Advanced Engineering Forum *Online: 2013-06-27*
ISSN: 2234-991X, Vols. 8-9, pp 480-489
doi:10.4028/www.scientific.net/AEF.8-9.480
© 2013 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.

Compressed domain computationally efficient processing scheme for JPEG image filtering

Camelia Florea, Mihaela Gordan, Bogdan Orza and Aurel Vlaicu

Technical University of Cluj-Napoca, Department of Communication,
Intelligent and Multimodal Image Processing and Analysis Group,
C. Daicoviciu 15, Cluj-Napoca, Romania.
{Camelia.Florea, Mihaela.Gordan, Bogdan.Orza, Aurel.Vlaicu}@com.utcluj.ro

Keywords: compressed domain processing; JPEG image filtering; Discrete Cosine Transform (DCT); basis images.

Abstract. Image filtering is one of the principal tools used in computer vision applications. Real systems store and manipulate high resolution images in compressed forms, therefore the implementation of the entire processing chain directly in the compressed domain became essential. This includes almost always linear filtering operations implemented by convolution. Linear image filtering implementation directly on the JPEG images is challenging for several reasons, including the complexity of transposing the pixel level convolution in the compressed domain, which may increase the processing time, despite avoiding the decompression. In this paper we propose a new computationally efficient solution for JPEG image filtering (as a spatial convolution between the input image and a given kernel) directly in the DCT based compressed domain. We propose that the convolution operation to be applied just on the periodical extensions of the DCT basis images, as an off-line processing, obtaining the filtered DCT basis images, which are used in data decompression. While this doesn't solve the near block boundaries filtering artefacts for large convolution kernels, for most practical cases, it provides good quality results at a very low computational complexity. These kind of implementations can run at real-time rates/ speeds and are suitable for developments of applications on digital cameras/ DSP/ FPGA.

Introduction

Image filtering operations are widely used in various real time applications of computer vision, due to their usefulness to various tasks, such as noise removal, edge extraction, sharpening, image resizing, feature extraction, image restoration, etc. Real systems manipulate the data in compressed formats, among which the most well-known standards for image and video compression (JPEG, MPEG1 to MPEG4, H.261 to H.264, HDTV, etc.) are based on the discrete cosine transform (DCT). These are also the preferred storage formats for the majority of digital cameras and a large amount of image/video acquisition boards, which means that the compression steps can be performed by embedded software and hardware blocks within an on-board image processor. Therefore, by embedding the processing step into the decompression scheme, the operation of fully decompression and recompression prior/after processing will be avoided - this means that the data is processed directly in the compressed domain.

Any compressed domain processing algorithm implies its formulation in the DCT image representation space and is typically applied on small size image blocks in which the image is prior decomposed by tiling (due to the compression standard [1]). Therefore, the processing can be realized after entropy decoding before inverse DCT step, which is computationally expensive (the concept is illustrated in Fig. 1). The advantages of compressed domain image processing are already acknowledged by the scientific community [2, 3, 4], in terms of the computational speed, not to mention the sometimes better performance, which explained the efforts for the re-formulation of many pixel-level algorithms directly in the JPEG compressed domain.

The formulation in the compressed domain of operations, in which the pixel intensity in the processed image is obtained as a combination of intensities from different positions in the original image aren't straightforward, but these can be computed in the compressed domain. Doing so, implementing these algorithms directly in the compressed domain, it is challenging not only because such operations often cross JPEG blocks boundaries, but also because of the complexity of the transposed formulation. Unfortunately, in some cases, the resulting formulation turns out to be no faster than pixel level processing version. Thus, in the literature, researchers have proposed and implemented several fast methods of these algorithms [5] - [14], where a trade-off between speed and quality was considered. Some implementations/methods reduce dramatically the computational cost, but degrade the processed image quality. In [5, 6] two reformulations of the convolution operation were proposed, exploiting the sparseness of the DCT-domain representation. The algorithm was applied on each block and for computing were considered 8 neighbouring blocks of 8×8 pixels. In [13] was implemented an algorithm for dominant edge direction extraction. It is based on usage of the average intensity values of four sub-blocks partition of an 8×8 pixels block. Using these 4 average intensity values (computed using the quantized DCT coefficients) is possible to compute the three parameters which indicate the block dominant direction as horizontal (H), vertical (V) or diagonal (D). The angle of the edge is decided by analysing the H, V and D values. In [10] the authors focused on feature point extraction in the compressed domain. Therefore, they proposed a compressed domain version of the Laplacian of Gaussian (LoG), and they extracted Harris corner/feature points or points showing a strong curvature directly from the DCT coefficients. In [11, 12], to improve efficiency of compressed image retrieval, they proposed different methods of edge histogram generation in DCT domain. Using the edge information provided by only two AC coefficients of DCT coefficients, they got edge directions and strengths directly in DCT domain [11]. In [12], the content descriptor is constructed by a run-length edge-block histogram with three patterns including horizontal edge, vertical edge and no-edge. While most of the implemented approaches are based on linear filters, they either are computationally complex ([5, 6]), or optimize the speed but at the cost of generality and precision (as the edge detection based solutions [[6] - [14]]).

In this paper we introduce a JPEG compressed domain formulation of the linear image filtering process and we prove mathematically and experimentally not only its numerical efficiency, but also the good quality of the results regardless the type of convolution mask (low pass or high pass filter). Our aim is to calculate efficiently a filtered image, as the result of spatial convolution between the image and a given filter (with explicit kernels, such as: Gaussian, Laplacian, Sobel, etc.). For direct processing in the compressed domain, we exploit the continuity/pattern of the DCT basis images, the convolution being apply on the periodical extensions of the DCT basis images. Therefore, the convolution step can be applied off-line on the DCT dictionary (DCT basis images) and saved in a file, and in the processing step we just load the filtered DCT bases (the modified dictionary) and use them when we decompress the image. This implementation can run at rates/ speeds that approach real-time, therefore, these can open the way to the development of computationally efficient real time image processing implementations on digital cameras/ DSP/ FPGA.

JPEG compressed domain image processing

Nowadays, since the image/video resolutions increased significantly, besides the requirements for high compression rates (in order to save storage space and transmission time) the *requirements for fast processing* (as developments directly in the compressed domain) have become essential. The most used image storage format is the DCT based JPEG format (over 95% of images on Internet are stored in this format), by reason of the advantages offered by this format: low storage capacity needed (while maintaining the image quality) and better performances in information transmission.

The basic steps used to compress/decompress the JPEG images [1] are presented in the following. First, the color image is represented in a decorrelated color space, usually the YUV color space. The YUV color space provides a clear separation between the luminance and chrominance representation, which provides the benefit of a much higher de-correlation of the color components, as compared to the default RGB (Red, Green and Blue) components, and thus allows their independent encoding and

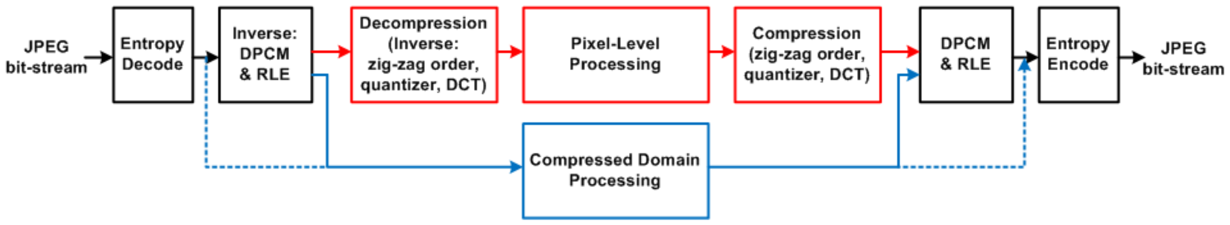
JPEG bit-stream
Entropy Decode
Inverse: DPCM & RLE
Decompression (Inverse: zig-zag order, quantizer, DCT)
Pixel-Level Processing
Compression (zig-zag order, quantizer, DCT)
DPCM & RLE
Entropy Encode
JPEG bit-stream
Compressed Domain Processing

Fig. 1: The two ways to process the image compressed JPEG

processing. The three color components (Y, U, V) in the image are independently scaled symmetrically towards 0 (from their initial range [0, 255] to the range [-128, 127]), and usually, the chrominance components U and V are down-sampled by a factor of two. As a result, the image is represented by three matrices (corresponding to the scaled luminance Y and scaled chrominance components, U and V), which are independently processed as follows: each matrix is divided into 8×8 pixels blocks by tiling, with each 8×8 block being further individually processed for compression. A discrete cosine transform (DCT) is applied on each block and the resulting DCT coefficients are quantized using quantization tables (by dividing them to the corresponding value in the quantization table followed by rounding the division result). Many small coefficients, usually the high frequency ones, are therefore quantized to zero, after this operation. A large among of the AC coefficients became zero after quantization therefore it is obtained a very sparse representation of the blocks of 8×8 pixels. The next step is to zigzag order the DCT coefficients matrix of each block; in this way, the AC coefficients are capable to indicate frequency components in an ascending order. The last steps in the coding algorithm are the DPCM (Differential Pulse Code Modulation) of the DC coefficient and the RLE (Run Length Encoding) of the AC coefficients, followed/ended by entropy coding (Huffman coding). The resulting compressed stream together with a header forms the JPEG file. In the decoder, the compressed image is decoded, inverse quantized and the inverse discrete cosine transform (IDCT) is applied.

There are two ways to process the JPEG compressed images (as illustrated in Fig. 1):

1. *The pixel-level processing* (spatial domain processing) requires the decompression of the JPEG bitstream until at pixel level, prior to the application of the processing algorithm on the pixels spatial matrix. Finally, the resulting image is recompressed.

2. *The compressed domain processing* does not require any decompression/ recompression, but the processing algorithm must be formulated in the DCT image representation space (the RLE vector - illustrated by blue dashed line in Fig. 1, or the zig-zag ordered quantized DCT coefficients - the blue solid line in Fig. 1) to provide the same result as in the pixel level processing case. Once such a formulation is given, it is only necessary to entropy decode the JPEG bitstream, obtaining the zigzag ordered quantized DCT coefficients, process these values, and entropy encode the new values obtaining the new JPEG bitstream. The processing in compressed domain not only reduces the computation time by avoiding the decompression before processing, but also can benefit from JPEG reduced data space (the majority of the AC coefficients are zero after the quantization). Furthermore, the application of the IDCT is avoided, which is computationally expensive.

Avoiding the double compression, by direct image manipulation in the compressed domain, is an important and of current interest aspect. By double compression [15, 16] it is understood that the JPEG image is decompressed to the spatial domain for different reasons and recompressed (at least one time) with the different or the same quantization tables. As it is well known, the DCT-based JPEG compression scheme is lossy. In general, when an image is decompressed and recompressed again some JPEG coefficients will be modified, and the obtained doubly compressed image would have some differences from the original image. The double compression can introduce three kinds of errors: *quantization error*, *truncation error* and *rounding error*. The DCT coefficients are divided by the quantization tables and then rounded to the nearest integer; this leads to *quantization errors*. *The truncation error* and *the rounding error* appear while reconstructing the image in the spatial domain: the float numbers obtained after applying IDCT to the de-quantized DCT coefficients have to be truncated to the interval [0; 255] and to be rounded to the nearest integer value.

If in the case of image processing at pixel level the new algorithms try generally to provide solutions with better results in quality than the existing ones, in the case of processing in the compressed domain the main aim is to achieve the same quality as in pixel level algorithms but at the same time to improve the numerical efficiency of processing, compared to the case of JPEG decompression based processing. As such, all the methods in this field of research are compared with pixel level methods and tend to achieve similar performance, and not necessarily an improvement in the quality of results. If a processing algorithm can be formulated directly in the JPEG compressed domain, it is recommended to implement it, because a local block description is available directly, at no extra computational cost.

The DCT basis images

The Discrete Cosine Transform (DCT) is an orthonormal transform widely used for image processing, analysis and compression due to its good performances (based on its properties: energy compaction and de-correlation of the coefficients of the resulting image representation). The DCT concentrates the spatial image content in relatively few coefficients, so we can treat images as sparse signals in the DCT domain. The first coefficient in the DCT coefficients matrix represents the DC coefficient (it represents the average intensity in a block of pixels), and the other DCT coefficients are the AC coefficients (which describe the variance of intensities changes in pixels within the same block). The DCT is appropriate for sparse representation mainly of natural images (on which JPEG format is successfully used) and typically it is applied on small size image blocks in which the image is priory decomposed by tiling.

Let us consider $\mathbf{X}[H \times W]$ to be a digital image. For the sake of simplicity, we assume $\mathbf{X}$ is a grey scale image. Let us now assume a decomposition of $\mathbf{X}$ into square blocks (by tiling) of size $N \times N$ pixels each: $\mathbf{X} = \{\mathbf{X}_{pq} | p \in [0, \lfloor H/N - 1 \rfloor], q \in [0, \lfloor W/N - 1 \rfloor]\}$, where (p, q) indicates the position of each patch, $\mathbf{X}_{pq}[N \times N]$. The 2D DCT of one block of $N \times N$ pixels, described by the intensity values, can be expressed in matrix form as:

$$\mathbf{Z}_{pq} = \mathbf{C}\,\mathbf{X}_{pq}\mathbf{C}^T,$$

$$\mathbf{C}(u,i) = \begin{cases} \frac{1}{\sqrt{N}}, & i = 0 \\ \sqrt{\frac{2}{N}}\cos(\frac{(2u+1)i\pi}{2N}), & i \neq 0 \end{cases} \tag{1}$$

where $\mathbf{Z}_{pq}$ is the representation of the block $\mathbf{X}_{pq}$ in the DCT domain (the DCT coefficients), and $\mathbf{C}[N \times N]$ denotes the discrete cosine transform matrix.

The DCT matrix $\mathbf{C}$ being unitary and real-valued (and therefore, the inverse of $\mathbf{C}$ being $\mathbf{C}^{-1} = \mathbf{C}^T$), the inverse two-dimensional discrete cosine transform (IDCT) of the block $\mathbf{X}_{pq}$ is given, in matrix form, by the expression:

$$\mathbf{X}_{pq} = \mathbf{C}^T\,\mathbf{Z}_{pq}\mathbf{C}, \tag{2}$$

which states that, once the DCT coefficients of the block are known, the spatial representation of the block can be recovered without error. Of course, if only an approximation of the DCT coefficients of the block is available, an approximation of the spatial representation of the block can be obtained; this is the case in the JPEG compressed images, where the DCT coefficients are only known in an approximate form (due to the quantization), but as long as most of the block energy is preserved in the quantization process, the information loss in the restored block is visually unperceivable.

In terms of the orthogonal transforms theory, the matrix **C** can also be seen as a complete set of orthonormal basis vectors of length N each. Each vector is represented by a line in **C** and denoted by $\mathbf{c}_u{}^T[N \times 1]$ (u stands for the line in the matrix **C**), $\mathbf{c}_u = [\mathbf{C}(u,0)\ \mathbf{C}(u,1)...\ \mathbf{C}(u,N-1)]^T$, $u = 0,1,...,N-1$. In this representation, the matrix **C** becomes:

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_0{}^T \\ \mathbf{c}_1{}^T \\ \cdots \\ \mathbf{c}_{N-1}{}^T \end{bmatrix}. \tag{3}$$

Using the description of the transform matrix **C** in terms of the basis vectors $\mathbf{c}_u$ introduced earlier, the IDCT yields the following form of the spatial block $\mathbf{X}_{pq}$:

$$\mathbf{X}_{pq} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{Z}_{pq}(i,j)\mathbf{c}_i\mathbf{c}_j{}^T, \tag{4}$$

which describes the spatial $N \times N$ pixels image block $\mathbf{X}_{pq}$ as the weighted sum of the matrices $\mathbf{c}_i\mathbf{c}_j{}^T[N \times N]$, over all $i,j = 0,1,...,N-1$. These matrices are called the *basis images* of the discrete cosine transform. The weights by which the basis images are combined to yield the reconstruction of the spatial representation of the image block $\mathbf{X}_{pq}$ are exactly the DCT coefficients of the block, $\mathbf{Z}_{pq}$.

One should note that although the equations above include apparently all the basis images, in practice, for a sparse representation, many of the DCT coefficients $\mathbf{Z}_{pq}$ are zero or very close to zero. Actually, the larger the number of zero coefficients, the better the sparsity property of the decomposition.

Just like the basis vectors found on the lines of the transform matrix **C** form a complete orthonormal set of vectors, the basis images associated to the two-dimensional discrete cosine transform form a complete orthonormal set of matrices. Denoting the basis image $\mathbf{c}_i\mathbf{c}_j{}^T$ by $\mathbf{D}_{C_{i,j}}[N \times N]$, the N^2 basis images can be seen as elements of a complete dictionary, that can be used for the representation of any spatial image block of size $N \times N$. The dictionary has a complete form and can be written as: $\mathbf{D} = \{\mathbf{D}_{C_{i,j}}[N \times N], i,j = 0,1,...,N-1\}$, with:

$$\mathbf{D}_{C_{i,j}} = \mathbf{c}_i\mathbf{c}_j{}^T, \tag{5}$$

which allows to re-write in a more compact form equation (4):

$$\mathbf{X}_{pq} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathbf{Z}_{pq}(i,j)\mathbf{D}_{C_{i,j}}. \tag{6}$$

The basis images of the DCT dictionary, for $N = 8$, scaled within the range [0; 255] and rounded to integers for visualization purposes, are shown in Fig. 2.a).

JPEG image filtering

An advantage offered by JPEG compressed domain for image filtering is that the color space used in the compression scheme decorrelates the luminance information (Y) and color components (U,V); therefore, each component can be processed separately. Furthermore, in the case of high-pass filtering, it is often enough to filter just the luminance component (Y) in order to detect the edges.

Let **G** be an odd-size convolution kernel, $\mathbf{G}[(2K+1) \times (2L+1)]$, with K, L positive integers. The convolution operation between the image **X** and the kernel **G**, formulated at pixel level, can be expressed as:

$$\begin{aligned} &\mathbf{F}(u,v) = \overline{\mathbf{X}}_{(u,v)} \otimes \mathbf{G} = \sum_{k=-K}^{K}\sum_{l=-L}^{L} \mathbf{X}(u-k,v-l)\mathbf{G}(k,l), \\ &\forall u = 0,1,...,H-1, v = 0,1,...,W-1, \end{aligned} \tag{7}$$

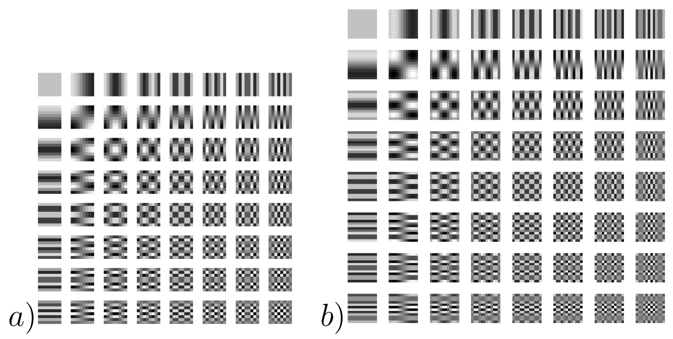

Fig. 2: a) The DCT basis images, forming the complete DCT dictionary of size $N = 8$; b) The periodical extensions of the DCT basis images of size $N = 8$, extended for a 5×5 convolution kernel.

where $\mathbf{F}[H \times W]$ represent the filtered image, and $\overline{\mathbf{X}}_{(u,v)}$ denotes a spatial window from $\mathbf{X}$ centred on (u, v) of the same size as $\mathbf{G}$.

Considering the application of the same formula on a single block from $\mathbf{X}$, $\mathbf{X}_{pq}[N \times N]$, one would get the filtered block $\mathbf{F}_{pq}[N \times N]$ as:

$$\mathbf{F}_{pq}(u,v) = \overline{\mathbf{X}}_{pq(u,v)} \otimes \mathbf{G} = \sum_{k=-K}^{K} \sum_{l=-L}^{L} \mathbf{X}_{pq}(u-k, v-l)\mathbf{G}(k,l), \quad \forall u = 0,1,...,H-1, v = 0,1,...,W-1, \tag{8}$$

with (u, v) the relative spatial locations in the block, from $\{0, 1, ..., N-1\} \times \{0, 1, ..., N-1\}$.

One can notice that, as long as any $u-k$ and $v-l$ is ≥ 0 and $< N$, only the brightness values in the block are enough to compute the filtering result. For the other cases, it will be impossible to estimate accurately $\mathbf{F}_{pq}(u, v)$ without error (i.e., for the boundaries of the block) using the corresponding $\mathbf{X}_{pq}$ alone. This is usually referred as cross boundary artefacts. However considering also the neighbour blocks yields the computation more complex and, as the experiments show, doesn't justify for small kernel sizes. In this paper we consider just the latter case, and attempt to provide a formulation of the linear block filtering independently from the other blocks.

Now let us recall that any $\mathbf{X}_{pq}$ can be written in terms of its DCT basis images dictionary, $\mathbf{D}_C$, expansion as a weighted sum of images known a-priori (as shown e.q. in Fig. 2.a)), with the scalar weights given by the DCT coefficients $\mathbf{Z}_{pq}$ (as expressed in equation (6)).

Therefore each spatial window $\overline{\mathbf{X}}_{pq(u,v)}[(2K+1) \times (2L+1)]$ has a corresponding decomposition of the same form over the basis images set, as long as $u \geq K$, $u \leq N-K$, $v \geq L$ and $v \leq N-L$ (that is, the block boundaries are not crossed):

$$\overline{\mathbf{X}}_{pq(u,v)} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{Z}_{pq}(i,j)\overline{\mathbf{D}}_{C_{i,j}(u,v)}, \tag{9}$$

with $\overline{\mathbf{D}}_{C_{i,j}(u,v)}[(2K+1) \times (2L+1)]$ is the spatial window centred on the location (u, v) in the basis image $\mathbf{D}_{C_{i,j}}$.

Replacing this expression into equation (8) yields:

$$\mathbf{F}_{pq}(u,v) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{Z}_{pq}(i,j)\overline{\mathbf{D}}_{C_{i,j}(u,v)} \otimes \mathbf{G}, \tag{10}$$

and since each $\mathbf{Z}_{pq}(i, j)$ is a scalar, independent of the spatial location in the block (u, v), the expression (10) becomes:

$$\mathbf{F}_{pq}(u,v) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{Z}_{pq}(i,j) \sum_{k=-K}^{K} \sum_{l=-L}^{L} \mathbf{D}_{C_{i,j}}(u-k, v-l)\mathbf{G}(k,l). \tag{11}$$

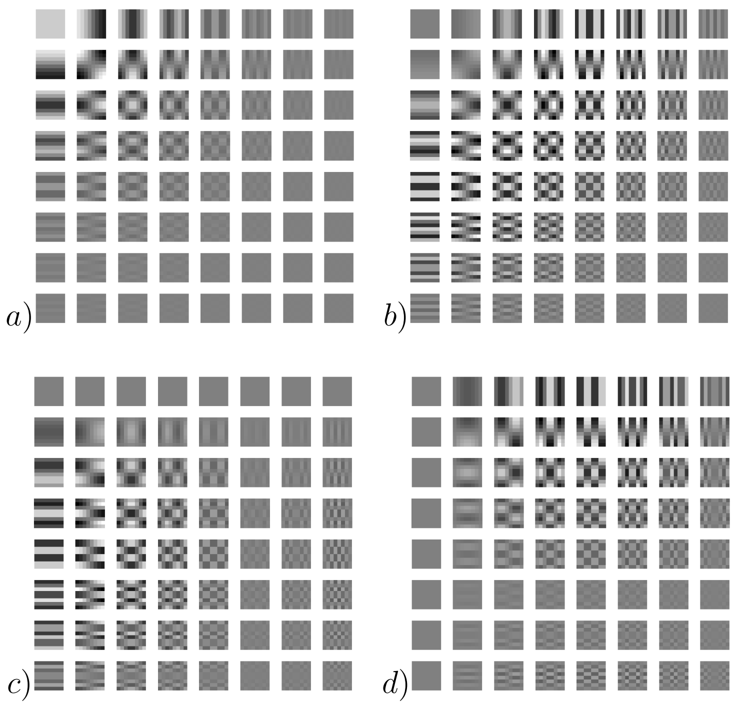

Fig. 3: The filtered DCT basis images, by: a) a Gaussian low-pass filter; b) a LoG high-pass filter; c)-d) two high-pass directional kernels: horizontal and vertical direction.

The convolution product on each basis image $\mathbf{D}_{C_{i,j}}$ will be denoted by $\mathbf{D}_{F_{i,j}}[N \times N]$, with the elements:

$$\mathbf{D}_{F_{i,j}}(u,v) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} \mathbf{D}_{C_{i,j}}(u-k, v-l)\mathbf{G}(k,l) = \overline{\mathbf{D}}_{C_{i,j}(u,v)} \otimes \mathbf{G}. \tag{12}$$

The equation (11) can be written at block level, as:

$$\mathbf{F}_{pq} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathbf{Z}_{pq}(i,j)\mathbf{D}_{F_{i,j}}. \tag{13}$$

Note that, for any $K > 0$ and $L > 0$, it is impossible to compute $\mathbf{D}_{F_{i,j}}$ in all the locations (u, v). To alleviate this problem, it is enough to use larger basis images in the equation (12) or (13), obtained as periodical expansions of $\mathbf{D}_{C_{i,j}}$ to give the size at least $[(N+2K) \times (N+2L)]$ (as illustrated for a 5×5 convolution kernel in Fig. 2.b). The filtered basis images, resulted after convolution of the periodical expansions of the DCT basis images of size $[(N + 2K) \times (N + 2L)]$, have the size $[N \times N]$.

Some examples of the filtered basis images from the DCT dictionary, forming the set $\mathbf{D}_F = \{\mathbf{D}_{F_{i,j}}[N \times N], i,j = 0, 1, ..., N-1\}$, are illustrated in Fig. 3 for different types of kernels: a LPF (Low-Pass Filter) kernel and some HPF (High-Pass Filter) kernels.

Experimental results

We used a set of standard images to evaluate the performance of the proposed method. From the experimental results, it can be seen that the proposed convolution method, besides the high computational speed benefit for JPEG compressed domain images, has good result. The quality of the filtered image (with medium size convolution kernels) is comparable to the pixel level convolution method. The implementation process is divided into two phases: (1) the dictionary of filtered basis images is computed and stored in a file; (2) the linear filtered image is computed, at block level, as a weighted summation

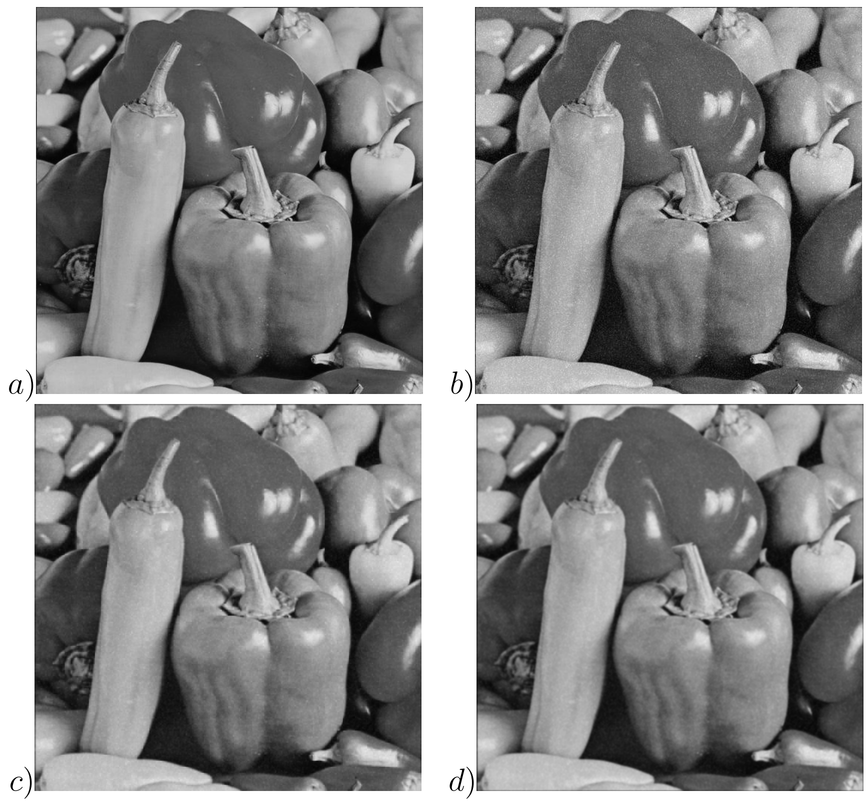

Fig. 4: a) Original image; b) Image affected by impulse noise; c) Low-pass filtered image by the proposed method; d) Low-pass filtered image by convolution at pixel level.

of filtered basis images, with the weights represented by the DCT coefficients. In the figures Fig. 4-Fig. 7 are illustrated some results for image linear filtering by our method in the compressed domain, and by pixel level convolution method.

Summary

In this paper we propose an approach for JPEG image filtering, directly in the DCT based compressed domain, as the result of spatial convolution between the image and a given filter (with explicit kernels, such as: Gaussian, Laplacian, Sobel, etc.). We propose to apply the convolution operation on the periodical extensions of the DCT basis images (we exploit the continuity/pattern of the DCT basis images), which can be computed off-line and just used to obtain directly the linear filtered image at pixel level through their weighted summation. The algorithm is suitable for real-time applications (as the ones on smart devices, digital cameras, mobile phones, IP cameras, etc.) due to the fact that at one moment just a single block of the compressed data it is necessary to be retained in memory for filtering process. Therefore, the image processing in the compressed domain is possible to be done during the JPEG file reading/writing operation. The proposed method is applicable to any DCT based data compression standard, not just JPEG, such as MPEG, H.26x, etc. The results are comparable in quality to the pixel level operations, for medium size convolution kernels.

Acknowledgement

This paper was supported by the project "Development and support of multidisciplinary postdoctoral programs in major technical areas of national strategy of Research - Development - Innovation" 4D-POSTDOC, contract no. POSDRU/89/1.5/S/52603, project co-funded by the European Social Fund through Sectoral Operational Program Human Resources Development 2007-2013.

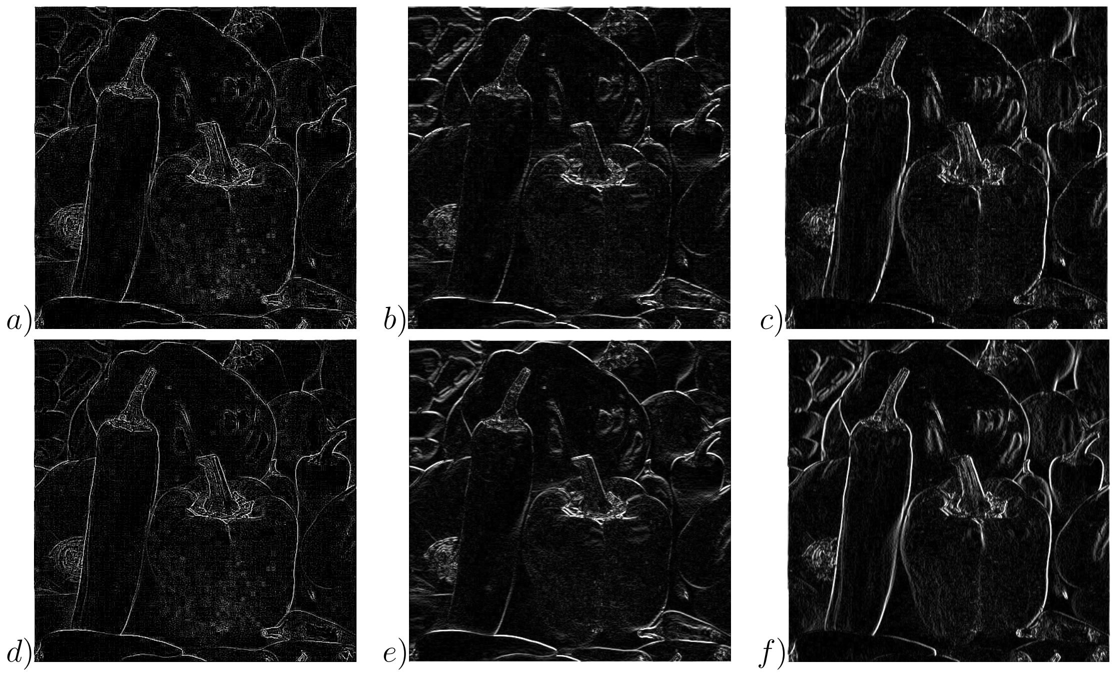

Fig. 5: High-pass filtered images; on the first row - with the proposed method, on the second row - with convolution at pixel level, using the kernels: a)-d) the LoG; b)-e) the Prewitt kernel for changes in the vertical direction; c)-f) the Prewitt kernel for changes in the horizontal direction.

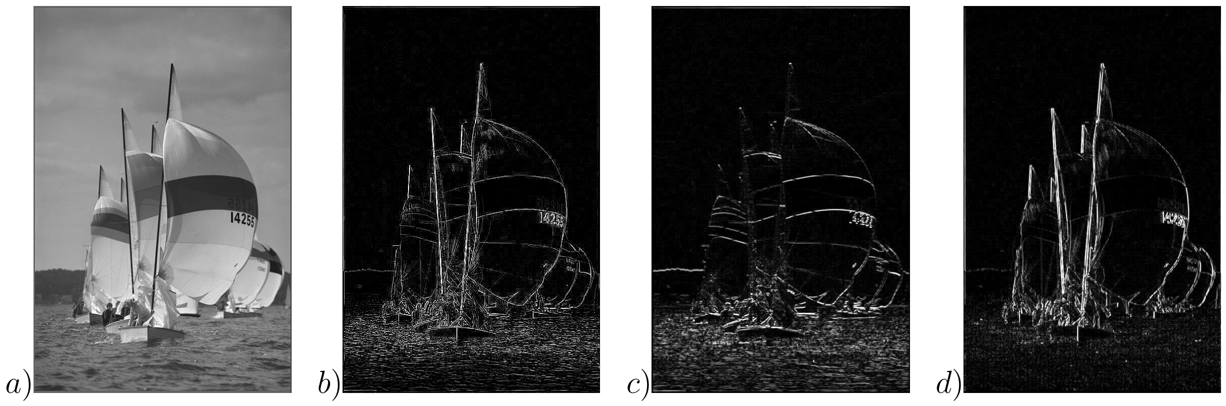

Fig. 6: a) Original image; b),c),d) High-pass filtered images: by the LoG kernel; the Prewitt kernel for changes in the vertical and horizontal directions.

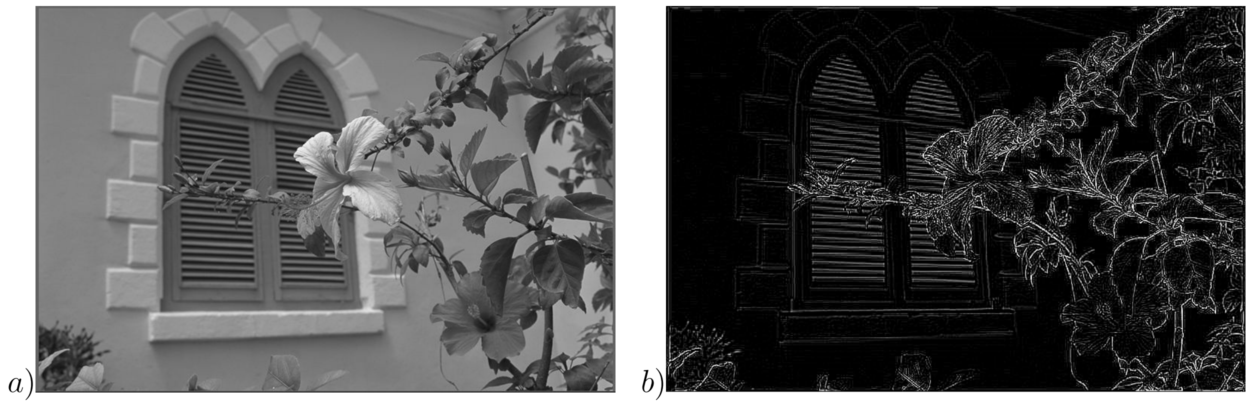

Fig. 7: a) Original image; b) High-pass filtered image by the LoG kernel.

References

[1] G.K. Wallace: The JPEG still picture compression standard. IEEE Transactions on Consumer Electronics, 38(1), 18-34, 1992.

[2] G.M. Farinella, S. Battiato: Scene Classification in Compressed and Constrained Domain. IEEE IET Computer Vision, Institution of Engineering and Technology, UK, 5(5), 320-334, 2011.

[3] R. Kakarala, R. Hebbalaguppe: A method for fusing a pair of images in the JPEG domain. Journal of Real-Time Image Processing, doi: 10.1007/s11554-011-0231-8, 2011.

[4] S. Battiato, G.M. Farinella, M. Guarnera, D. Ravi, V. Tomaselli: Instant Scene Recognition on Mobile Platform. European Conference on Computer Vision (ECCV), Lecture Notes in Computer Science (LNCS 7585), pp. 655-658, Florence, Italy, 7-13 October 2012.

[5] N. Merhav, V. Bhaskaran: A Fast Algorithm for DCT Domain Filtering. Hewlett Packard Laboratories Technical Report, HPL-95-56, May 1996.

[6] B. Shen, I.K. Sethi:Convolution-Based Edge Detection for Image/Video in Block DCT Domain. Journal of Visual Communication and Image Representation, 7(4), 411–423, 1996.

[7] B. Shen, I.K. Selthi: Direct feature extraction from compressed images. SPIE vol. 2670, Storage & Retrieval for Image and Video Databases IV, 1996.

[8] S.W. Lee, Y.M. Kim, S.W. Choi: Fast Scene Change Detection using Direct Feature Extraction from MPEG Compressed Videos, IEEE Transactions on Multimedia, 2(4), December 2000.

[9] S. Lee: Edge Statistics-based Image Scale Ratio and Noise Strength Estimation in DCT-coded Images. IEEE Transactions on Consumer Electronics, 55(4), 2009.

[10] R. Coudray, B. Besserer, P. Courtellemont : Feature point extraction in compressed domain. Storage and Retrieval for Media Databases (SPIESR), January 22, 2003.

[11] M. Eom, Y. Choe: Fast Extraction of Edge Histogram in DCT Domain based on MPEG7. World Academy of Science, Engineering and Technology, 9, 209-212, 2005.

[12] J. Jiang, K. Qiu, G. Xiao: A Block-Edge-Pattern based Content Descriptor in DCT Domain. IEEE Transactions on Circuits and Systems for Video Technology, 18(7), 994 - 998, July 2008.

[13] X. Wang, H. Wang, Y. Huang: Fast Block Edge Direction Analysis In DCT Domain. International Conference on Wireless Communications and Signal Processing (WCSP), Suzhou, China, Oct. 21-23, 703-707, 2010.

[14] F. Zhu: Blocking Artifacts Reduction in Compressed Data. International Conference on Computer Engineering and Applications (IPCSIT), vol.2, Singapore, 2011.

[15] F. Huang, J. Huang, Y.Q. Shi: Detecting Double JPEG Compression With the Same Quantization Matrix. IEEE Transactions on Information Forensics and Security, 5(4), 848 – 856, 2010.

[16] J. Lukas, J. Fridrich: Estimation of primary quantization matrix in double compressed JPEG images. In Proceedings of Digital Forensic Research Workshop, Cleveland, OH, 2003.