POST-PROCESSING FOR ARTIFACT REDUCTION IN JPEG-COMPRESSED IMAGES[1]

Jonathan K. Su and Russell M. Mersereau[2]

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250

ABSTRACT

An iterative post-processing algorithm for reducing blocking and ringing in block transform-coded images is presented. It smooths the decompressed image and uses convex projections theory to preserve the image's edges and bitstream. The algorithm exploits the coding scheme's block structure and ringing behavior. In experiments it outperforms an existing method and gives significant subjective and objective improvement over the decompressed image. Sufficient conditions for algorithm convergence are given.

1. INTRODUCTION

At low bit rates, images and image sequences compressed by block discrete cosine transform (DCT) methods (e.g., JPEG, MPEG) suffer from blocking and ringing. Post-processing to reduce these artifacts has appeal since it only requires the decompressed image. Proposed methods include block-boundary filtering [1], convex projections (CP) [2, 3], and maximum *a posteriori* (MAP) estimation [4]. The first two methods focused on reducing blocking but not ringing; the MAP approach addressed both artifacts but did not exploit the block structure of the coding scheme.

We present an algorithm that incorporates the known block structure to reduce both blocking and ringing. Our algorithm builds upon the work of Zakhor [2] and Reeves and Eddins [5].

2. MOTIVATION

2.1. Constraints

We take an image recovery approach employing CP. Let z_0 be the decompressed image. Then we specify constraints on the recovered image z^*. *Smoothness:* z^* should be (maximally) smooth to reduce blocking and ringing. *Edge preservation:* z^* should preserve edges found in z_0. *Bitstream consistency:* z^* should compress to the same bitstream as z_0. *Pixel limits:* z^* should only have values in $[0, 255]$.

For smoothing, we use a 3×3 FIR filter F. F may be linear shift-invariant (LSI) or linear shift-varying (LSV). CP imposes the other constraints.

2.2. Convex Projections Theory

In a Hilbert space with the inner-product norm, let C_1, C_2, $\ldots$, C_n be closed, convex sets with $C_0 = \cap_{i=1}^{n} C_i \neq \emptyset$. Let P_i be the projection operator onto C_i, for $i = 0, 1, \ldots, n$. (P_0 exists but may not be known.) CP theory [6] guarantees that the iteration

$$z_k = P_n P_{n-1} \cdots P_2 P_1 z_{k-1}, \quad k = 1, 2, \ldots \qquad (1)$$

converges to $P_0 z_0$, i.e., $\lim_{k\to\infty} z_k = P_0 z_0 = z^*$.

We apply CP to image recovery by expressing the constraints on z^* as C_1, $\ldots$, C_n. As long as $z^* \in C_0$, z^* is considered a satisfactory solution.

2.2.1. Closed, Convex Sets

Our remaining constraints define three closed, convex sets. To preserve edges, we limit how much $z^*(x,y)$ may deviate from $z_0(x,y)$ at each (x,y). That is, we restrict $z^*(x,y) \in [a_E(x,y), b_E(x,y)]$ to form the set C_E.

To maintain bitstream consistency, given z_0's quantization table (QT) and its quantized DCT coefficients, we can find the ranges of the original image's unquantized DCT coefficients. These ranges form C_Q. (For example, if the quantizer stepsize is 16 and the quantized DCT value is 3, then the unquantized DCT value is in [40,55].)

Finally, the interval $[0, 255]$ defines C_V.

2.2.2. Projections

C_E, C_Q, and C_V have simple projection operators:

$$P_i z(x,y) = \begin{cases} a_i(x,y), & z(x,y) < a_i(x,y); \\ b_i(x,y), & z(x,y) > b_i(x,y); \\ z(x,y), & \text{otherwise}, \end{cases} \qquad (2)$$

for $i = P, Q, V$. P_E and P_V operate in the spatial domain, P_Q in the block DCT domain.

3. ALGORITHM OUTLINE

We use variations of one basic algorithm, outlined here. Let T^m denote m consecutive applications of operator T; e.g., $T^2 z = T(Tz)$ and $T^m z = T(T^{m-1} z)$. Given z_0, we choose F, C_E, C_Q, and C_V; the way C_E, C_Q, and C_V are chosen ensures $(C_E \cap C_Q \cap C_V) = C_0 \neq \emptyset$. Then we apply

$$z_k = \lim_{m\to\infty} (P_V P_Q P_E)^m F z_{k-1}, \quad k = 1, 2, \ldots, k_{\max}. \quad (3)$$

From CP, the limit exists, so z_k is well-defined for all k.

3.1. Interpretation

With $k_{\max} = \infty$, (3) becomes a constrained optimization algorithm [5] that maximizes the smoothness of z^* under the constraint $z^* \in C_0$. If $k_{\max}$ is finite, then we have a partially-smoothed recovered image that lies in C_0.

[1]This research was supported in part by a grant from the Denon Corporation.

[2]The authors' electronic mail addresses are, respectively, su@eedsp.gatech.edu and rmm@eedsp.gatech.edu.

block type	No. of pixels of *pixel type* $N_{uniform}$	N_{edge}
uniform	50–64	0
uniform/texture	20–49	0
texture	0–19	0
edge/texture	$< 0.65(64-N_{edge})$	1–19
medium edge	$\geq 0.65(64-N_{edge})$	1–19
strong edge	0–44	20–64

Table 1: Guide to find block type from pixel types.

	pixel type			
block type	uniform	texture	edge	coastal
uniform	5	20	0	15
uniform/texture	5	10	0	15
texture	15	5	0	15
edge/texture	15	30	0	15
medium edge	10	50	0	15
strong edge	10	50	0	15

Table 2: Guide to find $\Delta(x, y)$ from pixel and block types.

3.2. Convergence

When $k_{\max} = \infty$, convergence of (3) becomes important. Define $G = \lim_{m\to\infty}(P_V P_Q P_E)^m F$; CP ensures that G is well-defined. By [7, Theorem 5.1.4], if G is nonexpansive (NE) and $\{Gz_k\}_{k=0}^{\infty}$ is bounded, then $\{Gz_k\}_{k=0}^{\infty}$ converges to a fixed point z^*. By CP, $z^* \in C_0$.

P_E, P_Q, and P_V are NE, and P_V is bounded. Thus, $\{Gz_k\}_{k=0}^{\infty}$ is bounded and F NE is a sufficient condition for convergence. Given F LSI, F is NE iff its Fourier transform $F(\omega_x, \omega_y)$ satisfies $|F(\omega_x, \omega_y)| \leq 1 \;\forall\; (\omega_x, \omega_y)$. Given F LSV with operator norm $\|F\|$, F NE iff $\|F\| \leq 1$.

If F is expansive, G may still be NE, ensuring convergence. Nonlinearity of P_E, P_Q, and P_V make this property difficult to check. In practice we have found that for F LSV, $1 < \|F\| < 1.002$ and convergence always occurred; the projections overcome F's expansivity and make G NE.

4. ALGORITHM IMPLEMENTATION

4.1. Initialization

We find C_Q directly from z_0, and C_V is always the same, so we detail the choice of C_E and F.

4.1.1. Edge-Preservation Set

Our specification of C_E preserves edges and incorporates JPEG's 8×8 block structure. We observe that ringing occurs near sharp edges but independently between blocks.

To exploit this idea, we label each pixel as edge, texture, or uniform and each 8×8 block as uniform, uniform/texture, texture, edge/texture, medium edge, or strong edge (cf. Table 1). In uniform and uniform/texture blocks, uniform pixels are already smooth but texture pixels are not: F may change the former slightly, the latter more. In texture blocks, the situation is reversed. In edge/texture blocks, non-edge pixels may reflect the true texture or ringing: F has a stronger effect on these pixels. Finally, in medium and strong edge blocks, ringing is likely in texture pixels but not in uniform pixels: F has the strongest effect on texture pixels and a moderate effect on uniform pixels.

1. Compute the local (3×3) variance $\sigma^2(x, y)$ at each pixel $z_0(x, y)$ and assign *pixel types* via

$$\textit{pixel type}(x,y) = \begin{cases} \text{uniform}, & \sigma^2(x,y) \leq T_u; \\ \text{texture}, & T_u < \sigma^2(x,y) \leq T_e; \\ \text{edge}, & T_e < \sigma^2(x,y). \end{cases}$$

   Thin the set of edge pixels. Edge pixels removed from this set become texture pixels. (In practice, $T_u = 100$ and $T_e = 900$. T_e is high enough so that blocking artifacts are not confused with edge pixels.)
2. For each block, use its pixel types and Table 1 to find its *block type*.
3. Introduce a fourth pixel type: *coastal*. A coastal pixel is a non-edge pixel that has an edge pixel among its eight nearest neighboring pixels.
4. Taking the pixel type of $z_0(x, y)$ and the block type of the block containing $z_0(x, y)$, refer to Table 2 to obtain $\Delta(x, y)$. Then $a_E(x, y) = z_0(x, y) - \Delta(x, y)$ and $b_E(x, y) = z_0(x, y) + \Delta(x, y)$.

4.1.2. Filter Selection

For F LSI, we use a 3×3 FIR filter with impulse response[3]

$$\begin{aligned} f(0,0) &= 0.2042; \\ f(\pm1,0) &= f(0,\pm1) = 0.1239; \\ f(\pm1,1) &= f(\pm1,-1) = 0.0751. \end{aligned} \tag{4}$$

This F is NE.

Our LSV F permits three types of filtering. Edge pixels are not smoothed. For non-edge, non-coastal pixels, we employ (4). Since coastal pixels lie near dissimilar regions, each coastal pixel is set to the mean of the non-edge pixels in its 3×3 neighborhood.

4.2. Iteration

The iterative part of the algorithm is simply (3). Let $d(\cdot,\cdot)$ be a closeness metric like mean-square error or maximum absolute pixel difference (MAPD). We specify tolerances δ, δ_{CP} and/or limits $k_{\max}$, $\ell_{\max}$. The algorithm becomes

```
1. Set $z_0 =$ the decompressed image and $k = 1$.
2. Compute $z' = Fz_{k-1}$.
3. Apply CP with $C_E$, $C_Q$, and $C_V$:
   (a) Set $z'_0 = z'$, $\ell = 1$.
   (b) Compute $z'_\ell = P_V P_Q P_E z'_{\ell-1}$.
   (c) If $d_{CP}(z'_\ell, z'_{\ell-1}) < \delta_{CP}$ or $\ell = \ell_{\max}$, set $z_k = z'_\ell$
       and go to Step 4. Otherwise, increment $\ell$ and
       go to Step 3a.
4. If $d(z_k, z_{k-1}) < \delta$ or $k = k_{\max}$, set $z^* = z_k$ and stop.
   Otherwise, increment $k$ and go to Step 2.
```

[3]This may be the same filter that Zakhor used. There is some confusion because Zakhor called for a 3×3 filter but then gave a 5×5 filter with 9 nonzero taps.

5. EXPERIMENTAL RESULTS

We implemented three versions of our algorithm: ($\mathcal{A}_\alpha$) F LSI, $\delta=10$; ($\mathcal{A}_\beta$) F LSV, $\delta=10$; and ($\mathcal{A}_\gamma$) F LSV, $k_{\max}=2$ (i.e., always 2 iterations). We chose T_u, T_e, and the parameters in Tables 1 and 2 by running $\mathcal{A}_\alpha$ on a decompressed version of the 256 × 256 8-bit gray image "Cameraman." The original image was compressed with standard JPEG using a QT equal to 3 × (JPEG example luminance QT) to produce considerable blocking and ringing. For comparison, we implemented Zakhor's algorithm ($\mathcal{A}_Z$): $z_k = P_Q F z_{k-1}$, $k = 1, 2, \ldots$, with F LSI, $\delta = 10$. For all algorithms, we used MAPD for $d(\cdot,\cdot)$ and $d_{CP}(\cdot,\cdot)$, and $\delta_{CP} = 10$.

We ran the algorithms on 10 other 256 × 256 8-bit gray images without changing the parameters. $\mathcal{A}_Z$ always converged after 2 or 3 iterations. $\mathcal{A}_\alpha$ and $\mathcal{A}_\beta$ converged in 2 or 3 iterations, with the CP step always converging in 2 subiterations. For each test image, $\mathcal{A}_\gamma$ gave the best results, pictorially and in peak signal-to-noise ratio (PSNR).[4] $\mathcal{A}_\beta$ outperformed $\mathcal{A}_\alpha$, which in turn usually outperformed $\mathcal{A}_Z$.

Figs. 1 and 2 show an original and a decompressed image, respectively. The result of $\mathcal{A}_Z$ appears in Fig. 3. Blocking has been reduced at the cost of blurring. Considerable ringing remains and the PSNR has worsened slightly. Figs. 4, 5, and 6 display results of $\mathcal{A}_\alpha$, $\mathcal{A}_\beta$, and $\mathcal{A}_\gamma$, respectively. All methods have reduced blocking as well as ringing. Textures have been blurred, though no more than by $\mathcal{A}_Z$. Edges remain sharp, and the PSNR has improved.

Fig. 7 shows the mean change in PSNR over the 11 images at each step in the algorithms. Upon convergence, the mean changes for $\mathcal{A}_Z$, $\mathcal{A}_\alpha$, and $\mathcal{A}_\beta$ are −0.14, +0.01, and +0.31 dB, respectively. $\mathcal{A}_\gamma$, which stops after 2 iterations, has a mean change of +0.49 dB. Filtering with F LSI always causes a PSNR drop. F LSV initially raises the PSNR, but repeated use lowers the PSNR. This PSNR decrease is likely due to oversmoothing, which explains why $\mathcal{A}_\gamma$ performs best.[5]

6. SUMMARY AND REMARKS

Smoothness, edge preservation, bitstream consistency, and pixel limits motivate the method. A filter provides smoothness, and the other constraints lend themselves to CP. Initializing C_E is the most complicated step; other techniques than Sec. 4.1.1 are possible. Given the filter, constraint sets, and stopping criteria, the algorithm is very simple and highly effective.

The algorithm can be efficiently implemented in parallel. After filtering the entire image (Step 2), we apply CP (Step 3) to each block in parallel. If $k_{\max}$ and $\ell_{\max}$ are used, computations can be done in-place because z_{k-1} is not required to check for convergence.

Finally, the algorithm can be applied to DCT video coders (MPEG, H.261) and could be adapted to reduce artifacts in subband coding.

[4] For an $N_x \times N_y$ original image z_0 and an image z, PSNR = $(N_x N_y (255)^2) \div \left(\sum\sum_{(x,y)}(z_0(x,y) - z(x,y))^2\right)$.

[5] Our algorithm was also tested with F LSV and $k_{\max} = 1$, but results were undersmoothed and still displayed ringing.

REFERENCES

[1] H. C. Reeve, III, J. S. Lim, *ICASSP '83*, pp. 1212–1215.

[2] A. Zakhor, *IEEE Tr. CAS Video Tech.*, v. 2, pp. 91–95, Mar. 1992.

[3] Y. Yang, N. P. Galatsanos, A. K. Katsaggelos, *IEEE Tr. CAS Video Tech.*, v. 3, pp. 421–432, Dec. 1993.

[4] R. L. Stevenson, *ICASSP '93*, pp. 401–404.

[5] S. J. Reeves, S. L. Eddins, *IEEE Tr. CAS Video Tech.*, v. 3, pp. 439–440, Dec. 1993.

[6] M. I. Sezan, *Ultramicroscopy*, v. 40, pp. 55–67, 1992.

[7] J. M. Ortega, W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables.* New York: Academic Press, 1970.

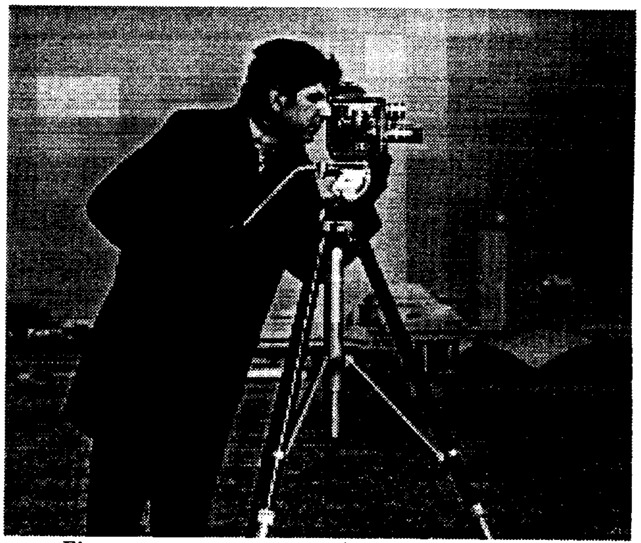

Fig. 1: 256 × 216 portion (for detail) of original 256 × 256 8-bit gray image "Cameraman"

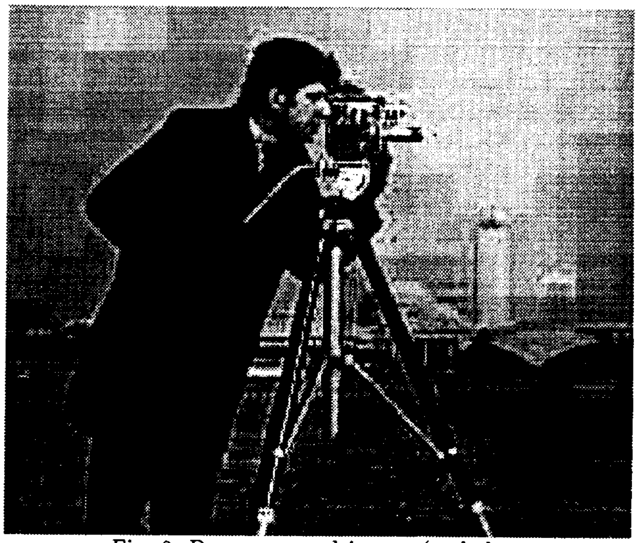

Fig. 2: Decompressed image (coded at 0.41 bits/pixel, PSNR 28.05 dB)

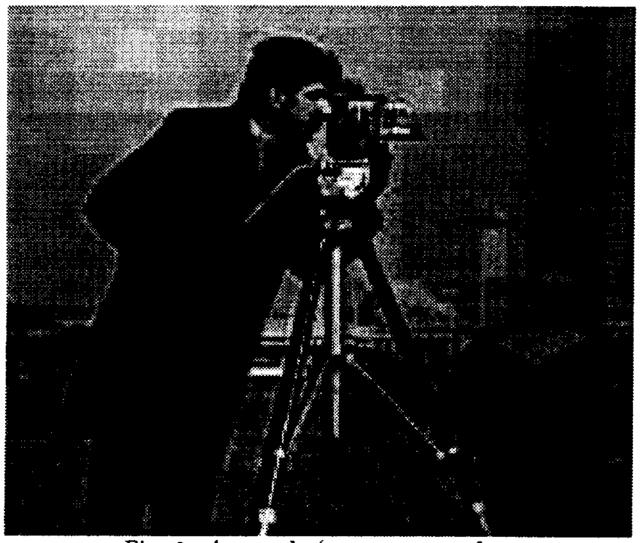
Fig. 3: $\mathcal{A}_Z$ result (convergence after 2 iterations, PSNR 27.72 dB)

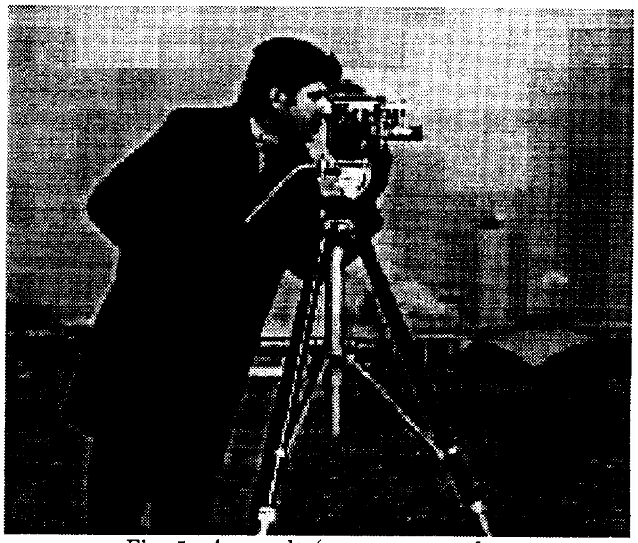
Fig. 5: $\mathcal{A}_\beta$ result (convergence after 3 iterations, PSNR 28.54 dB)

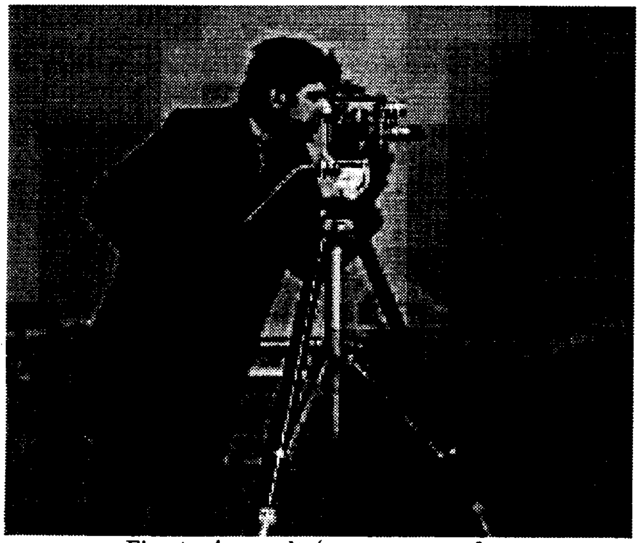
Fig. 4: $\mathcal{A}_\alpha$ result (convergence after 3 iterations, PSNR 28.26 dB)

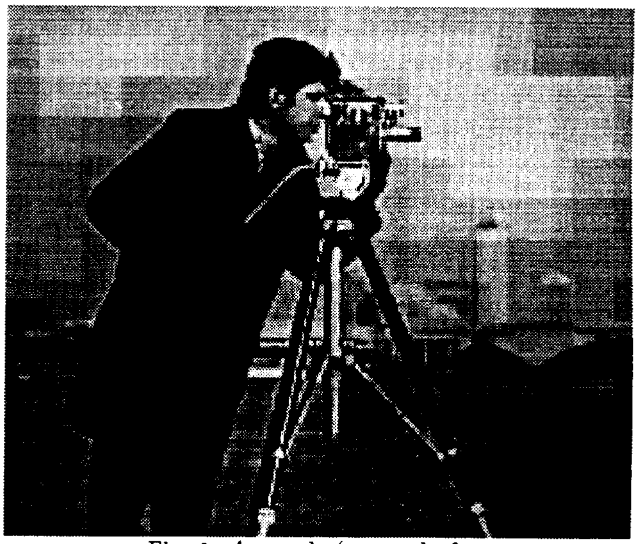
Fig. 6: $\mathcal{A}_\gamma$ result (stopped after 2 iterations, PSNR 28.59 dB)

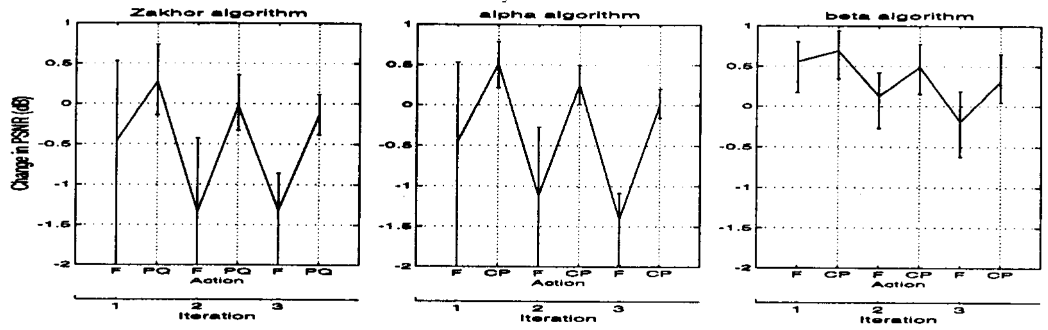

Fig. 7: Change in PSNR from simulations with 11 test images. Note that $\mathcal{A}_\gamma$ is just $\mathcal{A}_\beta$ stopped after 2 iterations. Solid line indicates the mean change in PSNR. Vertical bars indicate the range of the PSNR change results.