NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

IMAGE COMPRESSION SCHEME FOR NETWORK TRANSMISSION

by

Lim Seng

March, 1995

Thesis Advisor: Murali Tummala
Second Reader: Chin-Hwa Lee

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE : IMAGE COMPRESSION SCHEME FOR NETWORK TRANSMISSION

5. FUNDING NUMBERS

6. AUTHOR LIM SENG

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

In this thesis a scheme is proposed for still image compression for realtime transmission over a packet switched network. A layered image compression scheme based on the quad tree decomposition technique (QTD) is developed. The image is decomposed into three layers, where layers are added to a base image to incrementally improve the quality of the recontructed image. The performance of the QTD is enhanced by adding a DCT algorithm with adaptive bit allocation and a region of interest methodology. The quad tree is then extended to a nona tree whereby pixel blocks of size 3 x 3 are processed instead of 2 x 2. Results are presented to provide a visual subjective perception of the proposed scheme. Depending on the number of layers transmitted, the compression gains range from 60:1 to 16:1. It is shown that the proposed scheme exhibits graceful degradation of the reconstructed image and some error tolerant capabilities.

14. SUBJECT TERMS: Video Compression, Quad Tree Decomposition, Discret Cosine Transform.

15. NUMBER OF PAGES 68

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-10298-102

Approved for public release; distribution is unlimited.

IMAGE COMPRESSION SCHEME FOR NETWORK TRANSMISSION

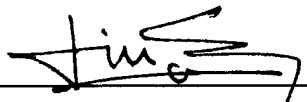
Lim Seng
Ministry of Defense Singapore
B.S., Institute de Chimie, Physique & Informatique de Lyon - July 1986

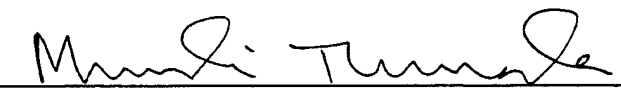
Submitted in partial fulfillment
of the requirements for the degree of

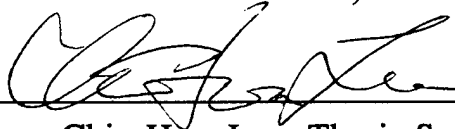
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

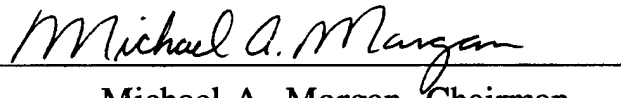
from the

NAVAL POSTGRADUATE SCHOOL
March 1995

Author: 
Lim Seng

Approved by: 
Murali Tummala, Thesis Advisor


Chin-Hwa Lee, Thesis Second Reader


Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

In this thesis a scheme is proposed for still image compression for real-time transmission over a packet switched network. A layered image compression scheme based on the quad tree decomposition technique (QTD) is developed. The image is decomposed into three layers, where layers are added to a base image to incrementally improve the quality of the recontructed image. The performance of the QTD is enhanced by adding a DCT algorithm with adaptive bit allocation and a region of interest methodology. The quad tree is then extended to a nona tree whereby pixel blocks of size 3 x 3 are processed instead of 2 x 2. Results are presented to provide a visual subjective perception of the proposed scheme. Depending on the number of layers transmitted, the compression gains range from 60:1 to 16:1. It is shown that the proposed scheme exhibits graceful degradation of the reconstructed image and some error tolerant capabilities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

The thesis objective is to propose an image compression scheme for real-time transmission over a packet switched network. The goal is to minimize the effects of errors due to traffic congestion and channel noise. To achieve this, an image compression scheme based on the quad tree decomposition technique is proposed. The scheme exhibits graceful degradation of the reconstructed image and some error tolerant capabilities.

The image is decomposed into multiple layers where layers added to a base image incrementally improve the image recontruction quality. This layering approach permits graceful degradation in the event of packet loss caused by traffic congestion. As the network's traffic fluctuates, the number of layers of the image to be transmitted adapts accordingly. When the traffic is congested, a base image of acceptable fidelity is transmitted at a reduced data bandwidth. As the traffic eases off, additional layers of details are transmitted to build up the image progressively. To counter the errors due to packet loss caused by either channel noise or burst errors, the packets are interleaved in a pseudo random fashion before transmission. This is to mitigate the localized patchy visual effect of such errors.

The thesis is organized as follows. Chapter II of the thesis describes the layering of the image. This is achieved using the quad tree decomposition methodology. The quad tree decomposition breaks down the image into several subimages with their regions of activity indicated. The scheme can be applied to all image types of various activity. Having formed the first layer of the image, the error image is computed from the difference of the base image and the original image. The error image constitutes the subsequent layers of subimages. Details of the quad tree decomposition algorithm and modifications for improvement are presented in Chapter II.

Chapter III describes the incorporation of adaptive discret cosine transform (DCT) into the quad tree decomposition so as to achieve better compression gain. The quad tree decomposition segments an image into regions of activity. Depending on the intensity of the activity of the region, the number of DCT coefficients is chosen. For regions of high activity, more DCT coefficients are used compared to those of low activity. Designated bit allocation masks are used to perform the DCT coefficient quantization and truncation.

Chapter IV extends the algorithm to interframe compression for full motion video. Here, the video frame rate is reduced from 30 frames per second to 15 frames per second. Additionally, by taking into account the interframe redundancy between two consecutive images, only the base image of the first frame is computed and transmitted. The error image of the second frame is derived from the difference between its original image and the first frame's base image. This is to gain further compression and computational speed.

Chapter V presents the conclusions. A summary of the thesis findings is provided. Possible benefits and applications of the proposed video compression scheme are also discussed.

II. QUAD TREE DECOMPOSITION (QTD)

A. QTD ALGORITHM

The quad tree decomposition (QTD) exploits the neighborhood spatial redundancy in an image; typically the probability of neighboring pixels having similar values is high. The QTD algorithm merges pixels, in groups of 4, with similar intensities.

Figure 1 shows the sequence of operations in QTD for an 8 x 8 image. In the first step, starting from the top left hand corner of the image, pixels in groups of four that have little or no pixel intensity differences are merged together. The merging is denoted by a tree node "0" with a leaf value obtained by averaging the four pixel values in the group. When the pixels cannot be merged, it is denoted by a tree node "1" with four leaf values representing the four pixel values. In the second step, the nodes of the tree from the first step are merged to obtain the next level of the tree. This process is repeated until we reach the top of the tree [1] [2].

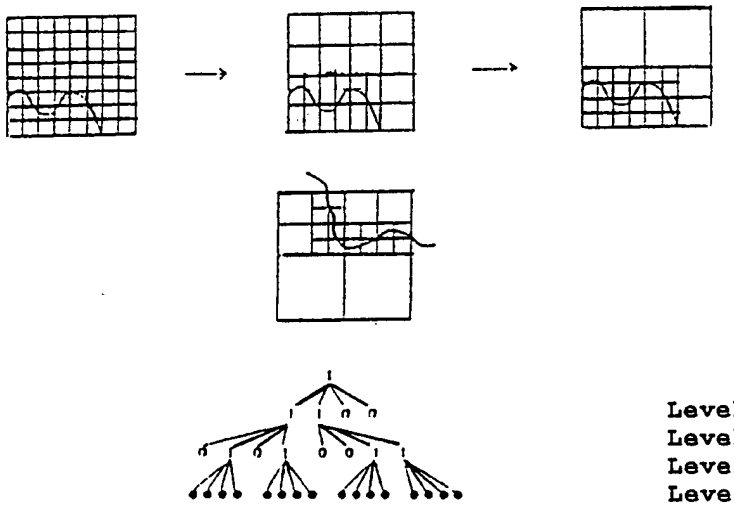


Figure 1 The sequence of operations in the quad tree decomposition.

The merging of the pixel group is determined by a homogeneity test. The homogeneity test used here is based on the absolute difference value:

$$| x_i(m,n) - x_{i-1}(2m+k,2n+l) | < T, \quad \text{for} \quad k=1,2, \; l=1,2$$

where $x_i(m,n) = [\sum_k \sum_l x_{i-1}(2m+1,2n+1)]/4$ and $x(2m+k,2n+l)$ are the individual pixel values in the group, and T is a threshold value to be assigned. The index i represents the resolution level, i=1,....P, for an image of size $2^P \times 2^P$. $x_0(m,n)$ is the pixel value of the original image at location (m,n); $x_i(m,n)$ represents the pixel values at resolution level i.

For a plain group with little pixel intensity variation, the homogeneity test is positive, i.e., the pixels are merged and represented as a single value --- the average pixel value. If the test is negative, the group remains unmerged and is represented by four distinct pixel values. Subsequently, for the next level, groups of 16 pixels are subjected to the test. The procedure continues iteratively till no merging is possible. Finally the tree structure and its leaves are encoded for transmission.

For a complete decomposition of a $2^P \times 2^P$ image, the tree consists of P levels of resolution, namely, levels 1 to P. Level 1 (as in Figure 2(b)) represents the image closest to the original where quad blocks are merged depending on the spatial redundancy of four neighbouring pixels. Hence, level 1 has the best image quality (compared with higher levels) but the lowest compression ratio. At level 2 the spatial redundancy of 4^2 pixel blocks is considered. Here, more compression is possible because of the merging of larger pixel blocks. In general as the level increases, the computation as well as the compression gain increases, but the reproduced image fidelity decreases.

Figure 2 provides an illustration of the QTD for an 8x8 image. In this illustration, we start at the top left hand corner of the image and sequentially process adjacent groups of 2 x 2 pixels, proceeding from left to right in a raster scan fashion. Starting from the top left hand corner of Figure 2(a), we merge the four pixels (all have a value of 1) as there is no pixel intensity variation. It is denoted by a tree node 0 with a single leaf value of 1 (see figure 2(b)). Similarly for the second group of four pixels (all have a value of 2), it is merged and has a tree node of 0 with a single leaf value of 2. For the third group of pixels, it is also merged as the pixel intensity variation is not significant. The leaf value of 4.5 is obtained by averaging the four pixel values. The fourth group of pixels are not merged as there is significant pixel intensity variation. This pixel group is denoted by tree node 1 with four leaf values: 10,20,30,40. After the first iteration we obtain an array of tree nodes and leaf values (indicated by the subscripts) as shown in Figure 2(b). The process is now repeated on the 4x4 array in Figure 2(b), and we obtain the 2 x 2 array in Figure 2(c). Repeating the process further on Figure 2(c) will take us to the top of the tree with node $1_{1.25,7.875,25,25}$.

1	1	2	2	3	4	10	20
1	1	2	2	6	5	40	30
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
10	20	10	20	10	20	10	20
40	30	40	30	40	30	40	30
10	20	10	20	10	20	10	20
40	30	40	30	40	30	40	30

(a)

0_1	0_2	$0_{4.5}$	$1_{10,20,30,40}$
0_1	0_1	0_1	0_1
$1_{10,20,30,40}$	$1_{10,20,30,40}$	$1_{10,20,30,40}$	$1_{10,20,30,40}$
$1_{10,20,30,40}$	$1_{10,20,30,40}$	$1_{10,20,30,40}$	$1_{10,20,30,40}$

(b)

$0_{1.25}$	$1_{4.5,25,1,1}$
0_{25}	0_{25}

(c)

Figure 2 Illustration of QTD scheme using an 8 x 8 array of pixel values.

B. PROPOSED MODIFICATIONS

It is observed that the QTD has several limitations. The compression gain achieved by encoding the QTD tree and its leaves is not substantial [1]. The size of the tree along with all the leaves is large, and the tree contains redundancies. Also merging at higher levels using larger pixel groups causes blockiness. We therefore propose to improve the performance of the QTD scheme with the following modifications.

The proposed modifications are explained with the help of the example considered in the previous section. Figure 3(a) shows an 8 x 8 array of pixel values. Level 1 representation is obtained by performing the quad tree merging with a threshold value of 3 (see Figure 3(b)). Blocks marked 1 indicate high activity and those marked 0 represent plain values. Note that Figures 3(a) and 3(b) are the same as Figures 2(a) and 2(b).

In the modified QTD, once a tree node is marked 1, nodes in subsequent levels which contain it are not allowed to merge. This avoids merging of high activity blocks at higher levels and ensures that high activity regions remain intact. With this modification, it is now possible to set a low threshold value for level 1 in order to mark out regions of high activity and retain the image quality. Larger threshold values can be set at higher levels to gain compression without causing degradation of the reconstructed image. In the unmodified QTD, a common threshold is set for all decomposition levels [1].

Figure 3(c) shows the QTD tree at level 2. Note that the elements in the second row of the array are not merged whereas these elements are merged in the unmodified case as shown in Figure 2(c).

The tree and the leaves generated by the basic QTD algorithm can be further condensed to improve the compression

performance. For coding the tree, the only information useful is a node value 0 that appears for the first time at the highest level. Corresponding 0s at lower levels are redundant for the image reconstruction process. In Figure 3(c), the node $0_{1.25}$ is the 0 that is needed to be transmitted. The corresponding four 0s in the top left corner of Figure 3(b) (marked by an * in the superscript) need not be transmitted as the node $0_{1.25}$ in Figure 3(c) effectively represents them, thereby reducing the tree size.

For coding the leaves, all the leaf values that correspond to a 1 in the tree are redundant except those belonging to tree level 1 as in Figure 3(b). All the leaf values corresponding to the 1s in Figure 3(c) are redundant (marked by an * in the subscript). Only the leaves in Figure 3(b) belonging to level 1 of the decomposition are used to reconstruct the original pixel intensities. The redundant leaf values at higher levels need not be transmitted which further improves the compression.

Figures 4 and 5 show the graphs of the compression gains and absolute error versus the different QTD levels and the threshold settings, respectively. These graphs are obtained from the QTD runs (about 100) with various images --- peppers, baboon, airplane etc. It is noted that the reconstructed image fidelity is acceptable by visual inspection as long as the total absolute error does not exceed 10^5. The total absolute error is defined as follows:

$$E=\Sigma_i\Sigma_j |x_{ij}-y_{ij}|$$

where x_{ij} and y_{ij} represent the pixel values of the original and reconstructed images, respectively. Beyond that, significant blockiness and blurring of images are observed. Based on this observation, from Figure 4(a), the maximum

resolution level required can be determined to be level 2 in order that the absolute error is less than 10^5. From Figure 4(b), for a resolution level of 2, we obtain a compression gain of 10. In turn, to achieve a compression gain of 10, from Figure 5(a), the threshold setting is between 20 and 25. These plots indicate that no significant gain is achieved by performing QTD at higher levels, such as from level 4 to level 9. In general, the decomposition can stop at level 2 because additional decomposition does not improve the quality nor the compression gain; besides, the decomposition at higher levels means larger computational time. It is also noted that the compression gain achieved by the QTD methodology, withstanding reasonable reconstruction quality, is not substantial.

1	1	2	2	3	4	10	20
1	1	2	2	6	5	40	30
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
10	20	10	20	10	20	10	20
40	30	40	30	40	30	40	30
10	20	10	20	10	20	10	20
40	30	40	30	40	30	40	30

(a)

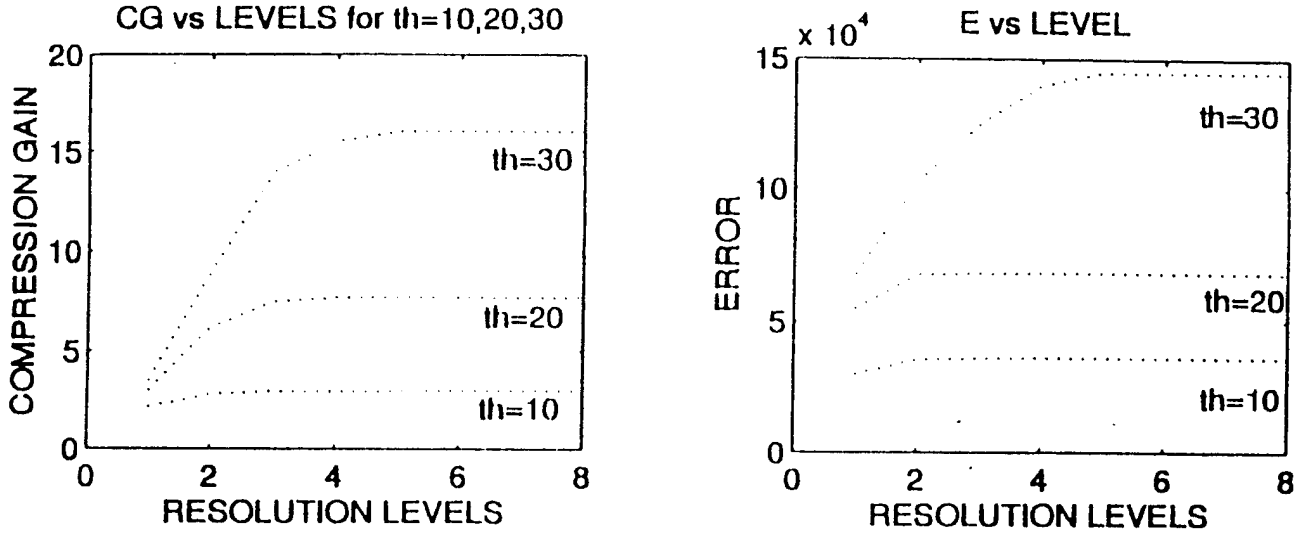
0^{*}_{1}	0^{*}_{2}	$0_{4.5}$	$1_{10,20,30,40}$
0^{*}_{1}	0^{*}_{1}	0_{1}	0_{1}
$1_{10,20,30,40}$	$1_{10,20,30,40}$	$1_{10,20,30,40}$	$1_{10,20,30,40}$
$1_{10,20,30,40}$	$1_{10,20,30,40}$	$1_{10,20,30,40}$	$1_{10,20,30,40}$

(b)

$0_{1.25}$	$1_{4.5,25,1,1^{*}}$
$1^{**}_{25,25,25,25^{*}}$	$1^{**}_{25,25,25,25^{*}}$

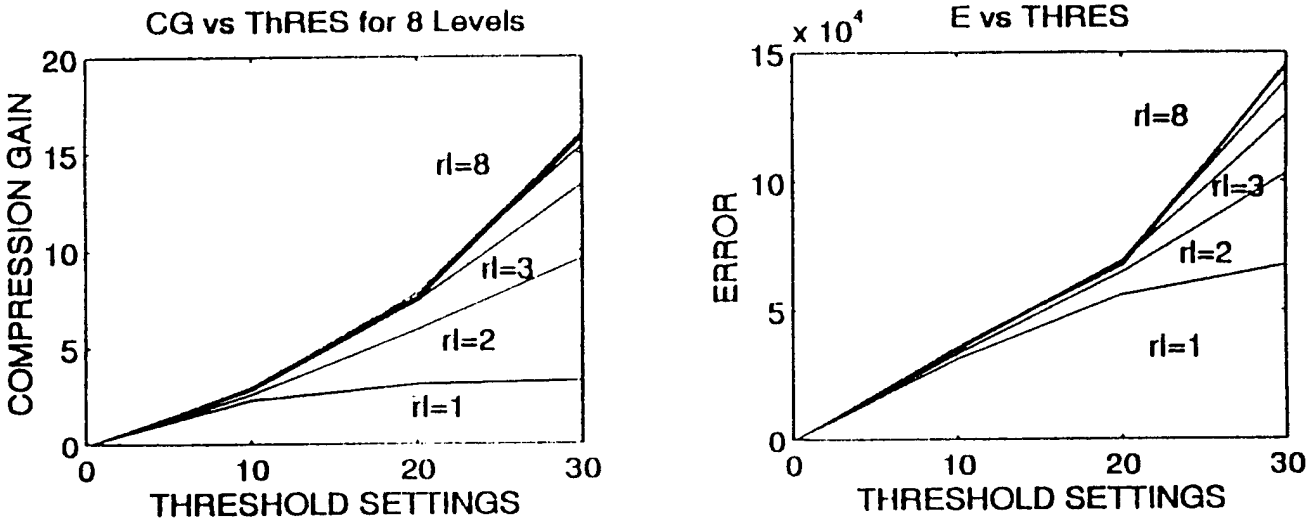
(c)

Figure 3 Improvements to Quad Tree Algorithm. * represents redundant tree and leaves. ** indicates that these nodes do not merge as the corresponding tree nodes in the previous level having a node value 1.



(a) (b)

Figure 4 Compression Gain (CG) and Total Absolute Error (E) vs Resolution Levels (rl)



(a) (b)

Figure 5 Compression Gain (CG) and Total Absolute Error (E) vs Threshold Settings (th)

C. OBSERVATIONS

The QTD segments an image into its regions of low and high activities. It performs this segmentation adaptively on any type of image. This information is useful in the classification of an image's activity. If the number of tree nodes marked 1 is substantial, the image has a higher activity compared to an image with fewer tree nodes marked 1. Based on this, it is sufficient to gauge if the image is of high activity type or not at level 1 without performing the complete QTD.

The QTD's performance depends mainly on two parameters: the selection of the desired level of the tree structure and the threshold setting for the homogeneity test. To improve the performance, for each resolution level, an optimum threshold can be assigned instead of an overall threshold value for all levels.

1. Threshold Setting

The threshold setting determines the image quality. The threshold selection at lower levels is critical in order to ascertain the quality of the reconstructed image. There is no need to fix a common threshold for all levels a priori. For optimum performance a more stringent (small) threshold is set for level 1, and the threshold values at higher levels are set higher to gain compression without compromising on quality. The threshold values at lower levels influence the quality of the reproduced image by marking out and protecting (refer to the modication mentioned above) the high activity regions.

To determine a threshold value, two approaches can be adopted. One way is by using the histogram of the pixel differences and setting the percentage of pixels to be merged. This is done by selecting the threshold value such that it

divides the histogram into a desired pixel population to be merged and a pixel population to be protected. Figure 6 shows the threshold selection by histogram approach. The section of the histogram to the left hand side of the threshold setting is merged (compressed). The pixel population to the right hand side is retained as high activity components. As the threshold setting increases (moves to the right), more pixel blocks merge giving higher compression gain, but the quality of the reproduced image is poor.

Another way is by setting a limit on the error value e.g., the absolute total error to be less than 10^5 (see Figures 4 and 5). The threshold setting is selected such that the error does not exceed the preassigned value for an acceptable image quality.

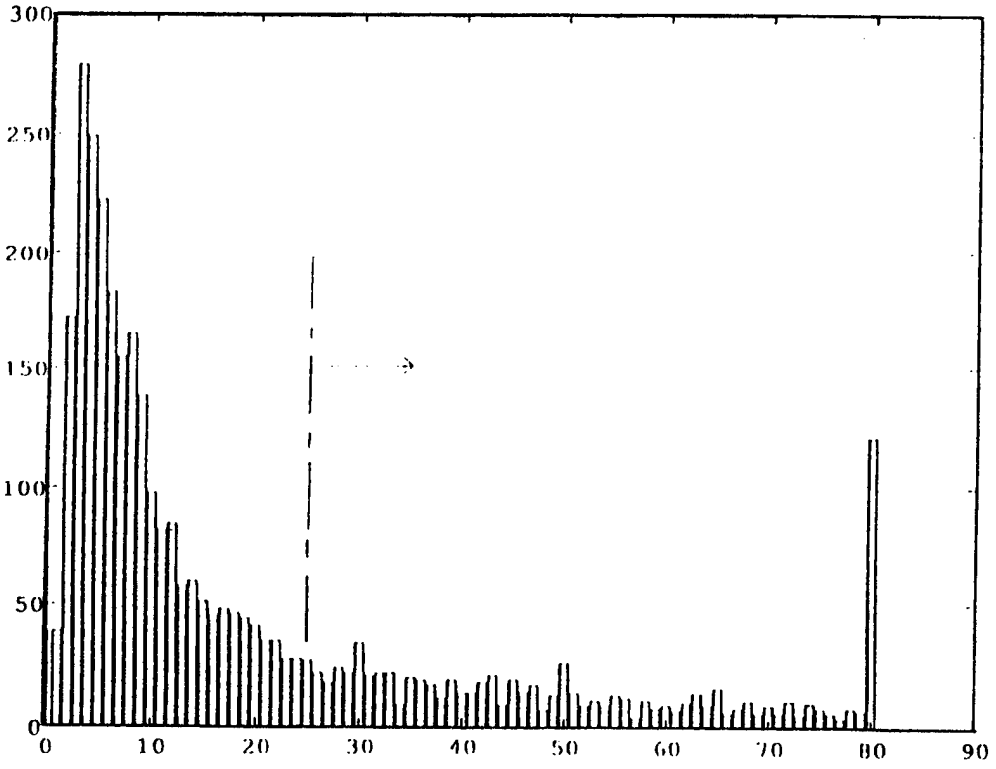


Figure 6 Threshold selection by histogram approach.

III. IMAGE COMPRESSION USING LAYERED APPROACH

A. PROPOSED QTD SCHEME

From Chapter II, it is shown that QTD has limited compression gain. Withstanding reasonable image fidelity, the maximum compression gain for resolution level 1 with a large threshold value tends toward 4. With higher decomposition levels included, the compression gain hardly exceeds 10 without substantial degradation in the reconstructed image. Hence, there is a need to improve the compression performance.

As we have seen from Chapter II, QTD has the inherent capability to decompose any image into regions of high and low activity. The tree information maps out the respective areas of activity; therefore, we could use this information to perform the DCT coding of the decomposed image sequences to gain further compression.

It is also observed from Chapter II that for any image with any threshold value, decomposition at level 1 always provides a visually acceptable reconstruction of the original image. To take the extreme case, by setting a very large threshold setting, we obtain an image that is averaged over quad pixel blocks. This is because with a very large threshold setting the homogeneity test is always positive. This causes all quad blocks to merge, and each block is represented by its average value. It is known that such an image provides a coarse replica of the original image [3]. Let us call the image so obtained the base image. This is advantagous not only for its simple computation (averaging blocks of 4 pixels) with an immediate compression gain of 4, but also for the reduction of the image size for subsequent processing. Given an image of size 512 x 512 pixels, the size of the base image is 256 x 256.

Based on the base image's quad tree structure which indicates its activity, we perform the DCT on the base image

to achieve further compression. Figure 7(a) shows a base image of size 8 x 8 pixels, which is assumed to have been derived from an image of size 16 x 16 pixels. The QTD of the base image in Figure 7(a) provides the quad tree activity information of the base image as shown in Figure 7(b). Groups of 2 x 2 pixels with little variation are merged and denoted by a 0; otherwise, the tree value is 1 with no merging. By summing all the tree node values in Figure 7(b), we estimate the degree of activity of the base image. In this case, we obtain a value of 9. We call this value the activity number.

We now perform DCT coding of the base image. The range of the activity number for the example in Figure 7 is from 1 to 16. We divide the range into 3 categories: 1 to 5, 6 to 11, and 12 to 16. Corresponding to each activity number category, we assign a bit allocation mask: low (activity number 1 to 5), medium (activity number 6 to 11), and high (activity number 12 to 16). The bit allocation mask performs both the DCT coefficient truncation and the allocation of qauntization level per DCT coefficient.

A high valued bit allocation mask retains a large percentage of the DCT coefficients of the transformed base image. Conversely a low valued bit allocation mask retains only a few of the DCT coefficients by masking out many of them. For the retained DCT coefficients, variable bit allocation is performed by assigning more bits to the low frequency coefficients. Figure 8 shows representative bit allocation masks. Bit allocation masks for DCT coefficient truncation and bit quantization are well known [4].

In Figure 7(b), we have an activity value of 9 which corresponds to a medium valued bit allocation mask shown in Figure 8(b). Correpondingly, for a high tree activity, e.g., an activity number of 16, the scheme assigns a high valued bit allocation mask (see Figure 8(c)) with less DCT coefficient truncation and more bit assignment.

Extending the above scheme to an image of size 512 x 512 pixels, a base image of size 256 x 256 pixels is obtained. The base image is then segmented into 8 x 8 pixel blocks, and the DCT is performed on each block. The DCT coefficients are truncated and quantized based on the activity number which allows further compression of the base image.

It is important to realize that the threshold value indirectly determines the overall DCT coefficient truncation and total bit expenditure. The threshold setting for the base image at level 1 (as described in chapter II, it suffices to decompose at level 1 to gauge the image activity) is selected such that at least half the image pixel population is merged (see Figure 6). That is to say, the tree has as many 1s as there are 0s. By setting a higher threshold value there is increased merging of pixel blocks, and the tree has more 0s than 1s indicating that the image is plain. This leads to larger compression gains with the low valued bit allocation mask assigned. This results in fewer retained DCT coefficients and a lower bit expenditure. On the other hand, with a low threshold value, there is less merging of pixel blocks, and the tree contains lots of 1s indicating a high activity image, which in turn requires more retained DCT coefficients with a higher bit expenditure. This produces a better quality image upon reconstruction but with increased bandwidth.

To increase the computational speed as well as the compression gain, instead of using 2 x 2 pixel blocks advocated by QTD, we have used 3 x 3 pixel blocks in the rest of the thesis. However, in the event of merging a block, the center pixel in the block is used to respresent the 3 x 3 pixel block instead of computing the average pixel value. This reduces the computational requirement. Correspondingly, the DCT block size is 9 x 9 instead of 8 x 8. For notational sake, we refer to the scheme using 3 x 3 pixel blocks as nona tree decomposition (NTD).

1	1	2	2	3	4	10	20
1	1	2	2	6	5	40	30
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
10	20	10	20	10	20	10	20
40	30	40	30	40	30	40	30
10	20	10	20	10	20	10	20
40	30	40	30	40	30	40	30

(a): Base image of size 8*8 pixels

0	0	0	1
0	0	0	0
1	1	1	1
1	1	1	1

(b): Quad Tree Activity Information

Figure 7 The base image and its QTD activity information.

8	7	0	0	0	0	0	0
7	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(a): Low Valued Bit Allocation Mask.

Figure 8 Low, Medium, and High Valued Bit Allocation Masks

8	7	6	5	4	0	0	0
7	6	5	4	0	0	0	0
6	5	4	0	0	0	0	0
5	4	0	0	0	0	0	0
4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b): Medium Valued Bit Allocation Map

8	7	6	5	4	3	2	1
7	6	5	4	3	2	1	0
6	5	4	3	2	1	0	0
5	4	3	2	1	0	0	0
4	3	2	1	0	0	0	0
3	2	1	0	0	0	0	0
2	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0

(c):High Valued Bit Allocation Map.

Figure 8(continued) Low, Medium, and High Valued Bit Allocation Masks

B. IMAGE LAYERING

In this section a scheme is proposed whereby the original image is decomposed into several layers. This is to resolve the packet loss problem due to traffic congestion over the network. A first layer consisting of a base image contains the core of the image. Additional layers incrementally improve the quality of the reconstructed image. A three layer scheme is described here for realtime transmission of image data over packet switched networks.

1. Region of Interest

To achieve further compression gains, a Region of Interest (ROI) approach is implemented. This is based on the premise that the middle portion of an image has higher interest for the viewer. Thus the bit allocation mask for the boundary regions is always designated a lower valued mask regardless of its activity level. By doing so, the compression gain of the base image is further enhanced without degrading the middle portion of the image.

2. Layers

The base image obtained by the above scheme (NTD+DCT+ROI) provides an acceptable replica of the original image. The base image is designated as the first layer. In the event of severe traffic congestion in the transmission channel, only the base image is transmitted. The compression ratio achieved using the combination of NTD, DCT and ROI is on the order of 60:1, which means that the bandwidth required for transmitting the base image is $1/60^{th}$ of that required for the original image. Subsequent "layers" of the image are added to the base image to enhance the quality of the reconstructed image. The generation of subsequent layers is explained in the following.

An error image is formed by substracting the base image from the original image. For this purpose, the base image is reconstructed (at the encoder) in order to compute the error image. To avoid the sign problem, the error image is offset by its minimum value to provide positive gray scale values.

The error image is then segmented into its middle region and boundary region. The middle region is designated to be layer 2 and the boundary region layer 3. Similar to the base image, the offset error image is DCT coded. However, this time, only the high valued bit allocation mask is used to approximate a lossless compression scenario. This is to ensure good quality for the reconstructed image. The DCT block size used is 27 x 27 instead of 9 x 9 for improved compression and computational speed [4]. Figure 9 illustrates the image layeral concept. Having decomposed the image into its layers, the data packets are transmitted layer after layer starting with the base image (layer 1), followed by layer 2 (error image's middle region), and layer 3 (error image's boundary region).

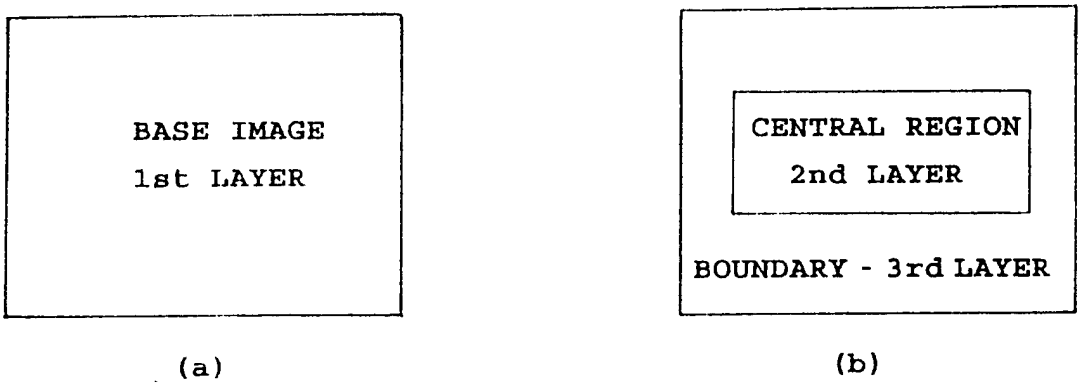


Figure 9 Definition and transmission of the layers of an image.

C. RESULTS

The scheme presented in the previous sections is now tested on real images using a combination of NTD with DCT and ROI. A 3-layer approach is implemented. The scheme is implemented in Matlab; the Matlab code is included in the Appendix.

Initially the first layer (base image) of a test image is transmitted as shown in Figure 9(a). This provides a coarse replica of the original image. Subsequently, the second layer (middle region of the error image) as depicted in Figure 9(b) is transmitted and added onto the base image. Lastly, provided more bandwidth is available, the third layer (boundary region of the error image) is transmitted to enhance the reconstructed image. Upon congestion in the transmission channel, at any point in time, the stream of packets belonging to layers 2 and 3 could be dropped with only the base image being sent. Though the image at the receiver suffers degradation, the effect is graceful.

Figure 10 shows the original image of size 512 x 512. Figure 11 shows the reconstructed base image at the receiver end, i.e., only layer 1 is transmitted. For this case the compression gain achieved in Figure 11 is 60:1. The compression gain is defined as the ratio between the number of bits in the original image and the number bits required to reconstruct the image from the DCT coefficients transmitted. For the various test images used in this study (peppers, airplane, baboon etc.), the compression gain computed ranges from 58 to 61. We have chosen 60 to be the typical compression gain for a base image.

Figure 12 shows the error image (scaled in magnitude by 10 for presentation purposes). This constitutes layers 2 and 3 as defined earlier. Figure 13 shows the reconstructed image at the receiver end with all three layers successfully

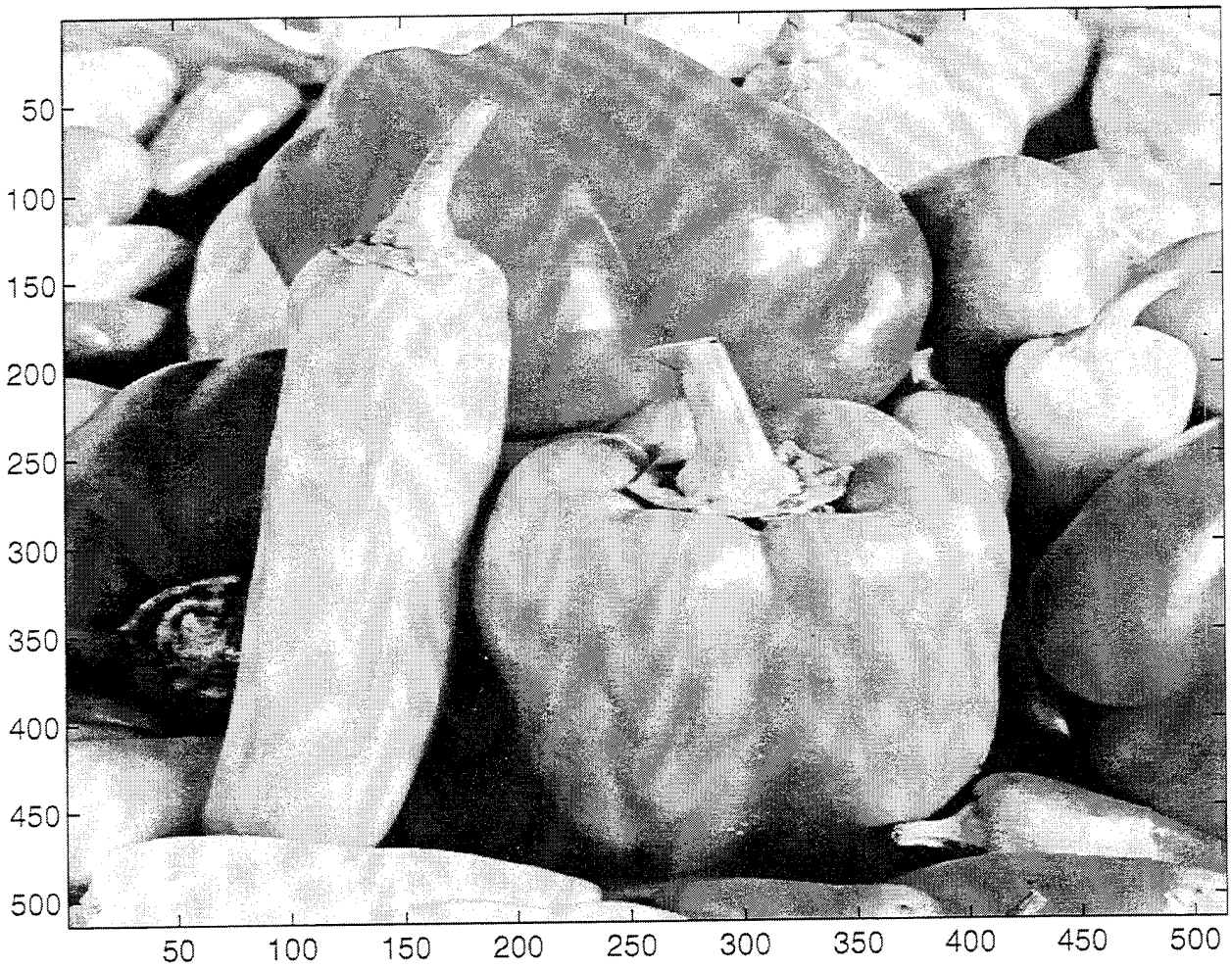


Figure 10 Original Image: Bit rate is 2.1 Mb/frame.

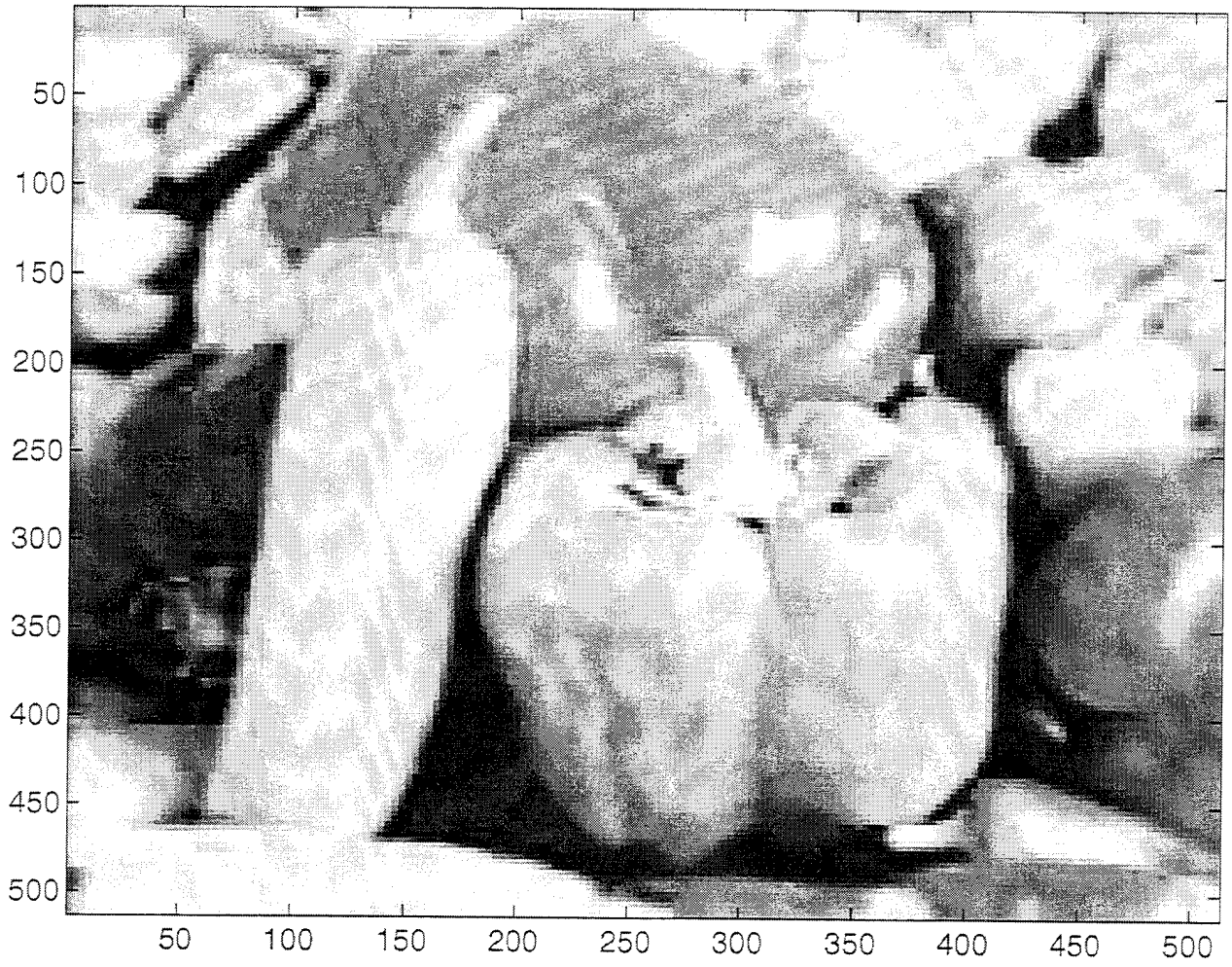


Figure 11 Reconstructed Base Image: The compression gain is 60:1. Bit rate is 35Kb/frame. The MSE is 98 units.

Figure 12 Error Image (magnitude scaled by a factor of 10).

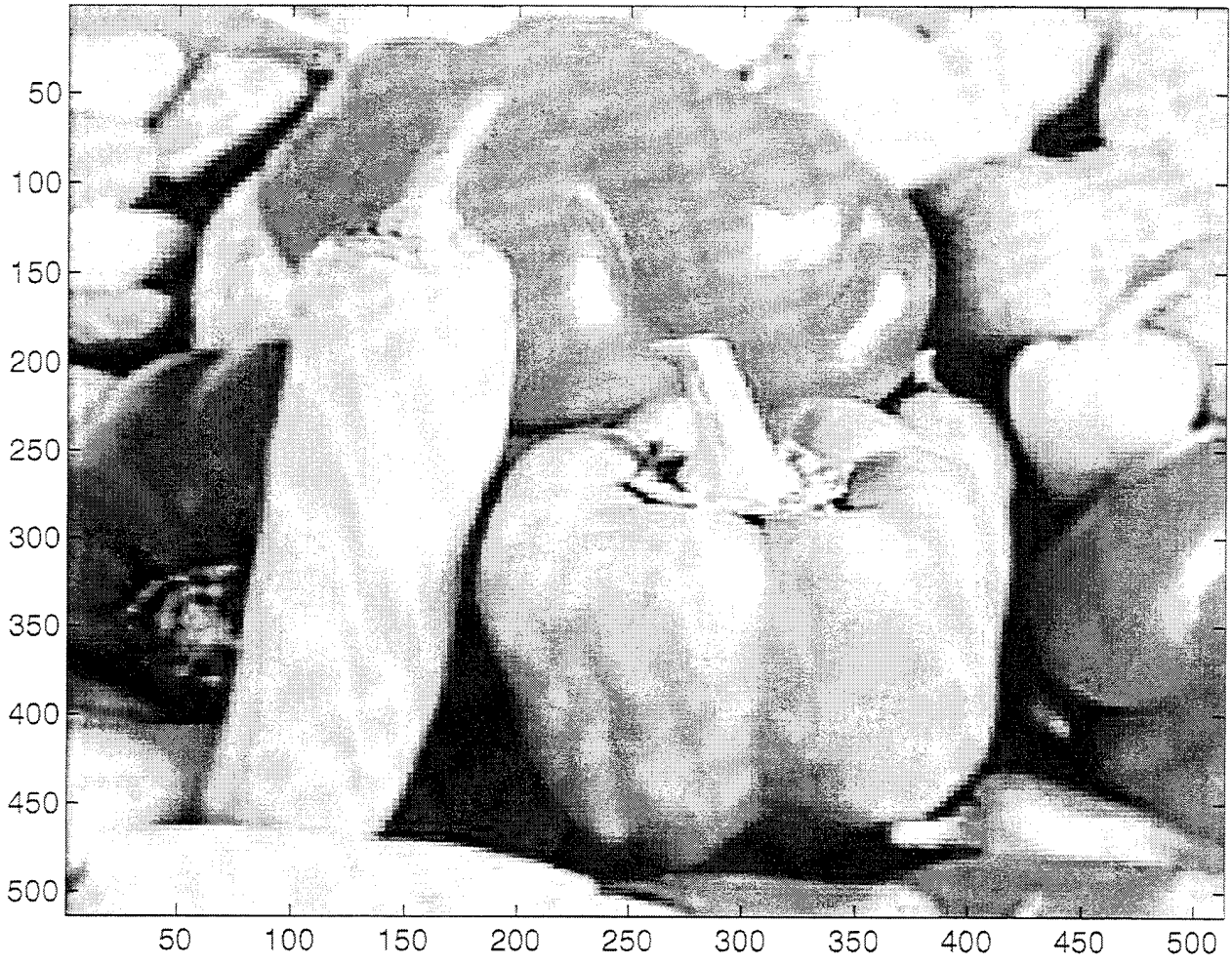


Figure 13 Reconstructed Image from Base Image and the Error Image. The compression gain is 20. The bit rate is 105Kb/frame. The MSE is 54 units.

received, i.e., the base image, the middle region of the error image and the boundary region of the error image. For the various test images, the compression gains of their error images ranged from 28 to 32. We have chosen 30 to be the typical compression gain for an error image. Consequently, the overall compression gain achieved for the base image (layer 1 with a compression gain of 60:1) and the error image (layers 2 and 3 with a compression gain of 30:1 together) is 20:1.

Figure 14 shows the reconstructed image at the receiver end using all the layers of the transmitted image. However, in this case, the error image is coded in a quasi-lossless fashion with high valued bit allocation masks for the entire error image.

Figure 15 shows the scenario whereby the 3rd layer packet stream is to be dropped off due to traffic congestion. In Figure 16 the resultant "survived" image is shown. Notice that the middle region maintains its image quality. Also the degradation of the reconstructed image in Figure 16 occurs in a graceful manner.

To provide a quantitative measurement for the quality of the reconstructed image, the mean square error (MSE) (between the original and the reconstructed images) has been used. The MSE is defined as

$$\xi = [\Sigma_i \Sigma_j |x_{ij} - y_{ij}|^2] / (512)^2$$

where x_{ij} and y_{ij} are pixel values of the original and reconstructed images of size 512 x 512, repectively.

For the images analyzed, we have obtained the compression gains and corresponding MSE values for the reconstructed images as shown in Table 1. The subjective quality for these images can be perceived from Figures 10 to 16.

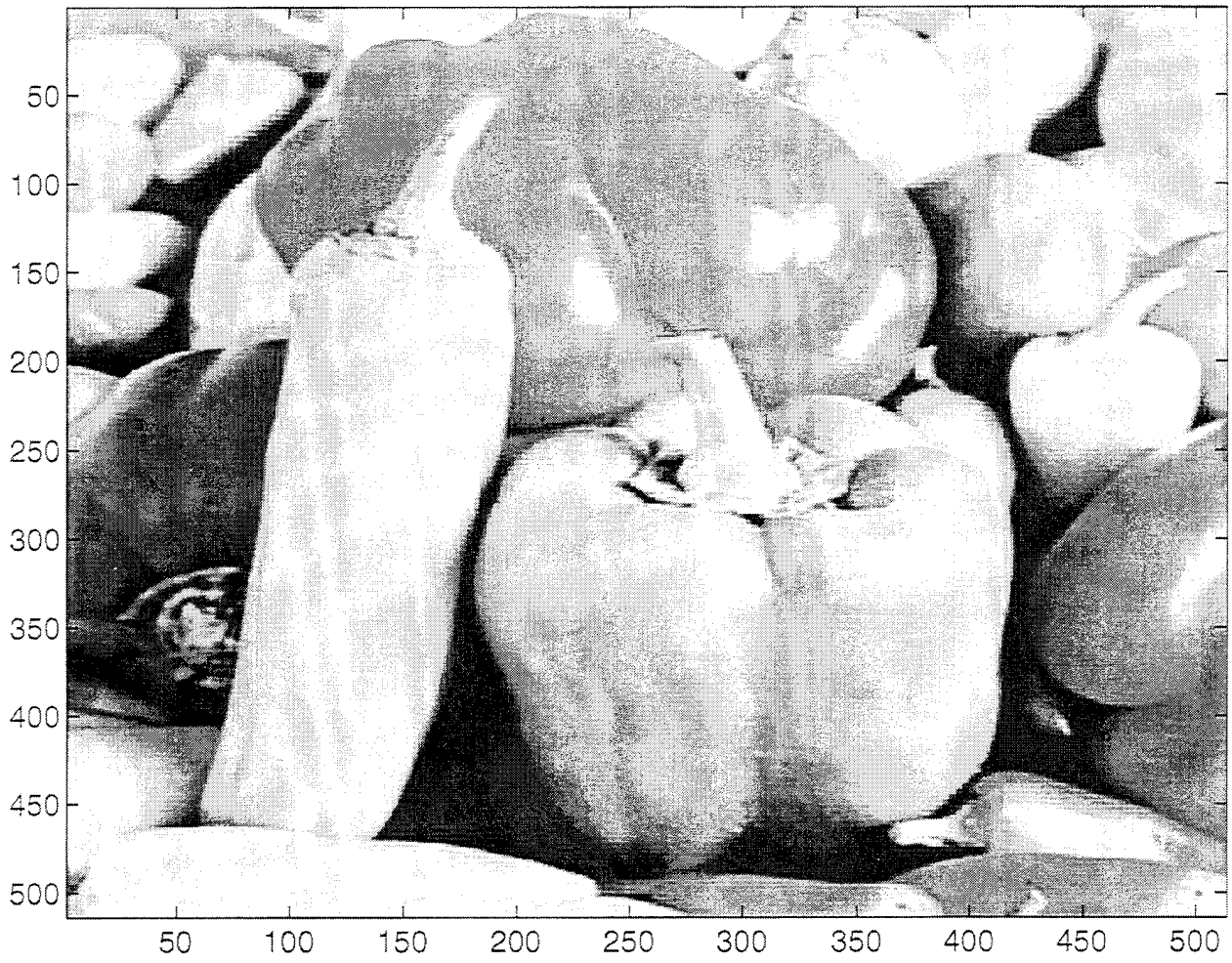


Figure 14 Reconstructed image from base image and quasi lossless error image. The compression gain is 16. The bit rate is 130kb/frame. The MSE is 34 units.

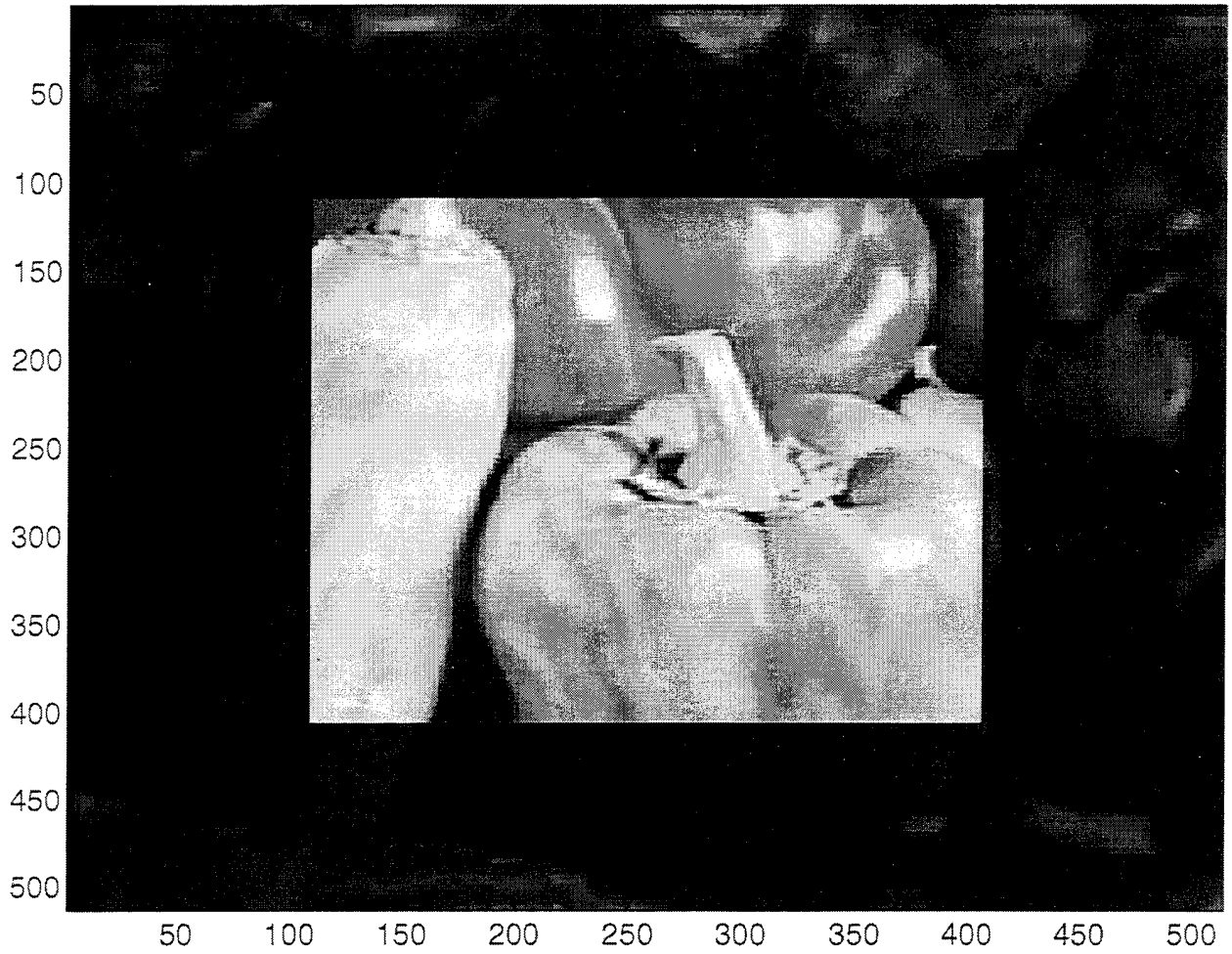


Figure 15 Congestion during the 3rd layer transmission. Shaded portion indicates the affected area of layer 3. (The resultant image is shown in Figure 16).

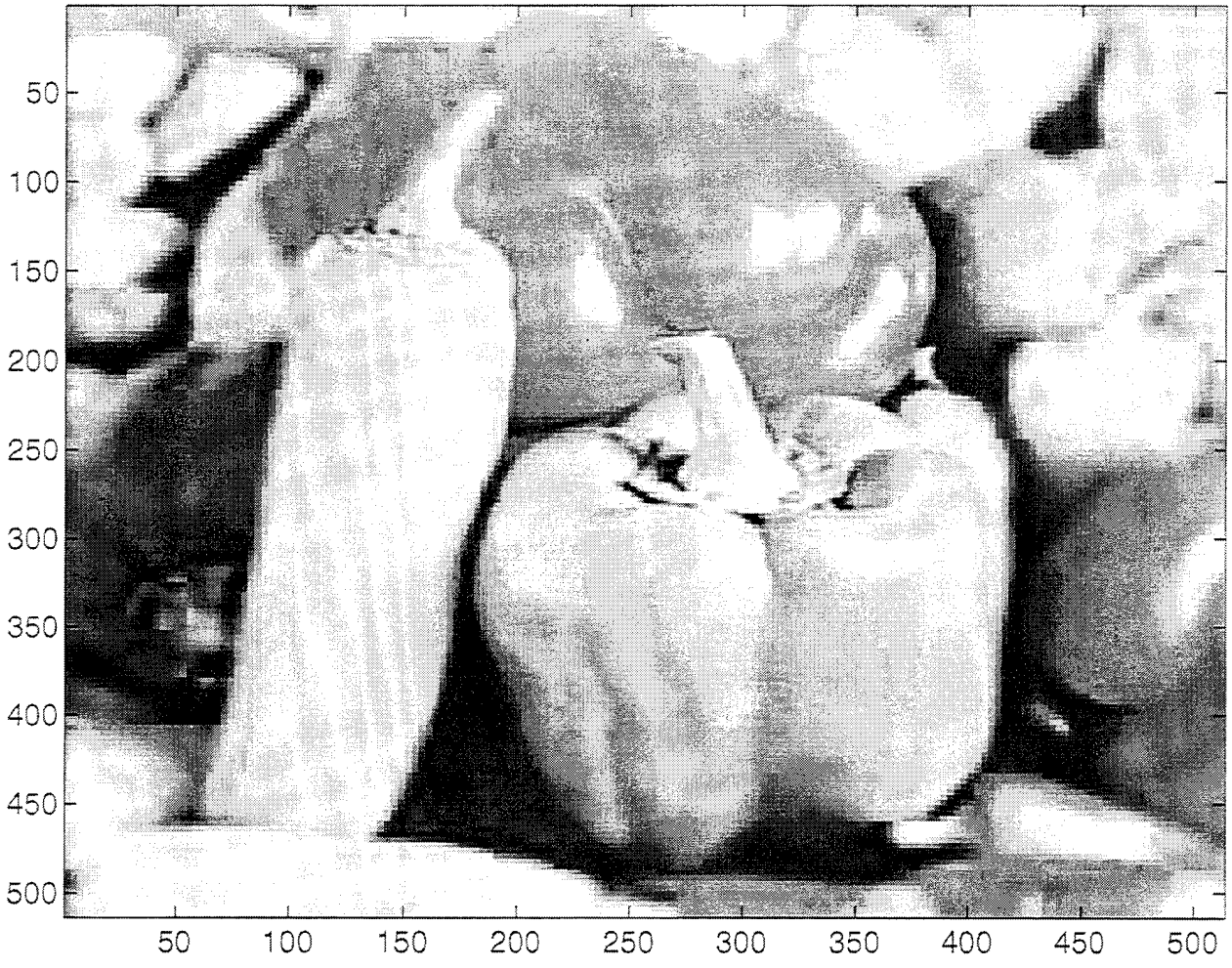


Figure 16 "Survived Image" with degraded boundary region. Reconstructed image uses only layers 1 and 2. The compression gain is 20. The MSE is 54 units.

COMPRESSED IMAGES	COMPRESSION GAIN	MEAN SQUARE ERROR
BASE IMAGE	60	98
BASE IMAGE + LAYERS 2 & 3	30	69
BASE IMAGE WITH IMPROVED LAYER 2	20	54
BASE IMAGE + QUASI-LOSSLESS ERROR IMAGE	16	34

Table 1: Compression gains and MSE values of the reconstructed images.

From Table 1, it can be seen that as more layers are transmitted, the reconstructed image quality improves incrementally. The MSE measurements range from 98 units to 10 units corresponding to a reconstruction of a coarse image to a quasi-lossless image, respectively.

D. A SCHEME TO REDUCE THE EFFECTS OF PACKET LOSS

Here a scheme is presented that reduces the effects of packet loss on the reconstructed image quality. Packets containing image data can be lost due to delays in routing and queueing. Packet loss can also occur due to excessive channel noise. Real-time transmission of image data may preclude retransmission of lost packets. In this discussion the packet loss is assumed to be due to burst errors. Figure 17 shows an image subjected to burst errors causing a large rectangular patch on the image. For the purpose of presentation, we simulated the burst errors by masking out the DCT blocks of the base image in the region where the patch occurs (layer 1).

To cope with such bursty errors, we propose to scramble the data packets before transmission in a pseudo-random manner. The idea is to make use of the fact that the error is correlated in the temporal domain [5]; deliberate jamming of the transmission channel is one source of such temporally correlated errors.

The scheme is explained with the help of an example as shown in Figure 18; each box in the figure corresponds to a DCT coded data block in a given image layer. The boxes are marked in a spiral sequence. For transmission, a sequence of four blocks marked 16 are transmitted first, followed by 15,14, ...etc. In the event where a series of packets is lost, the error is distributed over the entire image instead of being localized in a specified region. In Figure 17, the case of a normal non-interleaved transmission is shown with channel burst errors of 128 bytes (1K bits) in a row. Notice that the patchy effect of the burst errors is spatially concentrated. In Figure 19, the case of a spiral interleaved transmission is shown. This time with the interleaving, the patchy effect of the burst errors is "spread out", and their visual effect on the image quality is minimized.

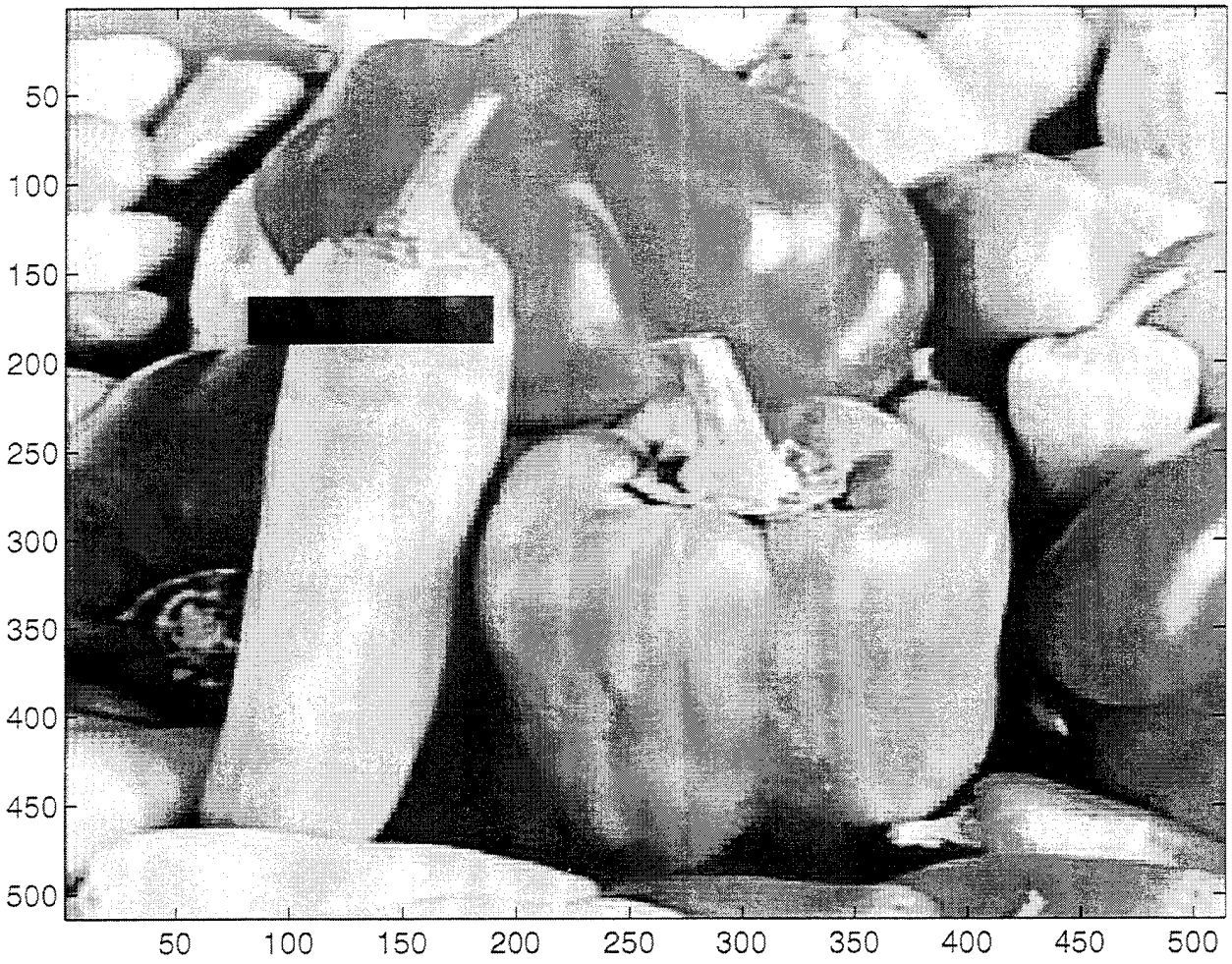


Figure 17 Burst Error of 128 bytes: Non Spiral Image transmission and reconstruction.

1	2	3	4	5	6	7	1
7	8	9	10	11	12	8	2
6	12	13	14	15	13	9	3
5	11	15	16	16	14	10	4
4	10	14	16	16	15	11	5
3	9	13	15	14	13	12	6
2	8	12	11	10	9	8	7
1	7	6	5	4	3	2	1

(a) Spiral interleaving before packet transmission

1 2 3 4 ..... 13 14 15 16 ------>

(b) Packet data stream

Figure 18 Spiral interleaving to minimize the effects of burst errors.

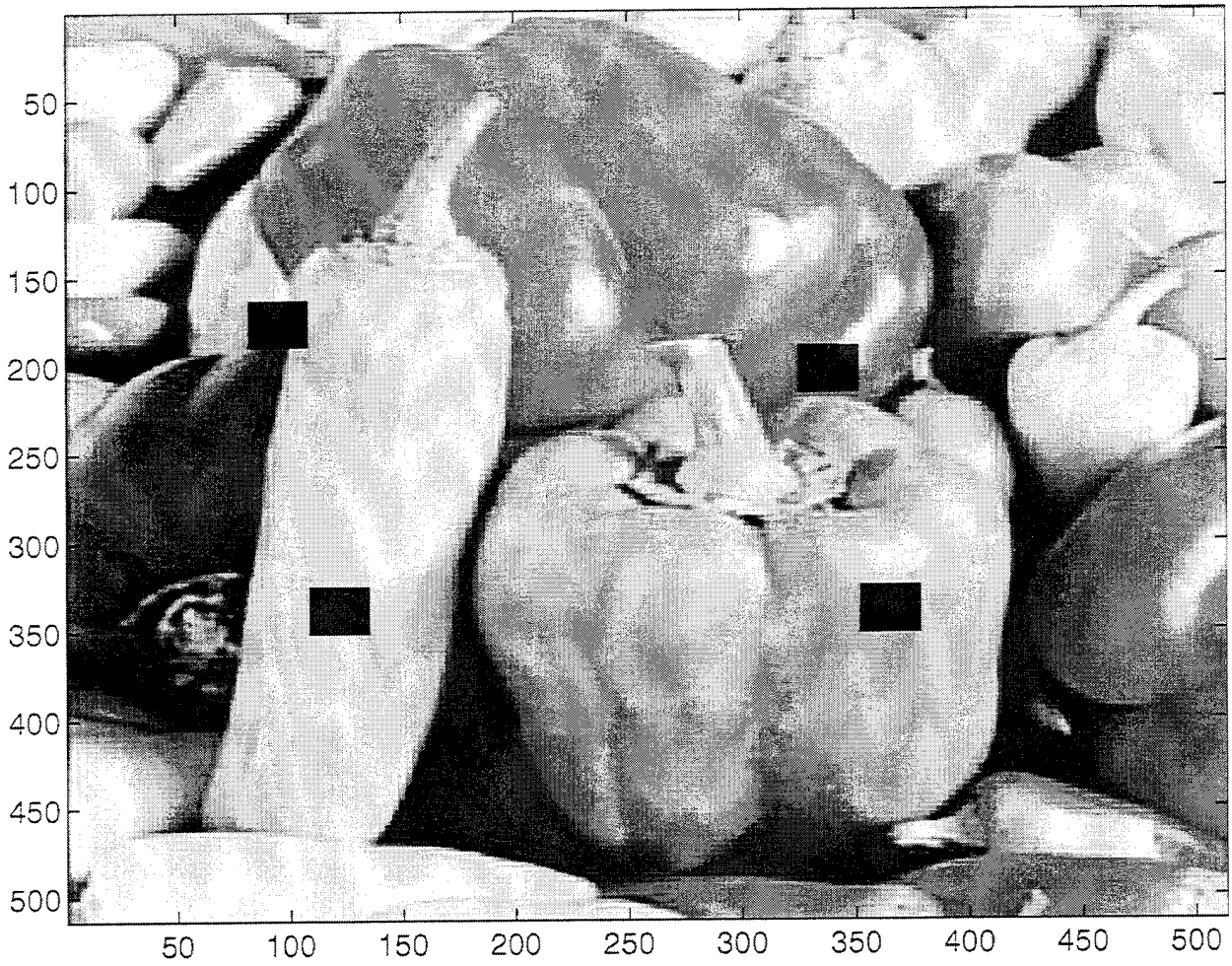


Figure 19 Burst Error of 128 bytes: Spiral Image transmission and reconstruction.

The layer concept is a viable method for real-time transmission of image data over a packet switched network [3]. The reconstructed image quality builds up gradually. Conversely in the event of packet loss, the reconstructed image degrades in a rather graceful manner. The spiral interleaving provides a means to conceal the patchy effect of burst errors. The overall compression gain of the proposed scheme is limited by the compression of the error image. As the error image does not possess spatial neighborhood redundancy as in a natural image, the QTD or the NTD methodology is not very effective. A more efficient way to compress the error image has to be investigated to achieve higher compression gains.

IV. FULL MOTION VIDEO

For full motion video transmission, a sequence of image frames are transmitted consecutively. In the previous chapters we have only considered the transmission of a still image frame. This chapter looks into the application of the proposed scheme for full motion video. The objective is to allow transmission of full motion video over a packet switched network such that the received video has graceful degradation and error concealment capabilities. The chapter presents an extension of the layered image transmission scheme to full motion video. The discussion is conceptual in nature, and no results are presented to support it. Further investigation is necessary before testing the scheme on real video sequences. Nevertheless, the estimated compression gains are realistic which makes the proposed scheme attractive for application to full motion video.

A. A LAYERED APPROACH FOR VIDEO COMPRESSION

The normal frame rate for video transmission is 30 frames per second. By taking the human psycho-visual characteristics into account, the frame rate can be reduced from 30 frame per second to 15 frame per second without significant degradation in perception [5]. In the proposed compression scheme for full motion video, the frame rate suggested is 15 frames per second. The still image layered approach presented in Chapter 3 can then be applied to a sequence of image frames transmitted at the reduced rate.

The compression of the video sequences exploits the fact that consecutive frames are correlated. To achieve enhanced compression gain, we consider pairs of consecutive frames for compression. The first frame of the pair is processed as if it were a still image. The three layers of the frame are computed, encoded, and transmitted.

The second frame is processed slightly differently to reduce processing, and gain compression. The base image is not computed for the second frame, instead the base image of the first frame is used to compute the error image, which is encoded for transmission. The error image of the second frame is now computed as the difference between the original second frame and the reconstructed base frame of the first image. This is to exploit the fact that consecutive frames are corelated. As a result only the base image of the first frame needs to be computed and transmitted. This is similar to applying DPCM between consecutive frames. The above compression scheme is repeated for frames 3 and 4, 5 and 6 etc. The error image is then segmented into two layers; a middle region and a boundary region. After which the layers are interleaved and transmitted in the same fashion as proposed in the still image case.

B. COMPRESSION GAIN ACHIEVEABLE

We now compute the achieveable compression gain for the proposed scheme for video transmission. To start with we compute the required bandwidth for a standard 30 frames per second full motion video with no compression. Consider an image frame of size 512 x 512 pixels with 8 bit quantization, and a frame rate of 30 frames per second. The bandwidth required for this case is 63 Mb/s.

In the proposed scheme for video, the frame rate is 15 frames per second, and two consecutive frames are processed as a pair. The bit rate calculations are different for each of the two frames. Using the layered compression scheme discussed in Chapter 3 to process the first frame, the total number of bits required is 105 kb/frame (20:1 compression gain). For the second frame, only the error image is transmitted (30:1 compression gain). The total number of bits required for the

second frame is 70 kb/frame. The total number of bits per frame pair is then 175 kb. The bit rate for video transmission, R, is given by

$$R = 175 \text{ kb/frame-pair} \times 15/2 \text{ frames/second} = 1.3\text{Mb/s}.$$

The compression gain achievable for full motion video using a frame rate of 15 frames per second and the layered frame pair processing approach is 48:1. Additional gain in compression is possible by using a post entropy coder, such as a Huffman or an arithmetic coder, prior to transmission. The proposed scheme can be considered for realtime transmission of video over packet switched networks. It has the characteristics of graceful degradation when traffic congestion occurs on the network and error concealment when the video sequence is subjected to temporally correlated burst errors; these were illustrated in Chapter III.

V. CONCLUSION

In this thesis a scheme is proposed for still image compression for real-time transmission over a packet switched network. A layered image compression scheme based on the quad tree decomposition technique (QTD) has been developed. The image is decomposed into three layers where layers are added to a base image to incrementally improve the quality of the recontructed image.

The basic QTD alogrithm has been shown to have certain limitations. A modified quad tree algorithm that overcomes these limitations has been proposed which improves the compression performance. This is accomplished by pruning the QTD tree and its leaves. The modified version allows variable threshold values to be set at different decomposition levels. Two methods of determining the threshold setting are presented: a histogram approach and an absolute error limit approach.

To achieve further gains in compression, a DCT algorithm with an adaptive bit allocation method and a region of interest methodology have been incorporated into the QTD. Finally we extended the quad tree to a nona tree whereby pixel blocks of size 3 x 3 are processed instead of 2 x 2.

The layered scheme is tested on real images using a combination of nona tree decomposition, DCT and ROI. Results of the reconstructed images of different cases are presented to provide a visual subjective perception of the proposed scheme. Also the mean square error values are provided as quantitative measures of the reconstructed images. Depending on the number of layers transmitted, the compression gains range from 60:1 to 16:1, providing gradual improvement in the quality of the recontructed image at the receiving end.

We have also incorporated within the compression scheme an interleaving methodology to minimize the effects of burst errors or channel errors on the reconstructed image. With

interleaving, the patchy effect of the burst errors is "spread out", and their visual effect on the image quality is minimized. In summary, It is shown that the proposed scheme exhibits graceful degradation of the reconstructed image and some error tolerant capabilities.

The still image compression scheme can be extended to the case of video transmission at a reduced 15 frames per second rate. Exploiting the interframe redundancy between two consecutive images, we can reduce the bandwidth requirement by sending the base image for alternative frames. The base image of the first frame is used to compute the error image of the second frame. Only the error image of the second frame needs to be transmitted which further improves the compression gain and the computational speed.

The proposed scheme constitutes a variable rate codec which has applications in HDTV and video conferencing. Additionally, applications can be found in video imagery archivals for security systems or hospitals and quick browse and retrival systems for hospitals or C^3I centers. Reduced bandwidth video transmission and imagery downlinks from satellites, surveillance aircraft, and unmanned aerial vehicles are the other possible application areas.

Further work can be directed on two fronts. First, the overall compression gain of the proposed scheme is limited by the compression of the error image. Given a better scheme to compress the error image, the performance could be further enhanced. Second, entropy coders, such as modified Huffman or arithmetic coders, can be applied to enhance the compression performance.

APPENDIX A

MATLAB CODES

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%compress.m;to create base image and error image %
%image: baborg513;baboon.mat;babbase;baberrimag; %
%print: baborgprt;babbaseprt;babdct18            %
%fn.m:  fn9thres;fncmt;fncom9;                   %
%       fndct9;fninvc9;fnreimag9;                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
clg;

%513 by 513 original image
%load peppers;
load baboon;
pos_image=x;
pos_image=[pos_image pos_image(:,512)];
pos_image=[pos_image ; pos_image(512,:)];

baborg513=pos_image;
save baborg513;
sz=513;
pmap='gray(256)';
colormap(pmap)
axis equal
image(pos_image)
pause
print baborgprt

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%creation of average by 9 pixel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  for k=1:sz/3
  for l=1:sz/3
    for j=1:3
    for m=1:3
    blk(j,m)=pos_image(3*(k-1)+j,3*(l-1)+m);
    end
    end
  erravg(k,l)=sum(sum(blk))/9;
  end
  end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%computation of tree to gauge activity
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=0;e318=[];
  for k=1:sz/9
  for l=1:sz/9
    for j=1:3
    for m=1:3
    err(j,m)=abs(erravg(3*(k-1)+j,3*(l-1)+m)-erravg(3*(k-1)+2,3*(l-1)+2));

    end
    end
      maxerr=max(max(err));
      errv=[errv maxerr];
```

```
        %e34=[e34 fn9thres(4,maxerr,t)];
        %e36=[e36 fn9thres(6,maxerr,t)];
        %e38=[e38 fn9thres(8,maxerr,t)];
        %e310=[e310 fn9thres(10,maxerr,t)];
        %e312=[e312 fn9thres(12,maxerr,t)];
        %e314=[e314 fn9thres(14,maxerr,t)];
        %e316=[e316 fn9thres(16,maxerr,t)];
        e318=[e318 fn9thres(18,maxerr,t)];
        t=t+1;
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%plainness of image histogram
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=1:1:80;
hist(errv,x)
pause
print erravg9ps

blksz=9;
a9cmt=fncmt(blksz);
%load a9base_image;
%load baborg513;
%[com0,ferr0]=fncom9(0,e30,blksz,a9cmt,erravg,a9base_image,baborg513);
%[com2,ferr2]=fncom9(2,e32,blksz,a9cmt,erravg,a9base_image,baborg513);
%[com4,ferr4]=fncom9(4,e34,blksz,a9cmt,erravg,a9base_image,baborg513);
%[com6,ferr6]=fncom9(6,e36,blksz,a9cmt,erravg,a9base_image,baborg513);
%[com8,ferr8]=fncom9(8,e38,blksz,a9cmt,erravg,a9base_image,baborg513);
%[com10,ferr10]=fncom9(10,e310,blksz,a9cmt,erravg,a9base_image,baborg513);

%[com12,ferr12]=fncom9(12,e312,blksz,a9cmt,erravg,a9base_image,baborg513);

%[com14,ferr14]=fncom9(14,e314,blksz,a9cmt,erravg,a9base_image,baborg513);

%[com16,ferr16]=fncom9(16,e316,blksz,a9cmt,erravg,a9base_image,baborg513);

[com18,a9base_image]=fncom9(18,e318,blksz,a9cmt,erravg);

err_base9=sum(sum((baborg513-a9base_image).^2))
mse_base9=err_base9/(513^2)

a9org513p=baborg513;
save a9org513p; % baboon org_image
a9err_imagep=baborg513-a9base_image;
save a9err_imagep; % baboon err_image
a9base_imagep=a9base_image;
save a9base_imagep; % baboon base_image

end

%%%%%%%%%%%%%%%%%%%% end %%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%comerr.m; to compress error image %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
clg;
load a9err_imagep;
load a9base_imagep;
load a9org513p;
pmap='gray(256)';
colormap(pmap)
axis equal
image(a9err_imagep*10)
pause
image(a9base_imagep)
pause
image(a9org513p)
pause

offset=min(min(a9err_imagep));
pos_image=a9err_imagep-offset;
image(pos_image)
pause
cmat=fncmt(27);
dim=513/(9*3);
maskf=ones(27);

mask1c=[1 1 1 1 1 1 1 1 1;
        1 1 1 1 1 1 1 1 1;
        1 1 1 1 1 1 1 1 1;
        1 1 1 1 1 1 1 1 1;
        1 1 1 1 1 1 1 1 1;
        1 1 1 1 1 1 1 1 1;
        1 1 1 1 1 1 1 1 0;
        1 1 1 1 1 1 1 0 0;
        1 1 1 1 1 1 0 0 0];

mask1d=[1 1 1 1 1 0 0 0 0;
        1 1 1 1 0 0 0 0 0;
        1 1 1 0 0 0 0 0 0;
        1 1 0 0 0 0 0 0 0;
        1 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0];

mask1u=[1 1 1 1 1 0 0 0 0;
        1 1 1 1 0 0 0 0 0;
        1 1 1 0 0 0 0 0 0;
        1 1 0 0 0 0 0 0 0;
        1 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0];

mask11=[mask1c   mask1u   zeros(9);
        mask1d   zeros(9) zeros(9);
        zeros(9) zeros(9) zeros(9)];
```

```
maska=[1 1 1 1 1 1 1 1 0;
       1 1 1 1 1 1 1 0 0;
       1 1 1 1 1 1 0 0 0;
       1 1 1 1 1 0 0 0 0;
       1 1 1 1 0 0 0 0 0;
       1 1 1 0 0 0 0 0 0;
       1 1 0 0 0 0 0 0 0;
       1 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0];

mask22=[maska zeros(9) zeros(9);
      zeros(9) zeros(9) zeros(9);
      zeros(9) zeros(9) zeros(9)];


maskb=[1 1 1 0 0 0 0 0 0;
       1 1 0 0 0 0 0 0 0;
       1 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0];

maskb=[maskb zeros(9) zeros(9);
      zeros(9) zeros(9) zeros(9);
      zeros(9) zeros(9) zeros(9)];


cntb=0;cnt1=0;cnt2=0;cnt4=0;cnt3=0;


 blksz=9*3;
 for k=1:dim
  for l=1:dim
    for j=1:27
    for m=1:27
    blk(j,m)=pos_image(blksz*(k-1)+j,blksz*(l-1)+m);
    end
    end


if (k>4) & (k<16) & (l>4) & (l<16)
mask=mask11;
cnt1=cnt1+1;
else
%mask=mask22;
%mask=maskb;
%mask=zeros(27);
mask=mask11;
cnt1=cnt1+1;
end

    tdct=(cmat*blk*cmat').*mask;
    dctcoef=[dctcoef tdct];
  end
 end

errinvc=fninvc9(blksz,cmat,dctcoef,513);
dcterr=a9base_imagep+offset+errinvc;
```

```
image(dcterr)
pause
%print pfinalrecon
%print psur
%print pspr
%print punspr
%print pzonal
%print pquasi
com_ratio=(8*(513^2))/((cnt1*234)+(cnt2*120))

final_err=sum(sum((a9org513p-dcterr).^2))
mse_dct=final_err/(513^2)

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%
% Functions %
%%%%%%%%%%%%%

function reimagd=fninvc9(blksz,cmat,dctcoef,bsz)
invc=[];slap=[];invcc=[];blk=[];ll=0;mod=0;t=0;
reimagd=zeros(bsz);
dim=171/9;
  for l=1:dim*dim
    for j=1:blksz
    for m=1:blksz
    blk(j,m)=dctcoef(j,blksz*(l-1)+m);
    end
    end
mod=mod+1;
if mod>dim
mod=1;
t=t+1;
end
reimagd(1+blksz*t:(t+1)*blksz,1+blksz*(mod-1):mod*blksz)=cmat'*blk*cmat;
 end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function reimage=fnreimag9(tree,leafval)
reimage=[];
yy=0;
mod=1;
jump=0;
for xx=1:length(tree)
 if mod>171
 jump=jump+1;
 mod=1;
 end
   for ii=1:3
   for jj=1:3
   reimage(ii+3*(jump),jj+3*(mod-1))=leafval(yy+1);
   end
   end
  yy=yy+1;
mod=mod+1;
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [e3]=fn9thres(thres_hold,maxerr,t)
if maxerr>thres_hold
e3=1;
else
e3=0;
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [c]=fncmt(blksz)
for k=1:blksz
 for l=1:blksz
  if (k==1)
    c(k,l)=blksz^(-0.5);
```

```
else
    c(k,1)=((2/blksz)^(0.5))*cos(pi*(k-1)*(2*1-1)/(2*blksz));
  end
 end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [com_ratio,a9base_image]=fncom9(no,errtree,blksz,a9cmt,erravg)
[errdctcoef,cntb,cnt1,cnt2,cnt3,cnt4]=fndct9(no,errtree,blksz,a9cmt,erra
vg);
errinvc=fninvc9(blksz,a9cmt,errdctcoef,171);
com_ratio=(8*(513^2))/((cntb*0)+(cnt1*120)+(cnt2*98)+(cnt3*80)+(cnt4*40))
new3=zeros(171);
new3=new3(:)';
errinvc=errinvc';
leafval=errinvc(:)';
errbig=fnreimag9(new3,leafval);

a9base_image=errbig;
image(errbig)
%pause
if no==18
print pbaseprt
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[dctcoef,cntb,cnt1,cnt2,cnt3,cnt4]=fndct9(no,tree,blksz,cmat,qim
age)
dctcoef=[];blk=[];mtree=[];act=[];
vact=[];mask=[];cntb=0;cnt1=0;cnt2=0;cnt3=0;cnt4=0;

mask1=[1 1 1 1 1 1 1 1 0;
       1 1 1 1 1 1 1 0 0;
       1 1 1 1 1 1 0 0 0;
       1 1 1 1 1 0 0 0 0;
       1 1 1 1 0 0 0 0 0;
       1 1 1 0 0 0 0 0 0;
       1 1 0 0 0 0 0 0 0;
       1 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0];

%[8 7 6 5 4 3 2 1 0;
% 7 6 5 4 3 2 1 0 0;
% 6 5 4 3 2 1 0 0 0;
% 5 4 3 2 1 0 0 0 0;
% 4 3 2 1 0 0 0 0 0;
% 3 2 1 0 0 0 0 0 0;
% 2 1 0 0 0 0 0 0 0;
% 1 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0]
%120 bits/blk;
```

```
 mask2=[1 1 1 1 1 1 0 0 0;
        1 1 1 1 1 0 0 0 0;
        1 1 1 1 0 0 0 0 0;
        1 1 1 0 0 0 0 0 0;
        1 1 0 0 0 0 0 0 0;
        1 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0;
        0 0 0 0 0 0 0 0 0];

%[8 7 6 5 4 3 0 0 0;
% 7 6 5 4 3 0 0 0 0;
% 6 5 4 3 0 0 0 0 0;
% 5 4 3 0 0 0 0 0 0;
% 4 3 0 0 0 0 0 0 0;
% 3 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0]
%98 bits/blk;

mask3=[1 1 1 1 1 0 0 0 0;
       1 1 1 1 0 0 0 0 0;
       1 1 1 0 0 0 0 0 0;
       1 1 0 0 0 0 0 0 0;
       1 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0];

%[8 7 6 5 4 0 0 0 0;
% 7 6 5 4 0 0 0 0 0;
% 6 5 4 0 0 0 0 0 0;
% 5 4 0 0 0 0 0 0 0;
% 4 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0]
%80 bits/blk;

mask4=[1 1 1 0 0 0 0 0 0;
       1 1 0 0 0 0 0 0 0;
       1 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0];

%[8 7 6 0 0 0 0 0 0;
% 7 6 0 0 0 0 0 0 0;
% 6 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0;
% 0 0 0 0 0 0 0 0 0]
```

```
%40 bits/blk;

mtree=reshape(tree,57,57);
dim=171/9;
  for k=1:dim
  for l=1:dim
    for j=1:9
    for m=1:9
    blk(j,m)=qimage(blksz*(k-1)+j,blksz*(l-1)+m);
    end
    end

act=mtree((1+(blksz/3)*(k-1)):(blksz/3)*k,(1+(blksz/3)*(l-1)):(blksz/3)*
l);
    vact=[vact sum(sum(act))];


if (k>4) & (k<16) & (l>4) & (l<16)
mask=mask1;
cnt1=cnt1+1;
elseif (k==1)|(k==dim)|(l==1)|(l==dim)
mask=mask3;
cnt3=cnt3+1;
elseif (sum(sum(act)) > 5)
     mask=mask1;
     cnt1=cnt1+1;
elseif (sum(sum(act)) > 3)
    mask=mask2;
    cnt2=cnt2+1;
else
     mask=mask3;
     cnt3=cnt3+1;
end

  tdct=(cmat*blk*cmat').*mask;
    dctcoef=[dctcoef tdct];
  end
  end
hist(vact)
title('DCT DISTRIBUTION')
%pause
if no==18
print airdct18
end
if no==4
print dct4
end
if no==8
print dct8
end
if no==10
print dct10
end
if no==14
print dct14
end
end

%%%%%%%%%%%%%%%%end%%%%%%%%%%%%%%%%%%%
```

LIST OF REFERENCES

1. Shusterman, E. and Feder, M., "Image Compression Via Improved QTD Algorithms," *IEEE Transactions on Image Processing,* Vol. 3, No. 2, March 1994, pp. 207-215.

2. W. Grosky and R. Jain, "Optimal Quadtrees For Image Segment," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-5, No. 1, January 1983, pp. 77-83.

3. Ken Knowlton, "Progressive Transmission of Grey-Scale and Binary Pictures," *Proceedings of the IEEE,* Vol. 68, No. 7, July 1980, pp. 885-896.

4. Jain K. Anil, *Fundamentals of Digital Image Processing,* Prentice Hall, Eaglewood Cliffs, NJ, 1989, pp. 505-506.

5. Ghanbari, M. and Seferidis, V., "Cell Loss Concealment in ATM Video Codecs," *IEEE Transactions on Circuits and Systems for Video Technology,* Vol. 3, No. 3, June 1993, pp. 238-247.

6. Taizo Kinoshita, Tomoko Nakahashi & Masanori Maruyama, "Variable Bit Rate HDTV CODEC," *IEEE Transactions on Circuits and Systems for Video Technology,* Vol. 3, No. 3, June 1993, pp. 230-237.