Journal of Electronic Imaging

SPIEDigitalLibrary.org/jei

Algorithm for JPEG artifact reduction via local edge regeneration

S. Alireza Golestaneh
Damon M. Chandler

SPIE

IS&T
imaging.org

Algorithm for JPEG artifact reduction via local edge regeneration

S. Alireza Golestaneh and Damon M. Chandler*
Oklahoma State University, School of Electrical and Computer Engineering, Laboratory of Computational Perception and Image Quality, Stillwater, Oklahoma 74078

Abstract. Transform coding using the discrete cosine transform is one of the most popular techniques for image and video compression. However, at low bit rates, the coded images suffer from severe visual distortions. An innovative approach is proposed that deals with artifacts in JPEG compressed images. Our algorithm addresses all three types of artifacts which are prevalent in JPEG images: blocking, and for edges blurring, and aliasing. We enhance the quality of the image via two stages. First, we remove blocking artifacts via boundary smoothing and guided filtering. Then, we reduce blurring and aliasing around the edges via a local edge-regeneration stage. We compared the proposed algorithm with other modern JPEG artifact-removal algorithms. The results demonstrate that the proposed approach is competitive, and can in many cases outperform, competing algorithms. *© 2014 SPIE and IS&T* [DOI: 10.1117/1.JEI.23.1.013018]

Keywords: artifacts removal for JPEG compressed images; blocking artifacts; edge artifacts; discrete cosine transform.

Paper 13323 received Jun. 13, 2013; revised manuscript received Dec. 27, 2013; accepted for publication Jan. 2, 2014; published online Feb. 6, 2014.

1 Introduction

Image and video compression continue to be in high demand. The block discrete cosine transform[1] is the most popular and widely used method in image/video compression standards, including JPEG[2] for images, MPEG[3] for videos, and H.261[4] for videophone/teleconference applications, because of its compaction property and ease of implementation. For JPEG, one widely known shortcoming is that at low bit rates, the compression can leave discontinuities of intensities between adjacent blocks (known as blocking artifacts). JPEG can also lead to other visual artifacts such as degraded textures, blurring, and distortion of edges. In general, decreasing the bit rate will increase the severity and prevalence of these visual artifacts.

Over the past several decades, numerous algorithms have been proposed to improve the visual quality of JPEG images by attempting to remove the artifacts. Two approaches are generally adopted: encoder-based methods and postprocessing-based methods. The encoder-based approaches work by making changes to the encoder, such as transform-domain methods,[5–7] interleaved block transform,[8] interactive methods,[9] lapped transform,[10] combined transform,[11] or wavelet-based filtering.[12] However, the drawback of this approach is that the resulting compression algorithms no longer conform to the JPEG standard.

Postprocessing attempts to improve the visual quality by removing artifacts via processing of the image after decoding. This approach does not require any modifications to the encoder or decoder, and can thus be used on existing JPEG images. Accordingly, most artifact-reduction algorithms follow this latter approach. Postprocessing can roughly be divided into spatial-domain techniques,[13–20] DCT-domain techniques,[21–26] projections onto convex sets (POCS),[27–31] and block-shift filtering.[32–41]

The spatial-domain techniques process the compressed image based on some prior knowledge and information about the original image, such as intensity smoothness or block boundaries of images, to improve the image quality. For instance, in Ref. 13, Reeve and Lim proposed a symmetric two-dimensional (2-D) Gaussian spatial filtering method to reduce the blocking artifacts. Other different methods of spatial-domain techniques, such as those based on gradients/thresholds and the histogram-based methods used in Refs. 15–20, classify the blocks as either high frequency or low frequency, and use filtering methods to remove artifacts and thus improve image quality.

In DCT-domain postprocessing algorithms, blocking artifacts are reduced by directly manipulating DCT coefficients. For example, in Ref. 23, Zeng proposed a DCT-domain method for blocking reduction by applying a zero-masking to the DCT coefficients of some shifted image blocks. By using the fact that visible boundaries between two adjacent blocks in the coded image are primarily oriented along the horizontal and vertical directions, Zeng generated a new data block which is modeled as a 2-D step function contaminated by an independently and identically distributed noise with zero mean and a small variance. Then, by performing a DCT on such blocks, Zeng reported that there always exist AC components of significant energy in a few fixed positions. Accordingly, the author zeroed out some of these AC components and demonstrated that doing so can make blocking artifacts much less visible. However, a loss of edge information caused by the zero-masking scheme can be noticed in his method. In Ref. 24, Jeon and Jeong proposed a postprocessing method to reduce discontinuities of pixel values over block boundaries by compensating for the loss of coefficients' accuracies in the transform domain. They defined the block boundary discontinuity as the sum of the squared differences of pixel values along the block boundary. More

*Address all correspondence to: Damon M. Chandler, E-mail: damon.chandler@okstate.edu

recently, in Ref. 25, Chen et al. proposed an algorithm based on three filtering modes in terms of the activity across block boundaries. They considered the masking effect of the human visual system and integrated adaptive filtering into the deblocking process.

There are also some methods which use both spatial-domain and DCT-domain approaches. For instance, in Ref. 26, Singh et al. proposed an adaptive postfiltering algorithm to remove blocking artifacts. They classify the boundary regions between the blocks as smooth, nonsmooth, or intermediate regions. Then, blocking artifacts in the smooth and nonsmooth regions are removed by modifying selected DCT coefficients while an edge preserving smoothing filter is applied to the intermediate regions.

In addition, there are typical postprocessing iterative methods based on the theory of projection onto convex sets (POCS)[27] and a maximum a posterior probability approach.[28] POCS based on recovery algorithm is a DCT-domain filtering approach. The basic idea is to optimize the value of the quantized transform coefficients, subject to some smoothness and quantization constraints. In Ref. 31, Reeve and Lim introduced a method based on the theory of POCS and proposed a postprocessing technique to reduce blocking artifacts in decoded images. They assumed that the input image is highly correlated so that similar frequency characteristics are maintained between adjacent blocks. The major drawback of this approach is the high computational complexity.

Block-shift filtering is an adaptive filtering algorithm for reducing image artifacts (see e.g., Refs. 40 and 41). Some algorithms have been proposed which attempt to reduce blockiness by using a quad-tree (QT) decomposition and block-shift filtering. QT decomposition[32–35] is a multiresolution image segmentation technique which can partition an image into many homogeneous regions based on some predefined rules. The variable block sizes generated by the QT decomposition facilitate the later block-shift filtering with low computational cost. In Ref. 38, Luo and Ward proposed an adaptive approach which reduced blocking artifacts in both the spatial-domain. For smooth regions, this method took advantage of the fact that the original pixels in the same block provide continuity. In Refs. 40 and 41, Zhai et al. proposed algorithms to preserve the image's details and reduce the effect of quantization noise. They integrated QT decomposition with the block-shift filtering.

Although previous algorithms can effectively suppress blockiness, JPEG images suffer from more than just blocking. At low bit rates, the compressed image also suffers from blurring and aliasing artifacts around the edges. In Refs. 12, 40, and 41, the authors have taken deblocking further, and shown attempts to compensate for degraded textures in the compressed image. However, this approach does not address blurring and aliasing around the edges. Together, all of these artifacts degrade the quality of JPEG images, and there is no single algorithm that can remove all of these artifacts significantly.

In this paper, we propose a technique to improve the visual quality of JPEG images via a two-stage approach. Our algorithm removes blocking artifacts by using existing techniques,[38,42] and it reduces blurring and aliasing artifacts around the edges by means of local edge regeneration. First, we remove blocking artifacts via boundary smoothing[38] and Guided filtering.[42] Then, we reduce blurring and aliasing around the edges via a local edge-regeneration stage. The main contribution of this work is a technique to enhance the quality of JPEG images not only by removing blocking artifacts, but also by reducing blurring and aliasing artifacts around the edges and via edge regeneration.

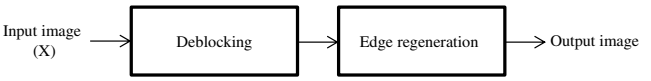


Fig. 1 Flow chart of our proposed algorithm.

This paper is organized as follows: In Sec. 2, we provide details of the two stages of the proposed algorithm. In Sec. 3, we present and discuss the results of the proposed algorithm. General conclusions are presented in Sec. 4.

2 Algorithm

The flow chart of our proposed algorithm is shown in Fig. 1. In our algorithm, given a JPEG image as the input, two stages are used to enhance the quality of the image. The first stage is designed to remove blocking artifacts. The second stage is designed to remove blurring and aliasing artifacts around the edges in an attempt to make the edges appear sharper.

2.1 *Deblocking*

A very well-known problem with JPEG images is blocking artifacts. This stage serves to remove blocking artifacts by smoothing the boundaries of blocks and by using guided filtering.[42] The flow chart of this stage is shown in Fig. 2. Although the technique employed for this deblocking stage is a combination of existing tools, as we will demonstrate later it is quite effective.

At low bit rates, there will be discontinuities between block boundaries due to independent quantization. We first reduce discontinuities between the neighboring blocks by using the method presented in Ref. 38 for smooth areas.

As explained in Ref. 38, their technique for smooth areas first identifies pairs of neighboring blocks whose shared boundary is not due to a genuine change in the intensities at that position. This condition is met if: (1) the two blocks share similar horizontal/vertical frequency properties and (2) a third block centered on the boundary is otherwise relatively smooth. After identifying such blocks, the block-to-block discontinuities are suppressed via DCT-based filtering, i.e., via strategic blending of the DCT coefficients of the two blocks and of the third block encompassing the boundary so as to remove blockiness but not introduce artifacts (for further details, see Ref. 38). By operating only on blocks which satisfy the above two criteria, the technique avoids modifying strong textures and edge regions, and thus attempts to preserve important information, true edges, and textures in the image.

After reducing the discontinuities between block boundaries, we apply a guided filter[42] on the image to reduce the blocking artifacts. According to the authors in Ref. 42, the guided filter utilizes the structures in the image and

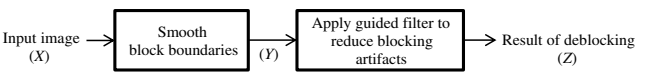


Fig. 2 Flow chart of the deblocking stage.

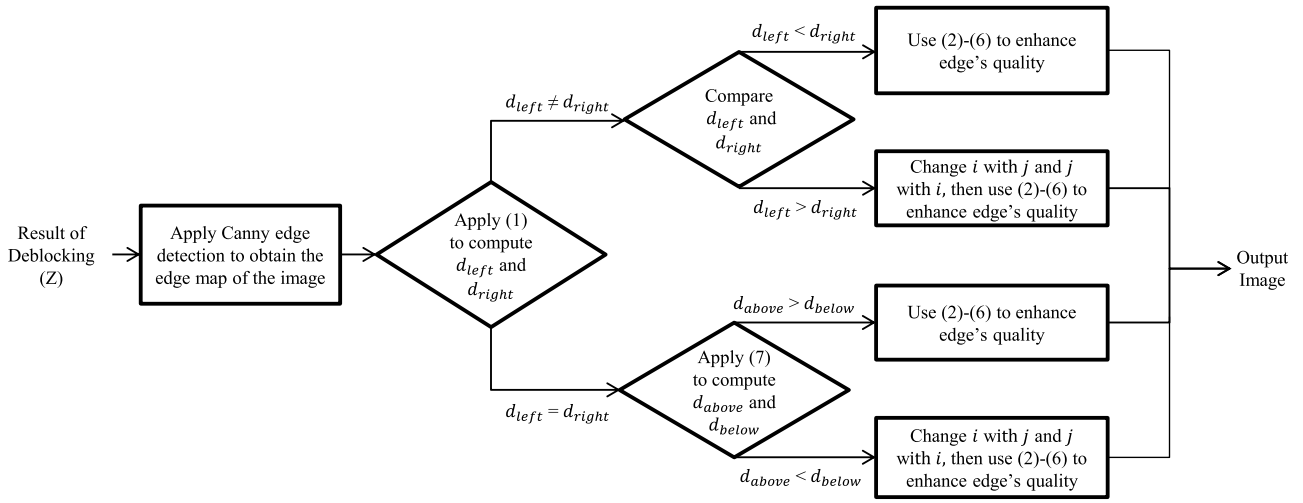


Fig. 3 Flow chart of edge-regeneration stage.

has a fast linear-time algorithm. The guided filter preserves the image's edges during deblocking. Unlike other edge-preserving filters (e.g., the bilateral filter), the guided filtering is computationally efficient [$O(N)$, linear time], and it does not introduce gradient-reversal artifacts found via the bilateral filter.

If we apply the guided filter without smoothing the boundaries, the image may still have blocking artifacts around the boundaries of 8×8 blocks [see Fig. 5(b)]. We assist the guided filtering by explicitly toning-down the block boundaries in the first stage of our deblocking scheme. We have found this first stage to be a simple, yet effective technique of boosting the ability of the guided filtering to remove blocking artifacts.

The guided filter has two parameters: ϵ and α. The parameter ϵ is a regularization parameter and α specifies the local window radius. Increasing ϵ and α generally results in more smoothing. Decreasing these parameters generally results in images which might still contain visible blocking artifacts. Following the same technique used in Ref. 40, we select the ϵ parameter of the guided filter based on the JPEG quality factor. To determine this relationship between ϵ and quality factor (Q), we tested our algorithm on images from the LIVE database[43] (29 original images). We compressed images using JPEG quality factors from 1 to 100 and recorded the ϵ that offered best structure similarity (SSIM)[44] as the optimal ϵ for each image at that setting. The measured optimal ϵ values were then plotted against the corresponding Q values, and these data were then fitted with the following power function:

$$\epsilon = 0.0067Q^{-0.7891} - 0.0003.$$

Note that Q is available at the decoder because the quantization table used in JPEG is a Q-scaled version of a predefined quantization table.

We also choose α empirically ($\alpha = 4$), which generally yields good results across a wide variety of images. However, the selection of this value is not critical, the results are very close when α is chosen within a $\pm 20\%$ range.

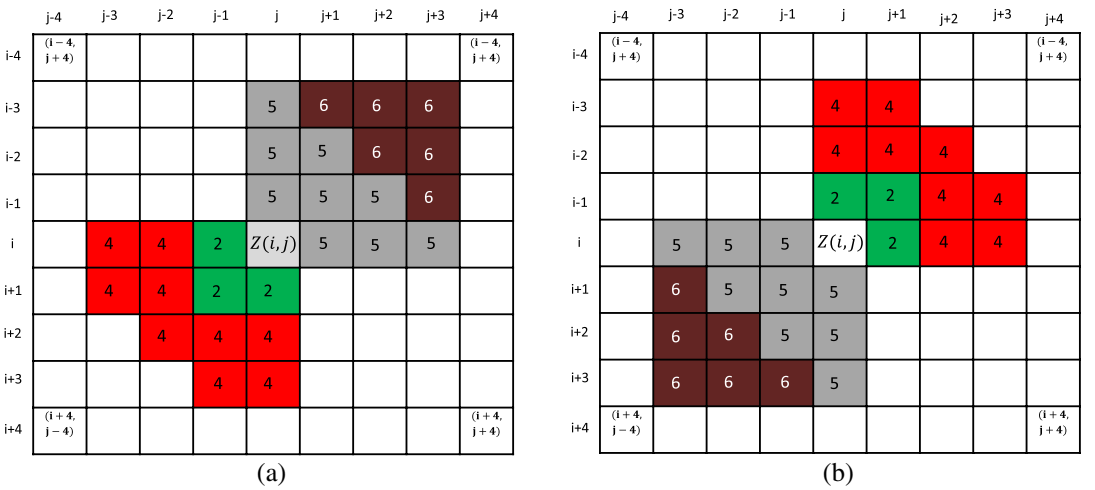


Fig. 4 Illustration of the edge regeneration process for a particular edge pixel $Z(i,j)$. (a) If $d_{\text{right}} > d_{\text{left}}$ or $d_{\text{above}} > d_{\text{below}}$. (b) If $d_{\text{right}} < d_{\text{left}}$ or $d_{\text{above}} < d_{\text{below}}$.

Fig. 5 Deblocking and edge regeneration results. (a) Input image PSNR = 30.40. (b) Result of guided filter without reducing discontinuities between the neighboring blocks. (c) Result after deblocking algorithm [applying Eqs. (1)–(4) and guided filter]. (d) Edge map of *Lena* by using Canny edge detection on (c). (e) Image after proposed edge regeneration, PSNR = 30.56 dB. (f) Shoulder, hat, and cheek of Lena before (left side) and after (right side) applying proposed algorithm.

Figure 5, provided later in the paper, will show results of the deblocking stage [see Figs. 5(a)–5(c)].

2.2 *Edge Regeneration*

Besides blocking artifacts, it is well known that JPEG causes aliasing and blurring artifacts that occur around the edges within the image. In this second and final stages, we use local area information in an attempt to regenerate the edges and thus remove these edge artifacts. The flow chart of this stage is shown in Fig. 3. Let Z denotes the output result of the deblocking stage.

To improve the distorted edges, we use the fact that the blurring and aliasing artifacts are located around the edges; thus, further away from the edges, the quality improves. Our edge-regeneration algorithm operates by regenerating edge pixels as well as pixels that are located slightly farther away from the edges.

In this stage, we first use Canny edge detection[45] to detect the strong edges [see Fig. 5(d)]. In order to compute the strong edges, we use the adaptive threshold parameters γT and βT for low threshold and high threshold of the Canny edge detection, respectively, where T indicates the threshold of the image computed by applying Otsu's method[46] to the gradient magnitude image used during Canny edge detection. Because we focus on improving the strong edges, so we chose $\gamma = 1.5$ and $\beta = 2.5$, and the kernel size parameter is set to 4 for the Canny edge detector. As we will demonstrate, these parameters generally yield excellent results across a variety of images and bit rates. However, the selection of these values is not critical. The results are very close when the values are chosen within a $\pm 20\%$ range.

Let denote an edge pixel value which is detected by Canny and is located at the positions i and j. We determine which area around the edges is more similar to the edges pixel by applying the following equation:

$$Z(i,j)d_{\text{left}} = \left| Z(i,j) - \frac{Z(i,j-1) + Z(i,j-2)}{2} \right|,$$
$$d_{\text{right}} = \left| Z(i,j) - \frac{Z(i,j+1) + Z(i,j+2)}{2} \right|. \quad (1)$$

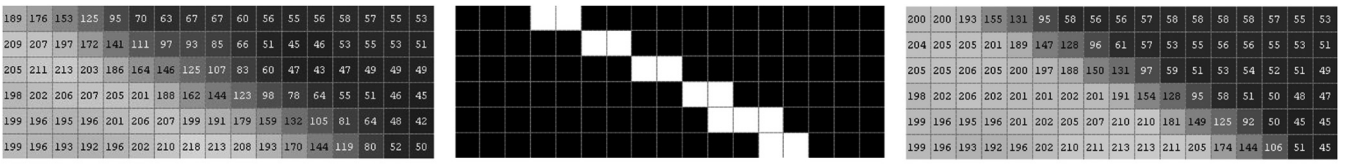
In Eq. (1), we chose to compare the edge pixel against the average of the two pixel values to the left and right. In this equation, d_{left} represents the difference between the edge pixel $Z(i,j)$ and the average of the two pixels horizontally to the left. Also, d_{right} represents the difference between the edge pixel $Z(i,j)$ and the average of the two pixels horizontally to the right. The smaller of the two measurements, d_{left} and d_{right}, signifies the side which is more similar to the edge pixel. By computing d_{left} and d_{right}, we measure the similarity of two sides of the edge pixel to the pixel on the edge.

After computing d_{left} and d_{right}, if $d_{\text{left}} < d_{\text{right}}$, this signifies that the left side of the edge pixel is more similar to the edge pixel than the right side. We would then use the side which is similar to the edge pixel to remake the edge pixel and its neighbors. Next, we use the pixels on the side which is different from the edge to reduce the artifacts around the edge.

To illustrate this process, Fig. 4(a) shows the 9×9 matrix centered around the edge pixel $Z(i,j)$. This matrix was designed around the fact that edge artifacts reduce in pixels further away from the edge.

In the following equation, we take the benefit of the area which is similar to the edge pixel to improve the edge pixel:

For $(i',j') = (i+1, j-1), (i, j-1), (i+1, j)$ [see the green blocks in Fig. 4(a)]:



(a) (b) (c)

Fig. 6 Close-up results for part of Lena's shoulder. (a) Before edge regeneration. (b) The result of Canny edge detection on (b). (c) The result after edge regeneration.

$$Z(i',j') = \frac{Z(i'+1,j') + Z(i',j'-1) + Z(i'+1,j'-1) + 4Z(i',j')}{7}. \tag{2}$$

For the edge pixel itself:

$$Z(i,j) = \frac{Z(i+1,j-1) + Z(i,j-1) + Z(i+1,j) + Z(i,j)}{4}. \tag{3}$$

For $(i',j') = (i,j-3),(i,j-2),(i+1,j-3),(i+1,j-2),(i+2,j-2),(i+2,j-1),(i+2,j),(i+3,j-1),(i+3,j)$ [see the red blocks in Fig. 4(a)]:

$$Z(i',j') = \frac{Z(i'+1,j') + Z(i',j'-1) + Z(i'+1,j'-1) + 4Z(i',j')}{7}. \tag{4}$$

In Eqs. (2)–(4), the algorithm begins by improving the pixels near the center pixel. Equation (2) describes the regeneration of the pixels neighboring the center block, as identified by the green blocks in Fig. 4(a). In this equation, we use the average of three neighboring pixels along with the pixel itself to regenerate the selected pixel. The algorithm then uses these improved pixel values to enhance the center block. Specifically, Eq. (3) uses the average of regenerated pixels as well as the previous value of the edge pixel to regenerate the edge pixel. Finally, Eq. (4) describes how to enhance the quality of other pixels on the similar side, but slightly further away [red blocks in Fig. 4(a)].

The next step is to reduce aliasing and blurring artifacts on the other side. By using the following equation, we regenerate pixels which are slightly further away from the edge pixel to reduce the artifacts around the edges:

For $(i',j') = (i-3,j),(i-2,j),(i-2,j+1),(i-1,j),(i-1,j+1),(i-1,j+2),(i,j+1),(i,j+2),(i,j+3)$ [see the gray blocks in Fig. 4(a)]:

$$Z(i',j') = \frac{Z(i'-1,j') + Z(i',j'+1) + Z(i'-1,j'+1)}{3}. \tag{5}$$

For $(i',j') = (i-3,j+1),(i-3,j+2),(i-3,j+3),(i-2,j+3),(i-2,j+2),(i-1,j+3)$ [see the brown blocks in Fig. 4(a)]:

$$Z(i',j') = \frac{Z(i'-1,j') + Z(i',j'+1) + Z(i'-1,j'+1) + 4Z(i',j')}{7}. \tag{6}$$

In Eqs. (5) and (6), by using the average of the pixels on the side which is different from the edge pixel, we regenerate pixels around the edge and reduce the aliasing and blurring to make the edge appear sharper. First, by using Eq. (5), we regenerate neighboring pixels [gray blocks in Fig. 4(a)]. Next, by using Eq. (6), we regenerate selected pixels [brown blocks in Fig. 4(a)] to make the area around the edge more smooth.

Equations (2)–(6) are applied to all edge pixels identified by the Canny edge detection algorithm. However, the algorithm will not regenerate an edge pixel until it is the center pixel in Fig. 4(a). For example, if the pixel directly below $Z(i,j)$ in Fig. 4(a) is identified as an edge pixel, it will be used only for calculating the result of Eq. (2); however, Eq. (2) will not be applied to that pixel. Only Eq. (3) is used in regenerating pixels identified as edge pixels by the Canny edge detector algorithm.

In Eq. (1), after calculating d_{left} and d_{right}, if $d_{\text{right}} < d_{\text{left}}$, this signifies that the right side of the edge pixel is more similar to the edge pixel than the left side. In this condition, the equations are similar to Eqs. (2)–(6), except that i is replaced with j and j is replaced with i, resulting in a 180 deg rotation in the image plane of the regeneration pattern about $Z(i,j)$ [see Fig. 4(b)]. We chose the basic structure for Eqs. (2)–(6) due to its relative simplicity and its ability to operate effectively on all edge orientations.

If $d_{\text{left}} = d_{\text{right}}$, it means that the edge is horizontal. As a result, we use the following equation to compute the side which is similar to the edge pixel:

$$d_{\text{above}} = \left| Z(i,j) - \frac{Z(i-1,j) + Z(i-2,j)}{2} \right|,$$
$$d_{\text{below}} = \left| Z(i,j) - \frac{Z(i+1,j) + Z(i+2,j)}{2} \right|. \tag{7}$$

In Eq. (7), d_{above} represents the difference between the edge pixel $Z(i,j)$ and the average of the two pixels vertically above. Also, d_{down} represents the difference between the edge pixel $Z(i,j)$ and the average of the two pixels vertically below. If $d_{\text{above}} > d_{\text{below}}$, this signifies that the region below the edge pixel is more similar to the edge pixel than the region above. Therefore, we use Eqs. (2)–(6) for the pixels which are shown in Fig. 4(a). On the other hand, if $d_{\text{above}} < d_{\text{below}}$, it means that the region above is more similar

Fig. 7 Eleven 512×512 test images. For the first row images, from left to right: *Lena*, *baboon*, *Barbara*, *peppers*, *plane*, and *bridge*. For the second row images, from left to right: *Zelda*, *redwood*, *actor*, *fruit*, and *Native American*.

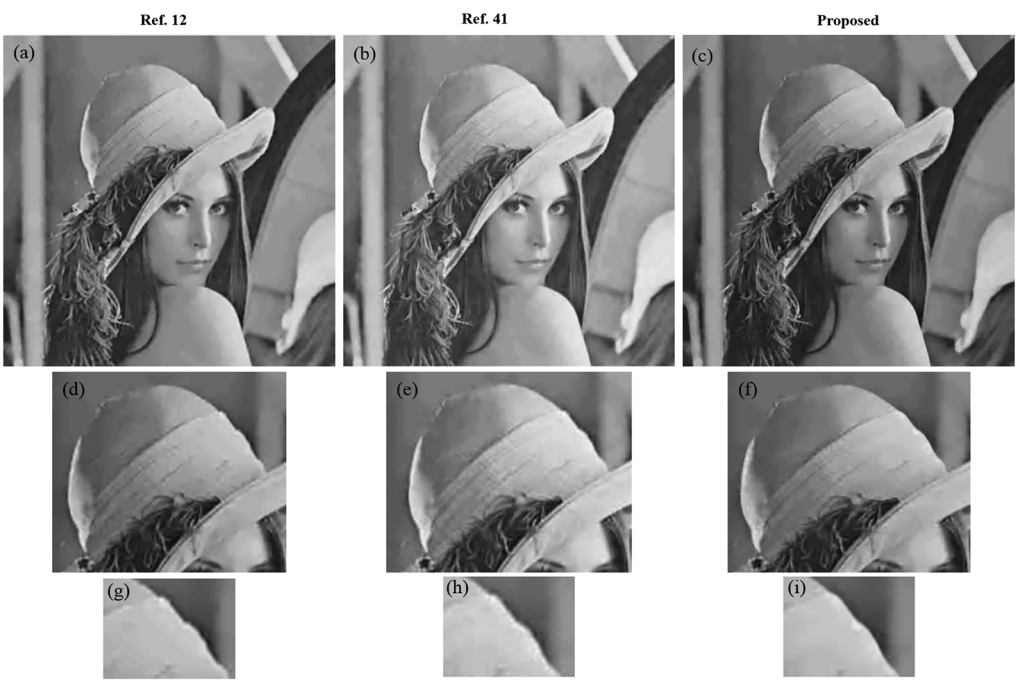


Fig. 8 Subjective quality comparison of *Lena* compressed at 0.24 bpp. (a) The result of Ref. 12, PSNR = 30.99 dB. (b) The result of Ref. 41, PSNR = 31.23. (c) The result of our proposed algorithm, PSNR = 30.56 dB. (d), (e), and (f) a close-up of *Lena* for the results shown in (a), (b), and (c), respectively. (g), (h), (i) a close-up of *Lena* for the results shown in (d), (e), and (f), respectively.

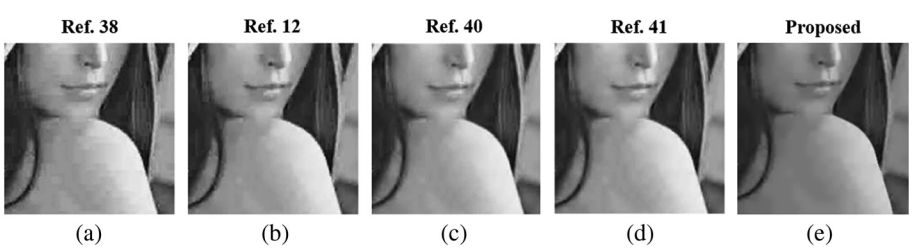


Fig. 9 Close-up of *Lena* compressed at 0.24 bpp. (a) Result of Ref. 38, (b) result of Ref. 12, (c) result of Ref. 40, (d) result of Ref. 41, and (e) result of our proposed algorithm.

Fig. 10 Subjective quality comparison of the images *Barbara*, *baboon*, and *peppers* compressed at quality factor $Q = 10$. (a), (b), and (c) JPEG compressed images, PSNR = 26.33, 23.43, and 30.72 dB. (d), (e), and (f) the result of Ref. 12, PSNR = 26.02, 23.53, and 30.91 dB. (g), (h), and (i) the result of Ref. 40, PSNR = 26.03, 23.67, and 30.90 dB. (j), (k), and (l) the results of our proposed algorithm, PSNR = 26.44, 23.68, and 31.28 dB. (m), (n), and (o) the two left images are close-up of the results from Refs. 40, 12 and right image is a close-up of the result from our proposed algorithm.

to the edge pixel than the region below. In this case, we use Eqs. (2)–(6) after changing i with j and j with i.

Equations (1)–(6) are applied to all the edge pixels which are detected by the Canny edge detector. The edge-regeneration stage operates on edges of all orientations. As long as a pixel is an edge pixel (as detected by the Canny edge detector), then the edge regenerator is applied at that pixel. The context structure shown in Fig. 4 is meant to illustrate only which surrounding pixels are used during the regeneration. So the algorithm will operate on edges of all orientation, but which surrounding pixels it uses to perform the regeneration is limited to the two choices shown in Fig. 4. Note that if the surrounding pixels are also part of the edge, those pixels are not used to regenerate the current edge.

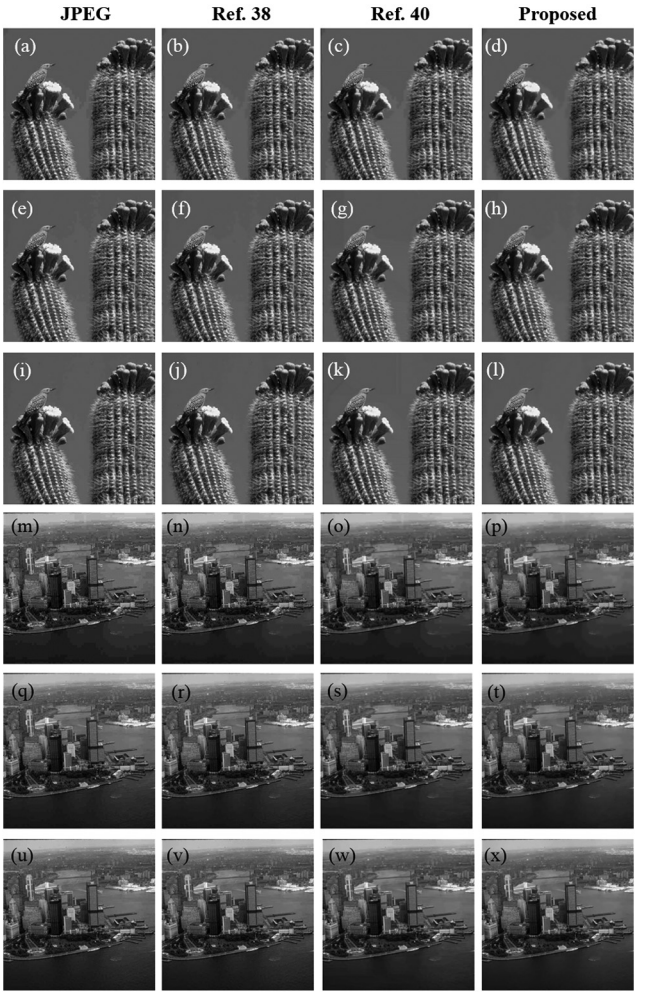


Fig. 11 Subjective visual quality comparison of the images *Cactus* and *Aerial city* compressed at quality factors $Q = 10$, 20, and 30. (a) and (m) compressed at $Q = 10$. (e) and (q) compressed at $Q = 20$. (i) and (u) compressed at $Q = 30$. (b), (f), (j), (n), (r), and (v) the result of Ref. 38. (c), (g), (k), (o), (s), and (w) the result of Ref. 40. (d), (h), (l), (p), (t), and (x) the results of our proposed algorithm.

Figure 5(e) shows the result of this stage. Additionally, in Fig. 5(f), we provide close-up results for the edges around the shoulder, hat, and cheek of Lena to demonstrate that this stage can significantly improve strong edges regardless of the edge orientations. Specifically, as shown in Fig. 5(f), our proposed method improves the quality of edges without considering the orientations of the edges. A further close-up for part of Lena's shoulder is shown in Fig. 6. Further results are provided in Sec. 3.

3 Results

In this section, we evaluate the performance of our proposed algorithm and compare it to competing algorithms, both in terms of qualitative results and in terms of quantitative results. For these evaluations, we have used the 11 standard images shown in Fig. 7.

3.1 *Qualitative Results*

We compared our proposed algorithm with Refs. 12, 38, 40, and 41. As mentioned in Sec. 2, all of these are modern, state-of-the-art postprocessing-based methods designed specifically for the use with JPEG images. Reference 12, by Liew and Yan, is a method based on a noniterative wavelet-based deblocking algorithm. Reference 38, by Luo and Ward, is an adaptive approach in both the spatial-domain and DCT-domain to reduce the block-to-block discontinuities. Reference 40, by Zhai et al., is a deblocking method for JPEG images via postfiltering in shifted windows of image blocks. Reference 41, by Zhai et al., employs a QT decomposition with block-shift filtering to reduce JPEG artifacts.

The visual results of our proposed algorithm on four images (Lena, peppers, Barbara, and baboon) are provided in Figs. 8–10 and are compared against the aforementioned algorithms. We chose Lena, Barbara, peppers, and baboon because the results of Refs. 12, 38, 40, and 41 on these images have been published in the respective papers. Note that the results for Ref. 12 were obtained from the authors, the results for Refs. 38 and 40 were generated by reimplementing their algorithms, and the results of Ref. 41 were obtained from their paper.

Figures 8(a) and 8(b) depict the results of Refs. 12 and 41 on Lena. For ease of examination, close-ups of Figs. 8(a) and 8(b) are provided in Figs. 8(d) and 8(e). The result of our proposed algorithm is shown in Fig. 8(c), and its close-up is provided in Fig. 8(f). Notice that whereas the results of Refs. 12 and 41 still exhibit aliasing and blurring (e.g., along the hat of Lena), these artifacts are better reduced by the proposed algorithm. Overall, the proposed algorithm has noticeably reduced the appearance of blocking and for edges blurring and aliasing. Figures 8(g), 8(h), and 8(i) show further close-ups of the results of the algorithms on the band in Lena's hat. Notice that the results from the proposed algorithm [Fig. 8(i)] exhibits the least ringing and the sharpest edge.

A portion of the shoulder in Lena is shown in Fig. 9. We chose this portion because of the smooth regions on the skin of the shoulder, as well as the strong edges of the shoulder rim. Our proposed algorithm yields superior distortion suppression performance in this region in comparison to others.

In Fig. 10, we compare our results for Barbara, baboon, and peppers with original JPEG compression and also the results of Refs. 40 and 12. Figures 10(a), 10(b), and 10(c) show the JPEG results of Barbara, baboon, and peppers. In Figs. 10(d) and 10(g), although Refs. 40 and 12 reduced many blocking artifacts, some noticeable blocking still remains. The result of our proposed algorithm in Fig. 10(j) shows that our algorithm removed blocking artifacts as well as removed aliasing and blurring around the edges. Figures 10(e) and 10(h) show the results of Refs. 40 and 12 for baboon. In Fig. 10(k), the edges along the nose of the baboon look sharper and better in comparison with Figs. 10(h) and 10(b). Also, Fig. 10(k) has less blocking artifacts compared to Fig. 10(e). In Fig. 10(l), the edges of peppers do not have ringing artifacts as compared to Figs. 10(c), 10(f), and 10(i). Figures 10(m), 10(n), and 10(o) provide close-ups of the aforementioned results from our proposed algorithm against Refs. 12 and 40. Figure 11 provides more visual comparison results of our algorithm to Refs. 12 and 40. Additional examples are available in the online supplement to this paper located at http://vision.okstate.edu/jpgregen/jpgregen.htm.

Table 1 Comparison of PSNR (dB) for various postprocessing techniques applied to different JPEG images. In this table, the bold numbers represent the best performing results and italicized numbers represent the second best performing results.

Images	Bitrate (bpp)	JPEG	Ref. 12	Ref. 38	Ref. 40	Ref. 41	Proposed
Lena	0.17	27.32	28.13	27.53	**28.46**	28.31	*28.33*
	0.24	30.40	30.99	30.31	*31.07*	**31.23**	30.56
	0.31	31.93	32.42	31.59	32.55	*32.57*	**32.61**
	0.36	32.95	33.34	32.43	*33.40*	33.39	**33.41**
Baboon	0.26	21.52	21.79	21.57	21.96	*21.98*	**22.06**
	0.46	23.43	23.53	23.30	*23.67*	23.62	**23.68**
	0.62	24.50	24.56	24.29	**24.73**	24.59	*24.61*
	0.76	25.26	25.29	24.97	**25.43**	*25.31*	25.24
Bridge	0.23	23.06	23.42	23.08	**23.45**	*23.43*	23.34
	0.41	25.13	*25.34*	24.90	25.32	**25.48**	24.98
	0.56	26.25	26.40	25.88	26.40	**26.51**	*26.49*
	0.69	27.01	*27.11*	26.54	**27.14**	27.10	**27.14**
Barbara	0.23	24.50	24.38	23.99	24.85	*24.87*	**24.91**
	0.35	*26.33*	26.02	25.66	26.03	26.07	**26.44**
	0.45	*27.64*	27.33	26.90	**27.75**	27.28	*27.64*
	0.54	*28.78*	28.47	27.95	**28.80**	28.41	28.51
Peppers	0.21	28.19	28.27	27.56	*28.61*	28.26	**29.08**
	0.31	30.72	*30.91*	30.13	30.90	30.90	**31.28**
	0.38	31.85	32.14	31.36	*32.18*	32.13	**32.29**
	0.46	32.71	32.91	32.10	*32.98*	32.85	**32.99**
Plane	0.18	26.54	27.14	26.68	*27.23*	**27.29**	27.02
	0.28	30.16	**30.61**	29.64	30.18	*30.43*	29.71
	0.36	31.94	*32.19*	31.06	31.82	31.95	**32.51**
	0.43	32.97	*33.23*	31.99	33.03	33.01	**33.40**

3.2 *Quantitative Results*

PSNR and SSIM[44] are two different, but widely accepted, methods for numerically representing image quality. Table 1 shows PSNR results of our proposed algorithm, as well as the results from Refs. 12, 38, 40, and 41. Table 2 shows a similar comparison using SSIM. Both Tables 1 and 2 show that the proposed algorithm gives competitive scores for both PSNR and SSIM.

To compare the algorithms' performances, we selected six images from Fig. 7 which represent a range of image content with different properties, such as textures and edges. In Table 1, we provide PSNR for these six images at four different ranges of bit rates, providing a total of 24 scores. Out of the 24 scores for PSNR, our algorithm had the best performance 13 times and the second best performance four times.

Table 2 shows SSIM results for the algorithms. In terms of SSIM, our proposed algorithm outperforms the results of Refs. 12, 38, and 40, and has comparable performance to Ref. 41. In this comparison, the proposed algorithm is a top performer as many times as any other existing state-of-the-art algorithm. However, it is important to note that neither PSNR nor SSIM are perfect measures of visual quality.

Table 2 Comparison of SSIM for various postprocessing techniques applied to different JPEG images. In this table, the bold numbers represent the best performing results and italicized numbers represent the second best performing results.

Images	Bitrate (bpp)	JPEG	Ref. 12	Ref. 38	Ref. 40	Ref. 41	Proposed
Lena	0.17	0.736	0.784	0.751	0.790	*0.791*	**0.794**
	0.24	0.817	0.844	0.820	0.843	**0.852**	*0.846*
	0.31	0.852	0.871	0.848	0.868	**0.877**	*0.872*
	0.36	0.873	0.884	0.836	0.884	**0.891**	*0.886*
Baboon	0.26	**0.539**	0.531	0.532	0.526	*0.535*	**0.539**
	0.46	*0.681*	0.666	0.666	0.660	**0.682**	0.673
	0.62	0.745	0.731	0.728	0.731	**0.750**	*0.732*
	0.76	*0.783*	0.771	0.765	0.774	**0.789**	0.776
Bridge	0.23	0.571	*0.572*	0.561	0.563	0.569	**0.573**
	0.41	*0.711*	0.703	0.693	0.694	**0.710**	0.702
	0.56	0.775	0.765	0.754	0.768	*0.780*	**0.782**
	0.69	*0.809*	0.799	0.788	0.800	**0.820**	0.803
Barbara	0.23	0.686	0.709	0.672	*0.711*	0.709	**0.719**
	0.35	0.787	0.790	0.769	*0.795*	**0.801**	*0.795*
	0.45	0.832	0.834	0.816	0.839	*0.844*	**0.845**
	0.54	0.863	0.863	0.847	0.869	*0.872*	**0.873**
Peppers	0.21	0.707	0.773	0.735	**0.780**	*0.778*	0.757
	0.31	0.771	**0.828**	0.794	0.820	*0.826*	0.792
	0.38	0.794	*0.845*	0.821	0.842	**0.847**	0.806
	0.46	0.813	*0.854*	0.836	*0.854*	**0.859**	0.819
Plane	0.18	0.775	*0.820*	0.778	0.810	0.815	**0.821**
	0.28	0.853	*0.876*	0.865	0.865	0.874	**0.878**
	0.36	0.889	0.899	0.873	0.891	*0.898*	**0.907**
	0.43	0.902	0.912	0.891	0.907	*0.913*	**0.916**

For example, Refs. 12, 40, and 41 all have higher PSNR on Lena at 0.24 bpp. However, as shown previously in Figs. 8 and 9, the proposed algorithm yields better visual results.

3.3 *Analysis of Computation Time*

To determine the effects of image size on our algorithm's runtime, we used the original images in the categorical image quality (CSIQ)[47] database (30 images) and generated different sizes of the images (256×256, 512×512, 768×768, and 1024×1024). To determine the effects of quality factor on our algorithm's runtime, for each size, we

Table 3 Average runtime (in s) of our proposed algorithm on different image sizes for each stage and totally.

	256×256	512×512	768×768	1024×1024
Deblocking	0.409	1.734	3.855	6.512
Edge regeneration	0.566	1.957	4.054	7.085
Total runtime	0.975	3.691	7.909	13.597

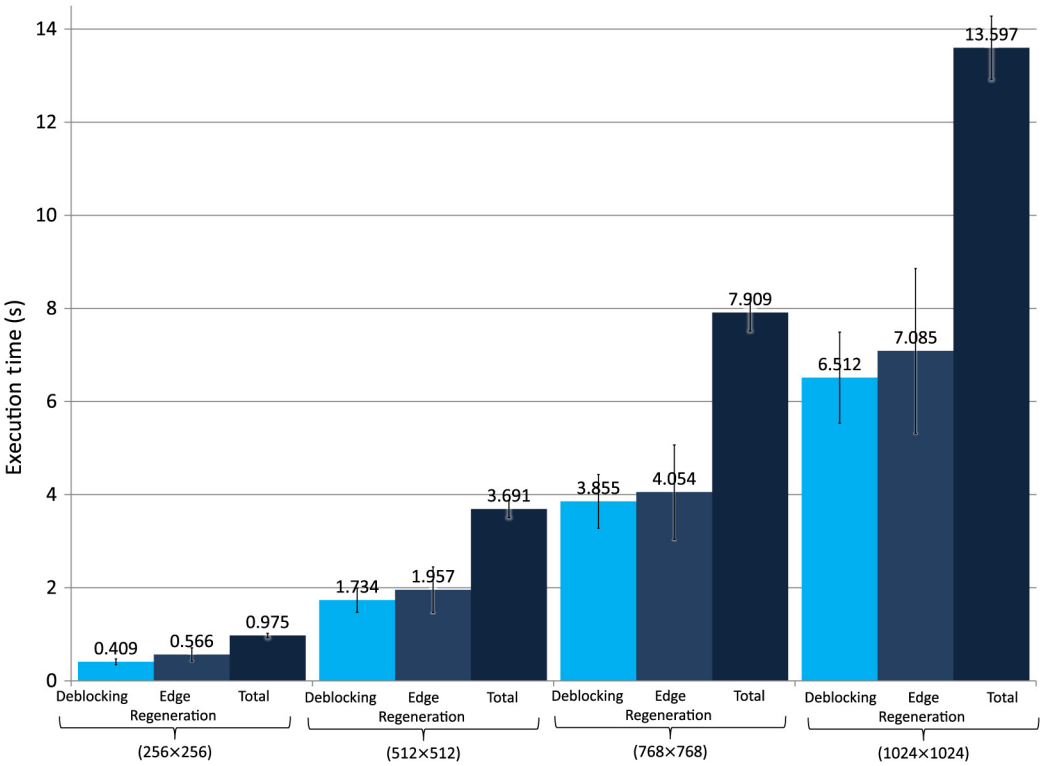


Fig. 12 Average runtime (in s) of our proposed algorithm for each stage as well as the total runtime of the algorithm on different image sizes.

used quality factors of 10, 30, and 50 to generate low-, medium-, and high-quality images, respectively. The test was performed on a modern desktop computer (AMD Phenom II ×4 965 Processor at 3.39 GHz, 4 GB RAM, Windows 7 Pro 64 bit, MATLAB 7.8.0), and the processing time was obtained by using MATLAB's time analysis tools.

Table 3 and Fig. 12 show the average runtime of our algorithm in seconds for each stage separately as well as the total runtime. Figure 13 shows the average execution time of our algorithm for images of different sizes of different qualities.

As shown in Table 3 and Fig. 12, the runtime of the algorithm increases approximately linearly with the number of pixels. As Fig. 13 shows, our proposed algorithm has consistent runtimes performance regardless of the image quality factor for each size.

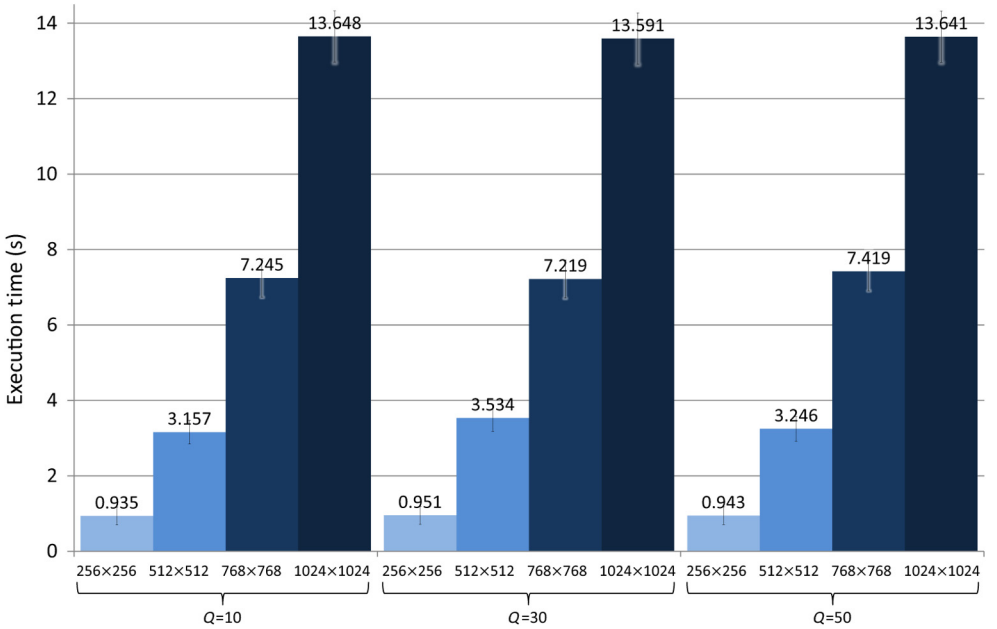


Fig. 13 Average runtime (in s) of our proposed algorithm on different qualities for each image size.

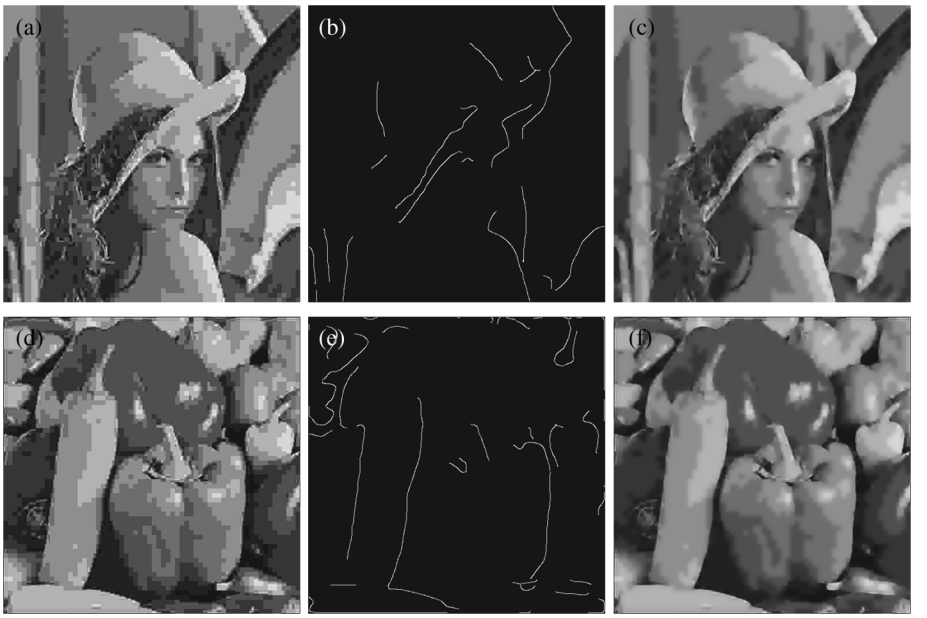


Fig. 14 Some failure cases of our algorithm. (a) *Lena* compressed at 0.13 bpp. (b) Edge map of *Lena* via Canny edge detection on (a). (c) Result after applying our algorithm on (a), (d) *peppers* compressed at 0.14 bpp. (e) Edge map of *peppers* via Canny edge detection on (d). (f) Result after applying our algorithm on (d).

3.4 *Limitations and Areas for Future Work*

Although the proposed algorithm generally works quite effectively at improving quality, there are limitations of the approach. As with most algorithms for artifact removal, the proposed approach is not quite effective for images coded at extremely low bit rates (<0.15 bpp). Generally, such low-rate images tend to have severe blocking artifacts and suffer from severe aliasing around the edges (particularly, diagonal edges), see Fig. 14.

To handle these types of failures, we would need either a better edge detector to detect strong edges accurately and/or some prior information about the map of the image's edges. Nonetheless, as shown in Fig. 14, the proposed algorithm does indeed make the images appear more natural, despite the fact that the overall quality is not significantly improved.

The proposed algorithm is also currently limited to processing only the grayscale information in an image; thus, artifacts induced via quantization of the DCT components of the chrominance channels are not addressed. Further improvements in performance could possibly be realized by designing artifact-removal techniques for such color artifacts.

4 Conclusions

In this paper, we proposed a new method to enhance the visual quality of JPEG images. Our algorithm address the three types of artifacts which are prevalent in JPEG images: blocking, and for edges blurring, and aliasing. Our proposed algorithm first reduces blocking artifacts by smoothing boundaries of blocks and via guided filtering, and it then improves the edges of the image by performing local edge regeneration. We demonstrated the performance of the proposed algorithm against several well-known JPEG artifact-removal methods. The results show that our algorithm is competitive, and in many cases outperforms, competing methods. We have made our codes and images available to the research community. A MATLAB implementation of the proposed method and the images used in this paper are available at http://vision.okstate.edu/jpgregen/jpgregen.htm.

Acknowledgments

This material is based upon the work supported by the National Science Foundation Award 1054612.

References

1. N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.* **C-23**(1), 90–93 (1974).
2. G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM* **34**(4), 30–44 (1991).
3. C. Fogg et al., *MPEG Video Compression Standard*, Springer, Boston, Massachusetts (1996).
4. J. Ostermann et al., "Video coding with h. 264/avc: tools, performance, and complexity," *IEEE Circuits Syst. Mag.* **4**(1), 7–28 (2004).
5. T. Chen, H. R. Wu, and B. Qiu, "Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts," *IEEE Trans. Circuits Syst. Video Technol.* **11**(5), 594–602 (2001).
6. J. Xu, S. Zheng, and X. Yang, "Adaptive video-blocking artifact removal in discrete hadamard transform domain," *Opt. Eng.* **45**(8), 080501 (2006).
7. H. S. Malvar and D. H. Staelin, "The lot: transform coding without blocking effects," *IEEE Trans. Acoust., Speech, Signal Process.* **37**(4), 553–559 (1989).
8. D. E. Pearson and M. W. Whybray, "Transform coding of images using interleaved blocks," *IEE Proc. Commun. Radar Signal Process., Part F* **131**(5), 466–472 (1984).
9. A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Trans. Circuits Syst. Video Technol.* **2**(1), 91–95 (1992).
10. H. S. Malvar, "Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts," *IEEE Trans. Signal Process.* **46**(4), 1043–1053 (1998).
11. Y.-Q. Zhang, R. L. Pickholtz, and M. H. Loew, "A new approach to reduce the blocking effect of transform coding [image coding]," *IEEE Trans. Commun.* **41**(2), 299–302 (1993).
12. A. W.-C. Liew and H. Yan, "Blocking artifacts suppression in block-coded images using overcomplete wavelet representation," *IEEE Trans. Circuits Syst. Video Technol.* **14**(4), 450–461 (2004).
13. H. C. Reeve, III and J. S. Lim, "Reduction of blocking effects in image coding," *Opt. Eng.* **23**(1), 230134 (1984).

14. B. Ramamurthi and A. Gersho, "Nonlinear space-variant postprocessing of block coded images," *IEEE Trans. Acoust., Speech, Signal Process.* **34**(5), 1258–1268 (1986).
15. J. D. McDonnell, R. N. Shorten, and A. D. Fagan, "An edge classification based approach to the post-processing of transform encoded images," in *1994 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-94)*, Adelaide, SA, Vol. 329, pp. V/329–V/332, IEEE (1994).
16. W. E. Lynch, A. R. Reibman, and B. Liu, "Post processing transform coded images using edges," in *1995 Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP-95)*, Detroit, Michigan, Vol. 4, pp. 2323–2326, IEEE (1995).
17. J. Hu et al., "Removal of blocking and ringing artifacts in transform coded images," in *1997 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-97)*, Munich, Vol. 4, pp. 2565–2568, IEEE (1997).
18. Y. L. Lee, H. C. Kim, and H. W. Park, "Blocking effect reduction of JPEG images by signal adaptive filtering," *IEEE Trans. Image Process.* **7**(2), 229–234 (1998).
19. H. Jiwu, Y. Q. Shi, and D. Xianhua, "Blocking artefact removal based on frequency analysis," *Electron. Lett.* **34**(24), 2323–2325 (1998).
20. J. G. Apostolopoulos and N. S. Jayant, "Postprocessing for very low bit-rate video compression," *IEEE Trans. Image Process.* **8**(8), 1125–1129 (1999).
21. T. Kasezawa, "Blocking artifacts reduction using discrete cosine transform," *IEEE Trans. Consumer Electron.* **43**(1), 48–55 (1997).
22. S. S. O. Choy, Y.-H. Chan, and W.-C. Siu, "Reduction of block-transform image coding artifacts by using local statistics of transform coefficients," *IEEE Signal Process. Lett.* **4**(1), 5–7 (1997).
23. B. Zeng, "Reduction of blocking effect in DCT-coded images using zero-masking techniques," *Signal Process.* **79**(2), 205–211 (1999).
24. B. Jeon and J. Jeong, "Blocking artifacts reduction in image compression with block boundary discontinuity criterion," *IEEE Trans. Circuits Syst. Video Technol.* **8**(3), 345–357 (1998).
25. Y.-Y. Chen, Y.-W. Chang, and W.-C. Yen, "Design a deblocking filter with three separate modes in DCT-based coding," *J. Visual Commun. Image Represent.* **19**(4), 231–244 (2008).
26. J. Singh et al., "A signal adaptive filter for blocking effect reduction of JPEG compressed images," *AEU Int. J. Electron. Commun.* **65**(10), 827–839 (2011).
27. Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images," *IEEE Trans. Circuits Syst. Video Technol.* **3**(6), 421–432 (1993).
28. R. L. Stevenson, "Reduction of coding artifacts in transform image coding," in *1993 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-93)*, Minneapolis, Minnesota, Vol. 5, pp. 401–404, IEEE (1993).
29. Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Projection-based spatially adaptive reconstruction of block-transform compressed images," *IEEE Trans. Image Process.* **4**(7), 896–908 (1995).
30. J. Chou, M. Crouse, and K. Ramchandran, "A simple algorithm for removing blocking artifacts in block-transform coded images," *IEEE Signal Process. Lett.* **5**(2), 33–35 (1998).
31. H. Paek, R.-C. Kim, and S.-U. Lee, "On the pocs-based postprocessing technique to reduce the blocking artifacts in transform coded images," *IEEE Trans. Circuits Syst. Video Technol.* **8**(3), 358–367 (1998).
32. G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Trans. Image Process.* **3**(3), 327–331 (1994).
33. R. Szeliski and H.-Y. Shum, "Motion estimation with quadtree splines," *IEEE Trans. Pattern Anal. Mach. Intell.* **18**(12), 1199–1210 (1996).
34. I. Rhee et al., "Quadtree-structured variable-size block-matching motion estimation with minimal error," *IEEE Trans. Circuits Syst. Video Technol.* **10**(1), 42–50 (2000).
35. B. A. Banister and T. R. Fischer, "Quadtree classification and TCQ image coding," *IEEE Trans. Circuits Syst. Video Technol.* **11**(1), 3–8 (2001).
36. A. S. Al-Fohoum and A. M. Reza, "Combined edge crispiness and statistical differencing for deblocking JPEG compressed images," *IEEE Trans. Image Process.* **10**(9), 1288–1298 (2001).
37. S. Liu and A. C. Bovik, "Efficient DCT-domain blind measurement and reduction of blocking artifacts," *IEEE Trans. Circuits Syst. Video Technol.* **12**(12), 1139–1149 (2002).
38. Y. Luo and R. K. Ward, "Removing the blocking artifacts of block-based DCT compressed images," *IEEE Trans. Image Process.* **12**(7), 838–842 (2003).
39. A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images," *IEEE Trans. Image Process.* **16**(5), 1395–1411 (2007).
40. G. Zhai et al., "Efficient image deblocking based on postfiltering in shifted windows," *IEEE Trans. Circuits Syst. Video Technol.* **18**(1), 122–126 (2008).
41. G. Zhai et al., "Efficient quadtree based block-shift filtering for deblocking and deringing," *J. Visual Commun. Image Represent.* **20**(8), 595–607 (2009).
42. K. He, J. Sun, and X. Tang, "Guided image filtering," in *Computer Vision–ECCV 2010*, pp. 1–14, Springer Berlin Heidelberg (2010).
43. H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Trans. Image Process.* **15**(11), 3440–3451 (2006).
44. Z. Wang et al., "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.* **13**(4), 600–612 (2004).
45. J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**(6), 679–698 (1986).
46. N. Otsu, "A threshold selection method from gray-level histograms," *Automatica* **11**(285–296), 23–27 (1975).
47. E. C. Larson and D. M. Chandler, "Most apparent distortion: full-reference image quality assessment and the role of strategy," *J. Electron. Imaging* **19**(1), 011006 (2010).

S. Alireza Golestaneh received his BSc degree in electrical engineering from the Shiraz University, Shiraz, Iran, in 2010. He is currently working toward the MSc degree in the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, Oklahoma. His research interests include image and video processing, multimedia compression, modeling, image and video quality assessment, computational vision, visual perception, computer vision, and machine learning.

Damon M. Chandler received his BS degree from Johns Hopkins University; and MS, and PhD degrees in electrical engineering from Cornell University. From 2005 to 2006, he was a postdoctoral researcher in Department of Psychology at Cornell. He is currently an associate professor in the School of Electrical and Computer Engineering at Oklahoma State University, where he heads the Laboratory of Computational Perception and Image Quality. His research interests include image processing, computational vision, and visual perception.