Tree-structured vector quantization with input-weighted distortion measures

Pamela C. Cosman, Karen Oehler, Amanda A. Heaton, Robert M. Gray

Pamela C. Cosman, Karen Oehler, Amanda A. Heaton, Robert M. Gray, "Tree-structured vector quantization with input-weighted distortion measures," Proc. SPIE 1605, Visual Communications and Image Processing '91: Visual Communication, (1 November 1991); doi: 10.1117/12.50308

SPIE.

Event: Visual Communications, '91, 1991, Boston, MA, United States

Tree-structured vector quantization with input-weighted distortion measures

Pamela C. Cosman Karen L. Oehler Amanda A. Heaton Robert M. Gray

Durand Building, Department of Electrical Engineering
Stanford University, Stanford, CA, 94305-4055

ABSTRACT

A greedy tree-growing algorithm is used in conjunction with an input-dependent weighted distortion measure to develop a tree-structured vector quantizer. Vectors in the training set are classified, and weights are assigned to the classes. The resulting weighted distortion measure forces the tree to develop better representations for those classes that are considered important. Results on medical images and USC database images are presented. A tree-structured vector quantizer grown in a similar manner can be used for preliminary classification as well as compression.

1. INTRODUCTION

Tree-structured vector quantization (TSVQ) is an image compression technique that is rapid for both the encoder and the decoder. A variable rate code can be implemented by an unbalanced tree, obtained either by growing a balanced tree and then pruning it back so that it becomes unbalanced, or by "greedily" growing an unbalanced tree directly[1,2]. In this paper we describe work in which this greedy growing algorithm is used in conjunction with an input-dependent, weighted distortion measure. Weights are assigned to the vectors in the training set according to a classification scheme; in our examples, the classification is based on brightness, texture, or on hand-labeled features in the training set. The weighted distortion causes the tree to have "growth spurts" for certain types of inputs that have been declared *a priori* to be important, and to become "stunted" for inputs that are less important.

A binary TSVQ consists of a tree with nodes labeled by candidate reproduction vectors. An input vector is compared to the labels of the two child nodes available at the root node, and the node with the minimum distortion label (the nearest neighbor) is selected. The encoder then performs a similar test for the new node's children and continues in this manner until a terminal node is reached. The label of the terminal node is the final reproduction, and the binary vector describing the sequence of encoder decisions is the codeword stored or sent to the decoder. The decoder then performs a table lookup to produce a local reproduction. A TSVQ is thus described by a tree (nodes and labels) and a distortion measure used to select the nodes in a nearest neighbor fashion (see, e.g., Gersho and Gray[3]).

2. THE GREEDY TREE GROWING ALGORITHM

A balanced TSVQ is grown one level at a time using, for example, the splitting method of the generalized Lloyd algorithm; this results in a fixed rate code[4]. To implement a pruned TSVQ (PTSVQ), a large fixed rate tree is grown and then pruned back to form a variable rate tree. Growing a fixed rate tree is unnecessarily constrained if the final code is to be variable rate. Instead of growing the tree one level at a time, we grow the tree one node at a time[1,2]. The algorithm is an extension of a common decision tree design technique[5] to VQ. An "impurity function" which measures the quality or penalty of a particular node in a tree is chosen. In our case, the impurity function is taken to be an average distortion,

$$d(j) = E[d|j] \equiv E[d(\mathbf{x}, \hat{\mathbf{x}}_j)|j], \tag{1}$$

0-8194-0742-9/91/$4.00

where $d(.,.)$ is a distortion measure and $\hat{\mathbf{x}}_j$ is the reproduction vector associated with node j. Typically,

$$d(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^t(\mathbf{x}-\mathbf{y}), \tag{2}$$

the squared error; but we will consider variations on this theme. The distortion measure $d(.,.)$ used to measure node impurity has been traditionally assumed to be the same as that used in the encoder nearest neighbor decisions. We will here consider codes where this need not be the case. This apparently unnatural decoupling of different uses of a distortion measure will be seen to provide additional freedom in code design allowing the code to better emphasize important features while maintaining a simple encoding rule.

The conditional expectation is computed as a sample average based on a training set, e.g., if $\mathcal{T}_j$ is the set of all training vectors $\mathbf{x}_k$ mapping into node j, then

$$d(j) = \frac{1}{\|\mathcal{T}_j\|} \sum_{k:x_k \in \mathcal{T}_j} d(\mathbf{x}_k, \hat{\mathbf{x}}_j) \tag{3}$$

where $\|\mathcal{T}_j\|$ is the number of vectors in $\mathcal{T}_j$. The "goodness" of a node split is defined as the decrease in node impurity:

$$\Delta d(s,j) = d(j) - p_L d(j_L) - p_R d(j_R). \tag{4}$$

Here, s is a binary test (a nearest neighbor selection in a TSVQ), $d(j)$ is the impurity (distortion) measured at node j of the tree, p_L is the proportion of the samples in node j that go to the left child, and p_R is the proportion that go to the right child. If $p(j), p(j_L)$, and $p(j_R)$ are the probabilities (as estimated by relative frequency on the training sequence) of nodes j, j_L, and j_R respectively, then $p_L = \frac{p(j_L)}{p(j)}$ and $p_R = \frac{p(j_R)}{p(j)}$.

Given a tree T, let $\tilde{T}$ stand for its leaves (or terminal nodes). Assume that we split $j \in \tilde{T}$ into two new leaves j_L and j_R. Let D and R stand for the distortion and rate, respectively, measured by T, and let D' and R' stand for the distortion and rate of the tree after j is split. Let $\Delta D = D' - D$ and $\Delta R = R' - R$ be the change in the distortion and rate, respectively, due to splitting j, and let $l(j)$ be the depth of node j. Then[1,2]:

$$D = \sum_{\substack{i \in \tilde{T} \\ i \neq j}} p(i)d(i) + p(j)d(j) \tag{5}$$

$$R = \sum_{\substack{i \in \tilde{T} \\ i \neq j}} p(i)l(i) + p(j)l(j) \tag{6}$$

$$D' = \sum_{\substack{i \in \tilde{T} \\ i \neq j}} p(i)d(i) + p(j_L)d(j_L) + p(j_R)d(j_R) \tag{7}$$

$$R' = \sum_{\substack{i \in \tilde{T} \\ i \neq j}} p(i)l(i) + p(j_L)l(j_L) + p(j_R)l(j_R), \tag{8}$$

and the ratio of the change in distortion to the change in rate due to splitting leaf j is

$$\lambda = -\frac{\Delta D}{\Delta R} = d(j) - p_L d(j_L) - p_R d(j_R) = \Delta d(s,j), \tag{9}$$

which is the goodness of split for leaf j.

As in decision tree design, we can design a TSVQ one node at a time, always splitting the node with the largest λ. For our purposes, the binary test s will be a nearest neighbor selection of node labels designed by the generalized Lloyd algorithm[6], although other tests could be used. The algorithm is "greedy" in the sense that each node is split without considering its later effect on the tree. This method results in an unbalanced tree, because the node that is split can be at any depth. There will be more codewords available to code high distortion events; this is where the tree will have been split the most.

The growing method optimally trades off rate and distortion for each new node in a greedy fashion. The resulting tree can then be pruned with the generalized Breiman, Friedman, Olshen, and Stone (BFOS) algorithm[5], an extension of an idea from classification tree design to coding. One can achieve a lower distortion for a given average rate by optimally pruning the tree with the generalized BFOS algorithm rather than by removing the nodes in the reverse order in which they were added. This is because the growing algorithm is greedy, whereas the BFOS pruning algorithm removes nodes optimally. Unbalanced trees are also able to code high distortion events at a higher resolution than can balanced trees which are limited by their initial depth.

3. USING A WEIGHTED DISTORTION MEASURE

In the original work on the greedy growing algorithm[1,2], the distortion measure was the usual squared error. If k indexes the training vectors that map into node j, then the sample average distortion measured at node j using the $d(\mathbf{x},\mathbf{y})$ of squared error is

$$d(j) = \frac{1}{\|\mathcal{T}_j\|} \sum_{k:x_k \in \mathcal{T}_j} (\mathbf{x}_k - \mathbf{y})^2, \tag{10}$$

where $\mathbf{y}$ is the centroid (with respect to d) of all the training vectors that map into node j. We propose to use instead an input-weighted distortion measure of the form

$$\rho(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^t \mathbf{W}_x (\mathbf{x}-\mathbf{y}), \tag{11}$$

where $\mathbf{W}_x$ is a strictly positive definite weighting matrix that depends on the input $\mathbf{x}$. This type of distortion measure is discussed by Gray and Karnin[7]. In our examples, $\mathbf{W}_x$ will equal $w_x\mathbf{I}$, where $\mathbf{I}$ is the identity matrix, and w_x is a weight assigned to each training vector $\mathbf{x}$ according to its classification. Training vectors which are more important according to some criterion will be assigned higher weights. In general, an input-weighted squared error can be substituted for the usual squared error in any or all of three aspects of the code and code design:

1. The weighted squared error can be used for the encoder nearest neighbor rule. This, however, can increase encoder complexity and requires the encoder to be able to classify the input vectors.
2. The weighted squared error can be used to compute the centroids giving the node labels: the centroid of a set S will be given by[7]

$$cent(S) = \left\{ \sum_{x \in S} \mathbf{W}_x \right\}^{-1} \left\{ \sum_{x \in S} \mathbf{W}_x x \right\}. \tag{12}$$

   Thus the centroid at each node will reflect the relative importance of the different types of vectors mapping into that node, i.e., the centroid will be located closer to the more important training vectors. If only one class of vectors is present at a node, then the weighted centroid will be equivalent to the usual unweighted centroid. The centroid computation is performed off-line during code design and does not affect encoder complexity.
3. The weighted error criterion can be used to define the impurity criterion implying which node to split during tree growth. The criterion to split the node yielding the largest value of

$$\lambda = -\frac{\Delta D}{\Delta R} \tag{13}$$

   will be altered compared to an unweighted distortion measure. Nodes having high concentrations of training vectors from important classes will have an inflated ΔD and therefore will split earlier and more often than they would in the unweighted case.

When using a weighting matrix of the form $w_x\mathbf{I}$, there is no difference between using the weighted squared error or the unweighted squared error for the encoder nearest neighbor rule. Every time the encoder calculates a distortion for the left child and one for the right child in order to choose its descent path, multiplying both such

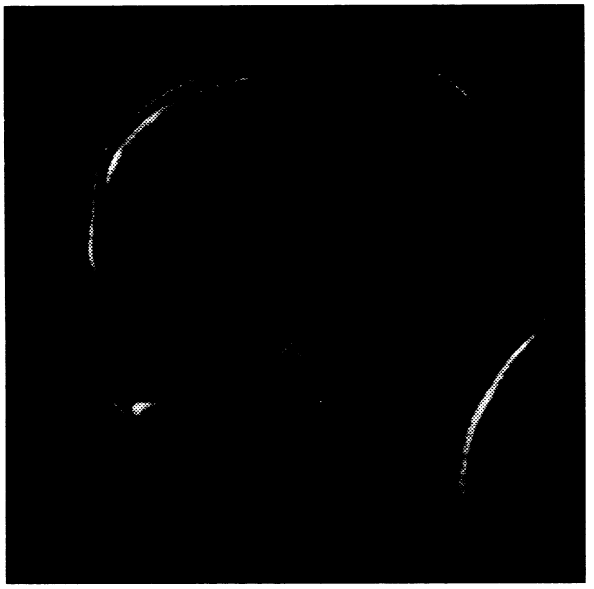

Figure 1: Original Test Image

distortions by the same weight will not affect the comparison. In this paper, the encoder rule is therefore always the unweighted squared error. If a more general weighting matrix were used or if the internal nodes of the tree themselves were associated with a class weight, then it might be useful to weight the encoder's distortion rule.

The choice of classes and their associated weights is clearly the key to making this approach useful. We present three different types of classifications: a brightness classification applied to magnetic resonance (MR) scans, a texture-based classification applied to images from the USC database, and a classification of man-made objects in satellite photos based on a hand-labeled training set.

4. CLASSIFICATION BASED ON INTENSITY

A classification based on intensity, in which bright training vectors are weighted more heavily than dark ones, is appropriate for MR brain scans because the bright parts of the image correspond typically to what is medically most important in the image. Certainly the dark background of the image is of no importance, and a high distortion can be tolerated there. The training sequence consisting of 8 MR brain scans was blocked into 2x2 pixel blocks, and each vector was assigned a weight proportional to its energy:

$$w_x = \left\lceil \frac{1}{20} \sqrt{\sum_{i=1}^{4} x_i^2} \right\rceil . \tag{14}$$

The images were 8-bit, and thus the range of possible weights was from 1 to 26. A tree was grown using the weighted distortion measure for centroid computation and for splitting. Use of the weighted distortion measure for both these purposes (2 and 3) was found to give no improvement perceptually over using it for splitting alone. The tree was grown to 2 bits per pixel (bpp) and pruned back to 0.75 bpp. The tree was evaluated on images not in the training sequence. The same training sequence without weights assigned was used to grow an unweighted unbalanced tree according to the original greedy growing algorithm. The original test image is shown in Figure 1. An example of the compressed images at 0.75 bpp produced by the two different trees is given in Figure 2. The image made from the weighted tree looks better in the bright regions (e.g., the cortex and cerebellum) which generally correspond to the diagnostically important part of these images. According to the usual unweighted squared error distortion measure,

Figure 2: Compressed images at 0.75 bpp: regular (left) and weighted distortion measure (right)

the image made from the classified tree has less distortion in the regions of the cortex and cerebellum, but more in other parts, resulting in a higher distortion overall.

5. CLASSIFICATION BASED ON TEXTURE

Due to pattern masking, the human visual system is generally less sensitive to noise in highly active or textured regions of an image. The greedy growing algorithm can be used to accomplish such a redistribution of quantization noise into regions of more texture by using a texture-based classification scheme. It is not appropriate to use MR scans for this purpose, because the most highly textured part of the image is the cortex, which is often the most important medically. Hence noise should not be placed in textured regions of MR scans. Instead we considered the USC database images for this application. The training sequence consisted of six USC database images, blocked into 4x4 vectors. The weights were assigned to the training vectors as follows: for each 4x4 block, 24 possible pairs of adjacent pixels were examined to see if the difference between them exceeded some threshold. The 24 pairs are shown in Figure 3 with curved lines. If the vector had k of the pairs exceeding the threshold, it is assigned weight $25 - k$. Thus highly textured vectors, which have many pairs exceeding the threshold, are assigned low weights. Highly homogeneous vectors are assigned large weights.

The original test image (not contained in the training sequence) is shown in figure 4. A tree was grown to 2 bpp using the weighted distortion for the splitting criterion, and then the tree was pruned back to 0.54 bpp. The compressed images at 0.54 bpp produced by a normal unweighted tree and by the weighted tree are shown in figures 5a and 5b. The compressed image from the weighted tree has less distortion in the cloud regions, where distortion is most noticeable, and it has more in the areas of trees, where the high texture masks the noise.

In another experiment, we examined whether this weighting could extend to predictive PTSVQ. Linear prediction coefficients were calculated from the training sequence using the blocks to the left, above, and diagonally to the left and above of a given block. Weights were assigned to the training sequence as before. Residual vectors were calculated from the prediction coefficients, and the weights that had been assigned to each training vector were transferred to the corresponding residual vector. A tree was grown on the residual vectors and pruned. The resulting encoded image quality was slightly better than that produced by unweighted predictive PTSVQ.

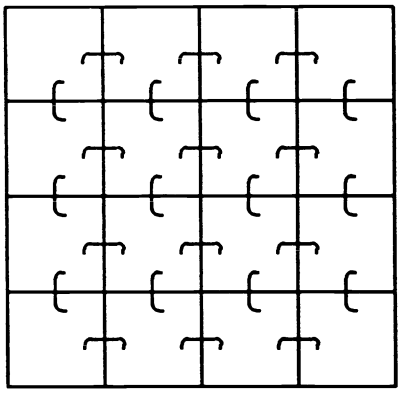

Figure 3: Differences between 24 pairs of adjacent pixels determine the texture class

6. HAND-LABELED CLASSIFICATION

In this section the emphasis is on classification rather than compression. The classification of the training set is done by hand labeling those features that are to be recognized in subsequent images. The VQ encoder described here classifies image subblocks while compressing the image. In our example aerial photographs are hand-labeled as regions of man-made and natural objects and are used to construct a VQ. Class information generated from the encoding process is used to highlight the reconstructed image to aid image analysis by human observers.

The new idea is that the classification is done simultaneously as the image is encoded. Hence, one tree-structured search is sufficient to both encode a particular subblock and to classify it into one of the previously determined categories. Stored with each codeword in the codebook is a class label representing the best class prediction for image subblocks that will be represented by that particular codeword. Thus, once the encoder selects the most appropriate codeword, the preliminary classification of the subblock at hand is simply a matter of memory lookup; no other computations are required. In effect, we are getting this classification knowledge "for free."

Images in the training sequence are divided into blocks, and each block is classified as man-made or natural by a human observer. This *a priori* knowledge is used when designing the codebook so that both small average distortion and accurate implicit classification are achieved. Several codebook design methods are possible with differing compression and classification abilities on the test images. The splitting criterion selection allows tradeoffs among compression rate, distortion, and misclassification rate.

The training set consisted of 5 images provided by ESL, Inc. The images were 512x512 pixels of 8 bit grayscale consisting of aerial photography of the San Francisco Bay area. Each 16x16 pixel subblock in the training set was assigned to be either man-made or natural based on the perceptions of a human observer. While the codebook construction and image encoding were carried out using 4x4 pixel subblocks as vectors, the training vectors were classified in 16x16 subblocks to simplify the task of the human classifier (even using the 16x16 subblocks, over 5000 decisions had to be made.) Using a training set classified with finer resolution might improve classification ability.

The best predictive class for a given codeword in a VQ codebook was determined by a majority vote of the *a priori* class assignments of the training vectors represented by that codeword. This makes sense if the costs of misclassifying the image subblocks of the various classes are equal. However, if the error costs are unbalanced, then a more sophisticated assignment function would be appropriate.

Codebooks were generated using three different splitting criteria:

Criterion 1 Ignore the classification information and split the node with the largest goodness of split as defined in Equation 4 where the distortion measure is mean-squared error (see Equation 10.) This design yields an ordinary VQ for comparison.

Criterion 2 Split the node that has the greatest percentage of misclassified training vectors, i.e. split

Figure 4: Original Test Image

Figure 5: Compressed images at 0.56 bpp: a) regular and b) weighted distortion measure

the node with the largest value of

$$\lambda_2 = \frac{\text{(Number of misclassified vectors in node)}}{\text{(Total number of vectors in node)}}. \tag{15}$$

This corresponds to measuring impurity by the Hamming distance between the node class, c_j, and the hand-labeled class, $c(x)$. Thus, if $d_H(x,y) = 0$ if $x = y$ and $d_H(x,y) = 1$ if $x \neq y$, then the impurity of node j is given by

$$d(j) = d_H(j) \equiv \frac{1}{\|\mathcal{T}_j\|} \sum_{k:x_k \in \mathcal{T}_j} d_H(c(\mathbf{x}_k), c_j) \tag{16}$$

as in Equation 3, and the node with the highest such impurity is split. Thus the encoder nearest neighbor mapping and the centroid reproduction levels are chosen to minimize squared error, but the tree is grown to reduce classification error.

Criterion 3 Split the node that has the greatest number of misclassified training vectors. This is equivalent to splitting the node with the largest impurity defined by the partial Hamming distortion $d(j) = \|\mathcal{T}_j\| d_H(j)$.

Sets of codebooks having either the same rate or same number of nodes were constructed to allow for comparison. In general, the first splitting criterion provided the lowest mean squared error in the encoded image at the expense of relatively poor classification ability. The latter two splitting methods provided poorer encoded images (much more blocky in appearance) but better classification ability. Criterion 3 classified more vectors as man-made than criterion 2. For a given number of nodes, criterion 2 produced a higher rate code than 1, and 3 higher than 2. Likewise, for a given rate the tree structure produced by criterion 1 has substantially more nodes than criterion 2, and 2 more nodes than 3. Choosing the splitting criterion involves a tradeoff between compression and classification quality; some splits serve one purpose better than the other.

Images outside the training sequence were encoded with the resulting codebooks. The compressed images are best viewed on a color monitor so that classification information can be indicated by color superimposed on the grayscale compressed image. For example, the classification information can be viewed by highlighting subblocks that were deemed man-made by the encoder with a red tint and subblocks deemed natural with a green tint. Such a contrast can make the natural and man-made features of the image easier for a human viewer to differentiate. Although this information is not as amenable to grayscale display, the images shown here reflect the accuracy of the classification encoder. The classification ability was modest; at 0.5 bpp the best classification encoder still had 25% misclassification error on the training sequence. However, this large error was partly due to the quality of the training sequence. The hand-labeling was affected by the human observer's resolution and consistency limitations.

Experimental data is shown for images encoded and classified with a codebook grown using criterion 2. Figure 6 shows the original test image (not in the training set) that was used to evaluate the different codebooks. Figure 7a shows the image after compression at 0.46 bpp. Figure 7b shows the subblocks in the compressed image that the encoder classified as natural, the man-made subblocks are replaced by solid white subblocks.

7. CLASSIFIED VECTOR QUANTIZATION

This work is related to the concept of a classified vector quantizer (CVQ)[8]. In CVQ, the training sequence is typically divided into classes and separate vector quantizers, either full-search or balanced or unbalanced trees, are developed for each class. The test image to be encoded is blocked into vectors, and each vector is classified in the same manner as was done for the training sequence vectors. The appropriate quantizer for that class is used for that vector, and the encoder must then transmit to the decoder both the index of the tree and the index corresponding to the path of the vector through that tree. CVQ allows one to expend more bits (e.g. grow a larger tree) for those classes that one judges important. One pays the price for this both with the extra bits (side information) needed to state which quantizer is being used, and with the added complexity needed to classify the vector at the encoder. An advantage is that one only has to search through the sub-codebook for that class, and not through the full codebook. There exist several variations upon this basic theme. An advantage of our method is that the incoming test vectors do not need to be classified, and no side information to specify the codebook is required; the classification is built into the tree.

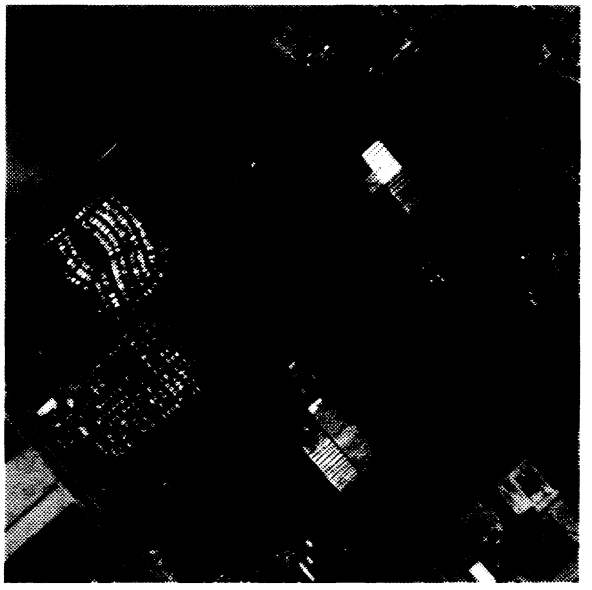

Figure 6: Original aerial image

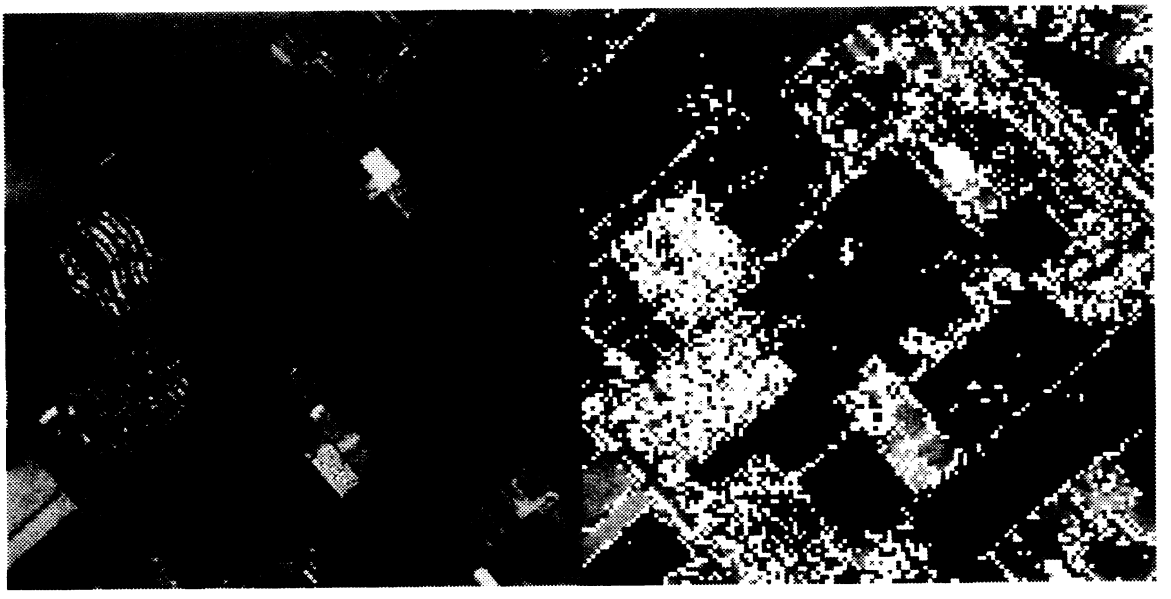

Figure 7: a) Image compressed at 0.46 bpp using compression/classification encoder, and b) Natural sublocks using compression/classification encoder. Man-made subblocks are replaced by solid white subblocks.

8. CONCLUSIONS

We have used an input-dependent, weighted distortion measure in conjunction with a greedy growing algorithm to grow unbalanced tree-structured vector quantizers. The weighting based on brightness for medical images caused more quantization noise to be hidden in the dark parts of the image, and less in the bright parts. This would seem to be useful for certain images in which the useful information is known *a priori* to be contained in the bright areas. A weighting based on texture, in which quantization noise was hidden in the more textured areas of the image, was used on an image of an outdoor scene. In both examples the weighted tree significantly improved the perceptual compressed image quality although it reduced the SNR as defined according to the standard unweighted squared error. Hand-labeling of aerial images was used to grow tree-structured vector quantizers which combined moderate classification ability with encoding, useful in enhancing the reconstructed image. The classification schemes used so far have classified each training vector without regard to its context in the training image. It might be possible to use edge detectors or texture classifiers that operate on a more global scale to classify the training vectors. As vector quantizers have frequently been accused of causing blockiness at edges[9], a vector quantizer that devotes a larger portion of the tree to edges might help with this problem. Perhaps other types of images would yield their own appropriate classification types, so that important structures in the image could be enhanced at the expense of the other parts.

9. ACKNOWLEDGEMENTS

This work was supported in part by the National Institutes of Health under Grant CA49697-02 and by Graduate Fellowships from the National Science Foundation and the Office of Naval Research.

10. REFERENCES

1. E. A. Riskin and R. M Gray, "A Greedy Tree Growing Algorithm for the Design of Variable Rate Vector Quantizers," *IEEE Transactions on Signal Processing,* in press.

2. E. A. Riskin and R. M Gray, "A Greedy Tree Growing Algorithm for the Design of Variable Rate Vector Quantizers," Presented at the 1990 Picture Coding Symposium, Cambridge, MA, 1990.

3. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression,* Kluwer Academic Publishers, Boston, in press.

4. A. Buzo, A. H. Gray Jr., R. M. Gray, and J. D. Markel, "Speech Coding Based upon Vector Quantization," *IEEE Transactions on Acoustics, Speech and Signal Processing,* 28:562-574, October 1980.

5. L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification and Regression Trees,* Wadsworth and Brooks, Monterey, CA, 1984.

6. Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications,* 28:84-95, January 1980.

7. R. Gray and E. Karnin, "Multiple Local Optima in Vector Quantizers," *IEEE Transactions on Information Theory,* 28(2):256-261, March 1982.

8. B. Ramamurthi and A. Gersho, "Classified Vector Quantization of Images," *IEEE Transactions on Communications,* 34(11):1105-1115, November 1986.

9. A. Gersho and B. Ramamurthi, "Image Coding Using Vector Quantization," *Proceedings ICASSP,* IEEE Acoustics Speech and Signal Processing Society, 1982.