$ refers to a pixel value of the reference block in the search area. Please note that the search area in our algorithm is equal to the size of a still image. When the size of the still image is 256×256 pixels, it contains of 32 non-overlap blocks with a size of 8×8 pixels. For a still image with a size of 512×512 pixels, the number of 8×8 non-overlap blocks is equal to 64. In order to decrease the computational cost, the block matching is carried out in raster order and only for 32 non-overlap blocks. After matching the current block against all non-overlap blocks, the positions of non-overlap blocks that their SAD values are lower than a block matching threshold (i.e. λ_{BM}), are selected as the best integer matched blocks. The threshold value for each current block is computed as follows.

As shown in Figure 3, we first divide an 8×8 current block into two 8×4 sub blocks and calculate the SAD of them (i.e. SAD1). Then, we divide that current block into two 4×8 sub blocks and compute their SAD (i.e. SAD2). The threshold value is equal to the average of

SAD1 and SAD2, as given in (2). Next current block is the previous current block which is shifted n pixels (i.e. $n=4$) to the right and the same process is repeated for it.

$$\lambda_{BM} = \text{average}(\text{SAD}_{8 \times 4}, \text{SAD}_{4 \times 8}) \quad (2)$$

To improve the performance of integer-pixel block-matching algorithm, recent video coding standards such as H.264 [15] and H.265 [16] use fractional block-matching at price of more computational cost. To the best of our knowledge, block-matching with fractional accuracy has not been used for image and video denoising. Therefore, we use fractional block-matching algorithm to improve the accuracy of block similarity which significantly increases the overall performance of image denoising, as will be shown in Section IV.

Figure 3: Illustration of different partitions of an 8x8 current blocks for calculation the threshold of block matching

After integer-pixel block-matching, first sub-pixel values are calculated using bilinear or bicubic interpolation [17]. Then, fractional motion estimation is carried out for all the best integer matched blocks, which is usually conducted in two steps, as shown in Figure 4 [18]. In the first step, half-pixel refinement is performed by searching eight half-pixel positions around the best integer matches. This means that the search area in this step is equal to $[-0.5, +0.5]$. Then, the quarter-pixel refinement is carried out in the same manner around the best half-pixel candidates with a search area of $[-0.25, +0.25]$. The result of quarter-pixel refinement determines the most similar block to the current block. The current block and its associated motion vectors of similar blocks are categorized and hold to use in the next steps.

Figure 4: Fractional motion estimation

It is worth to mention that there are many integer and sub-pixel block-matching algorithms in the literature [19], [20]. Among them all, the full search algorithm results in the best visual performance. However, full search algorithm needs higher computational requirement than other algorithms (i.e. fast algorithms).

• Wavelet filtering

After fractional block-matching, wavelet filtering is carried out. The main idea in this stage is to remove the noise of the image while preserving the perceptual fidelity of the visual information. Wavelet filtering conducted in three steps for each current block of noisy image including forward discrete wavelet transform, linear/nonlinear filtering and inverse discrete wavelet transform which are described here.

• Wavelet Transform

Wavelet transform is widely used for image and video denoising processing [21] because of its multiresolution nature and computational efficiency. Wavelet transform decomposes a signal into a set of basis functions. Wavelets are derived from a single prototype wavelet $\psi(t)$ called mother wavelet by scaling and translation as given in (3):

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right) \quad (3)$$

Here, s denotes the scale factor; τ is the translation factor and $\sqrt{s}$ refers to the energy normalization across the different scales. The presented continuous wavelets in (3) are very redundant. Discrete wavelets can solve this problem. Although they are not continuously scalable and translatable, they can only be scaled and translated in discrete steps hence piecewise continuous [22]. The discrete wavelet at level ' l ' is given in (4) and can be extended easily to 2D case for image processing.

$$\psi(2^j t) = \sum_k g_{l+1}(k) \varphi(2^{l+1} t - k) t \quad (4)$$

where $\varphi(\cdot)$ is the scaling function and $g(\cdot)$ is the wavelet filter.

Forward discrete wavelet transform of an image is obtained by passing it through a bank filter of high pass and low pass filters followed by the downscale sampling ($2\downarrow$), as depicted in Figure 5.

Figure 5: Forward discrete wavelet transform

This process can continue as far as needed. In this manner, the image decomposes to an approximation image and several smaller detail images. The outputs are the details coefficients (from the high pass filters) and approximation coefficients (from the low pass filters). The results suggest that details coefficients have high absolute value and carry more signal information than noise, while the amplitude of approximation coefficients are small and dominated by noise. In our algorithm, we have two levels of decomposition and perform filtering process for all produced subbands. In addition, among different basis functions that exist for wavelet decomposition [23], we use bi-orthogonal spline wavelet (bior1.5) in the first iteration and Daubechies wavelet (db8) in the second and third iterations.

• Threshold filter

The main idea behind the filtering process is to keep of high-amplitude coefficients that preserve most of image information and to remove the approximation coefficients (i.e. noisy coefficients). Generally there are different wavelet domain filters which are broadly categorized into linear filters and nonlinear filters. Nonlinear filters are based on thresholding technique which is further divided into hard thresholding and soft thresholding[24]. In hard thresholding method, the coefficients that are smaller than a threshold, are set to zero, otherwise, are kept. The "keep or kill" nature of hard thresholding generates image artifacts due to unsuccessful attempts of removing moderately large noise coefficients [25]. To reduce the image artifact, soft thresholding method is used that shrinks the coefficients above the threshold in absolute value. Soft thresholding improves the performance of image denoising in comparison to hard thresholding because of the continuity of soft thresholding, especially when noise level is significant. In addition, optimal soft thresholding yields a smaller risk than the optimal hard thresholding as discussed in [26].

In the wavelet domain, an observed image that is corrupted by additive white Gaussian noise of zero mean and variance $\hat{\sigma}_N^2$ can be represented by (5):

$$Y(i, j) = S(i, j) + N(i, j) \quad (5)$$

In (5), $Y(i, j)$ is noisy image, $S(i, j)$ is noiseless image and $N(i, j)$ is additive white Gaussian noise .

In (6) and (7), Hard and soft thresholding with a threshold equals to λ , are defined as follows: The hard thresholding operator is defined as:

$$\hat{Y}_{\lambda}^{Hard} = \begin{cases} Y(i, j) & |Y(i, j)| > \lambda \\ 0 & |Y(i, j)| \leq \lambda \end{cases} \quad (6)$$

The soft thresholding operator on the other hand is defined as:

$$\hat{Y}_{\lambda}^{Soft} = \begin{cases} \text{sgn}(Y(i, j))(|Y(i, j)| - \lambda) & |Y(i, j)| > \lambda \\ 0 & |Y(i, j)| \leq \lambda \end{cases} \quad (7)$$

In (6) and (7), the λ threshold is calculated from the average of the SAD values of the best matched blocks of each current block and its block matching threshold (i.e. λ_{BM}) as described in the first iteration step, as given in (8):

$$\lambda_{soft} = \text{average}(\lambda_1, \dots, \lambda_n, \lambda_{BM}) \quad n = 1 \dots 32 \quad (8)$$

After applying soft thresholding to each current block, we have its filtered version. Now each current block and all its similar blocks are replaced by the filtered version of current block using motion vector information of fractional block matching process.

Inverse discrete wavelet transform is last step in the wavelet filtering process. This process gives the denoised referenced block which is utilized in the next stage.

- **Weighted average function (WAF)**

In this step, the denoised current block (that is calculated after inverse discrete wavelet transform) is mapped to its original location and other overlapped locations associated blocks that had lowest SAD values in fractional block-matching stage. This process is carried out for all current blocks. The result of this process is different estimations for each block. The final estimate of the denoised image (in each iteration) is constructed from weighted averaging of overlapped blocks which are based on a weighted average function. This weighted average function ($Y : B_n \rightarrow B$) maps each denoised current block to locations of its similar blocks using their motion vectors from fractional motion estimation which is given in (9):

$$\hat{Y}(B_n) = \sum_{i=1}^n \sum_{j=1}^n w(i, j) \hat{Y}(i, j) \quad (9)$$

In (9), $\hat{Y}(B_n)$ is final estimate of current block, $w(i, j)$ is a weight function of overlapping blocks and $\hat{Y}(i, j)$ is an estimate of each current block which is obtained from denoised overlapped blocks. In first iteration that uses soft threshold filter, $w(i, j)$ is obtained from (10), where N_{soft} is the number of non-zero coefficients of each block after inverse discrete wavelet transform and σ^2 is variance of the current block.

$$w_{soft} = \begin{cases} \frac{1}{\sigma^2 N_{soft}} & \text{if } N_{soft} \geq 1 \\ 1 & \text{otherwise} \end{cases} \quad (10)$$

The main idea behind weighted average function is to use the average of several similar denoised blocks (that have lower SAD values in sub-pixel block matching process) instead of just one denoised block. In this way, the chance of preserving details and edges in the restored image is increased.

- **Basic estimate**

The outcome of the weighted average function is a first estimate of the noisy image. Although the basic estimate demonstrates acceptable image denoising performance, some of image details may be lost, especially for high level noise scenarios. To mitigate this problem, we use the first estimate as a pilot signal for using in the next iteration which leads to better image denoising results.

b) Second and afterward iterations

To improve the quality of denoised image and preserving more image details, we carry out one or more image denoising iteration at the price of more computational cost. In this way, we can make a good compromise between denoising performance and computational

requirements. However, as the number of iterations is increased and becomes greater than three, image denoising performance improvement is reduced and may lead to smoothing the details of denoised image. That is the reason we suggest to use three iterations for the proposed algorithm. The most parts of the second and the next iterations are similar to the first iteration and thus they are not described here again. However, there is a difference between them which is use of Wiener filter instead of soft threshold filter. A Wiener filter is a minimum mean square error filter in the sense of Gaussian noise [27]. It has been used in many image denoising algorithms including spatial and frequency domains. However, denoising performance of Wiener filter decreases in image details and edges. To overcome to this problem, Kazubek applied Wiener filter in Wavelet domain [28]. Yet, his work suffers from poor performance at high level noise. To achieve a good performance, Wiener filter needs to the noiseless (i.e. original) image and noise information which are not accessible in practice. To solve these problems, for each current block of noisy image, we use soft threshold filtering together with fractional block-matching in the first iteration to calculate an estimate of the noiseless image. Then, we utilize the estimate image and Wiener filter along with sub-pixel block-matching which result in much better performance in comparison to previous works, as will be shown in experimental result section.

In general, the denoised image is obtained from Wiener filter using (11). In (11), $\hat{S}(w)$ is power spectral density (PSD) of the restored image, $Y(w)$ is the PSD of the noisy image and $H(w)$ is the PSD of Wiener filter.

$$\hat{S}(w) = H(w)Y(w) \quad (11)$$

$H(w)$ is calculated from equation (12) and its optimal value (i.e. H_{opt}) is obtained from (13).

$$H(w) = \left[\frac{|\hat{S}(w)|^2}{|\hat{S}(w)|^2 + |N(w)|^2} \right] \quad (12)$$

$$H_{opt} = \begin{cases} H(w) = 1 & |\hat{S}(w)| > |N(w)| \\ H(w) = 0 & |\hat{S}(w)| \leq |N(w)| \end{cases} \quad (13)$$

In the proposed algorithm, Wiener filter is used in the second iteration based on (11) where $\hat{S}(w)$ is the *PSD* of the denoised image in the first iteration. $N(w)$ is the *PSD* of the estimated noise in the first iteration. H_{opt} in our algorithm is obtained from (14) where λ_{Wiener} is the threshold of Wiener filter which is calculated from (15).

$$H_{opt} = \begin{cases} H(w) = \lambda_{Wiener} & |\hat{S}(w)| > |N(w)| \\ H(w) = 0 & |\hat{S}(w)| \leq |N(w)| \end{cases} \quad (14)$$

$$\lambda_{wiener} = average(\lambda_1, \dots, \lambda_n, \lambda_{BM}) \quad n = 1 \dots 32 \quad (15)$$

In (15), $\lambda_1 \dots \lambda_n$ are the SAD values of similar blocks of current block and λ_{BM} is the SAD value of current block in the restored image achieved from first iteration. When performing third iteration, we must use the SAD values that calculated from the restored image of the second iteration. After denoising of each current block using Wiener filter, the

resulted denoised block is replaced on its position and the positions of its similar blocks using motion vectors from block matching process. The result of this stage is several estimates for each block. Finally, WAF of Wiener filter is used to calculate the weighted average of each block. Please note that due to use of Wiener filter in the second and next iterations, weighted average function is changed and is calculated from (16).

$$w_{Weiner} = \sigma^{-2} (E |Y(i, j) - \hat{Y}(i, j)|^2) \quad (16)$$

III. Extending the IFBM algorithm to color image denoising and video denoising

a) Color image denoising

The straightforward way for applying the proposed algorithm to color images is to separately apply our algorithm to the Red (R), Green (G) and Blue (B) channels of a noisy color image. While the implementation of this method is simple, it does not consider the strong spectral correlation between the R, G and B channels. In addition, all calculations are independently performed for each channel resulting in three times more computational cost. To reduce these problems, at first, we perform a color space transform (i.e. YCbCr) on the noisy image to reduce high correlation between the R, G, and B channels. Then, we only perform fractional block-matching for luminance and based on its results, the same positions of similar blocks are considered for chrominance channels. The reset part of our algorithm is similar for gray scale image and color image. In this way, the computational burden of fractional block-matching is reduced and the structural similarities are imposed from the luminance channel to the chrominance channels. The main reason for imposing of structural similarities from the luminance channel to the chrominance channels is that the luminance channel contains the most important parts of image information such as edge and details. In addition, the luminance channel generally has higher PSNR/SNR than chrominance channels [29] and [30].

b) Video noise removal based on the proposed algorithm

Noise removal of video sequences is an attractive and an important research field which has many applications such as digital video recording systems, video archive and medical imaging systems. This subsection describes our video denoising algorithm which is based on our image denoising algorithm. In general, successive frames of videos are very similar. This similarity is called temporal redundancy. Another kind of redundancy is spatial redundancy which is related to the similarities in an image/frame. In fact, the proposed image denoising benefits from the spatial redundancy. By using our image denoising algorithm and taking advantage of temporal redundancy of successive video frames, we can design a high performance video denoising algorithm. To aim this, for each current block of the current video frame, we apply our image denoising algorithm to the current frame and its neighborhood frames, as depicted in Figure 6. As seen in this figure, for each current block in the current frame, we perform block-matching motion estimation with quarter pixel accuracy within current frame, previous frame (i.e. Ref-1) and next frame (i.e. Ref+1). The other parts of the video denoising algorithm are similar to our image denoising algorithm including color video denoising techniques.

Figure 6: Block-matching motion estimation with two reference frames

IV. Experimental results and discussion

This section firstly explores the computation cost of the proposed algorithm. Then, quality metrics for image quality assessment are briefly reviewed. Finally, it gives the experiment results of the IFBM algorithm and compares its performance to state-of-the-art algorithms including different gray scale and color images, JPEG coded images [31] and gray scale videos.

a) Computational complexity analysis of the proposed algorithm

Computational complexity of our algorithm in terms of the number of arithmetic operations linearly depends on the size of input noisy image (i.e. $M \times N$) as well as the number of iterations. As shown in Figure 1, each iteration consists of two major parts including spatial domain portion (i.e. block matching process and weight averaging function) and transform domain part (i.e. wavelet filtering). Therefore, we model the total computational cost of the proposed algorithm with (17). In this equation, $M \times N$ is the dimensions of input noisy image and i parameter represents the number of iterations. In addition, CC_{BM} , CC_{WF} and CC_{WAF} terms represent the computation cost of the block matching, wavelet filtering and weighted averaging function, respectively.

$$CC_{total} \approx M \times N \sum_{i=1}^3 (CC_{BM}(i) + CC_{WT}(i) + CC_{WAF}(i)) \quad (17)$$

In (17), CC_{BM} is obtained from:

$$CC_{BM} = \frac{A_1 A_2 (N_{current_block}^2 + N_{similar_block}) N_{search}^2}{N_{shift}^2} \quad (18)$$

- Here, A_1 is the sub-pixel accuracy parameter and equals to 1, 2 and 4 for integer, half-pixel and quarter pixel accuracies, respectively.
- A_2 defines the interpolation method for half and quarter-pixel generation and equal to 8 and 16, correspondingly. Note that for integer case, $A_2=1$.
- $N_{current_block}$ denotes the size of current block that equals to eight with a power of two.
- $N_{similar_block}$ refers to the maximum number of blocks that are similar to current block. We can define an upper bound (i.e. 32) to reduce the computational requirement. If 32 similar blocks are found, the search is terminated and block matching for next current block is started.
- N_{search} is search area of current block within the noisy image. As stated, the whole of search area may not be scanned if the number of similar blocks hits the upper bound.
- N_{shift} is the number of pixels that the current block is shifted just before starting a search for next current block.

CC_{WT} is the next term in (17) which denotes the computational cost of the Wavelet filtering stage. It consists of three parts including forward wavelet transform, threshold/Wiener filters and inverse wavelet transform. The properties of these parts depend on many parameters such as decomposition level and basis function of wavelet transform, the value of thresholds (which is different for each block) and etc. Due to above mentioned properties; CC_{WT} is not constant value. Therefore and for simplicity, we consider it as constant value for each iteration.

CC_{WAF} is the last term in (17) that represents weighted averaging function which is denoted as:

$$CC_{WAF} = W(N_{current_block}^2 \times N_{similar_blocks}) \quad (19)$$

As illustrated in section II, WAF is used for averaging of several denoised similar blocks. The value of WAF depends on the number of estimated overlap denoised blocks (i.e. W), the size of current block and the number of similar blocks. Since W and $N_{similar_blocks}$ are not constant, we approximate WAF with a constant value.

b) Quality metrics for image quality assessment

To evaluate the performance of the proposed algorithm, we use peak signal-to-noise ratio (PSNR) and the Structural Similarity (SSIM) index measures as evaluation metrics [32], [33]. PSNR is a mathematical measure of image quality based on the pixel difference between reconstructed image compared with original image and is defined as in (20):

$$PSNR = 10 \log \frac{S^2}{MSE} \quad (20)$$

Where $S=255$ for an 8-bit image and MSE is Mean Square Error between the original and the distorted images.

Another measure for evaluating the performance of image quality is SSIM that was proposed by Wang et. al. It is calculated as follows: Firstly, the original and distorted images are divided into blocks of size 8×8 and then the blocks are converted into vectors. Secondly, two means and two standard derivations and one covariance value are computed from the images as in (21), (22), and (23).

$$\mu_x = \frac{1}{T} \sum_{i=1}^T x_i \quad \mu_y = \frac{1}{T} \sum_{i=1}^T y_i \quad (21)$$

$$\sigma_x^2 = \frac{1}{T-1} \sum_{i=1}^T (x_i - \bar{x})^2 \quad \sigma_y^2 = \frac{1}{T-1} \sum_{i=1}^T (y_i - \bar{y})^2 \quad (22)$$

$$\sigma_{xy}^2 = \frac{1}{T-1} \sum_{i=1}^T (x_i - \bar{x})(y_i - \bar{y}) \quad (23)$$

Thirdly, luminance, contrast, and structure comparisons based on statistical values are computed. The structural similarity index measure between images x and y is given by (24) where c_1 and c_2 are constants. Note that SSIM is applied locally using sliding window of size $B \times B$ that moves pixel by pixel horizontally and vertically covering all the rows and columns of the image, starting from top-left corner of the image.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (24)$$

c) Gray scale and color images denoising

To evaluate the performance of the proposed algorithm, we use a data set of gray scale and color images and use PSNR and SSIM measures as evaluation metrics. For this purpose, we firstly calculate the efficiency of our algorithm for different pixel accuracies including integer, half-pixel and quarter-pixel accuracies with three iterations. Then, we show the experiment results of the proposed algorithm with quarter-pixel accuracy with different level of iterations and different interpolation methods. In addition, we compare denoising performance of the IFBM algorithm with the BM3B, KSVD, LPG-PCA algorithms. The proposed algorithm is an iterative based algorithm and the one must choose the number of iterations how wish to examine. The higher the number of iterations is, the better the image denoising results are, but it will take longer time to calculate the denoised image.

Table 1 gives the simulation results of the proposed algorithm with different pixel accuracies for five gray scale images (i.e. Couple(512×512 pixels), Hill(256×256 pixels), Lena(256×256 pixels), Man(1024×1024 pixels) and Monarch(256×256 pixels) and two color images (including Lena(512×512 pixels) and Barbara(512×512 pixels) in terms of dB PSNR and SSIM for different noise levels (i.e. $\sigma = 10, 20, 30$ and 40) and three iterations. As seen in this table, the accuracy of block matching process has a significant effect on denoising performance. For example, increasing the accuracy from integer to quarter-pixel for the gray Couple and Lena images with $\sigma = 10$ result in 5.2 dB PSNR and 3.7 dB PSNR denoising improvements, correspondingly.

Table 1: Simulation results of the proposed algorithm for different gray scale and color images at different noise level and for integer, half-pixel and quarter-pixel accuracies of the proposed algorithm

method → Image	Integer pixel PSNR(SSIM)	Half-pixel Bilinear PSNR(SSIM)	Quarter-pixel Bilinear PSNR(SSIM)	Half-pixel Bicubic PSNR(SSIM)	Quarter-pixel Bicubic PSNR(SSIM)
Couple $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	34.2(0.9192) 30.9(0.8491) 28.7(0.7843) 27.3(0.7301)	35.3(0.9411) 32.1(0.9181) 29.3(0.8446) 28.2(0.8041)	36.5(0.9636) 34.5(0.9315) 32.6(0.8990) 30.9(0.8676)	36.1(0.9581) 34.0(0.9211) 30.8(0.8943) 29.2(0.8390)	39.4(0.9762) 35.8(0.9404) 33.2(0.9054) 31.2(0.8712)
Hill $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	32.1(0.8612) 29.9(0.8438) 27.6(0.7218) 26.3(0.6895)	32.9(0.8879) 31.8(0.8755) 28.7(0.8418) 27.9(0.8233)	33.8(0.9437) 33.0(0.9243) 31.6(0.8936) 29.7(0.8596)	34.9(0.9491) 32.3(0.9031) 30.4(0.8775) 28.8(0.8480)	36.2(0.9658) 34.9(0.9467) 32.6(0.9100) 30.4(0.8725)
Lena $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	34.3(0.9382) 30.5(0.9135) 28.7(0.8349) 27.4(0.7909)	34.8(0.9451) 31.9(0.9333) 29.4(0.8812) 28.2(0.8201)	35.4(0.9672) 33.2(0.9516) 30.9(0.9153) 29.0(0.8818)	36.3(0.9606) 33.4(0.9587) 30.7(0.9055) 28.6(0.8410)	38.0(0.9769) 34.5(0.9600) 31.3(0.9194) 29.1(0.8835)
Man $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	36.0(0.9074) 30.9(0.8490) 29.0(0.8073) 27.3(0.7301)	36.3(0.9333) 32.2(0.8718) 29.8(0.8309) 28.2(0.7943)	36.8(0.9663) 33.7(0.9363) 30.7(0.8961) 28.7(0.8453)	37.7(0.9699) 33.0(0.9143) 30.1(0.8750) 28.1(0.7909)	39.1(0.9772) 34.4(0.9445) 30.9(0.9008) 28.4(0.8458)
Monarch $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	34.3(0.9663) 30.6(0.9354) 28.3(0.8978) 27.0(0.8731)	34.8(0.9712) 32.7(0.9516) 29.2(0.9212) 27.7(0.8909)	35.4(0.9826) 33.6(0.9709) 30.7(0.9566) 28.2(0.9332)	37.6(0.9868) 33.9(0.9737) 30.4(0.9428) 27.9(0.9192)	39.4(0.9892) 35.5(0.9767) 31.3(0.9615) 28.3(0.9364)
Lena color $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	36.8(0.9408) 33.6(0.9145) 31.6(0.8712) 29.4(0.8507)	37.9(0.9593) 34.1(0.9201) 31.8(0.8875) 29.5(0.8707)	39.3(0.9627) 34.8(0.9282) 32.4(0.9015) 30.3(0.8877)	39.7(0.9638) 34.3(0.9255) 31.9(0.8891) 29.8(0.8715)	40.3(0.9670) 35.0(0.9285) 32.0(0.8999) 30.2(0.8746)
Barbara color $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	34.8(0.9253) 31.4(0.9107) 29.3(0.8875) 27.8(0.8853)	35.5(0.9593) 32.8(0.9265) 30.1(0.9032) 28.6(0.8829)	36.0(0.9617) 34.7(0.9462) 32.0(0.9199) 29.5(0.8856)	36.7(0.9685) 34.9(0.9401) 31.2(0.9067) 28.6(0.8829)	38.5(0.9766) 36.2(0.9579) 32.5(0.9281) 29.6(0.8900)

Table 2 shows the performance of our algorithm for different iteration numbers and for the same images and noise levels that are used for Table 1. As seen in Table 2, the best performance is when the iteration number is three. That is because the pilot signal for the third iteration is more accurate than the second pilot signal. The higher the number of iterations is, the better the image denoising results are, but it will take longer time to calculate the denoised image.

In Table 2, we give the simulation results for bilinear and bicubic filters which are used as interpolation methods for calculating sub-pixel values where the bicubic method outperforms the bilinear version. The reason is that, for calculating the sub-pixels, the bilinear filter uses four neighborhood pixels whereas the bicubic filter needs to sixteen neighborhood pixels. As a result, the bicubic method leads to more accurate sub-pixels at the price of more computational cost.

Table 2: Simulation results of the proposed algorithm for different gray scale and color images at different noise level and for bicubic and bilinear interpolation methods

method → ↓image	First Iteration Bilinear PSNR(SSIM)	Second Iteration Bilinear PSNR(SSIM)	Third Iteration Bilinear PSNR(SSIM)	First Iteration Bicubic PSNR(SSIM)	Second Iteration Bicubic PSNR(SSIM)	Third Iteration Bicubic PSNR(SSIM)
Couple $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	33.1(0.9001) 29.3(0.8104) 27.5(0.7632) 25.9(0.7093)	34.1(0.9114) 30.9(0.8498) 29.0(0.7992) 27.5(0.7456)	36.5(0.9636) 34.5(0.9315) 32.6(0.8990) 30.9(0.8676)	32.6(0.9011) 29.5(0.8111) 27.8(0.7656) 26.5(0.7294)	34.3(0.9154) 31.2(0.8562) 29.2(0.8032) 27.8(0.7656)	39.4(0.9762) 35.8(0.9404) 33.2(0.9054) 31.2(0.8712)
Hill $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	30.2(0.8682) 27.0(0.7200) 25.3(0.6543) 24.1(0.6133)	31.9(0.8819) 28.9(0.7879) 27.1(0.7151) 25.9(0.6613)	33.8(0.9437) 33.0(0.9243) 31.6(0.8936) 29.7(0.8596)	30.7(0.8711) 27.5(0.7250) 26.1(0.6690) 24.8(0.6201)	32.2(0.8921) 29.0(0.7900) 27.3(0.7198) 26.3(0.6703)	36.2(0.9658) 34.9(0.9467) 32.6(0.9100) 30.4(0.8725)
Lena $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	32.3(0.9083) 28.7(0.8271) 26.0(0.7955) 25.5(0.7474)	34.2(0.9302) 30.3(0.8719) 28.3(0.8231) 27.6(0.7799)	35.4(0.9672) 33.2(0.9516) 30.9(0.9153) 29.0(0.8818)	32.7(0.9143) 29.1(0.8305) 26.6(0.8016) 25.2(0.7578)	34.5(0.9322) 30.6(0.8769) 29.0(0.8291) 28.1(0.7827)	38.0(0.9769) 34.5(0.9600) 31.3(0.9194) 29.1(0.8835)
Man $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	32.4(0.8811) 28.3(0.7761) 28.2(0.7498) 25.7(0.6703)	36.1(0.9094) 30.6(0.8328) 28.9(0.7805) 27.7(0.7386)	36.8(0.9663) 33.7(0.9363) 30.7(0.8961) 28.7(0.8453)	33.1(0.8928) 29.1(0.7825) 27.7(0.7336) 26.0(0.6810)	36.4(0.9174) 31.2(0.8428) 29.1(0.7825) 28.2(0.7466)	39.1(0.9772) 34.4(0.9445) 30.9(0.9008) 28.4(0.8458)
Monarch $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	32.5(0.9401) 28.1(0.8977) 26.0(0.8399) 24.4(0.8160)	40.0(0.9617) 30.3(0.9222) 28.1(0.8877) 26.6(0.8500)	35.4(0.9826) 33.6(0.9709) 30.7(0.9566) 28.2(0.9332)	33.2(0.9594) 28.7(0.9002) 26.4(0.8469) 24.9(0.8278)	34.5(0.9677) 30.7(0.9301) 28.5(0.8910) 27.0(0.8562)	39.4(0.9892) 35.5(0.9767) 31.3(0.9615) 28.3(0.9364)
Lena color $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	34.8(0.9153) 29.5(0.8699) 29.0(0.8575) 27.9(0.8491)	36.6(0.9395) 33.9(0.9200) 32.1(0.8911) 29.6(0.8712)	39.3(0.9627) 34.8(0.9282) 32.4(0.9015) 30.3(0.8877)	35.2(0.9243) 30.1(0.8757) 29.4(0.8646) 28.5(0.8548)	37.2(0.9445) 34.2(0.9230) 32.5(0.8981) 30.1(0.8757)	40.3(0.9670) 35.0(0.9285) 32.0(0.8999) 30.2(0.8746)
Barbara color $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	32.9(0.9253) 29.0(0.8828) 27.1(0.8705) 24.9(0.8481)	34.9(0.9492) 31.3(0.9090) 29.4(0.8891) 27.7(0.8800)	36.0(0.9617) 34.7(0.9462) 32.0(0.9199) 29.5(0.8856)	33.1(0.9332) 29.5(0.8905) 27.5(0.8756) 25.3(0.8517)	35.2(0.9512) 31.8(0.9110) 29.7(0.8921) 28.1(0.8897)	38.5(0.9766) 36.2(0.9579) 32.5(0.9281) 29.6(0.8900)

Table 3 gives the performance comparison results of the IFBM algorithm with three iterations and BM3B, KSVD, LPG-PCA algorithms for the same images and noise levels that are used for Table 1. As seen in Table 3, our algorithm achieves the much better denoising performance relative to BM3B, KSVD, LPG-PCA algorithms in terms of PNSR and SSIM measures even at heavy noise levels. For two iterations and based on Table 2, our algorithm still shows better results compared with previous algorithms.

Figures 7 and 8 show the denoised version of the Couple and Barbara images corrupted with Gaussian white noise (i.e. $\sigma = 40$), respectively. As seen in these figures, the denoised images of our algorithm have best visual quality retaining the edges and details. For example, our algorithm preserves the details of faces in the Barbara and Couple denoised images while faces are blurred in the other algorithms. In addition, BM3B, KSVD, LPG-PCA algorithms leads to image artifacts while the proposed algorithm shows no or negligible artifacts as seen in Figures 7 and 8.

Table3: Comparison of image denoising performance between the KSVD, LPG-PCA, IFBM(bilinear) and IFBM(bicubic) algorithms for gray scale and color images

method → image	KSVD PSNR(SSIM)	LPG-PCA PSNR(SSIM)	BM3D PSNR(SSIM)	IFBM(Bilinear) PSNR(SSIM)	IFBM(Bicubic) PSNR(SSIM)
Couple $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	33.4(0.9006) 29.7(0.8211) 27.6(0.7498) 26.2(0.6713)	33.6(0.9011) 30.1(0.8232) 28.0(0.7533) 26.5(0.6894)	34.0(0.9084) 30.8(0.8462) 28.9(0.7932) 27.5(0.7456)	36.5(0.9636) 34.5(0.9315) 32.6(0.8990) 30.9(0.8676)	39.4(0.9762) 35.8(0.9404) 33.2(0.9054) 31.2(0.8712)
Hill $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	31.5(0.8802) 28.0(0.7587) 26.3(0.6843) 25.1(0.6193)	31.7(0.8811) 28.2(0.7690) 26.5(0.6860) 25.3(0.6201)	31.9(0.8819) 28.6(0.7820) 27.0(0.7131) 25.9(0.6613)	33.8(0.9437) 33.0(0.9243) 31.6(0.8936) 29.7(0.8596)	36.2(0.9658) 34.9(0.9467) 32.6(0.9100) 30.4(0.8725)
Lena $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	33.5(0.9203) 29.7(0.8571) 27.8(0.8055) 26.2(0.7504)	33.7(0.9243) 29.7(0.8605) 27.6(0.8066) 26.0(0.7578)	33.9(0.9272) 30.2(0.8699) 28.3(0.8231) 27.3(0.7727)	35.4(0.9672) 33.2(0.9516) 30.9(0.9153) 29.0(0.8818)	38.0(0.9769) 34.5(0.9600) 31.3(0.9194) 29.1(0.8835)
Man $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	33.4(0.9011) 30.1(0.8211) 28.2(0.7498) 27.0(0.6713)	33.7(0.9028) 30.1(0.8144) 28.2(0.7436) 27.0(0.6850)	36.0(0.9074) 30.6(0.8328) 28.8(0.7795) 27.6(0.7366)	36.8(0.9663) 33.7(0.9363) 30.7(0.8961) 28.7(0.8453)	39.1(0.9772) 34.4(0.9445) 30.9(0.9008) 28.4(0.8458)
Monarch $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	33.5(0.9501) 29.6(0.9077) 27.4(0.8663) 25.9(0.8260)	34.2(0.9594) 30.0(0.9202) 27.4(0.8769) 25.9(0.8378)	33.9(0.9577) 30.1(0.9222) 28.0(0.8850) 26.6(0.8462)	35.4(0.9826) 33.6(0.9709) 30.7(0.9566) 28.2(0.9332)	39.4(0.9892) 35.5(0.9767) 31.3(0.9615) 28.3(0.9364)
Lena color $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	35.3(0.9303) 32.0(0.8999) 30.0(0.8855) 28.5(0.8601)	36.2(0.9343) 32.1(0.9005) 31.1(0.8866) 29.6(0.8718)	36.5(0.9365) 33.7(0.9170) 31.9(0.8891) 29.4(0.8697)	39.3(0.9627) 34.8(0.9282) 32.4(0.9015) 30.3(0.8877)	40.3(0.9670) 35.0(0.9285) 32.0(0.8999) 30.2(0.8746)
Barbara color $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	34.0(0.9403) 30.0(0.9028) 27.9(0.8755) 26.6(0.8601)	34.6(0.9432) 30.7(0.9035) 28.5(0.8866) 26.9(0.8647)	34.7(0.9462) 31.1(0.9070) 29.1(0.8791) 27.5(0.8797)	36.0(0.9617) 34.7(0.9462) 32.0(0.9199) 29.5(0.8856)	38.5(0.9766) 36.2(0.9579) 32.5(0.9281) 29.6(0.8900)

(a)

(b)

(c)

(d)

(e)

(f)

Figure 7: Visual performance results for (a) noiseless Couple image, (b) noisy Couple image ($\sigma = 40$ and PSNR=22) , (c) denoised Couple image using KSVD(PSNR=26.2), (d) denoised Couple image using LPG-PCA(PSNR=26.5), (e) denoised Couple image using BM3D(PSNR=27.5) and (f) denoised Couple image using IFBM (PSNR=31.2 and bicubic interpolation method)

(a)

(b)

(c)

(d)

(e)

(f)

Figure 8: Visual performance results for (a) noiseless Barbara image, (b) noisy Barbara image ($\sigma = 40$ and PSNR=26.5), (c) denoised Barbara image using KSVD(PSNR =26.6), (d) denoised Barbara image using LPG-PCA(PSNR=26.9), (e) denoised Barbara image using BM3D(PSNR=27.5) and (f) denoised Barbara image using IFBM(PSNR=29.6 and bicubic interpolation method)

d) Denoising of JPEG images

JPEG is a block-transform compression method for digital images where tradeoff between image quality and storage size can be made by determining the degree of compression. While block-transform compression method of JPEG is simple and efficient, it also leads to undesirable artifacts including blocking and ringing artifacts [34]. Due to popularity of JPEG images, we investigate denoising performance of the proposed algorithm for JPEG images. To evaluate the performance of our algorithm, we use the gray Lena (256×256 pixels) and gay Hill (256×256 pixels) images which are encoded with JPEG [35]. For these images different compression ratios and noise levels are used. Figure 9 shows fragments of the gray scale Lena, coded Lena (compression ratio=2.5), noisy version of coded Lena($\sigma=10$), denoised of coded Lena using the proposed bilinear algorithm and denoised of coded Lena with the proposed bicubic algorithm from left to right. Similar example with different compression ratio (i.e. 3.3) and noise level (i.e. $\sigma=40$) is shown in Figure 10.

Figure 9. Fragments of the gray scale Lena, coded Lena (compression ratio=2.5), noisy coded Lena($\sigma=10$ and 28.13 dB PSNR), denoised coded Lena (bilinear and 37.63 dB PSNR) and denoised coded Lena (bicubic and 40.60 dB PSNR) and from left to right.

Figure 10. Fragments of the gray scale Lena, coded Lena (compression ratio=3.3), noisy coded Lena($\sigma=40$ and 16.09 dB PSNR), denoised coded Lena (bilinear and 29.36 dB PSNR) and denoised coded Lena (bicubic and 29.68 dB PSNR) and from left to right.

In addition to Lena, We perform the simulation for the gray Hill image with a compression ratio of 2.9 and $\sigma=10$, as given in Figure 11. As seen in figures, the proposed algorithm demonstrates good visual performance for both images with different levels of noise and compression ratios.

Figure 11. Fragments of the gray scale Hill, coded Hill (compression ratio=2.9), noisy coded Hill ($\sigma= 10$ and 28.13 dB PSNR), denoised coded Hill (bilinear and 38.49 dB PSNR) and denoised coded Hill (bicubic and 42.61 dB PSNR) and from left to right.

(d) Video denoising results

This subsection gives the results of video denoising for our algorithm (i.e. Video IFBM(V-IFBM)) with different pixel accuracies and compares its performance with well-known Video BM3D (V-BM3D) algorithm [36]. Table 4 presents the experimental results of the V-IFBM and V-BM3D algorithm at different noise levels for Coastguard (288×352 and 300 frames), Foreman (288×352 and 300 frames), Football (240×352 and 125frames), Hall monitor (288×352 and 300 frames), Tennis (240×352 and 112 frames), Foreman color (288×352 and 300 frames) and Tennis color (240×352 and 112frame) image sequences in terms of PSNR and SSIM measures. As seen in Table 4, the proposed algorithm results in much better denoising performance compared with V-BM3D algorithm especially with quarter-pixel accuracy in terms of dB PSNR and SSIM Criteria.

Figure 12 shows the denoising performance of the proposed and V-BM3D algorithms for 40 frames of Tennis and Football image sequences in terms of dB PSNR. From Figure 12, we can see that our algorithm outperforms the V-BM3D algorithm while the proposed algorithm uses only two reference frames whereas the V-BM3D algorithm four reference frames. This means that with lower computational cost, our algorithm obtains better denoising performance than V-BM3D algorithm.

Table 4. Video denoising performance comparison between the V-IFBM and V-BM3D algorithms for different noise levels and image test sequences

Video → ↓method	coastguard $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	foreman $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	football $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	hall monitor $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	tennis $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	foreman color $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$	tennis color $\sigma = 10$ $\sigma = 20$ $\sigma = 30$ $\sigma = 40$
V-BM3D PSNR(SSIM)	33.1(0.9403) 29.5(0.8499) 27.8(0.8349) 26.3(0.8203)	36.6(0.9548) 33.9(0.9494) 32.0(0.9285) 30.4(0.9001)	31.5(0.9386) 27.8(0.8289) 25.9(0.8069) 24.6(0.7804)	36.3(0.9523) 32.5(0.9487) 30.2(0.9367) 28.5(0.9032)	33.9(0.9411) 31.0(0.9315) 29.3(0.9085) 27.8(0.8731)	37.6(0.9423) 34.9(0.9088) 32.9(0.8935) 31.3(0.8741)	34.0(0.9650) 31.1(0.9359) 29.3(0.8929) 27.8(0.8699)
V-IFBM PSNR(SSIM) Integer pixel	33.2(0.9435) 29.5(0.8499) 28.0(0.8377) 26.5(0.8291)	36.7(0.9611) 34.3(0.9445) 32.1(0.9367) 30.4(0.9002)	31.6(0.9432) 29.2(0.8954) 27.2(0.8572) 26.6(0.8321)	36.4(0.9564) 32.8(0.9511) 30.6(0.9379) 28.7(0.9086)	33.9(0.9411) 31.0(0.9315) 29.0(0.8977) 28.2(0.8850)	38.0(0.9564) 35.3(0.9132) 33.1(0.9005) 32.2(0.8873)	33.9(0.9644) 31.2(0.9372) 29.5(0.9014) 27.8(0.8699)
V-IFBM(Bilinear) PSNR(SSIM) Half pixel	34.5(0.9586) 32.3(0.9183) 29.4(0.8862) 28.7(0.8291)	37.8(0.9725) 36.5(0.9562) 33.8(0.9401) 31.4(0.9174)	31.7(0.9446) 30.2(0.9180) 30.1(0.9093) 26.8(0.8256)	36.6(0.9586) 33.0(0.9540) 31.2(0.9425) 29.1(0.9133)	34.0(0.9450) 31.1(0.9354) 29.3(0.9085) 28.4(0.8850)	38.6(0.9512) 37.8(0.9470) 35.3(0.9309) 34.7(0.8677)	34.1(0.9664) 31.4(0.9172) 29.5(0.8964) 28.5(0.8847)
V-IFBM(Bilinear) PSNR(SSIM) Quarter pixel	35.9(0.9739) 34.6(0.9513) 32.8(0.9140) 30.7(0.8558)	39.6(0.9829) 38.2(0.9671) 35.5(0.9474) 32.7(0.9332)	31.9(0.9454) 31.6(0.9317) 30.9(0.8856) 28.8(0.8396)	36.8(0.9617) 33.3(0.9586) 32.0(0.9453) 29.7(0.9277)	34.1(0.9523) 31.3(0.9433) 29.5(0.9112) 28.9(0.8911)	39.8(0.9838) 39.6(0.9829) 38.9(0.9781) 38.2(0.9671)	34.1(0.9674) 31.5(0.9493) 31.2(0.9476) 31.0(0.9433)
V-IFBM(Bicubic) PSNR(SSIM) Half pixel	37.3(0.9832) 35.0(0.9567) 33.1(0.9205) 30.8(0.8593)	39.3(0.9788) 38.0(0.9634) 35.1(0.9394) 32.3(0.9289)	32.6(0.9517) 31.6(0.9363) 31.3(0.8918) 29.1(0.8476)	37.1(0.9617) 33.4(0.9624) 32.1(0.9465) 29.7(0.9277)	34.0(0.9505) 31.6(0.9492) 29.7(0.9155) 28.9(0.8911)	40.5(0.9885) 41.3(0.9867) 40.9(0.9817) 39.2(0.9695)	34.2(0.9682) 33.3(0.9653) 33.2(0.9637) 32.7(0.9614)
V-IFBM(Bicubic) PSNR(SSIM) Quarter pixel	40.9(0.9903) 37.4(0.9654) 34.2(0.9247) 31.2(0.8625)	42.8(0.9895) 40.0(0.9717) 36.1(0.9504) 34.0(0.9348)	35.3(0.9706) 34.3(0.9557) 32.0(0.9045) 29.6(0.8539)	37.6(0.9789) 35.0(0.9633) 32.9(0.9489) 30.0(0.9304)	34.4(0.9697) 34.0(0.9632) 31.0(0.9270) 30.1(0.9011)	43.2(0.9906) 42.8(0.9896) 41.4(0.9842) 39.9(0.9719)	34.4(0.9700) 34.3(0.9697) 34.2(0.9680) 33.9(0.9632)

(a)

(b)

Figure 12: Denoising performance of the V-IFBM (bicubic and bilinear interpolation methods) and V-BM3D algorithms for 40 frames of Tennis (a) and Football (b) image test sequences in terms of dB PSNR

Figures 13 and 14 show an original and its noisy and denoised frames of Football and Tennis image sequences for the V-IFBM and V-BM3D algorithms, respectively. The evaluation of denoised frames shows that visual performance of the proposed algorithm is much better than V-BM3D algorithm. For example, the details of the texts on the poster and

texture of table Tennis net in Tennis video are preserved in the V-IFBM algorithm while they are not clear in V-BM3D algorithm.

(a)

(b)

(c)

(d)

Figure 12: visual performance comparison of Football image test sequence between the V-BM3D and V-IFBM algorithms (a) one noiseless frame (b)noisy frame (PSNR=22.1 and $\sigma = 40$) (c) denoised frame with the V-BM3D algorithm(PSNR=24.6) (d) denoised frame with the V-IFBM algorithm and bicubic interpolation(PSNR=29.6)

(a)

(b)

(c)

(d)

Figure 13: Visual performance comparison between the V-BM3D and V-IFBM algorithms for Tennis image test sequence (a) one noiseless frame (b) noisy frame (PSNR=22.1 and $\sigma = 40$) (c) denoised frame with the V-BM3D algorithm (PSNR=27.8) (d) denoised frame with the V-IFBM algorithm and bicubic interpolation (PSNR=30.1)

V. Conclusion

In this paper we introduce the IFBM algorithm which is very effective for gray scale image denoising. The proposed algorithm benefits from spatial domain and transform domain advantages to improving sparsity representation by using several techniques including fractional block-matching with quarter-pixel accuracy, wavelet domain filtering and WAF. Since the IFBM algorithm is iterative, can tradeoff between the computational complexity and the quality of the denoised image. In addition to gray scale images denoising, the proposed algorithm has been extended for color image and video denoising as well. The experimental results show that the IFBM algorithm achieves much better denoising performance compared with the state of the art algorithms in terms of subjective and objective measures.

References

- [1] R. Yang, L. Yin, M. Gabbouj, J. Astola, and Y. Neuvo, "Optimal weighted median filters under structural constraints," *IEEE Trans. Signal Processing*, vol. 43, pp. 591–604, Mar. 1995.
- [2] R. C. Hardie and K. E. Barner, "Rank conditioned rank selection filters for signal restoration," *IEEE Trans. Image Processing*, vol. 3, pp.192–206, Mar. 1994.
- [3] A. Buades., B. Coll., & J. M. Merol. (2005). A Review of Image Denoising Algorithms,With A New One Multisc. Multiscale Modeling & Simulation (Vol. 4, pp. 490-530).
- [4] A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37:3327–3338, 1997.
- [5] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, 2002.
- [6] D. D. Muresan and T. W. Parks. Adaptive principal components and image denoising. In *Proc. IEEE Int. Conf. Image Process.*, volume 1, Sept. 2003.
- [7] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.
- [8] Mukesh C. Motwani, Mukesh C. Gadiya, Rakhi C. Motwani and Frederick C. Harris, Jr.Survey of Image Denoising Techniques, DATASET · SEPTEMBER 2014, DOI: 10.1109/NRSC.2011.5873625.
- [9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.
- [10] Jeremias Sulam, Boaz Ophir, Michael Elad, IMAGE DENOISING THROUGH MULTI-SCALE LEARNED DICTIONARIES, CONFERENCE PAPER · OCTOBER 2014 DOI: 10.1109/ICIP.2014.7025162.
- [11] L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern Recognit.*, vol. 43, pp. 1531–1549, Apr. 2010.
- [12] Ehab F. Badran & Ammar M. Alhosainy. (2011). Image Denoising Using Non Local Means Algorithm and Subbands Mixing . 28th National Radio Science Conference.
- [13] Shawetangi kala and Raj kumar sahu ,ICA based Image denoising for Single-Sensor Digital Cameras Shawetangi kala, Raj kumar sahu, *IOSR Journal of Engineering* Mar. 2012, Vol. 2(3) pp: 446-450.
- [14] K. Dabov thesis Image and Video Restoration with Nonlocal Transform-Domain Filtering. PhD thesis, Tampere University of Technology, Finland, September 2010.
- [15] Draft ITU-T recommendation and final draft International standard of Joint video specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC C.F.R. (2003).
- [16] High Efficiency Video Coding, ITU-T Rec. H.265 and ISO/IEC 23008–2 (HEVC), ITU-T and ISO/IEC.
- [17]Thomas M. Lehmann, Claudia G'onner, and Klaus Spitzer, Survey: Interpolation Methods in Medical Image Processing, *IEEE TRANSACTIONS ON MEDICAL IMAGING*, VOL. 18, NO. 11, NOVEMBER 1999.
- [18] M.R.H. Fatemi. Algorithm Optimization and Low Cost Bit-Serial Architecture Design for Integer-Pixel and Sub-Pixel Motion Estimation in H.264/AVC. PhD thesis, University of Malaya Kuala Lumpur, Malaysia, 2012.
- [19] Huang, Y.-W., Chen, C.-Y., Tsai, C.-H., Shen, C.-F., & Chen, L.-G. (2006). Survey on block matching motion estimation algorithms and architectures with new results. *The Journal of VLSI Signal Processing*, 42(3), 297-320.
- [20] M. R. H. Fatemi, H. F. Ates, R. Saleh, "A Survey Of Algorithms And Architectures For H.264 Sub-Pixel Motion Estimation", *Journal of Circuits, Systems, and Computers* , vol.21, no.3, May 2012.
- [21]G. Y. Chen, T. D. Bui and A. Krzyzak, "Image denoising using neighbouring wavelet coefficients," *Acoustics, Speech, and Signal Processing*, 2004. Proceedings. (ICASSP '04). IEEE International Conference on, 2004, pp. ii-917-20 vol.2.
- [22]C. Valens, A really friendly guide towavelets ,1999.
- [23] M. Srivastava, C. L. Anderson and J. H. Freed, "A New Wavelet Denoising Method for Selecting Decomposition Levels and Noise Thresholds," in *IEEE Access*, vol. 4, no. , pp. 3862-3877, 2016.
- [24] D. L. Donoho and I. M. Johnstone, "Denoising by soft thresholding", *IEEE Trans. on Inform. Theory*, Vol. 41,pp.613-627,1995.
- [25] MS.ANJALY CHAUHAN, Comparative Analysis of Wavelet Thresholding for Image Denoising Using Discrete Wavelet Transform, *Mathematical and Computational Methods in Electrical Engineering*, ISBN: 978-1-61804-329-0.
- [26] S. Grace Chang, Bin Yu and Martin Vetterli, Adaptive Wavelet Thresholding for Image Denoising and Compression, *IEEE Transactions on Image Processing*, VOL. 9, NO. 9, September 2000.
- [27] R.C Gonzales & R.E Woods. (2008). *Digital Image Processing*. Prentice Hall, New Jersey.
- [28] M. Kazubek, Wavelet domain image denoising by thresholding and Wiener filtering, *IEEE Signal Processing*, VOL. 10, NO. 11, pp. 324 - 326Nov. 2003.
- [29] K. N. Plataniotis and A. N. Venetsanopoulos, *Color Image Processing and Applications*. New York: Springer-Verlag, 2000.

- [30] Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian, Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 16, NO. 5, MAY 2007.
- [31] Wang, Z., Sheikh, H.R., Bovik, A.C.: No-reference perceptual quality assessment of JPEG compressed images. In: IEEE Proceedings International Conference on Image Processing, vol. 1, I-477-I-480, Sept 2002
- [32] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh and Eero P. Simoncelli, Image Quality Assessment: From Error Visibility to Structural Similarity, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 13, NO. 4, APRIL 2004.
- [33] Y. Al-Najjar, D. Chen Soong, Comparison of Image Quality Assessment: PSNR, HVS, SSIM, UIQI, International Journal of Scientific & Engineering Research, Volume 3, Issue 8, August-2012.
- [34] Zhou Wang and Alan C. Bovik, "A Universal Image Quality Index," IEEE SIGNAL PROCESSING LETTERS, vol. 9, pp. 81-84, 2002.
- [35] Aria Nosratinia: Enhancement of JPEG-Compressed Images by Re-application of JPEG. VLSI Signal Processing 27(1-2): 69-79 (2001).
- [36] <https://www.mathworks.com/matlabcentral/fileexchange/38518-jpeg-compression>
- [37] Dabov, K., Foi, A., Egiazarian, K.: Video denoising by sparse 3D transform-domain collaborative filtering. In: European Signal Processing Conference EUSIPCO 2007.