A Novel Variable Block-Size Image Compression Based On Edge Detection

Thuniki Yashwanth Reddy
Department of Electronics and Communication
IIIT Allahabad
India

Satish Kumar Singh
Assistant Professor
Department of Information Technology
IIIT Allahabad
India

***Abstract*— Image compression techniques using a fixed block-size cannot exploit the uniformity or redundancy of an image. To take advantage of this, we can vary the block size until it accommodates a certain minimum detail. Based on this, we present our novel variable block-size compression technique that estimates the detail of a block using edge detection. We consider representation of edges to be vital in the reconstruction of the image as they characterize sharp changes in the pixel values of the image. By classifying blocks into edges and nonedges using a criterion and merging only the nonedge blocks, the block-size is varied. The merged blocks are then transformed into a single block to form the source vector for quantization. This method represents the compressed image at a lower rate for the same amount of distortion. Satisfactory results are achieved when compared to fixed block-size quantization for bitrates between 0.1 and 0.6 bpp depending on the nature of the image.**

Keywords—image compression, variable block-size, vector quantization, edge detection, block-classification

I. INTRODUCTION

Vector Quantization takes the source data, groups the individual source samples into vectors and the so formed source vectors are classified into clusters. The number of source samples grouped to form a source vector is called dimensionality of that vector. On quantization of these vectors, we arrive at clusters, each represented by a reconstruction vector which is assigned to the source vectors belonging to that cluster after quantization [1]-[2].

Vector quantization can be applied on an image by dividing it into square blocks of a certain size and grouping the pixels in them to form source vectors [3]-[5]. Each block represents a source vector. All such source vectors are inputs to the quantization algorithm. The problem with using blocks of a fixed size is that there may be many blocks which are fairly similar and they only add to the redundancy of the data to be quantized. This is evident in images with uniform background or plain images.

DJ Vaisey and A Gersho took up this problem and introduced a variable block-size image coding technique which varies the size of blocks according to the local detail of the image [3]. An image, in this method, is divided into blocks of varying size using Quad-Tree segmentation. Quad-Tree segmentation divides a block of image into four uniform-sized blocks and evaluates the detail in each of them. The block is further divided into four equal sub-blocks if it meets the criterion of detail. Else, it is not divided any further. Here, a block of size, say 64×64, is divided into four blocks, each of size 32×32, thus leaving the intermediate block-sizes. The higher block-sizes are encoded using a hybrid of Transform Coding and Vector quantization, whereas the 4×4 blocks are encoded using Vector quantization.

Another method, called the Classified Vector Quantization (CVQ) [4], exploits the redundancy in an image by classifying the blocks into various classes based on the extent of presence of edges in them. Each class uses a specific codebook for coding the blocks. The codevectors of these codebooks preserve the variations present in the blocks and thus tackle the problem of edge degradation.

In our novel approach, we mitigate the redundancy which is evident in some images when using fixed block-size. We classify blocks of an image into two classes - edge and nonedge - based on edge detection technique, and then, vary the block-size by merging consecutive nonedge blocks and later transforming them to form a single block. The change in block-size due to merging is gradual as opposed to the sharp change in block-size seen in Quad-Tree segmentation. Here, we start with a basic block-size, usually 4×4 and divide an image into blocks of this size. We perform edge-detection on the image and classify each block as edge block or nonedge block based on the criterion applied to that block. Edge blocks have sharp variations among the pixel values and it is considered vital for them to be represented as they are (without any changes) for image encoding. On the other hand, a nonedge block is fairly uniform, so two or more consecutive nonedge blocks can be merged and transformed into a block whose size equals that of a single block. For example, if two consecutive horizontal $p \times p$ blocks are merged, this results in a $p \times 2p$ block, and this is to be transformed back into a $p \times p$ block. Irrespective of the number of blocks merged, they are transformed back to this base size. While this may induce distortion in the reconstructed or compressed image, it can also reduce the number of source vectors to be quantized, thereby decreasing the bitrate. This method - in contrast to fixed block-size quantization - also requires side information to be transmitted to the decoder about the edge and nonedge blocks of the image, which is known to the encoder on performing edge detection. The amount of side information is same irrespective of the codebook size. For higher codebook sizes, the share of the side information in the overall data transmitted to the decoder decreases. Therefore, the effect of the overhead due to side information is subdued for higher codebook sizes, as discussed in the further sections.

The remainder of the paper is set up in the following manner. The classification criterion for a block to be considered an edge or nonedge is described in Section II, the proposed algorithm in Section III, the performance of the proposed algorithm in Section IV, and concluding remarks in Section V.

II. BLOCK CLASSIFICATION

In this section, we formulate the block-classifying criterion based on edge detection. Edge detection produces a binary image, with each pixel being either an edge or a nonedge. We

have used Canny edge detection here [7]. Fig. 1 shows the image obtained after canny edge detection to the adjacent grayscale Lenna image.

(a) (b)

Fig. 1. (a) Grayscale Lenna image (b) After Canny edge detection on (a).

We take blocks, normally square, across the image and classify them based on the image obtained after Canny edge detection. Let N be the total number of edge pixels in a selected area of the image and T be the total number of pixels in that selected area. The subscript to these denote the selected area. We find the probability of a pixel to be an edge pixel in the image, P(e) as follows.

$$P(e) = N_{image}/T_{image} \quad (1)$$

Similarly, the probability of a pixel to be an edge pixel in a block B, $P_B(e)$ is

$$P_B(e) = N_B/T_B \quad (2)$$

We classify the block B as per the following criterion.

$$B = \begin{cases} \text{edge block, if } P_B(e) > \mu P(e) \\ \text{nonedge block, otherwise} \end{cases} \quad (3)$$

μ is a constant which lies between 0 and $1/P(e)$ inclusive.

μ=0.5 μ=1 μ=2

Fig. 2. Grayscale Lenna image after block-classification for μ=0.5, 1, and 2.

Fig. 2 shows the image after classifying 4 x 4 blocks of Fig. 1(b) into edges and nonedges for μ=0.5, 1, and 2. White blocks represent edge blocks and black blocks represent nonedge blocks. As we increase μ from 0, the number of edge blocks decrease as shown in Table I and this results in a higher number of blocks to be merged at a time and therefore higher distortion, which is discussed in section IV.

TABLE I
VARIATION IN THE NUMBER OF EDGE BLOCKS AND NONEDGE BLOCKS FOR A 512 × 512 GRAYSCALE LENNA IMAGE USING BASE SIZE 4 × 4 AS μ VARIES

μ	Edge blocks	Nonedge blocks
0.5	6880	9504
1	6112	10,272
2	4616	11,768

III. PROPOSED ALGORITHM

The proposed algorithm consists of three stages. First, we merge consecutive nonedge blocks and transform them to the size of a single block and thereby, reduce the number of source vectors to be quantized. Second, we perform vector quantization using LBG algorithm on source vectors obtained in the previous step. Third, the decoder looks into its codebook and obtains the reconstruction vectors which are replicated and filled in the blocks which are merged. We assume that the encoder and the decoder agree beforehand about the base-size of the blocks.

A. Merging and Transforming of nonedge blocks

The image is divided into blocks of a certain size called the base size. Each block is classified as edge or non-edge based on the criterion specified by Eq. 3. Consecutive nonedge blocks, each of the base size, say p × p are merged in a row-wise (left to right) fashion until an edge block or end of the row is encountered, as we traverse the image. We start again from left to right in the next row when traversing the previous row completes. Let m consecutive horizontal nonedge blocks be merged to form a block M′ of size p × mp. The block M′ is transformed into a block M″ of base size p × p as per the following equation.

$$M'' = \sum_{i=\{0,m,\ldots,m(p-1)\}} \left(\frac{1}{m}\sum_{j=\{0,1,\ldots,m-1\}} M'_{i+j}\right) \quad (4)$$

where M'_{i+j} represents the $(i+j)^{th}$ column of M′. We are doing this transformation to preserve the base size of the merged blocks. This single block of base size replaces m blocks and is used as a source vector. The edge blocks are not merged as they contain more detail than the nonedge blocks. They are individually used as source vectors.

Table II shows the decrease in number of source vectors due to merging and transforming on a 512 × 512 Lenna image. Source vectors are reduced approximately by a factor of 2 which is significant in reducing the bitrate. This is because the encoder now has to transmit the indices of fewer number of source vectors to the decoder.

The algorithm for transforming the merged image blocks is presented below.

Algorithm 1 Transforming a set of **m** blocks into a single block of base size

1. function TRANSFORM(**M′**, **m**);
   Input : Set of **m** consecutive horizontal nonedge blocks **M′**, each of base size **p × p**.
   Output: Block $\mathbf{M''_{p\times p}}$ after merging.
2. Let **M″**[**0** ... **p**] be a new array.
3. **for i = 0 to p − 1 do**
4. $\mathbf{s} = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}_{\mathbf{p\times 1}}$
5. **for j = 0 to m − 1 do**
6. $\mathbf{s} = \mathbf{s} + \mathbf{M'_{mi+j}}$
7. **end for**
8. $\mathbf{M''_i} = \mathbf{s/m}$
9. **end for**
10. **return M″**

TABLE II
COMPARISON OF NUMBER OF SOURCE VECTORS BEFORE AND AFTER MERGING AND TRANSFORMING OF NONEDGE BLOCKS FOR A 512 × 512 LENNA IMAGE USING BASE SIZE 4 × 4

μ	Number of source vectors after merging	Number of source vectors without merging	Percentage decrease in source vectors due to merging
0.5	8486	16,384	48.21
1	7896	16,384	51.81
2	6612	16,384	59.64

B. Vector Quantization

We have used the Linde-Buzo-Gray algorithm using splitting technique on the source vectors to obtain the codebook of reconstruction vectors [8]. The encoder transmits the indices of reconstruction vectors corresponding to the source vectors and side information to the decoder. As the number of source vectors is decreased due to merging, the encoder has to transmit fewer information to the decoder. We assume that the decoder has the same codebook as the encoder. In order to generate the reconstructed image, the reconstruction vector accounting for the merged blocks should replicate itself to the size of the merged blocks. This is explained in the next section.

C. Replication of reconstruction vectors

The decoder with the help of side information and the indices reconstructs the image. The decoder, using the side information, determines the number of nonedge blocks merged and transformed to form a single block. This number is called the replication factor. The decoder, using the indices, finds the reconstruction vector corresponding to this single block. This vector is to be replicated to account for the data lost in merging and transformation. Reconstruction vectors of edge blocks are treated as they are - without any replication - because they are not merged at the encoder. Let m be the replication factor and reconstruction vector R of size p × p (the base size) is to be transformed to match the size of m merged blocks p × mp by replicating each column of R m times to form R′. This way, we generate m blocks $M_i, i = 0,1, \dots, m-1$, each of base size p × p as per the following equation.

$$M_i = \left[R'_{pi} \quad R'_{pi+1} \quad R'_{pi+2} \quad \dots \quad R'_{pi+p-1}\right] \qquad (5)$$

R'_k denotes the k^{th} column of R′ and is of size p × 1. As we have reduced the size of the merged blocks at the encoder by transformation, it is only logical to replicate columns at the decoder. The algorithm of replicating a reconstruction vector $R_{p \times p}$ is presented below.

Algorithm 2 Replication of a reconstruction vector **R** by a replication factor **m**

1. function REPLICATE(**R**, **m**);
   Input : Reconstruction vector $\mathbf{R_{p \times p}}$ which is to be replicated and replication factor **m**.
   Output: Block $\mathbf{M_{p \times mp}}$ after replication.
2. Let $\mathbf{R'}[\mathbf{0} \dots \mathbf{mp}]$ and $\mathbf{M}[\mathbf{0} \dots \mathbf{m}]$ be the new arrays.
3. **for i = 0 to p – 1 do**
4. **for j = 0 to m – 1 do**
5. $\mathbf{R'}[\mathbf{mi + j}] = \mathbf{R_i}$
6. **end for**
7. **end for**
8. **for i = 0 to m – 1 do**
9. $\boldsymbol{M_i} = [\, \boldsymbol{R'}[\boldsymbol{pi}] \quad \boldsymbol{R'}[\boldsymbol{pi+1}] \quad \dots \quad \boldsymbol{R'}[\boldsymbol{pi+p-1}] \,]$
10. **end for**
11. **return M**

The algorithm of the proposed method is presented below (algorithm 3). The encoder takes an image as input and generates the codebook of reconstruction vectors. The indices of reconstruction vectors corresponding to the source vectors are transmitted to the decoder along with the side information. The decoder uses this to generate the compressed image.

IV. EXPERIMENTS, RESULTS, AND OBSERVATIONS

We discuss the performance of the proposed algorithm by experimenting it on various types of images.

A. Results over a Lenna image

We have taken a 512 × 512 grayscale Lenna image and evaluated our proposed algorithm by comparing it with fixed block-size quantization using LBG algorithm by generating initial codevectors using splitting technique. We used 4 × 4 as the base size. Tables III and IV compare the compression ratios and distortions respectively of fixed block-size quantization and the proposed method.

From Table III, the proposed method has better compression ratio except for K = 2. From Table IV, distortion increases with μ and is least for fixed block-size quantization for a given codebook size. Fixed block-size quantization has C=32.00 at K=16 which is around the same for μ=0.5 and 1 at K=64. On comparing their respective distortions, μ=0.5 has D=81.41 and μ=1 has D=84.36, whereas fixed block-size quantization has D=116.87. This implies the variable block-size quantization actually performs better for the above values. The advantages we get by the proposed method can be better explained by the Bitrate-PSNR curve.

TABLE III
COMPARISON OF COMPRESSION RATIOS BETWEEN FIXED BLOCK-SIZE AND VARIBLE BLOCK-SIZE QUANTIZATION FOR VARIOUS μ FOR A 512 × 512 GRAYSCALE LENNA IMAGE USING BASE SIZE 4 × 4

Codebook size, K	Compression ratio, C			
	Variable block-size			Fixed block-size
	μ=0.5	μ=1	μ=2	
2	84.32	86.37	91.20	128.00
4	62.87	65.18	70.83	64.00
8	50.12	52.33	57.90	42.67
16	41.67	43.72	48.96	32.00
32	35.66	37.54	42.41	25.60
64	31.16	32.89	37.41	21.33
128	27.67	29.27	33.46	18.29
256	24.89	26.36	30.27	16.00
512	22.60	23.98	27.63	14.22
1024	20.71	22.00	25.42	12.80

Algorithm 3 Proposed Variable Block-Size Quantization Algorithm

1. function ENCODER(**I**, **p**, μ);
   Input : A grayscale image ***I*** of size **u** × **v**, the base size **p** × **p** which is the size of the blocks the image is divided into, and a constant μ**.**
   Output: Indices of reconstruction vectors after quantization $\mathbf{R_i}$, side information **S**, and the image width **u**.
2. Let B be the set of **p** × **p** blocks the image **I** is divided into in a row-wise left to right fashion.
3. Let **V**, $\mathbf{V_{ne}}$ and **M** be empty arrays.
   *// **V** is the set of source vectors after merging*
4. Let **S** be an empty bit array.
   *// **S** is transmitted to the decoder as side information*
5. Let $\mathbf{m} = \mathbf{0}$.
6. Calculate **P**, the probability of an edge pixel in the image **I**.
7. **for each b** in **B do**
8. Calculate $\mathbf{P_b}$, the probability of an edge pixel in block ***b***.
9. **if** $\mathbf{P_b} > \mu\mathbf{P}$ **then**
10. $\mathbf{M}$ = TRANSFORM($\boldsymbol{V_{ne}}$, $\boldsymbol{m}$)
11. $\mathbf{m} = \mathbf{0}$
12. Append **M** and **b** to **V** in the corresponding order.
13. Empty $\mathbf{V_{ne}}$.
14. Append 1 to **S**.
15. **else**
16. **if b** is start of the row **then**
17. $\mathbf{M}$ = TRANSFORM($\mathbf{V_{ne}}$, **m**)
18. $\mathbf{m} = \mathbf{0}$
19. Append **M** to **V**.
20. Empty $\mathbf{V_{ne}}$.
21. **end if**
22. Append **b** to $\mathbf{V_{ne}}$.
23. Append 0 to **S**.
24. $\mathbf{m} = \mathbf{m} + \mathbf{1}$
25. **end if**
26. **end for**
27. Perform vector quantization on **V** using LBG-splitting technique. Let **R** be the reconstruction vectors and $\mathbf{R_i}$ be the set of indices of reconstruction vectors corresponding to the vectors in **V**.
28. **return** $\mathbf{R_i}$, **S**, **u**

1. function DECODER(R_i, S, u);
   Input : Indices of the reconstruction vectors R_i, side information S, and width of the image, u.
   Output: Compressed image I′.
2. Let I′ be an empty array.
3. $j, k, c = 0$
4. **while** $j <$ length(S) **do**
5. **if** $S[j] = 0$ **then**
6. M = REPLICATE($R[R_i[k]]$, c)
7. Append M to I′.
8. $k = k + 1$
9. Append $R[R_i[k]]$ to I′.
10. $\mathbf{c} = 0$
11. **else**
12. **if** $\left(j\%\frac{u}{p}\right) = 0$ **and** $j! = 0$ **then**
13. M = REPLICATE($R[R_i[k]]$, c)
14. $k = k + 1$
15. Append M to I′.
16. $c = 0$
17. **end if**
18. $c = c + 1$
19. **end if**
20. $j = j + 1$
21. **end while**
22. **return** I′

TABLE IV
COMPARISON OF DISTORTIONS (MSE) BETWEEN FIXED BLOCK-SIZE AND VARIBLE BLOCK-SIZE QUANTIZATION FOR VARIOUS μ FOR A 512 × 512 GRAYSCALE LENNA IMAGE USING BASE SIZE 4 × 4

Codebook size, K	Distortion, D in MSE			
	Variable block-size			Fixed block-size
	μ=0.5	μ=1	μ=2	
2	704.95	707.54	751.89	670.76
4	317.75	324.30	385.59	241.07
8	199.52	205.47	267.10	142.39
16	152.64	141.00	201.00	116.87
32	104.71	110.15	150.67	77.10
64	81.41	84.36	122.20	59.86
128	64.93	68.06	101.14	46.40
256	53.71	55.85	85.84	36.56
512	43.38	46.15	72.68	29.25
1024	36.39	38.43	64.65	23.40

We have plotted the Bitrate-PSNR curve (see Fig. 3) to compare these two methods. Bitrate is calculated in bits per pixel (bpp) and quality of reconstructed image is measured by its Peak Signal-to-Noise Ratio (PSNR). For initial rates, the effect of side information is clearly seen. The maximum size of side information in bits equals the number of blocks the image is divided into. Each block is represented either by 0 or 1, therefore representing either an edge block or a

nonedge block. The overhead due to side information for μ =1 is shown in Table V. This should be fairly similar for other values of μ as well. For lower codebook sizes, the side information is a significant part of the data transmitted to the decoder. This hinders the performance of the variable block-size quantization for lower bitrates. As the codebook size increases, the overhead per pixel decreases and thus, outperforms fixed block-size quantization for certain range of values of bpp.

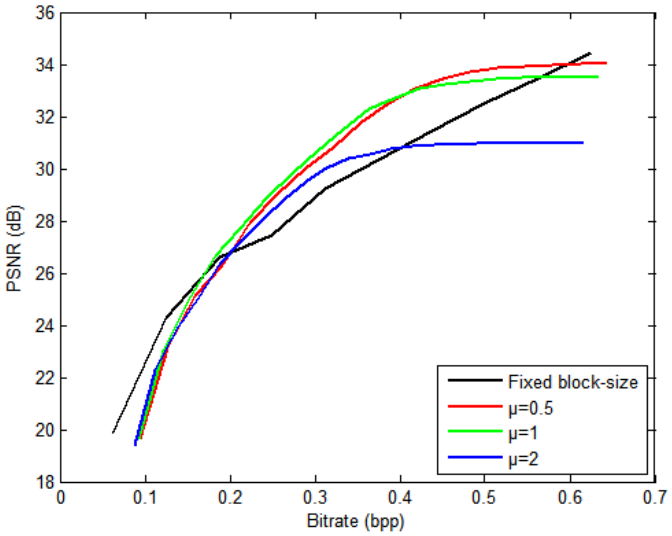


Fig. 3. Comparison of performance of fixed block-size vector quantization and variable block-size vector quantization by horizontal merging for μ=0.5, 1, and 2.

TABLE V
PERCENTAGE OF OVERHEAD PER PIXEL IN TRANSMITTING THE SIDE INFORMATION FOR μ =1

Codebook size, K	Bits per pixel used to transmit codevectors to decoder	Overhead per pixel in percentage to transmit side information
2	0.0301	207.50
4	0.0602	103.75
8	0.0904	69.17
16	0.1205	51.87
32	0.1506	41.50
64	0.1807	34.58
128	0.2108	29.64
256	0.2410	25.94
512	0.2711	23.06
1024	0.3012	20.75

For higher bpp, the performance of variable block-size quantization matches that of fixed-size and as bpp increases, our method underperforms compared to fixed block-size quantization. This is because the PSNR is saturated and cannot be improved any further by increasing codebook size or bitrate due to the loss of information due to merging. For μ=2, the range of bitrate values for which variable block-size quantization outperforms fixed-size quantization is considerably lesser than that for other values of μ. This is due to greater loss of information contained in the edges and this helps the PSNR reaching the saturation point earlier than for other values of μ. For bitrates between 0.2 and 0.5 bpp, for μ=0.5 and 1, the proposed method gives better PSNR values.

B. *Results over Cartoon, Medical, and Natural images*

We have taken 30 images belonging to each of the mentioned categories, ensuring each image of a category is of same size. Fig. 4 shows the sample images used. Cartoon images are of size 225 × 225 and 5 × 5 is used as the base size. Medical images are of size 512 × 512 and 4 × 4 is used as the base size. Natural images are of size 256 × 256 and 4 × 4 is used as the base size. This way consistency is maintained within each class.

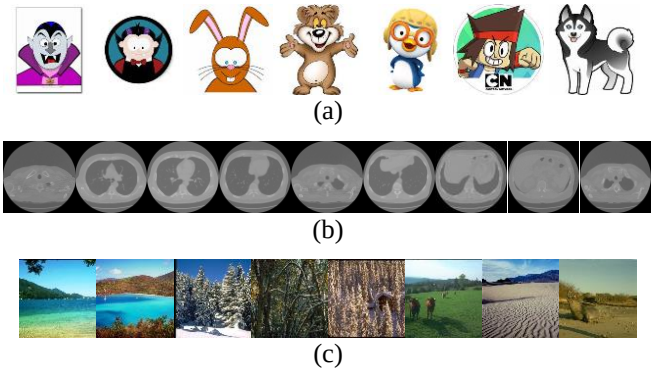


Fig. 4. Examples of (a) cartoon images, (b) medical images, and (c) natural images used for testing.

The images are converted into grayscale and then quantized according to our proposed algorithm. The average number of image vectors obtained after merging for different classes of images is shown in Table VI. On an average, natural images have the least reduction of image vectors on merging. Natural or cartoon images have the least, depending on μ. This is due to the uniformity in background and continuity in pixel values for consecutive horizontal blocks of cartoon and medical images.

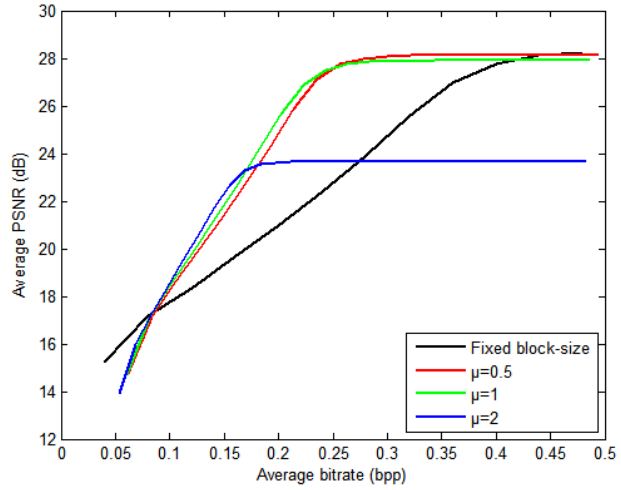
TABLE VI
AVERAGE NUMBER OF IMAGE VECTORS FOR IMAGES BELONGING TO A PARTICULAR CLASS AFTER MERGING AND TRANSFORMATION

Image class	Number of image vectors after merging and transformation			Number of image vectors without merging and transformation
	μ=0.5	μ=1	μ=2	
Cartoon	1096	1028	725	2025
Medical	9920	8924	4746	16,384
Natural	3130	2685	637	4096

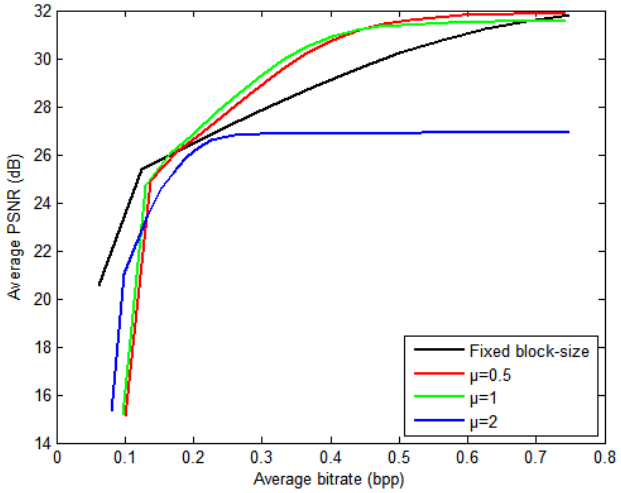
For a given value of μ, the average bits per pixel (bpp_{avg}) is calculated from the average number of vectors (V_{avg}) to be quantized for all the images belonging to the same class and the size of the codebook K as per Eq. 6. The numerator also features the number of blocks to account for the side information. For a given codebook size, the average PSNR, $PSNR_{avg}$ is calculated from the average PSNR for all the images of same class as per Eq. 7. Once we get these two parameters, the Bitrate-PSNR curve can be plotted for all classes of images (see Fig. 5).

$$bpp_{avg} = \frac{V_{avg}K + \text{number of blocks}}{\text{number of pixels}} \quad (6)$$

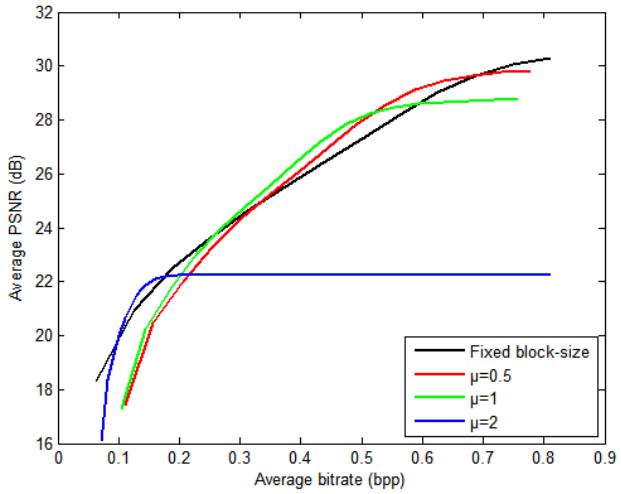
$$PSNR_{avg} = \frac{\text{Sum of PSNR for images of the same class}}{\text{Number of images of that class}} \quad (7)$$



(a)



(b)



(c)

Fig. 5. Bitrate-PSNR curves for (a) cartoon images, (b) medical images, and (c) natural images.

On examination of the curves, irrespective of the class of the image, the variable block-size quantization for μ=2 has PSNR value lower than those for other μ and the fixed block-size quantization. This and the anomaly of the Bitrate-PSNR curve at lower bpp is on lines with the result of the Lenna image and the same reason can be attributed. The number of image vectors are cut short by a factor of 3-6 for μ=2 from Table VI and several blocks containing edges are merged as they are classified nonedges. This results in several edge degradation and the PSNR reaches its saturation at a smaller bitrate. We observe that the curves, irrespective of μ, converge before the fixed block-size quantization does due to the loss of data induced upon merging. Generally, increasing the codebook size – which results in an increase of bpp - after a certain point no longer affects the PSNR. This point reaches prematurely for the proposed fixed block-size quantization.

The proposed method performs remarkably better for cartoon images. From Table VI, cartoon images have the largest reduction of the image vectors due to merging. This reduces the bitrate by a fair amount without having any significant toll on the distortion as carton images have definite edges and continuous stream of uniform pixel values. The same trend follows for medical images also. Bitrates between 0.1 and 0.4 give better results for cartoon images and between 0.2 and 0.6 for medical images. The performance gap is reduced for natural images which have the least reduction in number of image vectors compared to other classes, thus resulting in a higher bitrate. For natural images which have the highest percentage of edges, bpp between 0.3 and 0.6 give better results.

V. CONCLUSION

A novel approach to vector quantization based on edge detection and variable block-size has been proposed and the corresponding algorithm has been presented and evaluated. The algorithm includes: a block-classifying criterion to distinguish between edge and nonedge blocks based on Canny edge detection, an algorithm to merge and transform consecutive nonedge blocks at the encoder, and a replicating algorithm to generate the compressed image at the decoder. This algorithm was tested on various images and gave satisfactory results compared to the fixed block-size method for a certain range of bitrates depending on the image. For bitrates above 0.8, this method performs as good as the conventional fixed block-size quantization. This is attributed to the loss of information of the image due to merging of the nonedge blocks.

REFERENCES

[1] Gray, Robert. "Vector quantization." *IEEE Assp Magazine* 1.2 (1984): 4-29.

[2] Sayood, Khalid. *Introduction to data compression*. Newnes, 2012.

[3] A. Gersho, B Ramamurthi, "Image Coding Using Vector Quantization", *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, 1 pp. 428-431 (April 1982).

[4] Ramamurthi, Bhaskar, and Allen Gersho. "Classified vector quantization of images." *IEEE Transactions on communications* 34.11 (1986): 1105-1115.

[5] Nasrabadi, Nasser M., and Robert A. King. "Image coding using vector quantization: A review." *IEEE Transactions on communications* 36.8 (1988): 957-971.

[6] Vaisey, D. J., and Allen Gersho. "Variable block-size image coding." *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*. Vol. 12. IEEE, 1987.

[7] Canny, John. "A computational approach to edge detection." *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986): 679-698.

[8] Linde, Yoseph, Andres Buzo, and Robert Gray. "An algorithm for vector quantizer design." *IEEE Transactions on communications* 28.1 (1980): 84-95.