A Generic Postprocessing Technique for Image Compression*

S. Chen[1], *Z. He*[2] *and B.L. Luk*[3]

[1] Department of Electronics and Computer Science, University of Southampton, Highfield, Southampton SO17 1BJ, U.K.

[2] LTX Corporation, WW2-32, University Avenue, Westwood, MA 02090, USA

[3] Center for Intelligent Design, Automation and Manufacturing, City University of Hong Kong, Kowloon, Hong Kong, P.R. China

Abstract

A postprocessing technique is developed for image quality enhancement. In this method, a distortion-recovery model extracts multi-resolution edge features from the decompressed image and uses these visual features as input to estimate the difference image between the original uncompressed image and the decompressed image. Coding distortions are compensated by adding the model output to the decompressed image. Unlike many existing postprocessing methods, which smooth blocking artifacts and are designed specifically for transform coding or vector quantization, the proposed technique is generic and can be applied to all of the main coding methods. Experimental results involving postprocessing four coding systems show that the proposed technique achieves significant improvements on the quality of reconstructed images, both in terms of the objective distortion measure and subjective visual assessment.

1 Introduction

Image coding is a principal technique for reducing the requirements on bandwidth and storage capacity. The majority of current image coding methods cause distortions in reconstructed images. In practice, it is always a trade-off between the coding bit rate and the coded image quality. Generally speaking, increasing coding bit rate can improve the quality of the reconstructed image, but this is limited by channel bandwidth or storage capacity. Alternatively, postprocessing, which improves the quality of the reconstructed image after coding and decoding have been completed, can be an effective approach to achieve a good-quality reconstructed image without requesting extra bit rate. Existing postprocessing methods can roughly be divided into two

*This work is partly supported by the UK EPSRC under grant GR/M16894

categories: those employing filtering to smooth blocking artifacts in reconstructed images [1]–[5] and those formulating postprocessing as an image recovery problem [6]–[11].

The filtering approach is mainly designed to achieve better viewing quality because smoothing basically provides some artificial “make-up” on reconstructed images. This is acceptable and adequate for applications where obtaining pleasant viewing quality for entertainment is the main purpose. Since filtering also causes unwanted over-smoothing on image edges, this class of methods are not appropriate for applications which require genuinely good image quality with minimum distortions. The second class of methods rely heavily on the accuracy of *a priori* image models used and the optimisation algorithms adopted. These methods often involve adaptive filtering to limit excessive smoothing. In addition, existing postprocessing methods are specifically designed for block-based coding methods with fixed coding block sizes, such as transform coding (TC) and vector quantization (VQ), where blocking artifacts are serious sources of distortions. They cannot be applied to non-block-based predictive coding (PC), where blocking artifacts do not exist and blurred edges are main coding distortions. These methods are also impractical to use for quadtree (QT) coding [12], which has variant block sizes.

The motivation of this research is the need for a postprocessing technique which is able to correct the actual coding distortions and applicable to all the major coding systems. The key here is the ability to recover the distortion image, defined as the difference between the original and decoded images. It can be shown that main coding losses are due to edge distortions, including blurred edges and blocking artifacts (The latter can be regarded as spurious edges). This suggests that the basic task of postprocessing is to correct these edge distortions. Our distortion-recovery model consists of a visual feature extractor to extract edge information from the decoded image, and a mapping to map the visual features of the decoded image onto the distortion image. Specifically, visually important edge features are computed as multi-scale first-order derivatives. Interestingly, this gradient extractor imitates certain characteristics of visual cortex [13]–[15]. As the exact relationship between the gradient features of the decoded image and the distortion image is unknown, a neural network, referred to as the neural network visual model (NNVM), is trained to learn this relationship.

We demonstrate the advantages of the proposed postprocesing technique on four coding systems, namely TC, VQ, QT coding and PC. Our experimental results confirm that the NNVM achieves significant improvements on the quality of reconstructed images, in both the objective distortion measure and subjective viewing assessment. We also implement two typical existing postprocessing methods [1],[8] to compare them with the proposed approach in identical coding environments. The simulation results show that the proposed technique generally has better postprocessing gains over the two existing methods.

2 The proposed approach

The schematic of the proposed approach is depicted in Fig.1. In our approach, a distortion-recovery model estimates the distortion image from the decompressed image and adds this estimate to the decoded image to compensate coding distortions. Obviously, relationship between the decoded image and the distortion image is highly complicated, and it may be impossible or unadvisable to correct all the coding distortions. We start with a brief discussion on main sources of coding distortions for the three major coding methods, PC, TC and VQ. This will point to the way for building up an efficient distortion-recovery model.

2.1 Main coding distortions

The coding error of the most popular PC system, the differential pulse code modulation (DPCM), consists of the slope overload and granular noise [16]. The slope overload causes visual blurring at the edges in the reproduced image, and these edge distortions are visually more annoying than granular noise [17]. In a TC system such as the DCT coding, high spatial frequency components are either coded with very few bits or deleted completely [18]. This helps to achieve significant data compression but also causes distortions mainly at the edges in the reproduced image. The process of VQ is to find a representative codeword in the codebook for each input vector [19]. Since a codeword is the centroid (average) of all the vectors in a class, the process of averaging leads to smoothed edges in the reproduced image. Coding methods based on nonoverlapping blocks, such as TC and VQ, also give rise to the blocking artifacts, namely visible discontinuities between adjacent blocks [20]. Blocking artifacts may be viewed as exotic or spurious edges.

Perceptually, edges and contours of objects in an image belong to the most important features which characterize the picture. Errors at edges have more influence on the picture quality than errors in other image regions [21]. Therefore, the quality of the reproduced image relies very much on the fidelity of the reconstructed edges. Main coding distortions are edge distortions, including blurred edges and blocking artifacts. These edge distortions are main visual disturbances for human observers viewing images. Reducing these distortions can significantly improve visual quality of reproduced images. The distortion-recovery model in Fig. 1 is a realisation or approximation of the underlying functional relationship between the decoded image and the distortion image. It is clear that the main task of this model is to correct edge distortions. To achieve this aim, we adopt the following strategy. A decoded image of size $N \times N$ is divided into blocks of size $n \times n$, and pixels of each block are fed into a visual feature extractor, which extracts edge features of the block. These edge features are then mapped onto the corresponding block in the distortion image. We will refer to $n \times n$ as the postprocessing block size. Fig. 2 illustrates this distortion-recovery model. The basic idea is that, after learning, the output of the model will be a good estimate of the distortion image, which can then be added to the decoded image to compensate actual coding distortions.

2.2 Edge feature extractor

Edge features of an image block are extracted as multi-scale first-order directional derivatives, since gradients are known to represent edges well. Psychovisual experiments have demonstrated that stimuli in vertical and horizontal directions have more visual sensitivity than stimuli in other directions [17]. The horizontal and vertical derivatives are therefore chosen to represent edges along vertical and horizontal directions, respectively. Furthermore, the combination of these two directional derivatives can represent edges in any other directions. To calculate derivatives for an $n \times n$ block in different scales, the block is recursively divided into 4 equal-size sub-blocks until the sub-block size is reduced to 2×2. For a generic sub-block X_s of size $n_s \times n_s$, a pair of horizontal and vertical derivatives (d_h, d_v) are calculated as:

$$d_h = \sum_{i=1}^{n_s} \sum_{j=1}^{n_s/2} X_s(i,j) - \sum_{i=1}^{n_s} \sum_{j=1+n_s/2}^{n_s} X_s(i,j) \quad (1)$$

$$d_v = \sum_{i=1}^{n_s/2} \sum_{j=1}^{n_s} X_s(i,j) - \sum_{i=1+n_s/2}^{n_s} \sum_{j=1}^{n_s} X_s(i,j) \quad (2)$$

where $X_s(i,j)$ is the pixel value at position (i,j) in X_s. The outputs of the visual feature extractor, the multi-scale derivatives, can be arranged in a vector form

$$\mathbf{d} = [d_1 \ d_2 \ \cdots d_M]^T \quad (3)$$

The total number of derivatives, M, for an $n \times n$ block is determined by the formula

$$M = 2 \sum_{i=1}^{\log_2 n} \left(\frac{n}{2^i}\right)^2 \quad (4)$$

It is worth emphasizing that this edge feature extractor incorporates certain characteristics of visual cortex. It is known that there are visual feature detectors in visual cortex, called simple, complex and hypercomplex cells, which are sensitive to edge patterns of various orientations in different scales [13]–[15]. A simple cell responds maximumly to edges with particular orientation in its receptive field. A complex cell also responds maximumly to edges but has a larger receptive field. A hypercomplex cell responds mostly to edge patterns and can generalize its response over several complex cells. In the proposed edge feature extractor, an image block of certain size is divided into smaller sub-blocks. Derivatives of small sub-blocks can model visual features detected by simple cells, and derivatives of larger sub-blocks can represent features detected by complex cells. The collection of multi-scale derivatives can mimic the response of a hypercomplex cell, which is able to generalize all the details over the given area in the visual field.

Obviously, the choice of the postprocessing block size $n \times n$ has important influence on the complexity and performance of the model. Ideally, the block size should be as large as possible. However, too large a block size would make computation and storage impractical. From (4), it can be seen that the total number of derivatives increases exponentially as n increases. The size of the model therefore increases dramatically as the postprocessing block size increases. On the other hand, increasing n can provide more gradient information, and this can result in a better performance, provided that enough training data are available to train the model properly. For postprocessing of block-based coding systems, such as TC and VQ, the postprocessing block size should be larger than the coding block size, so that blocking artifacts at coding block boundaries can be corrected. Experimental results on choosing block size will be given later.

2.3 Neural network visual model

We use a one-hidden-layer neural network to learn the relationship between the edge features and distortion patterns, and call the resulting distortion-recovery model the NNVM. Specifically, the inputs to the NNVM, the edge features, are normalized to the range $(-1,\ 1)$; the hidden-layer outputs of the NNVM are given by

$$h_k = f\left(\sum_{l=1}^{M} V_{l,k} \cdot d_l + V_{0,k}\right) \quad 1 \le k \le H_n, \tag{5}$$

where H_n is the number of hidden neurons; and the outputs of the NNVM are given by

$$\hat{Y}(i,j) = \alpha \cdot f\left(\sum_{k=1}^{H_n} W_k(i,j) \cdot h_k + W_0(i,j)\right) + \beta, \quad 1 \le i,j \le n \tag{6}$$

where α and β are fixed scaling and shifting constants for mapping the model outputs onto the range of pixel values. The activation function f is the bipolar sigmoid function:

$$f(x) = \frac{2}{1+e^{-x}} - 1 \tag{7}$$

The number of the hidden-layer neurons, H_n, is determined during training using the following procedure. Given an appropriate block size $n \times n$, we start with a small hidden layer and gradually increase the size of the hidden layer until the performance stops improving. The network weights $V_{l,k}$ and $W_k(i,j)$ are learned using the stochastic gradient algorithm [22],[23]. The total number of adjustable parameters, P_{NNVM}, for the NNVM is

$$P_{NNVM} = n \times n \times (H_n + 1) + H_n \times \left(1 + 2\sum_{i=1}^{\log_2 n} \left(\frac{n}{2^i}\right)^2\right) \tag{8}$$

With $H_n = 40$ and $n \times n = 16 \times 16$, for example, $P_{NNVM} = 17,336$. To collect training data from a coding system, a training image of size $N \times N$ is compressed and then decompressed. The corresponding distortion image is obtained by subtracting the decoded image from the original image. The decoded and distortion images are divided into $n \times n$ blocks. A pair of blocks gives rise to a pair of input and desired output. As an image can only provide $\frac{N \times N}{n \times n}$ pairs of training data, many images should be used in order to collect sufficient training data samples.

3 Experimental results and comparison

The proposed postprocessing technique was applied to four coding methods, TC, VQ, QT coding and PC. The coding algorithms employed were the JPEG [24] for TC, the algorithm based on the Kohonen self-organizing feature map [25] for VQ design, the improved QT algorithm [12] for QT coding, and the algorithm using a neural network predictor [26] for PC. Sixteen images of size 512×512 with 8 bits per pixel (bpp), as shown in Fig.3, were involved in the experiment. The first eight images were used to provide training data, and the other eight images were used as test images. We also implemented two typical existing postprocessing methods, Reeve's filtering method [1] and Paek's modified projection onto convex set (PCS) method [8], to compare them with our approach in the identical TC and VQ coding environments. It should be pointed out that these two existing algorithms are impractical for postprocessing of QT and PC systems.

Fig.4 shows the postprocessing gains averaged over the 8 test images as a function of the postprocessing block size, obtained by the the NNVM with $H_n = 40$. In Fig.4, when $n \times n$

increased to 32×32, a sharp drop in peak signal to noise ratio (PSNR) gain occurred. This was because the training data set was too small, compared with the model size. The number of hidden neurons for the NNVM, H_n, also had to be determined. Fig.5 depicts the postprocessing gains averaged over the 8 test images versus the number of hidden neurons, given the postprocessing block size 16×16 for JPEG, VQ and QT coding, and 8×8 for PC. The results in Fig.4 suggest that a block size of 16×16 is adequate for postprocessing of JPEG, VQ and QT coding. The JPEG algorithm used had a standard 8×8 coding block size and the VQ employed in the study had a coding block size of 4×4. The QT had variable block sizes, depending on image activities, and majority of the blocks were 4×4 and 8×8. A postprocessing block size larger than coding block sizes ensures that the distortions at coding block boundaries can be corrected. The PC is non-block based and a smaller postprocessing block size of 8×8 appears sufficient. The results of Fig.5 suggest that $H_n = 40$ is sufficient for the NNVM to achieve adequate performance.

Tables 1 to 4 compare the postprocessing gains obtained using the NNVM and two existing algorithms for the JPEG and VQ. Tables 5 to 8 list the postprocessing gains obtained using the NNVM for the QT coding and PC. Fig.6 depicts the original and JPEG coded images of "Lena" together with the three post-improved images. Fig.7 shows the VQ coded image and the three corresponding post-improved images of "Lena." Figs. 8 and 9 compare the QT and PC coded images of "Lena" with the post-improved images obtained using the NNVM, respectively. These pictures give face portions of their corresponding images for a clearer visual evaluation. The results given in Tables 1 to 4, and Figs. 6 and 7 demonstrate that the NNVM has superior performance over Reeve's and Paek's algorithms for postprocessing of TC and VQ systems, in terms of both the objective PSNR measure and subjective visual evaluation. For the VQ case, the performance of Paek's algorithm was particularly poor, as it is designed for the TC with an 8×8 coding block size. The results shown in Tables 5 to 8, and Figs. 8 and 9 confirm that the NNVM is particularly effective for post-improving QT coding and PC systems.

The Reeve's and Paek's methods were used in the comparative study, as they represents two typical approaches of the existing postprocessing methods and can readily be implemented. For many other existing postprocessing methods, we can make some comparison using the results reported in the literature. For the sophisticated space-variant filtering method [2], the improvement given by the authors was 0.40 dB for a VQ coded "Lena" image at an original coding PSNR of 29.90 dB. The NNVM achieved a postprocessing gain of 0.80 dB for a VQ coded "Lena" image at an original coding PSNR of 30.20 dB. For Tien and Hang's postprocessing methods [3], their experimental results gave a postprocessing gain of 0.36 dB at most for TC coded images while the NNVM achieved average improvements up to 0.77 dB for TC coded images. For the adaptive α-filtering method [5], the varying postfiltering method [4] and another modified PCS method [9], the authors did not provide any numerical distortion measurements.

4 Conclusions

A generic postprocessing technique for image coding has been developed. Unlike many existing postprocessing methods which basically smooth blocking artifacts to achieve better viewing quality, the proposed technique corrects actual coding losses. Our model is inspired by the mechanism of visual perception in visual cortex. It uses gradient features to estimate coding distortions. This has been shown to be very effective in dealing with blurred edges and blocking artifacts, the two main coding distortions. An important advantage of our approach is that the same simple design can be employed for postprocessing of different coding systems. This is in contrast to existing postprocessing methods which are limited to TC or VQ. Experimental results of applying the proposed technique to four coding systems confirm that the proposed technique has better postprocessing gains and wider applications over existing methods.

References

[1] H.C. Reeve and J.S. Lim, "Reduction of blocking effect in image coding," in *Proc. ICASSP* (Boston, USA), 1983, pp.1212–1215.

[2] B. Ramamurthi and A. Gersho, "Nonlinear space-variant postprocessing of block coded images," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol.ASSP-34, pp.1258–1267, 1986.

[3] C.N. Tien and H.M. Hang, "Transform-domain postprocessing of DCT-coded images," in *Proc. SPIE*, Vol.2094, 1993, pp.1627–1638.

[4] C. Derviaux, F.X. Coudoux, M.G. Gazalet, P. Corlay and M. Gharbi, "A postprocessing technique for block effect elimination using a perceptual distortion measure," in *Proc. ICASSP*, 1997, pp.3001–3004.

[5] J.D. McDonnell, R.N. Shorten and A.D. Fagan, "Postprocessing of transform coded images via histogram-based edge classification," *J. Electronic Imaging*, Vol.6, pp.114–124, 1997.

[6] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Trans. Circuits and Systems for Video Technology*, Vol.2, pp.91–95, 1992.

[7] 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 and Systems for Video Technology*, Vol.3, pp.421–432, 1993.

[8] H. Paek, J.W. Park and S.U. Lee, "Non-iterative post-processing technique for transform coded image sequence," in *Proc. ICIP*, 1995, pp.208–211.

[9] Y.K. Lai, J. Li and C.C.J. Kuo, "Image enhancement for low bit-rate JPEG and MPEG coding via postprocessing," in *Proc. SPIE*, Vol.2727, 1996, pp.1484–1494.

[10] J. Luo, C.W. Chen, K.J. Parker and T.S. Huang, "Artifact reduction in low bit rate DCT-based image compression," *IEEE Trans. Image Processing*, Vo.5, No.9, pp.1363–1368, 1996.

[11] S. Comes, B. Macq and M. Mattavelli, "Postprocesing of image by filtering the unmasked coding noise," *IEEE Trans. Image Processing*, Vol.8, No.8, pp.1050–1062, 1999.

[12] E. Shusterman and M. Feder, "Image compression via improved quadtree decomposition algorithms," *IEEE Trans. Image Processing*, Vol.3, pp.207–215, 1994.

[13] D.H. Hubel and T.N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiology*, Vol.160, pp.106–154, 1962.

[14] D.H. Hubel and T.N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *J. Physiology*, Vol.195, pp.215–243, 1968.

[15] E.R. Kandel, J.H. Schwartz and T.M. Jessell, *Principles of Neural Science.* Prentice-Hall, 3rd edition, 1991.

[16] N.S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice-Hall, 1984.

[17] S. Comes and B. Macq, "Human visual quality criterion," in *Proc. SPIE Visual Communications and Image Processing*, Vol.1360, 1990, pp.2–13.

[18] R.J. Clarke, *Transform Coding of Images.* London: Academic Press, 1985.

[19] R.M. Gray, "Vector quantization," *IEEE ASSP Magazine*, April, 1984, pp.4–29.

[20] S.A. Karunasekera and N.G. Kingsbury, "A distortion measure for blocking artifacts of images based on human visual sensitivity," *IEEE Trans. Image Processing*, Vol.4, pp.713–724, 1995.

[21] H. Sakata, "Effect of contour components on picture quality," *Society of Motion Picture and Television Engineers Journal*, Vol.90, pp.1075–1084, 1981.

[22] S. Chen, C.F.N. Cowan, S.A. Billings and P.M. Grant, "Parallel recursive prediction error algorithm for training layered neural networks," *Int. J. Control*, Vol.51, pp.1215–1228, 1990.

[23] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: Macmillan College Publishing Company, 1994.

[24] W.B. Pennebaker and J.L. Mitchell, *JPEG Still Image Data Compression Standard.* New York: Van Nostrand Reinhold, 1993.

[25] N.M. Nasrabadi and Y. Feng, "Vector quantization of images based upon the Kohonen self-organizing feature maps," in *Proc. IEEE Int. Conf. Neural Networks*, 1988, pp.101–105.

[26] S.A. Dianat, N.M. Nasrabadi and S. Venkataraman, "A non-linear predictor for differential pulse-code encoder (DPCM) using artificial neural networks," in *Proc. ICASSP*, 1991, pp.2793–2796.

Coding image	JPEG coded PSNR (dB)	PSNR gain (dB)		
		NNVM	Reeve's	Paek's
Lena	28.85	0.81	0.69	0.36
Littlegirl	29.04	0.79	0.77	0.26
Zelda	29.98	0.80	0.73	0.68
Boats	28.23	0.81	0.52	0.28
Cablecar	27.84	0.79	0.69	0.16
Hatgirl	30.60	0.75	0.72	0.47
Kids	28.19	0.69	0.68	0.24
Soccer	27.34	0.73	0.64	0.21
Average±standard deviation		0.77±0.04	0.68±0.07	0.33±0.16

Table 1: PSNR values (dB) of JPEG coding (Quality=7) and postprocessing gains (dB). The postprocessing block size for the NNVM is 16×16 and the NNVM has 40 hidden neurons.

Coding image	JPEG coded PSNR (dB)	PSNR gain (dB)		
		NNVM	Reeve's	Paek's
Lena	31.67	0.71	0.30	0.15
Littlegirl	32.26	0.69	0.59	0.08
Zelda	33.22	0.55	0.38	0.19
Boats	31.01	0.77	0.07	0.13
Cablecar	30.96	0.71	0.23	0.07
Hatgirl	34.15	0.58	0.21	0.24
Kids	31.56	0.64	0.21	0.05
Soccer	30.76	0.68	0.54	0.05
Average±standard deviation		0.67±0.07	0.32±0.17	0.12±0.07

Table 2: PSNR values (dB) of JPEG coding (Quality=14) and postprocessing gains (dB). The postprocessing block size for the NNVM is 16×16 and the NNVM has 40 hidden neurons.

Coding image	VQ coded PSNR (dB)	PSNR gain (dB)		
		NNVM	Reeve's	Paek's
Lena	26.53	1.13	0.64	0.02
Littlegirl	27.34	1.10	0.69	0.06
Zelda	28.79	1.01	0.67	0.11
Boats	25.19	0.97	0.45	0.07
Cablecar	24.51	0.95	0.33	0.06
Hatgirl	27.37	1.17	0.61	0.03
Kids	25.34	0.93	0.37	0.05
Soccer	23.85	1.19	0.56	0.08
Average±standard deviation		1.06±0.10	0.54±0.13	0.06±0.03

Table 3: PSNR values (dB) of VQ coding (bit rate=0.25 bpp) and postprocessing gains (dB). The postprocessing block size for the NNVM is 16×16 and the NNVM has 40 hidden neurons.

Coding image	VQ coded PSNR (dB)	PSNR gain (dB)		
		NNVM	Reeve's	Paek's
Lena	30.20	0.80	0.25	0.01
Littlegirl	31.50	0.83	0.43	0.02
Zelda	32.49	0.81	0.35	0.01
Boats	29.74	0.76	0.10	-0.01
Cablecar	28.72	0.58	0.06	0.00
Hatgirl	32.51	0.76	0.17	0.00
Kids	29.73	0.65	0.03	0.00
Soccer	28.65	0.78	0.38	0.01
Average±standard deviation		0.75±0.08	0.22±0.14	0.01±0.01

Table 4: PSNR values (dB) of VQ coding (bit rate=0.5 bpp) and postprocessing gains (dB). The postprocessing block size for the NNVM is 16×16 and the NNVM has 40 hidden neurons.

Coding image	QT coded PSNR (dB)	NNVM PSNR gain (dB)
Lena	29.66	0.91
Littlegirl	29.87	0.96
Zelda	31.38	1.00
Boats	28.72	0.85
Cablecar	27.83	0.91
Hatgirl	33.29	1.14
Kids	28.38	0.92
Soccer	25.83	1.11
Average±standard deviation		0.98±0.10

Table 5: PSNR values (dB) of QT coding (bit rate=0.25 bpp) and postprocessing gains (dB). The postprocessing block size for the NNVM is 16×16 and the NNVM has 40 hidden neurons.

Coding image	QT coded PSNR (dB)	NNVM PSNR gain (dB)
Lena	32.39	0.84
Littlegirl	32.42	1.01
Zelda	33.90	0.89
Boats	31.80	0.77
Cablecar	30.59	0.93
Hatgirl	36.86	0.80
Kids	31.14	0.98
Soccer	28.32	1.23
Average±standard deviation		0.93±0.14

Table 6: PSNR values (dB) of QT coding (bit rate=0.5 bpp) and postprocessing gains (dB). The postprocessing block size for the NNVM is 16×16 and the NNVM has 40 hidden neurons.

Coding image	PC coded PSNR (dB)	NNVM PSNR gain (dB)
Lena	24.17	2.87
Littlegirl	27.51	2.01
Zelda	29.59	1.20
Boats	23.84	3.42
Cablecar	22.38	2.88
Hatgirl	23.71	3.52
Kids	23.88	2.66
Soccer	21.88	2.97
Average±standard deviation		2.69±0.71

Table 7: PSNR values (dB) of PC (coding bit=1, quantizing step=4) and postprocessing gains (dB). The postprocessing block size for the NNVM is 8×8 and the NNVM has 40 hidden neurons.

Coding image	PC coded PSNR (dB)	NNVM PSNR gain (dB)
Lena	28.42	1.43
Littlegirl	34.07	0.51
Zelda	34.79	0.38
Boats	28.39	2.32
Cablecar	25.95	2.39
Hatgirl	28.42	2.84
Kids	27.83	1.95
Soccer	26.03	2.05
Average±standard deviation		1.73±0.83

Table 8: PSNR values (dB) of PC (coding bit=2, quantizing step=4) and postprocessing gains (dB). The postprocessing block size for the NNVM is 8×8 and the NNVM has 40 hidden neurons.

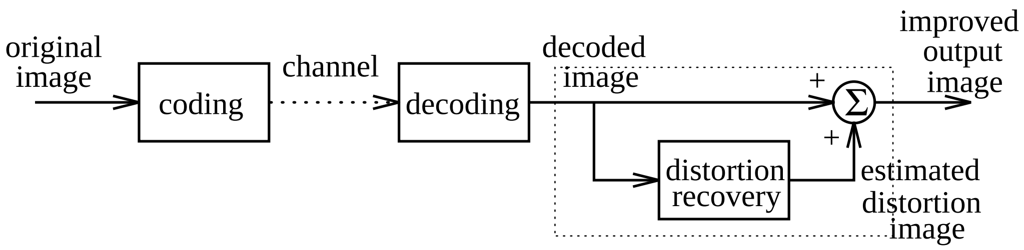


Figure 1: A generic postprocessing approach for image coding.

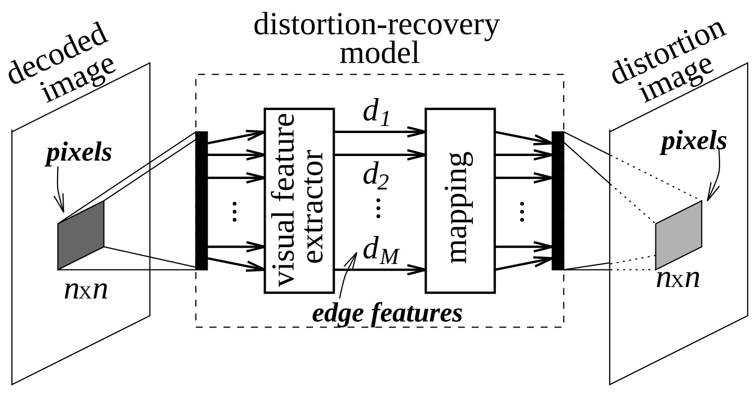


Figure 2: Schematic of the proposed distortion-recovery model. The postprocessing block size is $n \times n$ and the total number of gradient features is M.

Peppers Airplane Goldhill Lake

Announcer Cornfield Windows Yacht

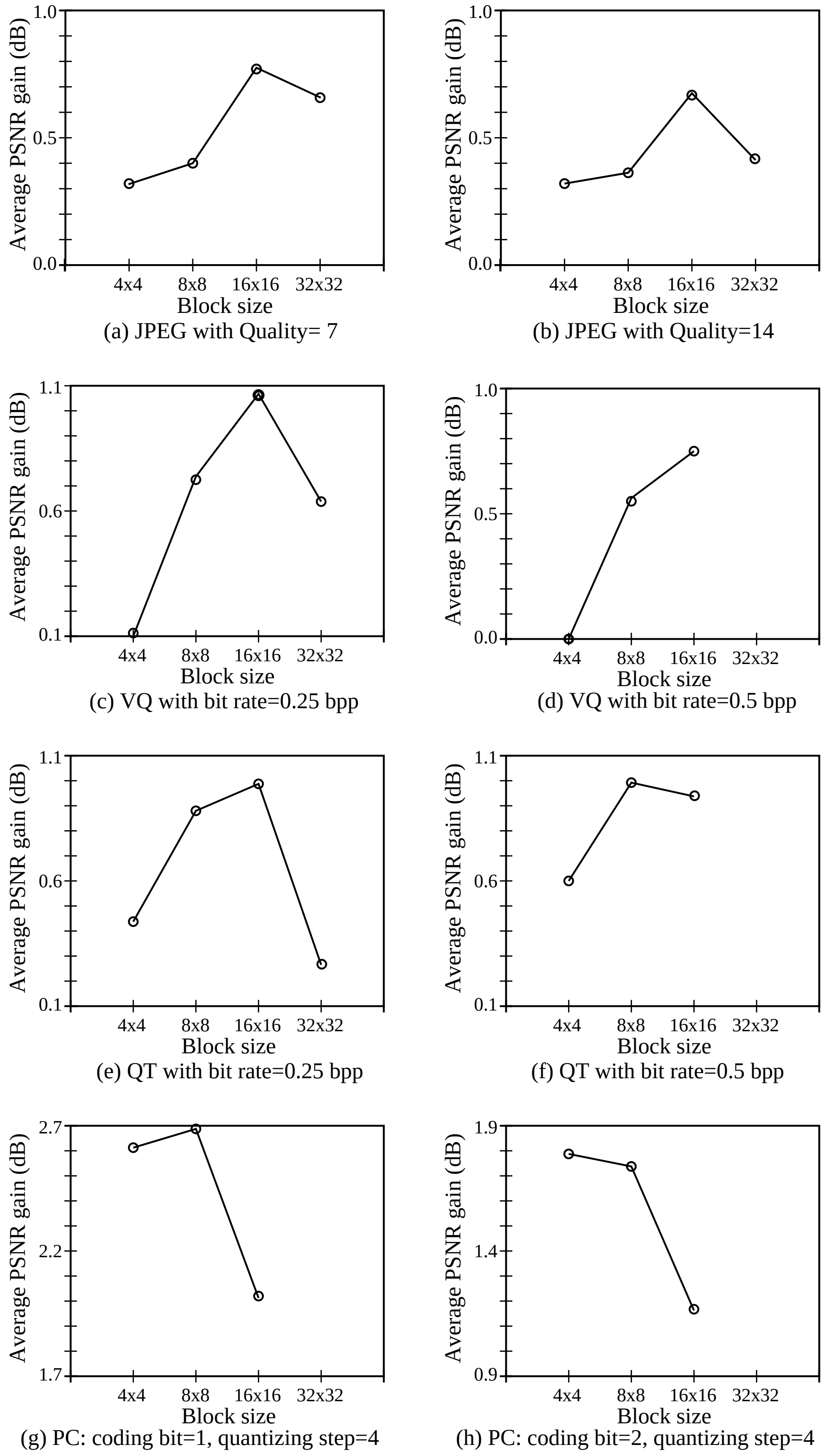
(a) Training images

Lena Littlegirl Zelda Boats

Cablecar Hatgirl Kids Soccer

(b) Testing images

Figure 3: Images used for (a) training (b) testing.



(a) JPEG with Quality= 7

(b) JPEG with Quality=14

(c) VQ with bit rate=0.25 bpp

(d) VQ with bit rate=0.5 bpp

(e) QT with bit rate=0.25 bpp

(f) QT with bit rate=0.5 bpp

(g) PC: coding bit=1, quantizing step=4

(h) PC: coding bit=2, quantizing step=4

Figure 4: Postprocessing gain averaged over 8 test images as a function of postprocessing block size. The NNVM has 40 hidden neurons.

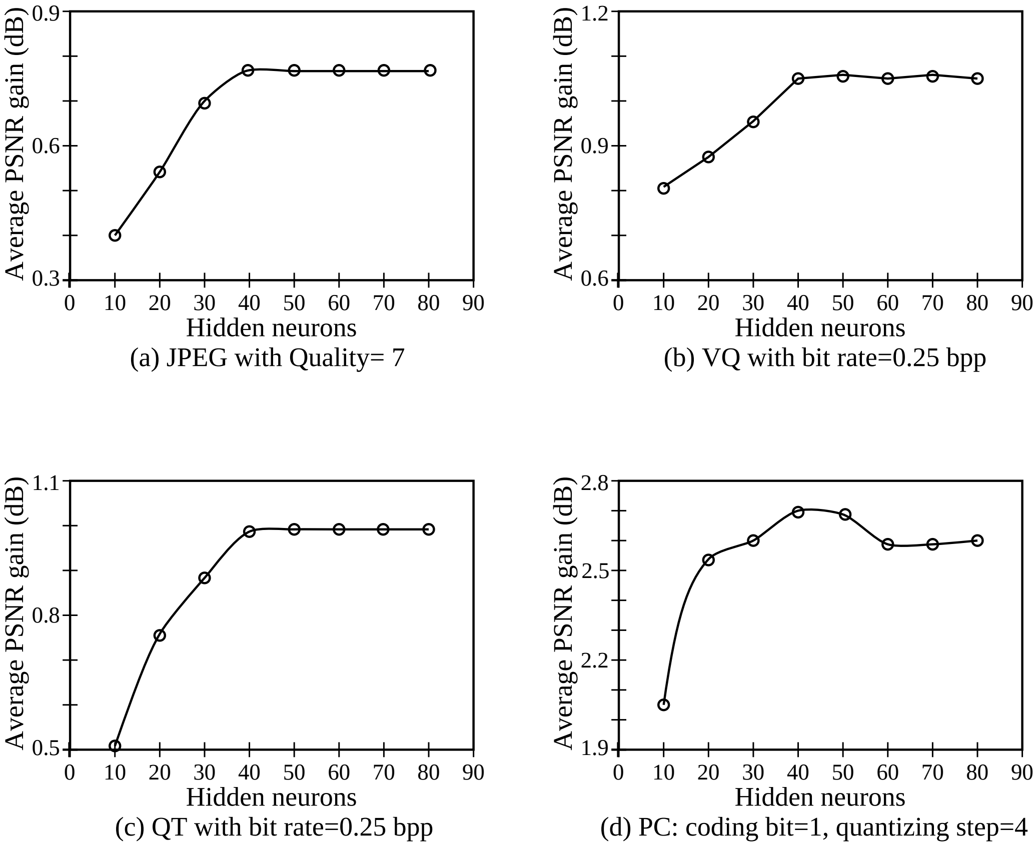
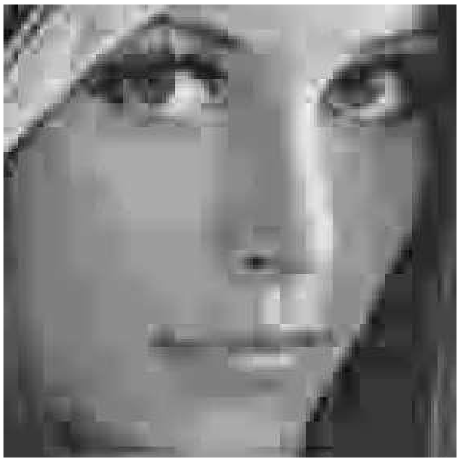


Figure 5: Postprocessing gain averaged over 8 test images as a function of the number of hidden neurons in the NNVM. The postprocessing block size is 16×16 for JPEG, VQ and QT, and 8×8 for PC.

(a) original image

(b) JPEG coded (PSNR=28.85 dB) (c) NNVM (PSNR gain=0.81 dB)

(d) Reeve's (PSNR gain=0.69 dB) (e) Paek's (PSNR gain=0.36 dB)

Figure 6: Face portions of original, JPEG coded (Quality=7) and post-improved images of "Lena."

(a) VQ coded (PSNR=26.53 dB)

(b) NNVM (PSNR gain=1.13 dB)

(c) Reeve's (PSNR gain=0.64 dB)

(d) Paek's (PSNR gain=0.02 dB)

Figure 7: Face portions of VQ coded (bit rate=0.25 bpp) and post-improved images of "Lena."

(a) QT coded (PSNR=29.66 dB) (b) NNVM (PSNR gain=0.91 dB)

Figure 8: Face portions of QT coded (bit rate=0.25 bpp) and post-improved images of "Lena."

(a) PC coded (PSNR=24.17 dB) (b) NNVM (PSNR gain=2.87 dB)

Figure 9: Face portions of PC coded (coding bit=1, quantizing step=4) and post-improved images of "Lena."