REDUCTION OF BLOCKING EFFECT

IN IMAGE CODING

by

HOWARD CLIFFORD REEVE III

B.S.E.E. University of Cincinnati
(1981)

Submitted to the Department of Electrical
Engineering and Computer Science
in Partial Fulfillment of the
Requirements of the Degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 1983

Signature of Author: Signature redacted
Department of Electrical Engineering and Computer Science
November 18,1982

Certified by: Signature redacted
Jae S. Lim, Thesis Supervisor

Accepted by: Signature redacted
Chairman, Departmental Committee on Graduate Students

REDUCTION OF BLOCKING EFFECT IN IMAGE CODING

by

HOWARD C. REEVE III

Submitted to the Department of Electrical Engineering and Computer Science on November 18,1982 in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering

ABSTRACT

In some important image coding techniques, such as transform coding, an image is first broken up into subimages, and then each subimage is coded independently. The segmentation procedure has significant advantages, but when used in a low bit rate scheme, an undesirable side effect can occur. Specifically, when the image is reconstructed at the decoder, a "blocking effect" can develop due to discontinuities between the subimages. The thesis addresses the problem by introducing and evaluating two methods to reduce the blocking effects. One method works by modifying the segmentation procedure, whereas the other method consists of an enhancement procedure.

The two methods are applied to a Discrete Cosine Transform (DCT) image coder. A thorough study indicates that both reduction methods do very well to alleviate blocking effects, but that the enhancement procedure is slightly better. Preliminary results on a video-conferencing application are presented.

Thesis Supervisor: Jae S. Lim
Associate Professor of Electrical Engineering

ACKNOWLEDGEMENTS

I wish to extend my sincere gratitude to Jae S. Lim for his technical expertise, guidance, and dedication of time and effort to long discussions. I would also like to thank Tom Osborne, Rudolf Hecken and my family, especially my parents, for their support and encouragement during the past year. Finally, I wish to thank Neal Bergano and Gil Bristol for their assistance and friendship throughout the development of the thesis.

TABLE OF CONTENTS

CHAPTER I - INTRODUCTION

Digital image coding is a practical problem that arises in a variety of contexts, including conservation of bandwidth in transmitting images, and conservation of memory space in storing images. A desirable feature in coding is to represent the image with as few bits as possible, while still retaining sufficient picture quality in reconstruction. A variety of coding schemes and approaches have been developed in the past. Some of the more promising approaches involve segmenting the image into smaller subimages before coding. More specifically, the original image is divided into some number of subimages, usually of equal size, and then each subimage is coded independently of the others. Reassembly of the separate subimage blocks is performed by the decoder to reproduce the full image. The reasoning behind segmenting an image is to exploit the local characteristics of the image and to simplify the hardware implementation of the coding algorithm. Transform coding is a prime example of a subset of coding techniques that usually makes use of image segmentation.

There is, of course, a tradeoff in image coding between two conflicting entities, namely, low bit rate and high image quality (The term "bit rate" will be used as a synonym for the number of bits used to represent an image). In

several important applications, some sacrifice in image quality is acceptable if very low bit rates can be achieved. Of course, the definition of acceptable image quality depends on the particular application at hand. The objective of the research performed in this thesis is to improve existing low bit rate coding schemes that contain image segmentation procedures.

In the past, image coding systems that divide an image into subimages had problems in very low bit rate applications. The major drawback is a "blocking effect" that stems from the image segmentation process itself. Specifically, the reconstruction of the coded segments creates artificial boundaries that form a predictable grid pattern in the image. The blocking effect is noticeable due to discontinuities between the subimages, which appear because of a loss of information in each subimage through the coding process. An example of a coded image that exhibits blocking effects is shown in figure I-1. The details of the coding algorithm are unimportant at this point, but they will be presented in Chapter III. It should be noted that the blocking effects are not noticeable if enough bits are used in coding. The effect becomes more and more prominent as the bit rate is reduced in the coding process.

Decoded images that exhibit blocking effects can be

Figure I-1: Blocking Effect Example.
Image Size=256×256 Pixels.
Subimage Size=16×16 Pixels.

very unpleasant to a viewer. The result is that coding systems using image segmentation do not attain further bit rate reduction gracefully when blocking effects start to prevail. Two simple and viable solutions are explored in this thesis that help remedy the blocking effect problem quite effectively. The thesis will focus on testing and evaluating the two methods in a variety of situations.

The thesis is comprised of seven chapters. Chapter II introduces the two methods for reducing blocking effects. Chapter III describes the coding system that is used. Chapter IV contains the details of the blocking effect study, while Chapter V summarizes the results and draws some conclusions. In Chapter VI, the capability of our coding system is tested with video-conferencing as the application. Finally, Chapter VII presents a brief summary of the thesis, which is then followed by the bibliography and two appendices.

CHAPTER II - METHODS FOR REDUCING BLOCKING EFFECT

Let us assume that an image is to be segmented into smaller uniform blocks before it is coded, and that each block is then to be processed independently. This is not unlikely, since segmenting the image can have useful advantages (see Chapter III). Furthermore, if minimization of the number of bits used to represent an image is important to the extent that some image degradation is tolerable, then blocking effects may become very noticeable in the reconstructed image. Again, the blocking effects arise because of a loss of information in each segment through the coding process. When the segments are reconstructed at the decoder, boundaries form at the edges due to intensity jumps that were not originally present. In many cases, the appearance of blocking effects can be the limiting factor in reducing the bit rate further. This is because the blocking effect can be unacceptable to the viewer, even though each subimage still contains acceptable quality.

From the above discussion, it is clear that some method for reducing blocking effect problems would be useful. There is one advantageous property of the problem that makes its solution easier. If the images are segmented the same way every time, then the boundary locations are known a

priori. Two methods that exploit this property were developed for dealing with blocking effects, namely, the overlap method and the filtering method.

II.1 The Overlap Method

In the overlap method, the blocking effect problem is dealt with at its source, namely, the segmentation process. A typical segmentation procedure divides an image into mutually exclusive regions, each containing some number of pixels. Discontinuities between regions can then appear in low bit rate schemes. Instead of forcing the regions to be exclusive of each other, it is reasonable that a slight overlap around the perimeter of each region could reduce the problem. The pixels at the perimeter would then be coded in two or more regions. In reconstructing the image, a pixel that was coded more than once would use an average of the values that it has from each region that it was included. Thus, abrupt discontinuities at boundaries from coding are reduced because the segmentation process in effect "tacks" the subimages together.

An example of the overlap method would be useful at this point to clearly illustrate the segmentation procedure. Figure II-1a shows how a group of pixels might be segmented by a conventional procedure. Figure II-1b shows how the same

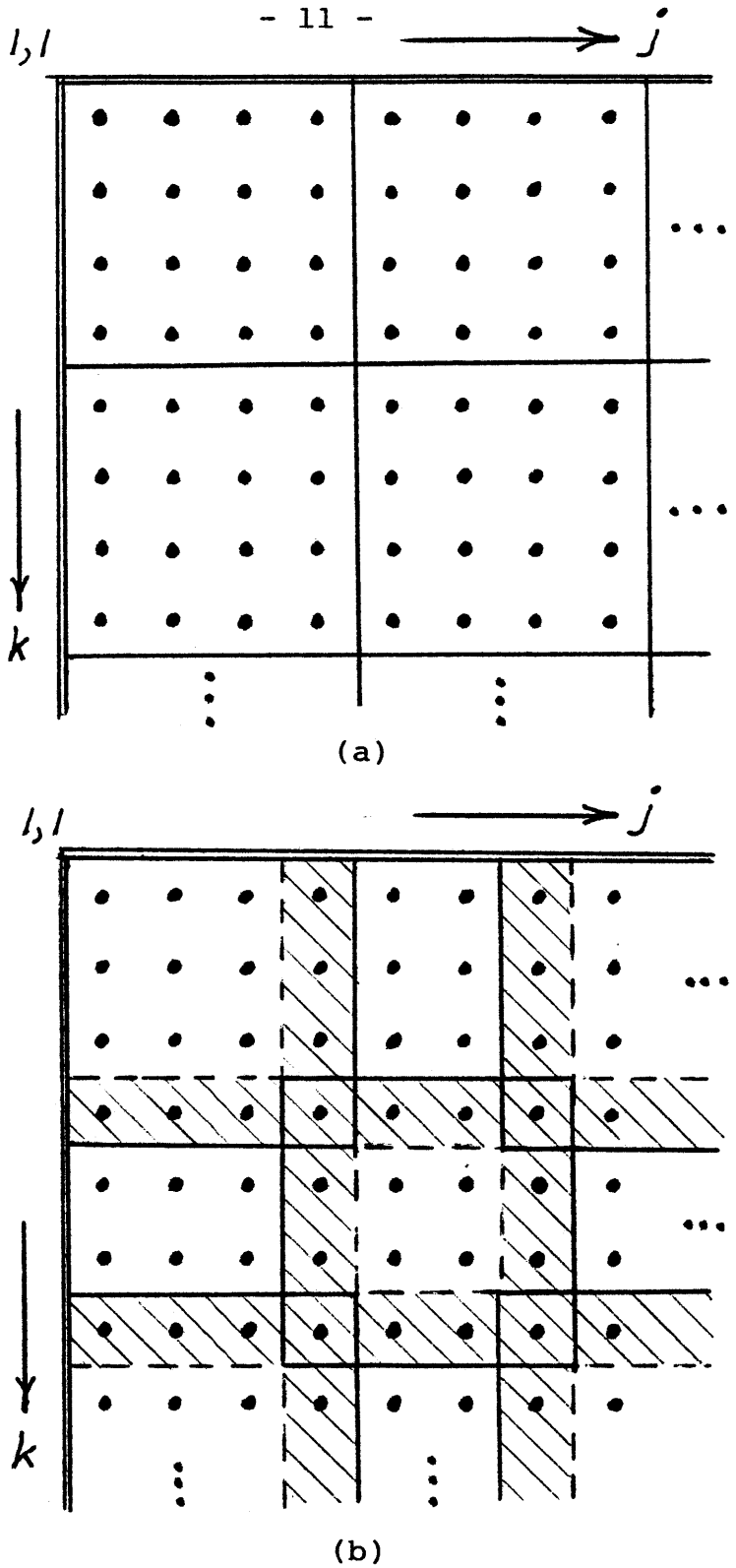


Figure II-1: Segmentation Procedures. a) The normal method - no overlap. b) The one pixel overlap method. Pixels in shaded area would be processed more than once.

group of pixels would be segmented in a one pixel overlap procedure. Of course, any number of pixels can be overlapped, but it will be seen later that a one pixel overlap scheme is probably the best. The effectiveness of the overlap method along with its drawbacks will subsequently be examined in Chapters IV and V.

II.2 The Filtering Method

The filtering method is a completely different approach from the overlap method in reducing the blocking effect problem. No changes are made in the coding or segmentation process, which means the subimage regions are left mutually exclusive. Instead, the filtering method is an image enhancement procedure, and is thus applied after reconstruction of the coded image.

It is well known that any sharp edges in an image represent high frequency content. Boundary region discontinuities caused by the segmentation procedure are similar to very sharp edges. Furthermore, the location of these edges is known exactly, since the segmentation procedure is predetermined. Thus, it is evident that lowpass filtering the image at and near the boundaries would smooth the unwanted discontinuities. This, in fact, is the basis for the filtering method.

Figure II-2 serves to illustrate how a decoded image might be processed by the filtering method in order to reduce blocking effects. The center of each region remains untouched, while the boundaries are smoothed by a lowpass filter. Of course, there are advantages and disadvantages to this method also, but a discussion of these will be saved for Chapter V.

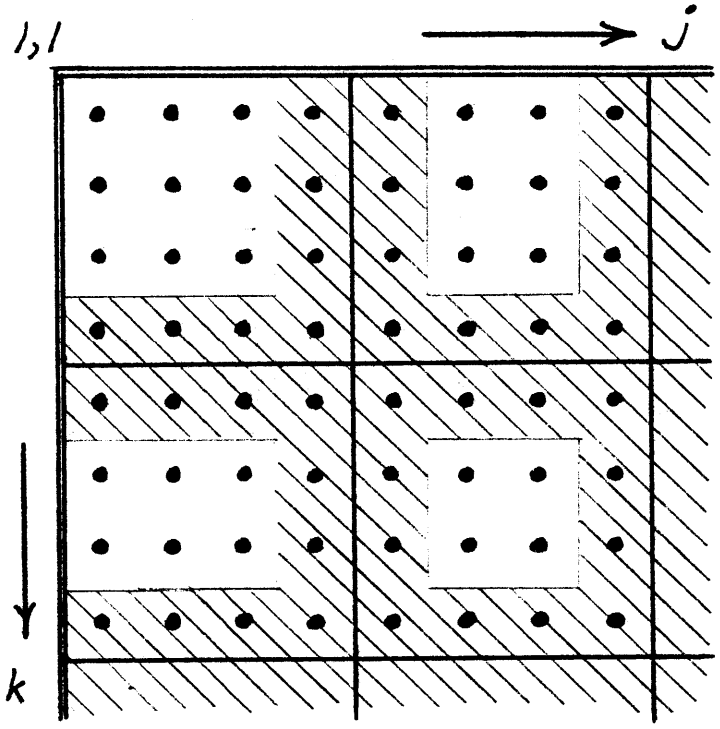


Figure II-2: The filtering method. Pixels in shaded region are processed by the reduction filter.

CHAPTER III - THE CODING SYSTEM

In order to study the two methods proposed in Chapter II for reducing blocking effect problems, an applicable coding system needs to be developed. One successful approach among a wide variety of possible coding schemes, uses two dimensional transforms. Transform coding systems typically use a segmentation procedure, which makes them a natural and likely choice for our purposes. In particular, the Discrete Cosine Transform (DCT) has been shown to be especially attractive with respect to low bit rate schemes [1]. Thus, the decision was made to use the two dimensional DCT as a basis for creating a coding system. The coding system will be developed for black and white images, but the coding techniques could be applied to color images as well.

In developing a coding system, it was decided to incorporate some control over the number of bits used to represent an image. A natural tradeoff exists between minimizing the coded bits and preserving image quality. A control over this tradeoff will be advantageous in the study of blocking effect issues. The control parameters should ideally force the tradeoff to be as graceful and continuous as possible. With the application of transmitting a sequence of images in mind, the control parameters could be used in a feedback scheme to produce a variable bit rate

system.

III.1 The Discrete Cosine Transform

The Discrete Cosine Transform (DCT) is an offspring of the Fourier Transform. It was chosen for several reasons. First, the DCT coefficients are always real, and thus complex computation is unnecessary. Next, the two dimensional DCT does extremely well in decorrelating a typical image, which is a good measure of a transform's data compression efficiency. An ideal transform would extract all of the redundancy in an image, or in other words, completely decorrelate the image. Such a transform can exist, namely, the Karhunen-Loeve Transform, however, its success hinges on some very restrictive conditions. Even if the conditions are met, the computations are impractical and unwieldy. The DCT usually does well in decorrelating a typical image. This fact, along with its simplicity was the reason behind choosing it for our coding system. There are also efficient algorithms available for computation of the DCT.

There are several possible variations of cosine transforms that could be used in the coding system, but the DCT seems to be the most popular in the literature. The forward DCT can be written as follows;

$$F^*(u,v)=\frac{2}{N}c(u)c(v)\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}F(j,k)\cos\left[\frac{\pi}{N}u(j+0.5)\right]\cos\left[\frac{\pi}{N}v(k+0.5)\right]$$

where $c(x)=1/\sqrt{2}$ for $x=0$

$=1$ for $x=1,2,\ldots N-2,N-1$

F(j,k) denotes the two dimensional intensity array of the original image, and F*(u,v) denotes the respective DCT coefficient array. The reverse DCT is then defined as;

$$F(j,k)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}c(u)c(v)F^*(u,v)\cos\left[\frac{\pi}{N}u(j+0.5)\right]\cos\left[\frac{\pi}{N}v(k+0.5)\right]$$

The objective behind using transform coding, in general, is to use only a limited number of the transform coefficients available. Ideally, one would want to save the highest energy coefficients and truncate the rest to zero. This is where the data compression is achieved, since less coefficients are used in the transform domain than in the spatial domain. For example, let us assume that a 16x16 pixel image digitized to 8 bits is to be coded. A DCT is taken, and it is decided that only 30 DCT coefficients have sufficient energy to be saved. These will also be quantized to 8 bits, and it is assumed that acceptable image quality is preserved when the inverse DCT is taken using the 30 DCT coefficients. Thus, instead of coding 256 spatial domain coefficients, only 30 DCT coefficients would be coded, which

saves a significant number of coding bits. The above example serves to illustrate how image data is compressed using transform coding. More visual and concrete results will be presented later.

Of course, the major problem in a DCT coder is determining which coefficients should be saved, and then remembering their respective locations. The coding system should perform this efficiently without being overly complicated. The details of the coding system are presented in the next section.

III.2 The DCT Coder

For the purpose of studying blocking effect issues, a coding system that only makes use of intraframe coding could be developed, since the proposed blocking effect reduction methods operate on only one frame at a time. However, a coding system would not be complete unless it allowed for some interframe coding in anticipation of working with a sequence of images. Thus, a DCT coder was developed that utilizes both interframe and intraframe coding. A convenient input threshold determines whether or not the system will use interframe coding. In the blocking effect study, this threshold will be set in such a way so as to force only intraframe coding to simplify the conclusions.

After the study is complete, interframe coding will be introduced into the system in Chapter VI to analyze the overall efficiency of the DCT coder equipped with a blocking effect reduction method.

Figure III-1 shows a block diagram of the DCT coder. Each of the components of the coder will now be described in the order in which they appear. It will be assumed that the image is presented to the coder in black and white, and is digitized to 8 bits (256 intensities).

The first component of the DCT coder segments the image into subimages. There are several useful advantages to doing this. First, segmenting the image into smaller blocks allows one to exploit the localized characteristics of the image more effectively to reduce the coded bits. Next, coding several smaller subimages is much easier than forcing the system to code a large image all at once. This is especially true when one considers the hardware implementation of taking a two dimensional DCT. A 16x16 DCT is much more feasible to implement than a 256x256 DCT when considering hardware and computational efficiency in coding an entire image. Finally, one must consider the possibility of coding errors. If many subimages are coded independently to represent an image, a bit error will only affect the segment where it is located. Chances are that the overall image will still be acceptable. However, if a whole image

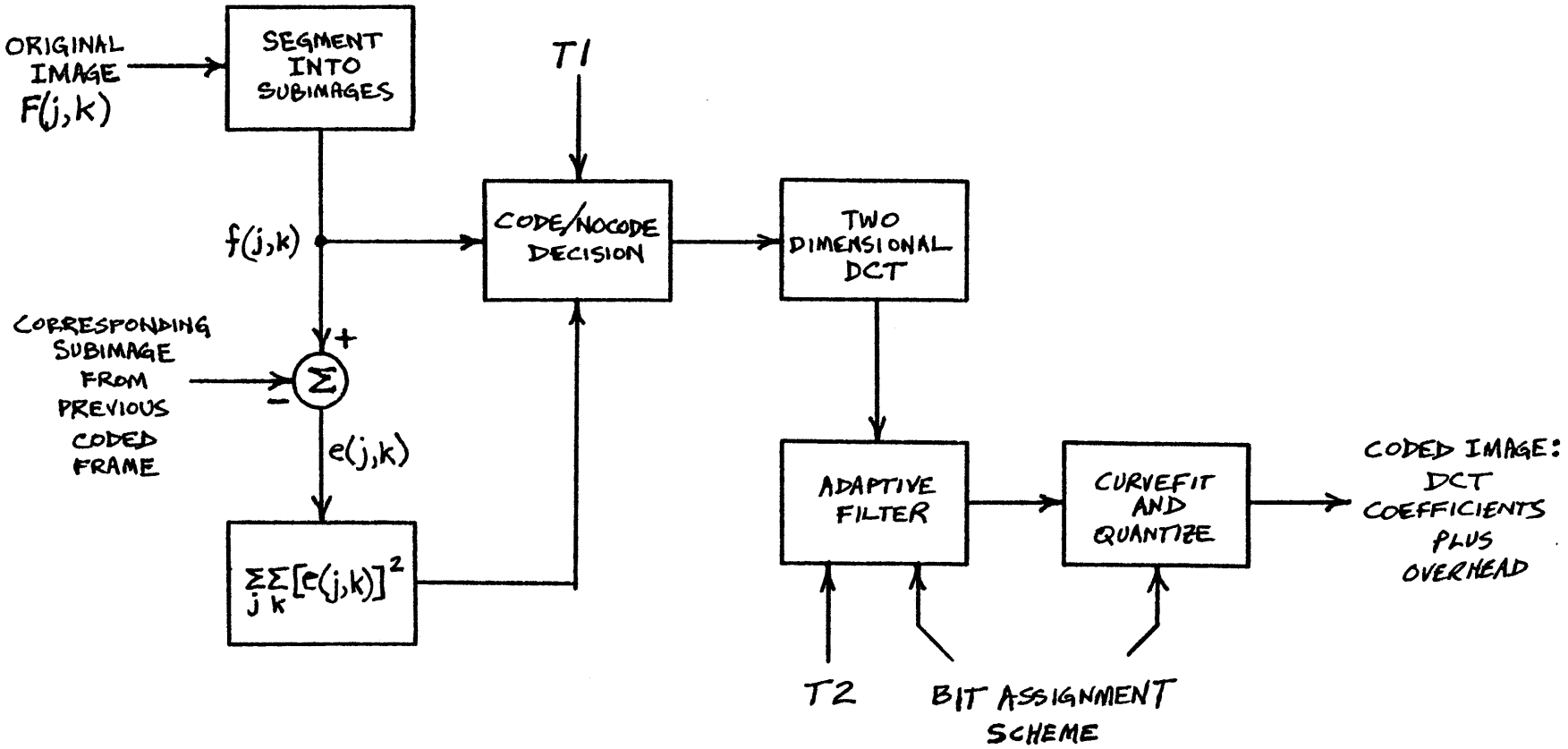


Figure III-1: The DCT Coder.

is transform coded in one big block, a single bit error could conceivably destroy the entire image upon decoding. Thus, segmentation of an image has useful advantages, while the appearance of blocking effects is its major disadvantage. The choice of subimage size should be based on computational efficiency, practical implementation, data compression efficiency, and final image quality to the viewer. With these in mind, it was found that 16x16 pixel subimage blocks work most effectively for an image of size 256x256 pixels.

There are three inputs to the system that indirectly control the number of coded bits, the first of which coordinates the interframe coding potential of the system. This threshold, called T1, controls a code/nocode decision for each subimage. Its mode of operation is that if there is not a significant difference between the previously coded image and the present image, then that subimage section is not coded. In the decoder, the subimage from the previous frame would be repeated. This is the only part of the system that deals with interframe coding, and it works by searching for the sections of an image sequence that remain fairly fixed over some time period, e.g., background scenery. With T1 set to zero, all of the subimages are coded, and no interframe coding takes place. If T1 is set sufficiently high, none of the subimages will be coded, and

the previously coded frame will be repeated at the decoder. Thus, the interframe coding in the system can easily be eliminated by setting T1 equal to zero. This would be done if only one frame is to be coded, and it will be done for the blocking effect study in Chapter IV.

If the decision is made to code a subimage, then a two dimensional fast DCT of the segment is performed. Appendix A describes the computational advantages of using a fast algorithm, as opposed to computing the DCT directly. It also contains the actual fortran programs used for taking the forward and inverse fast DCT in the coding system.

At this point in the coder, it is desirable to filter out some number of DCT coefficients, and preserve only the ones that contain sufficient image energy. This is not a trivial task. The locations of high energy coefficients tend to be extremely spurious, depending on the characteristics of the subimage coded. Each coefficient represents a particular spatial frequency, with the DCT origin representing a DC value of the image. Thus, it is impossible to know a priori where the high energy DCT coefficients will occur for an arbitrary subimage. It is possible, however, to make an educational guess as to the area where there is a high probability of finding the desirable coefficients. A typical subimage will usually be dominated by low spatial frequency components. Figure III-2

indicates the somewhat fuzzy area that usually contains high energy coefficients, and thus, a lowpass filter seems appropriate.

It is clear that for maximum coding efficiency, some type of adaptive filter would be useful to preserve the "right" DCT coefficients. However, the filter cannot be overly complicated, as this would imply costly overhead bits. Two types of filters are commonly used, a threshold filter or a zonal filter. A threshold filter is sensitive to each coefficient, and saves only the higher magnitude ones. It is efficient in saving the "right" coefficients but requires overhead bits to bookkeep the correct locations. A zonal filter, on the other hand, is only sensitive to coefficients located in a prearranged zone, which limits its coding efficiency. Its advantage lies in the fact that it has no overhead, since the locations of the saved coefficients are known a priori. An adaptive filter was developed for the DCT coder that utilizes a combination of threshold and zonal filters, and it was found to be very effective without being too complex. First, a description of the zonal portion of the filter will be discussed, and then the threshold portion will be described.

The first decision in creating a zonal filter is choosing a zonal shape. There did not seem to be any unanimous choice represented in the literature, so a brief

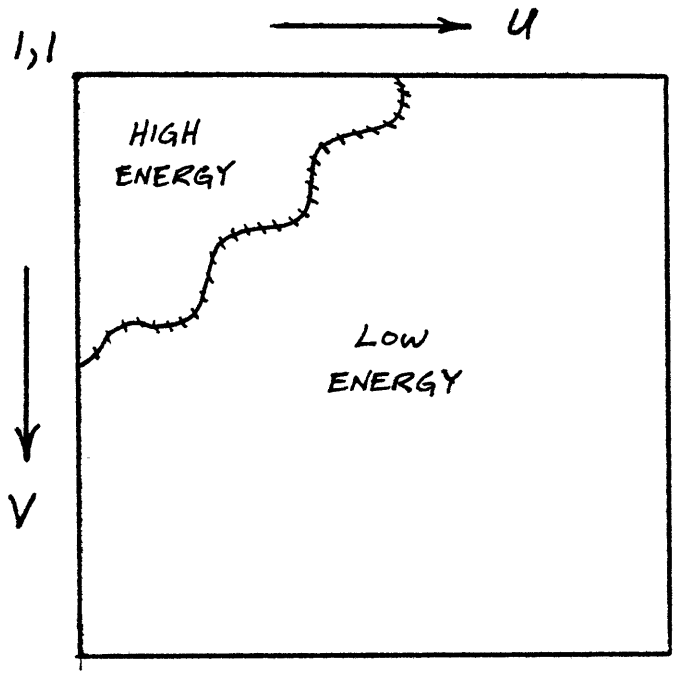


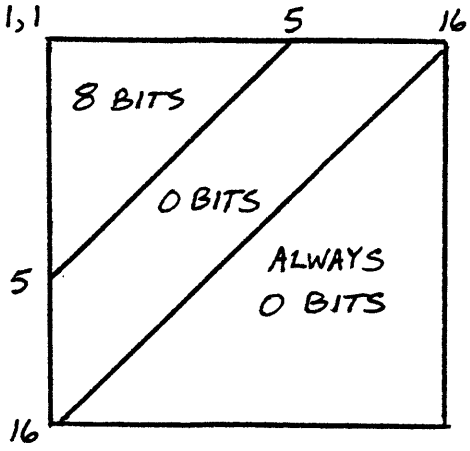
Figure III-2: The likely subimage energy distribution for a typical image.

study was made. Three zonal shapes were examined to determine which shape retains the most image energy for some typical images. The shapes were based on three simple curves, namely, a hyperbola, a circle and a straight line. The study resulted in choosing the linear cutoff region for the coding system. The details of the study are contained in Appendix B, where figure B-1 displays the linear cutoff region shape. With the basic shape of the zonal filter chosen, it was decided to subject the DCT coefficients located inside the zone to a threshold in order to settle on which coefficients to preserve.

The other two inputs that control the coding strategy can now be introduced, namely, T2 and the bit assignment scheme. The threshold, T2, is used for the threshold portion of the adaptive filter, whereas the bit assignment scheme is in command of the bit allocation to each DCT coefficient saved. The threshold, T2, controls the adaptive filter in two ways. In the first way, it compares itself to each DCT coefficient it encounters in a prearranged order. If the coefficient is below the threshold, it is not coded. A one bit code/nocode flag must be used for each coefficient. It should be noted that the DCT origin is treated as a special case and is always coded to 8 bits. The second way that T2 controls the adaptive filter is that it checks itself against each value of a group of pixels.

Pixels are considered in the same group if they all lie in the same diagonal parallel to the linear cutoff region. The adaptive filter keeps the DCT coefficients inside and including the farthest diagonal that has a maximum magnitude above T2, and the zone of the filter is therefore, shrunk to the last diagonal with a sufficient maximum. The filter basically acts as a selective lowpass filter. It is important to note that the bit assignment scheme has priority over T2 in determining the farthest diagonal to code. Figure III-3 shows an example of which coefficients would be saved in a hypothetical 8x8 DCT, with T2 and the bit assignment scheme shown. It will be assumed throughout this thesis that the bit assignment scheme will never code coefficients past the halfway diagonal in the adaptive filter. The filter was designed with low overhead in mind, and in the case of the example shown in figure III-3, 3 bits are needed for the location of the last diagonal coded and 6 bits are needed for the coefficient code/nocode monitor. The four coefficients saved would use another 32 bits, which is determined by the bit assignment scheme.

The general effect of the adaptive filter is to spread the error in the coded image "equally" among the subimages. The subimages that have more activity in them will retain more DCT coefficients, whereas nonactive subimages will code less coefficients.



Bit Assignment

This bit assignment and T2=10 leads to the coefficient coding shown below.

□ - Saved and Coded

△ - Saved but not Coded

All Others not Saved

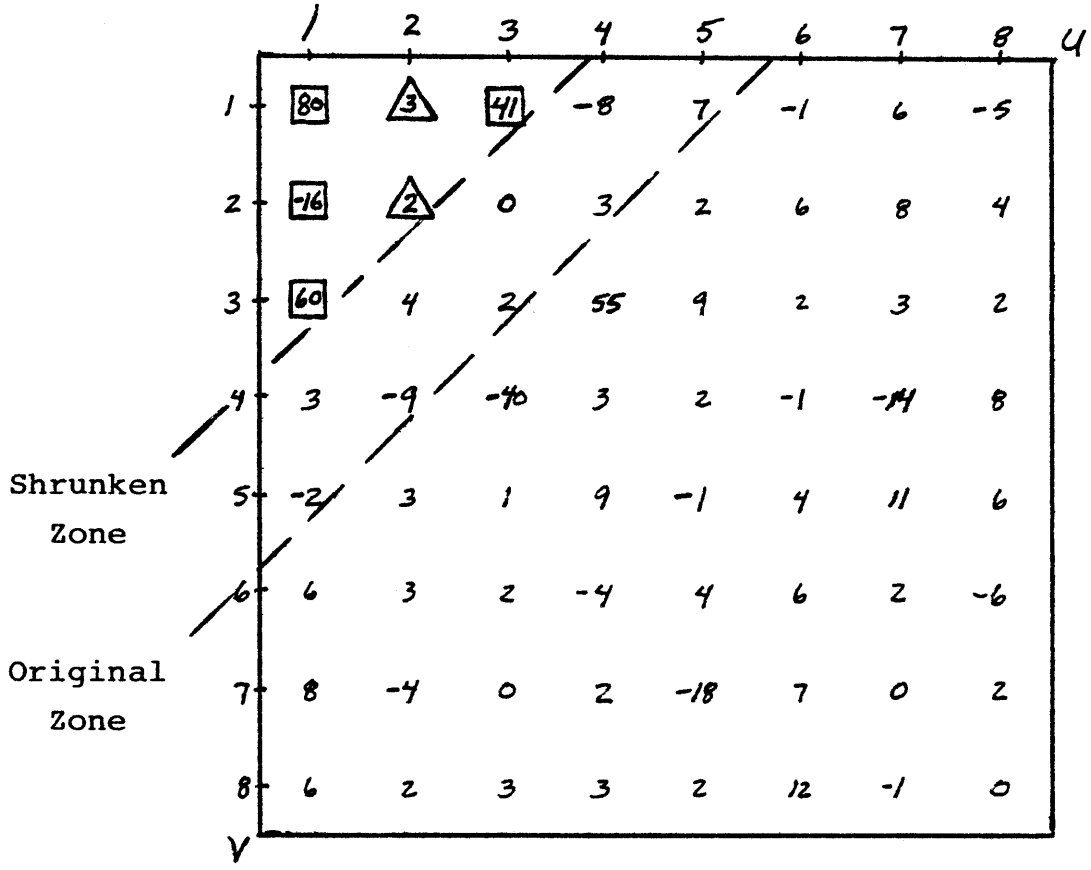


Figure III-3: An example of the coefficients that would be saved in a hypothetical 8x8 DCT subimage.

The final building block of the DCT coder is the quantizer. A simple curvefit is applied to the saved coefficients to utilize the quantizer scale more efficiently. Briefly, a triangular plane is determined that fits over the saved DCT coefficients. In almost all cases, the magnitudes of only two DCT coefficients actually touch the triangular plane, whereas the rest of the coefficients lie below it in magnitude. These two coefficients will define the plane entirely. It would normally take three points to represent a plane, but one degree of freedom is lost because the plane is always assumed to be unbiased towards either frequency axis. Before the DCT coefficients are quantized they are first multiplied by the inverse of the curvefit plane to "equalize" their magnitudes. The uniform quantizer then allocates bits according to the bit assignment scheme. It should be noted that the quantizer scale is modified on both the positive and negative side by T2, since it is known that none of the coefficients saved will have a magnitude less than T2. The curvefit procedure that is incorporated reduces quantization error significantly. It does however create some overhead that is needed for decoding purposes.

Now that the DCT coder has been described, the overhead of the system for each subimage will be assessed. If the system handles subimages in a prearranged order from a fixed

size image, then no overhead is necessary for locations of subimages. The interframe coding block represents one bit of overhead for each subimage. The adaptive filter requires one bit of overhead for each DCT coefficient coded, plus 4 bits for the location of the last diagonal coded, assuming that the subimages are 16x16 pixels. The quantizer uses 12 bits to code the maximum magnitude in the DCT coefficients saved, plus 8 bits for the curvefit procedure. Finally, the system uses four possible prearranged bit assignment schemes, which requires only 2 bits of overhead per frame if it is to remain consistent throughout an image.

Of course, inverse curvefits and inverse DCTs, along with image reconstruction would be required at the decoder. With the DCT coding system developed, it is now possible to study the blocking effect reduction methods using the system. Afterwards, in Chapter VI, some examples will be given that represent the success of the DCT coder equipped with a method to reduce the blocking effects.

CHAPTER IV - THE BLOCKING EFFECT REDUCTION STUDY

Now that a coding system has been developed, the blocking effect study can begin, but first, some important details have to be developed in sections 1, 2 and 3 of this chapter. Then, sections 4, 5 and 6 contain the results of the actual study. Some measurements of image quality will be necessary to intelligently compare and evaluate the blocking effect reduction methods. Section 1 of this chapter discusses the measurements that will be used in the study.

Chapter II presented the two methods proposed for reducing blocking effects. The details of the one pixel overlap method are straightforward and have already been presented. However, the details of the filtering method have not yet been described. Many unanswered questions remain as to the specifications of the lowpass reduction filter to be implemented. A discussion of the development of this filter is presented in section 2 of this chapter.

There are two basic types of error that one may encounter when low bit rate is achieved in the image coding system. They shall be referred to as quantization error and high frequency error. A discussion of these errors and their characteristics is necessary and is presented in section 3 of this chapter. It will help in the

understanding of the individual effectiveness of the two blocking effect reduction methods.

Finally, sections 4, 5 and 6 of this chapter focus on making a coherent study. Section 4 evaluates the reduction methods when the blocking effect is primarily due to high frequency error, whereas section 5 evaluates the methods when the blocking effect is primarily due to quantization error. Section 6 evaluates the methods when a combination of both errors is present. All in all, 16 separate cases are examined. Each case has a photograph associated with it showing a total of four images, namely, the original image, the coded image using no reduction method, the coded image using the overlap method, and the coded image using the filtering method. An image display system was the subject of the photographs. As a side note, the display system had some horizontal linearity problems, causing the images on the left of each photo to be slightly wider than the ones on the right. This is strictly a display problem and has absolutely nothing to do with the coding algorithms. The following diagram illustrates the orientation of the four

images in the photographs for each of the 16 cases.

ORIGINAL IMAGE	NO REDUCTION METHOD
OVERLAP METHOD	FILTERING METHOD

Photo Orientation

IV.1 Measurements of Image Quality

A good measure of image quality is certainly needed if a meaningful blocking effect study is to be performed. There has to be some way of determining how useful the blocking effect reduction methods are. There should also be a means of comparing the two methods to decide which one, if either, is more advantageous. The measurements should ideally be simple and efficient. Unfortunately, due to many undefined variables, there is no known measure of image quality that can decide absolutely how good a decoded image is to a random observer. However, we can use a mean square error measurement, calculated from the original and decoded image, to guide our judgements.

It will be assumed that the DCT coder will never be

able to improve on an original image, or in other words, any original image is the best that it can be. Therefore, the coder cannot enhance an image, and any deviation in a decoded image from the original is considered error. With this in mind, we can make use of a standard measurement, called normalized mean square error (NMSE), which is defined as;

$$NMSE=\frac{\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}\left[F(j,k)-F'(j,k)\right]^2}{\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}\left[F(j,k)\right]^2}$$

where F(j,k) is the intensity array of the original image, and F'(j,k) is the intensity array of the decoded image after it has been processed by the DCT coder. In words, this is the error energy divided by the total energy in the image.

The NMSE is just a number, and in theory, the lower the NMSE is, the better the coded image. However, it is unwise to judge images by this measure alone. Two processed images could have identical NMSE, but their appearance could be totally different. Therefore, the study will also involve a visual inspection of the processed images. A subjective, visual evaluation will accompany the NMSE as a quality measurement in this study. The NMSE will serve as a guide, while the visual inspection will be the deciding factor in choosing the superior image. Also, the bits per pixel that a coded image requires is an important entity that will be

monitored in the study.

In most cases, the NMSE is an excellent indication of the quality a decoded image possesses when compared to the original. In general, if the NMSE is below 0.0025, the decoded image still has fairly good quality. However, image quality degrades rapidly above this, and the image is not very good when the NMSE is greater than 0.0036, although it can still be recognizable. Examples of images will be included in the study, so that the reader can make their own judgements of quality. Note that the resolution and size of the image photographs presented in this thesis are limited, so that certain stated aspects may be difficult to discern. The conclusions and comments in the thesis, however, are based on observation of the images when displayed on a full 18 inch screen.

IV.2 Details of the Filtering Method

The filtering method was introduced in Chapter II, and operates by filtering out the blocking effect discontinuities. Since the discontinuities represent high frequency content, the most straightforward two dimensional filter to use is a lowpass filter. The next issue to be considered is the design of the lowpass filter that will be used. Ideally, the filter should eliminate the blocking

effect discontinuities without affecting the image content itself. Obviously, this is not entirely possible, for if an image contains a very sharp edge, it too, will be altered by a lowpass filter.

There are many approaches that could be taken in designing the lowpass filter. Since we are only interested in focusing on the feasibility of the filtering method, the lowpass filter design will be simple. There is no claim that the filter will be the most optimal for the situation at hand.

Several lowpass filter designs were examined, including both spatial and transform domain filters. Also, the number of pixels around the discontinuities that were filtered was varied. At one extreme, the entire image was filtered, whereas at the other extreme, only the pixels that were directly adjacent to the subimage boundaries were filtered. Several 256x256 pixel, black and white images that exhibited blocking effects from 16x16 pixel subimages were the test subjects for the filters. The images were obtained from the DCT coder. The results on which filter worked best, and how many pixels to filter were mainly based on the relative NMSE, and the final appearance of the image itself.

Three filter formats were applied to the images, namely, a 3x3 gaussian spatial filter, a 5x5 gaussian

spatial filter, and a 256x256 transform domain filter. The transform domain filter was implemented using the DCT. Various cutoff frequencies in all three filters were examined.

It was found in all cases, that only the pixels adjacent to the discontinuities should be filtered to obtain the most improvement in NMSE of the coded image. All three filter formats did very well, with appropriate cutoffs, to reduce blocking effects. The 3x3 and 5x5 gaussian spatial filters obtained almost identical results, while the transform domain filter did slightly better. However, because of the extreme computational effort required by the transform domain filter and the fact that the spatial filters were computationally efficient, it was decided to use the 3x3 gaussian spatial filter for the study. Its simplicity and effectiveness combined made it the best choice for our purposes.

The exact specifications of the 3x3 gaussian spatial filter that will be used are given by;

$$h(m,n)=0.2042\exp[-0.5(m^2+n^2)]$$

$$\text{for } m=-1,0,1$$

$$\text{and } n=-1,0,1$$

Notice that $\sum_{m=-1}^{1}\sum_{n=-1}^{1} h(m,n)=1$.

The filter is convolved with the coded image at the desired pixels. The center point location of the filter (m=n=0) determines which pixel in the image is being filtered. Throughout the study, this filter is only applied at pixels located directly adjacent to the blocking effect discontinuities, since this results in the best success.

IV.3 Error Classification in Coded Images

As was mentioned previously, two distinct types of error are encountered when coding an image with the DCT coder. If enough bits are used to code the image, neither of the errors is noticeable. However, to achieve low bit rates, a cutback in the number of bits coded has to be made. There are two ways to reduce the number of bits used to code the DCT coefficients, each producing its own type of error.

The first way to reduce bit rate is to save less DCT coefficients. This can be accomplished by either dropping the cutoff region or raising the threshold, T2, in the adaptive filter. Since low frequency content is so vital in typical images, the high frequency coefficients are the ones usually sacrificed. This leads to high frequency error, which is characterized by a blurring effect on the image.

The high frequency content of the sharp edges is essentially lost in the adaptive filter of the coder.

The second way to reduce bit rate is to save the same number of DCT coefficients, but reduce the bit allocation to some or all of the coefficients. This results in quantization error, and is noticeable in the image by a sinusoidal rippling effect of intensity in originally solid areas. Edges appear to remain fairly sharp, but they are distorted and a ghost effect can be present.

The two types of error discussed above, namely, high frequency error and quantization error, will be important in the blocking effect study. They represent two distinct situations that can arise, and each is capable of creating a blocking effect. The reduction methods will need to be tested on blocking effect caused by either type of error, as well as a combination of the two.

IV.4 Blocking Effects Resulting from High Frequency Error

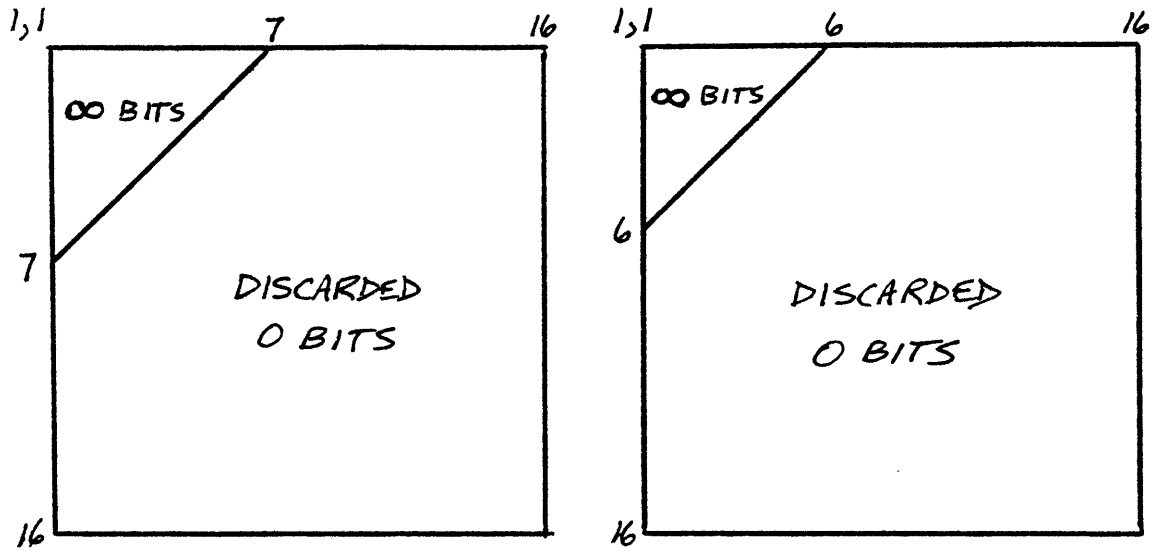
In this part of the chapter, a study of the reduction methods will be made when the blocking effect is caused only from high frequency error. To do this, the DCT coder was set up with perfect quantization. The image that will be used is a 256x256 pixel picture of Walter Cronkite segmented

into 16x16 pixel subimages. Three different maximum cutoff diagonals were used in the adaptive filter, leading to various degrees of blocking effect. The cutoff regions were the 7, 6 and 5 diagonals shown in figure IV-1 for each subimage, which we will refer to as cases 1, 2 and 3, respectively. Note that if the cutoff region is at the 5 diagonal, not more than 15 DCT coefficients could be saved for each segment. In general, less coefficients would probably be salvaged because of the nature of the adaptive filter in the DCT coder.

For each of the three cases, the image was coded with no overlap in segmentation, with one pixel overlap in segmentation, and with no overlap using the filtering method. These represent, respectively, no blocking effect reduction, reduction by the overlap method, and reduction by the filtering method. This comparison procedure will be consistent throughout the blocking effect study. Each of the three cases with its particular cutoff region represented a different severity of blocking effect. Tables IV-1, IV-2 and IV-3 summarize the numerical results.

Table IV-1 - Case 1

Reduction Method	Cutoff Diagonal	NMSE	Bits/Pixel
none	7	0.00153	not applicable
overlap	7	0.00138	not applicable
filtering	7	0.00147	not applicable



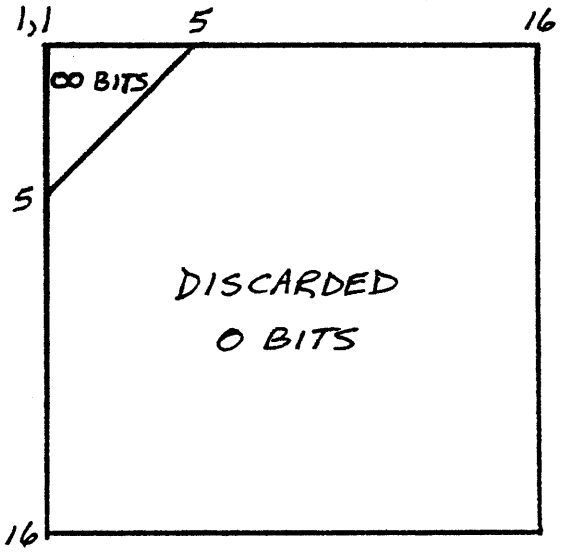


Figure IV-1: The DCT subimage cutoff regions with perfect quantization for cases 1, 2 and 3.

Table IV-2 - Case 2			
Reduction Method	Cutoff Diagonal	NMSE	Bits/Pixel
none	6	0.00257	not applicable
overlap	6	0.00217	not applicable
filtering	6	0.00238	not applicable

Table IV-3 - Case 3			
Reduction Method	Cutoff Diagonal	NMSE	Bits/pixel
none	5	0.00438	not applicable
overlap	5	0.00389	not applicable
filtering	5	0.00399	not applicable

To obtain these results, the threshold T1 was set to zero and T2 was fixed slightly above zero. The bit assignment allowed perfect quantization for the coefficients saved inside the cutoff region.

The results of this part of the study indicate that both methods work extremely well in reducing blocking effects. Figures IV-2, IV-3 and IV-4 show the coded images plus the original in all three cases for comparison (see page 32 for the photo orientation diagram). The blocking effects are definitely present in the three cases, and both reduction methods improved the images. The improvement is indicated slightly by NMSE, whereas it is quite significant upon visual inspection. In cases 1 and 2, the blocking effects were eliminated almost entirely, whereas in case 3, they were reduced, but still somewhat apparent due to the

Figure IV-2: Case 1

Figure IV-3: Case 2

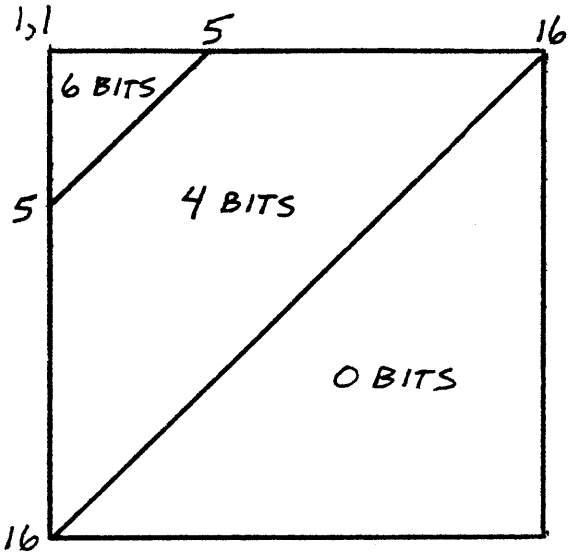
Figure IV-4: Case 3

large amount of distortion in the image. As a final note for this part of the study, both methods worked very well, although the NMSE indicate that the overlap method worked slightly better than the filtering method. This was not apparent in the visual inspection, as both methods seemed to possess equally successful results.

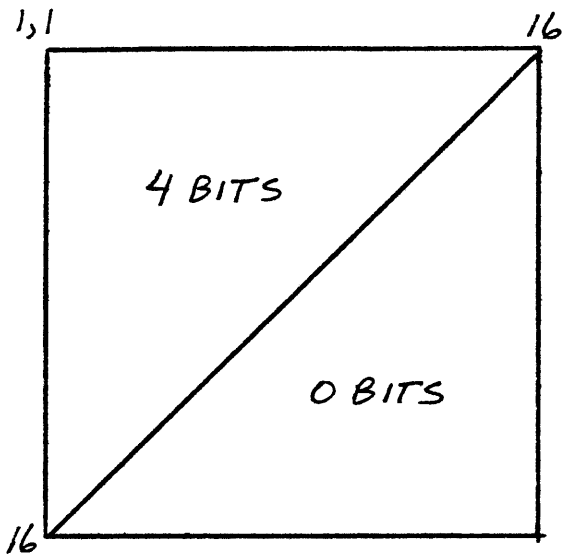
IV.5 Blocking Effects Resulting from Quantization Error

Now we will study the effectiveness of the reduction methods when blocking effect is mainly due to quantization error. Ideally, lowpass filtering should not take place to ensure that the error is strictly quantization error. For our purposes, however, the cutoff region in the adaptive filter was set to the sixteenth diagonal. This means that roughly half of the high frequency coefficients are filtered out in each segment, but for this test image, those filtered are virtually insignificant. The threshold, T2, was fixed at the same value as in section 4 of this chapter to be consistent. This value only filters very small energy coefficients also, so that we can safely conclude that almost all of the error will be from quantization.

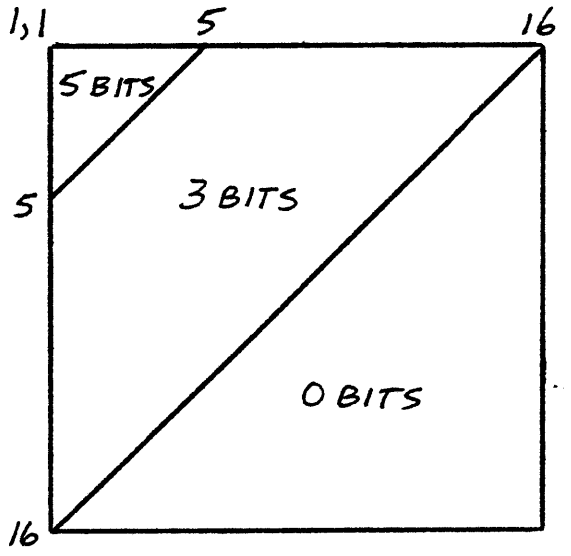
Four different bit assignment schemes were tested, all of which produced blocking effects when no reduction method was used. The bit maps are illustrated in figure IV-5, and



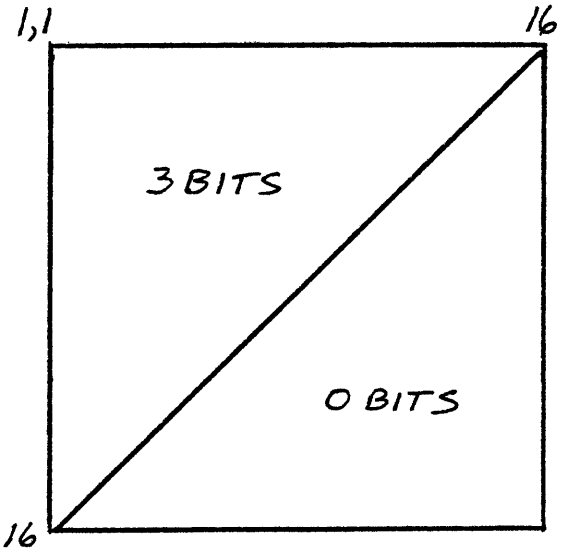
Bit Map 1 - Case 4



Bit Map 2 - Case 5



Bit Map 3 - Case 6



Bit Map 4 - Case 7

Figure IV-5: The subimage bit maps for cases 4 through 7. The 1,1 coefficient is always coded with 8 bits.

will be referred to as cases 4 through 7, respectively. Again, the characteristics in the coded image that indicate the presence of quantization error alone, are the appearance of relatively sharp, but distorted edges and a rippling effect in originally solid areas.

Tables IV-4 through IV-7 summarize the numerical results for the four cases, whereas figures IV-6 through IV-9 illustrate the actual coded images.

Table IV-4 - Case 4

Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	1	0.00062	.836
overlap	1	0.00056	.954
filtering	1	0.00064	.836

Table IV-5 - Case 5

Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	2	0.00121	.775
overlap	2	0.00104	.886
filtering	2	0.00118	.775

Table IV-6 - Case 6

Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	3	0.00144	.742
overlap	3	0.00126	.848
filtering	3	0.00133	.742

Figure IV-6: Case 4

Figure IV-7: Case 5

Figure IV-8: Case 6

Figure IV-9: Case 7

Table IV-7 - Case 7			
Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	4	0.00391	.682
overlap	4	0.00338	.779
filtering	4	0.00354	.682

The results of the study as to which reduction method works better when quantization error alone is present clearly show that the overlap method is more effective. The NMSE indicate this quite readily, and upon close inspection of the images, the reason is evident. The filtering method not only eliminates sharp edges due to discontinuities, but it also smoothed edges in the coded image near the discontinuities. The blurring effect that the filtering method has at the segment boundaries is not readily noticeable on this test image, but it might be more apparent on a different type of image. To expand on this point, another test was performed to judge the degradation that the filtering method might cause.

As a hypothetical situation, suppose that there is an infinite number of bits available for each of the reduction methods. In the case of the overlap method, the original image will also be the coded image. This will not be the case if the filtering method is employed. When the filtering method is used it will not enhance the image, since no blocking effects are present, but instead, it will

degrade the image by blurring the grid pattern that it filters. If the original image is passed through the filtering method, an NMSE of 0.00011 results, which serves as a loose upper bound of degradation that could take place in this particular image using the filtering method.

From this part of the blocking effect study, it is clear that the overlap method does better than the filtering method in terms of image quality. However, an important point that cannot be ignored, is the fact that the overlap method requires more bits when the same bit map is used because of the fact that more segments have to be coded. The reason is that the overlap method requires some pixels to be processed more than once, causing, in this case, approximately a 13 percent increase in the number of segments processed and therefore, bits used. The bits per pixel is an important factor since coding efficiency is our ultimate objective. A full discussion of this important point will be presented later.

IV.6 Blocking Effects Resulting from Both Types of Error

At this point, a comparison of the two reduction methods will be made when the blocking effect is due to a combination of high frequency error and quantization error. This shall be referred to as "combinational" error.

Achieving both types of error in the coded images is easily accomplished, since DCT coefficients can be lowpass filtered, and also have nominal bit allocations. Three different test images that exhibit significantly different characteristics will be used and referred to as Walt, Clock and Village. The images that go along with these names will become obvious. To make this study more fruitful, an attempt was made to equalize the number of coded bits used with each reduction method. Consequently, the overlap method employed a slightly more severe adaptive filter for eliminating higher frequencies, or in other words, less DCT coefficients were coded per segment in the overlap method. This was done to make selection of the best method an easier task because the coding efficiency factor (number of bits) has now been somewhat equalized.

To start with, let us consider the combinational errors on the test image Walt, which was chosen because it represents a typical picture of a person. Many applications would code this type of image, so that it can serve as a fair measure of the coding system. In testing the reduction methods with combinational errors present in Walt, three cases were examined. The bit map was the variable in the cases, which will be referred to as cases 8 through 10. The overlap method used slightly different bit maps (more severely filtered), so as to equalize the coded bits.

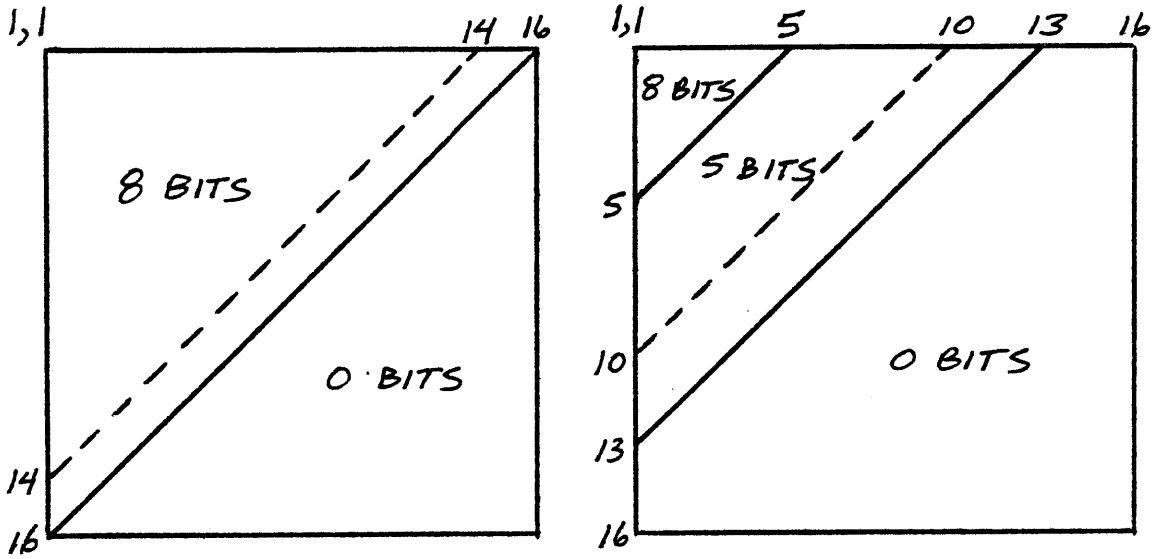
Figure IV-10 shows the three bit maps used, with the dotted lines showing the modified cutoff frequency of the overlap method. Figures IV-11 through IV-13 show the coded images for the three cases. It is clear that case 8 does not contain much error, and thus does not have any noticeable blocking effects. It is useful, however, to see the influence of the reduction methods when no blocking effect is present. Cases 9 and 10 do exhibit blocking effect, with case 10 being the most severe. Tables IV-8 through IV-10 summarize the numerical results for the three cases, respectively.

Table IV-8 - Case 8

Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	5	0.00010	2.23
overlap	5 modified	0.00012	2.28
filtering	5	0.00021	2.23

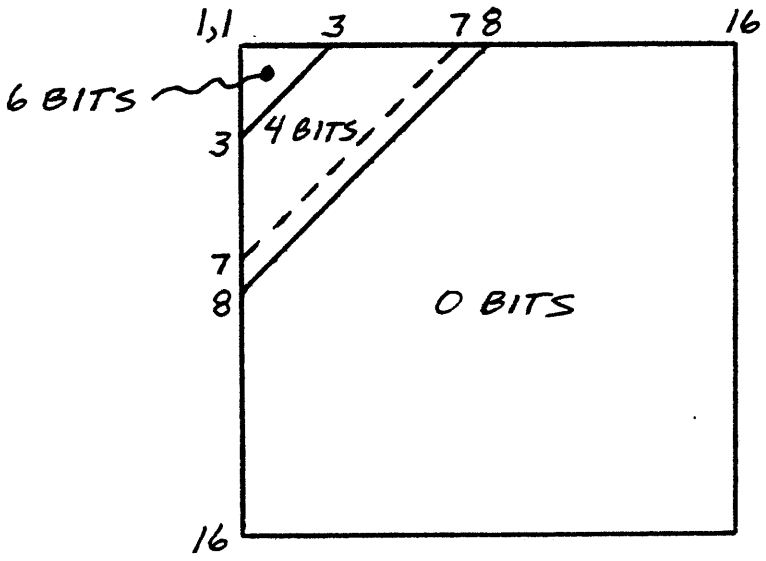
Table IV-9 - Case 9

Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	6	0.00051	.703
overlap	6 modified	0.00060	.724
filtering	6	0.00057	.703



Bit Map 5 - Case 8

Bit Map 6 - Case 9



Bit Map 7 - Case 10

Figure IV-10: The subimage bit maps for cases 8 through 10 on Walt. Dotted lines are cutoffs for overlap method. The 1,1 coefficient is always coded to 8 bits.

Figure IV-11: Case 8

Figure IV-12: Case 9

Figure IV-13: Case 10

Table IV-10 - Case 10			
Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	7	0.00162	.341
overlap	7 modified	0.00178	.364
filtering	7	0.00151	.341

For the test image Walt, it is apparent that the filtering method does better for the lower bit rate schemes. In case 8, the filtering method degrades the high quality coded image more that the overlap method, as is indicated by the NMSE. Again, the reason is that a grid pattern in the image was blurred slightly from application of the reduction filter. It should be noted that the overlap method would not have degraded the image if it was not for the fact that an attempt was made to equalize the coding efficiency (bits/pixel). In case 9, the filtering method seems to work better than the overlap method. The NMSE indicates that using no method would be best in this case, but upon a visual inspection, blocking effect is present, and using a reduction method is justified. In case 10, a low bit rate example, there is no question that the filtering method is best.

The next image used in the study is the image Clock. This image was chosen because of the presence of sharp, vertical and horizontal edges. Three cases were again examined, and are represented by the bit maps shown in

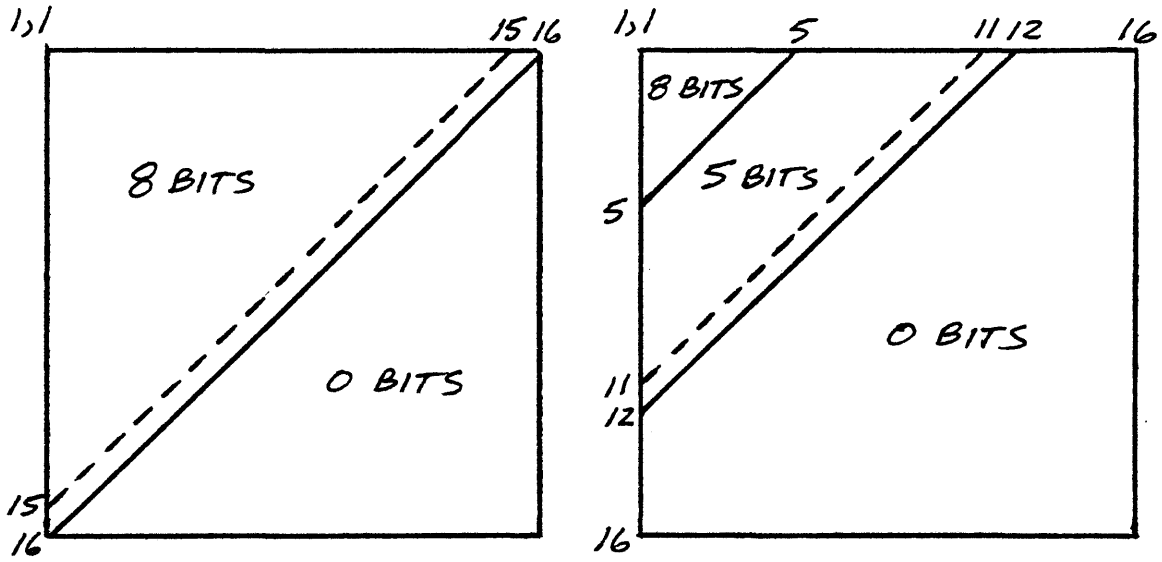
figure IV-14. Coded images for cases 11 through 13 are shown in figures IV-15 through IV-17, respectively. Tables IV-11 through IV-13 summarize the numerical results. Again, an attempt was made to equalize coding efficiency in the overlap method.

Table IV-11 - Case 11			
Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	8	0.00026	2.43
overlap	8 modified	0.00041	2.53
filtering	8	0.00056	2.43

Table IV-12 - Case 12			
Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	9	0.00105	.752
overlap	9 modified	0.00128	.797
filtering	9	0.00123	.752

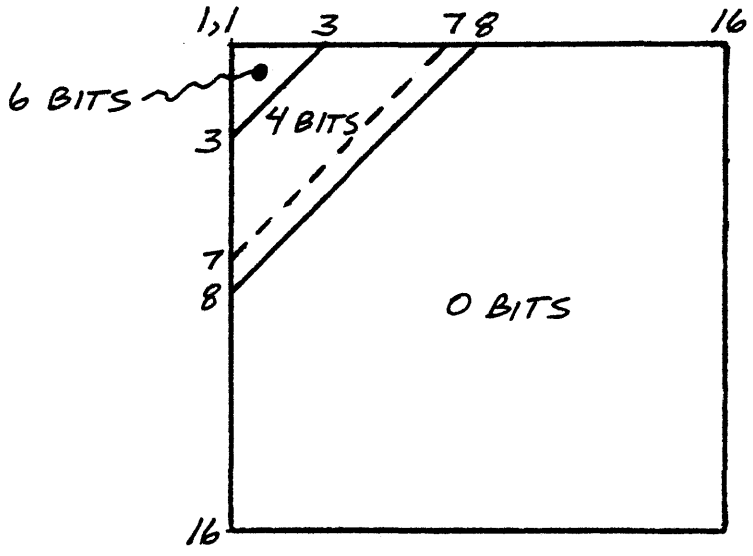
Table IV-13 - Case 13			
Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	10	0.00252	.348
overlap	10 modified	0.00295	.358
filtering	10	0.00255	.348

Next, the results of the study from the image Village will be presented and then a discussion of the results from Clock and Village will be combined. The image Village was



Bit Map 8
Cases 11 & 14

Bit Map 9
Cases 12 & 15



Bit Map 10
Cases 13 & 16

Figure IV-14: The subimage bit maps for cases 11 through 16 on Clock and Village. Dotted lines are cutoffs for the overlap method. The 1,1 coefficient is always coded to 8 bits.

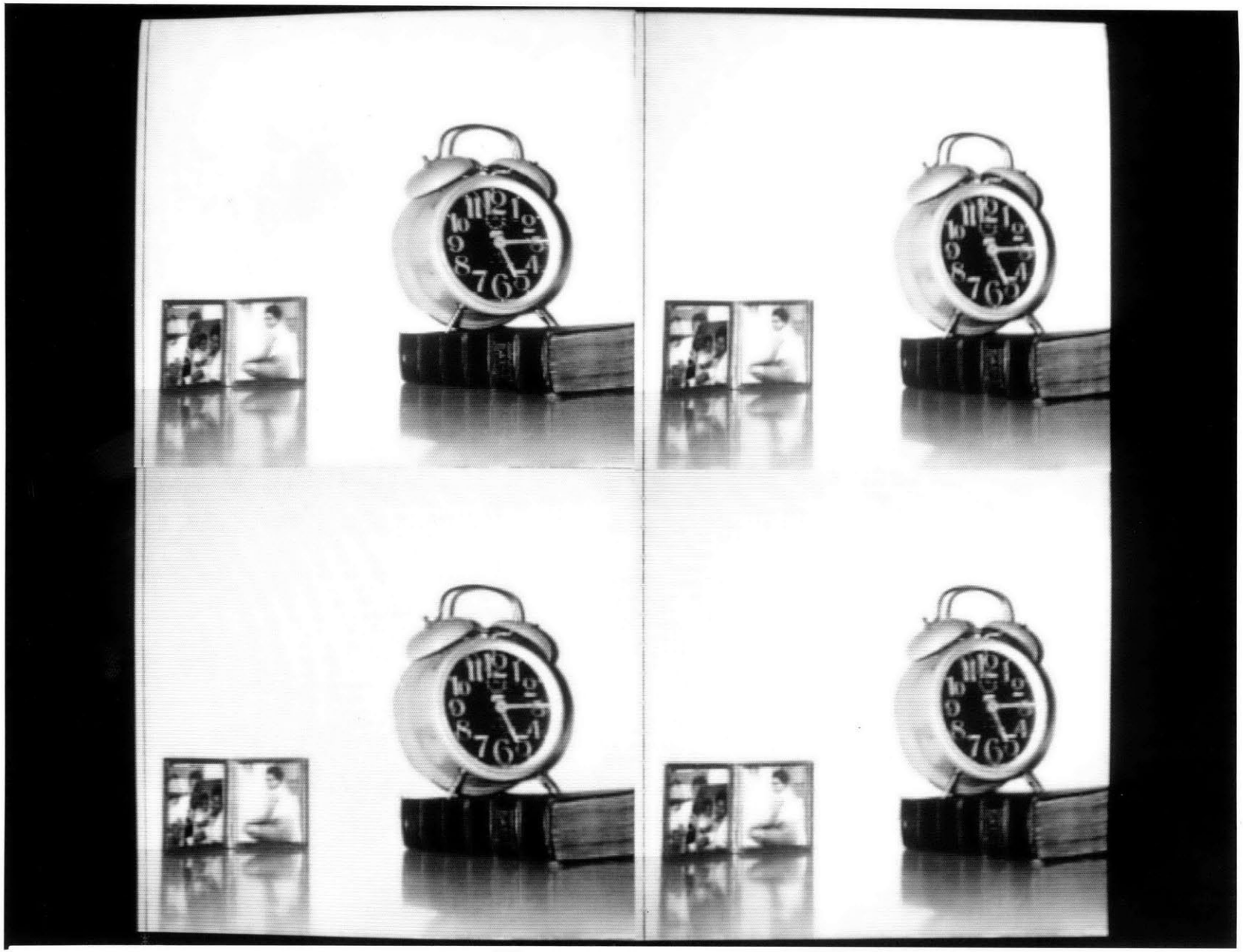


Figure IV-15: Case 11

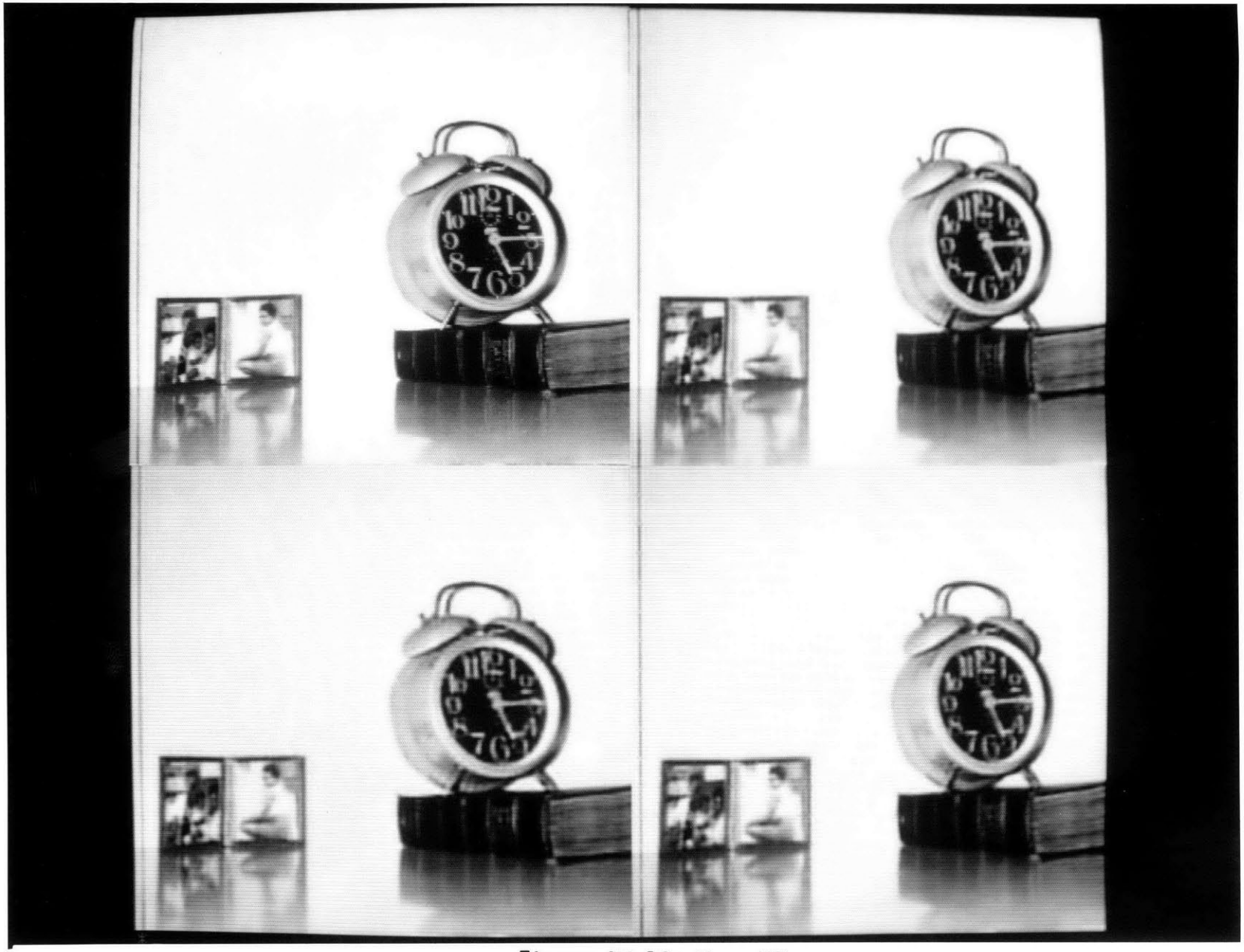


Figure IV-16: Case 12

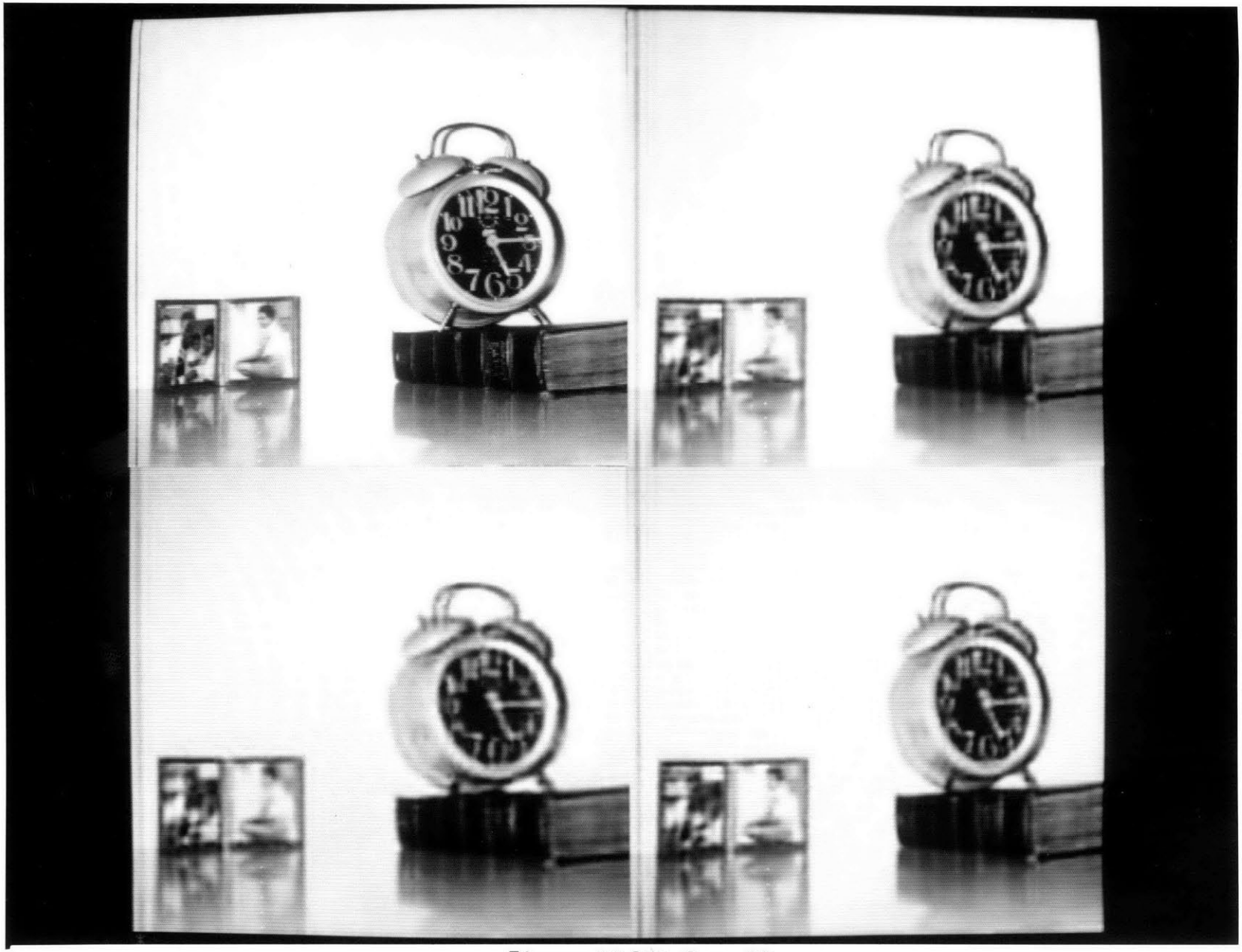


Figure IV-17: Case 13

chosen because it contains an extreme amount of high spatial frequencies. Three cases were again examined, referred to as cases 14 through 16, and are identified again by the bit maps shown in figure IV-14. Coded images for the three cases are shown in figures IV-18 through IV-20, while Tables IV-14 through IV-16 summarize the numerical results.

Table IV-14 - Case 14

Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	8	0.00059	2.81
overlap	8 modified	0.00080	2.80
filtering	8	0.00102	2.81

Table IV-15 - Case 15

Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	9	0.00168	.939
overlap	9 modified	0.00203	.938
filtering	9	0.00195	.939

Table IV-16 - Case 16

Reduction Method	Bit Map Number	NMSE	Bits/Pixel
none	10	0.00411	.404
overlap	10 modified	0.00487	.397
filtering	10	0.00412	.404

The test images Clock and Village produce similar results in terms of the blocking effect study. For low bit

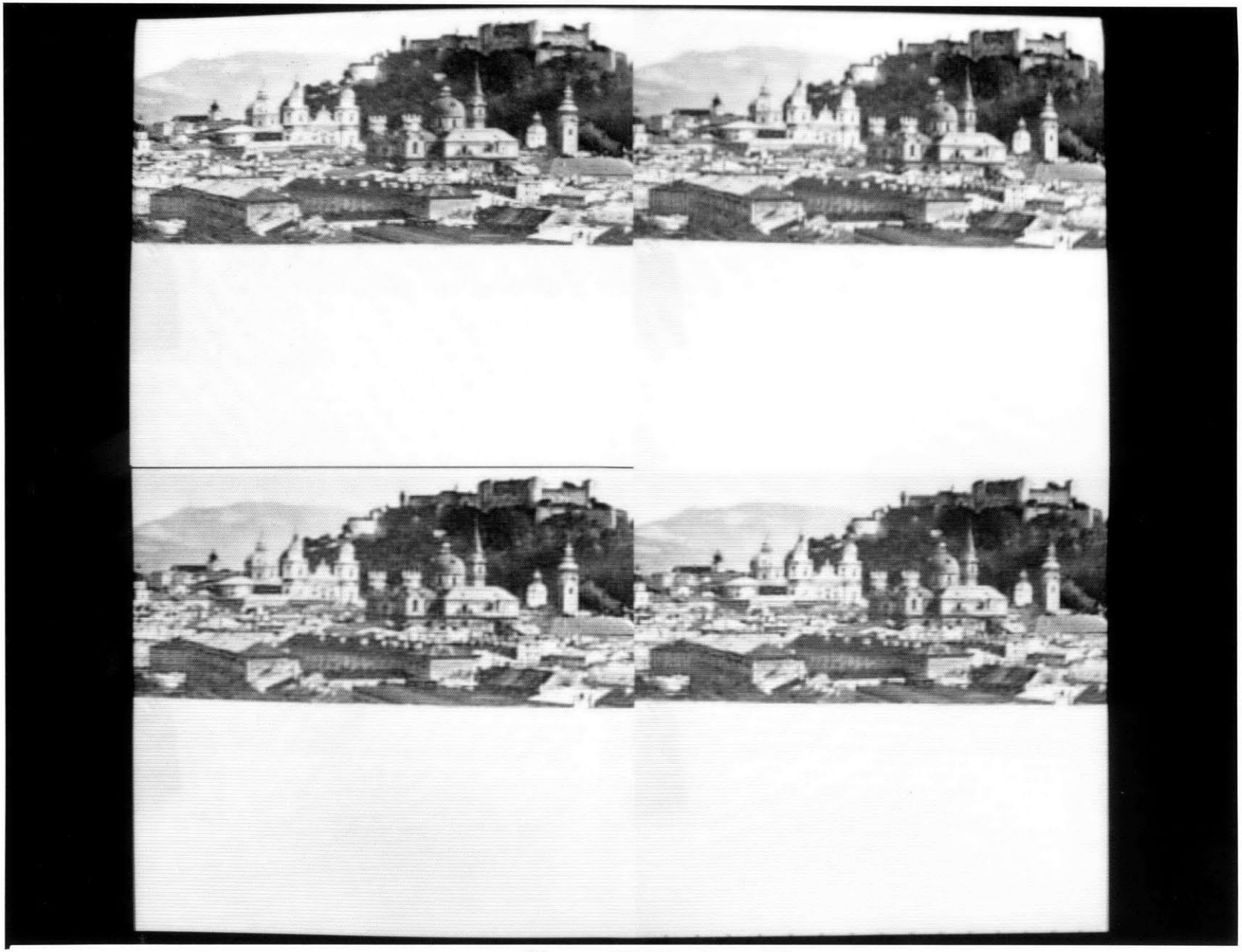


Figure IV-18: Case 14

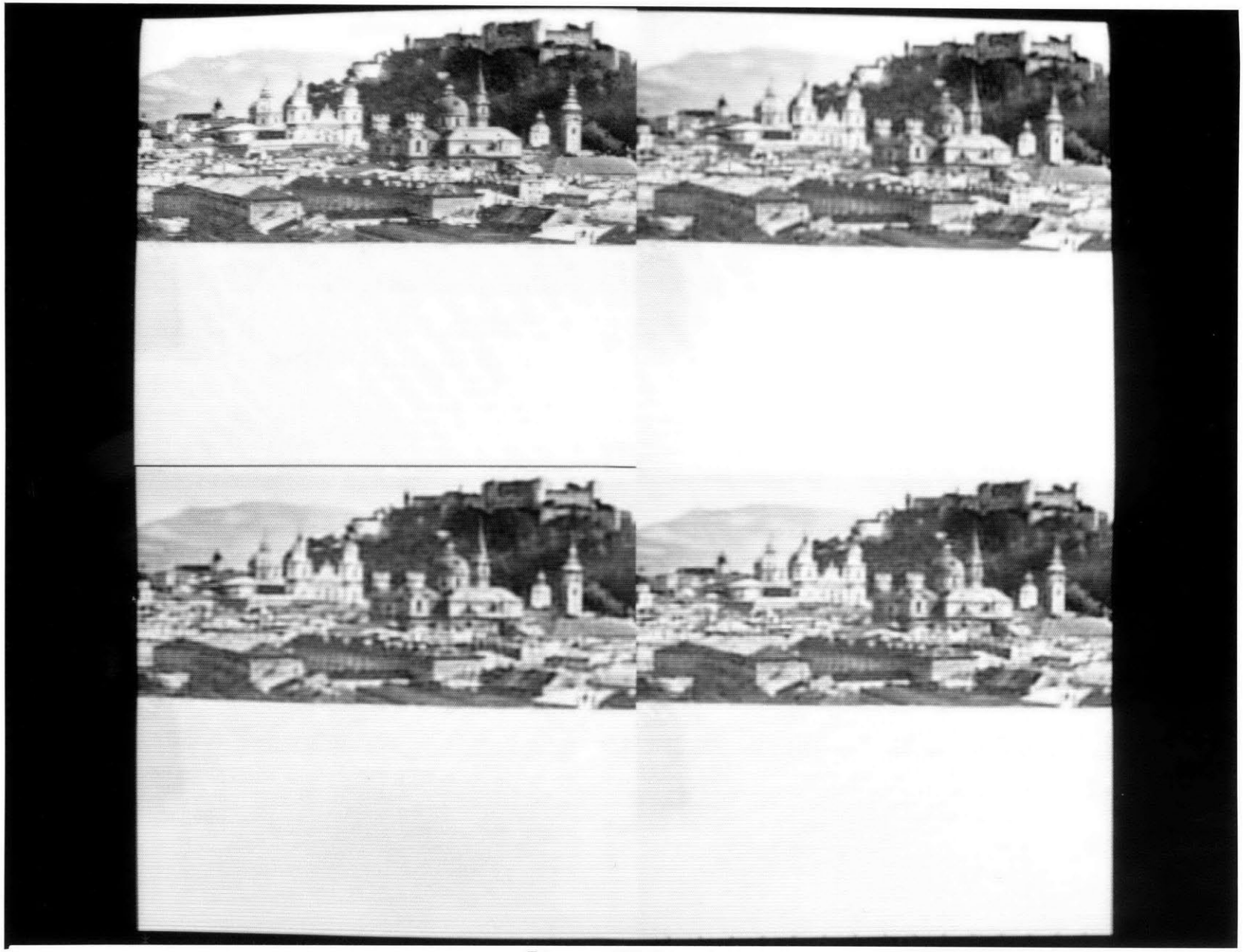


Figure IV-19: Case 15

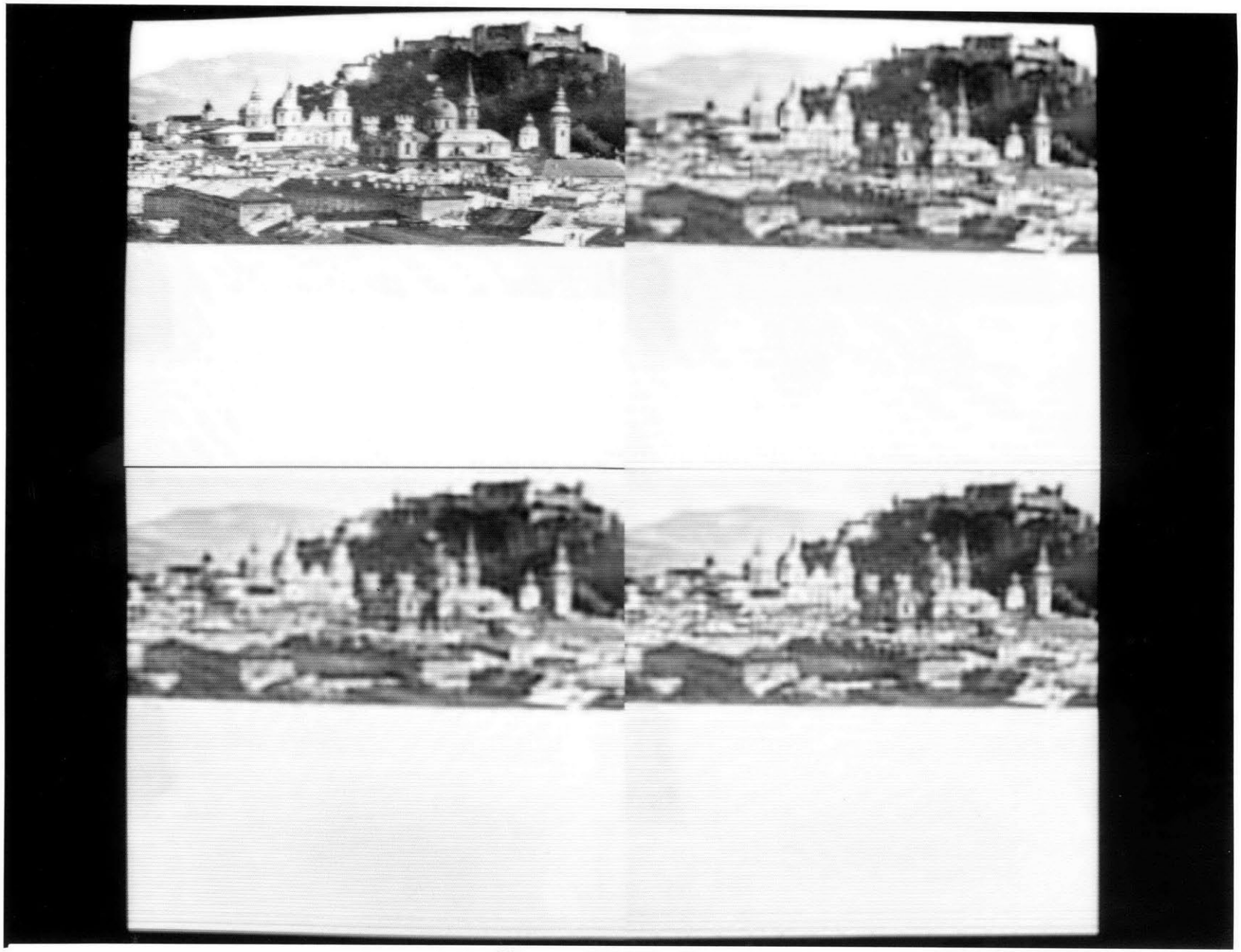


Figure IV-20: Case 16

rate schemes producing combinational blocking effects, the filtering method was clearly superior to the overlap method, as is obvious from cases 13 and 16. However, for higher bit rate schemes that exhibit blocking effects, both methods work very well, but the overlap method may have a slight advantage since it does not directly degrade a sharp high quality image.

Again, if a hypothetical situation were created where an infinite amount of bits were available for coding, the overlap method would have zero error, whereas the filtering method would produce error. More specifically, the image Clock would have an NMSE of 0.00033 if the filtering method were used, and the image Village would have an NMSE of 0.00053.

All in all, the results from the images Clock and Village served to confirm the results obtained from Walt in the blocking effect study. The overall conclusions that can be drawn from the study of the two reduction methods will be discussed in the following chapter.

CHAPTER V - EVALUATION OF THE REDUCTION STUDY RESULTS

The purpose of this chapter is to examine the advantages and disadvantages of the two reduction methods, and then reach some appropriate conclusions. Presentation of this chapter will consist of three parts. The first two sections discuss the advantages and disadvantages of the overlap method and the filtering method, respectively. The final section presents the conclusions of the study.

V.1 Summary of the Overlap Method

In this section, the advantages and disadvantages of the overlap method will be discussed. First, a list of advantages will be given, followed by a description of each.

Advantages of Overlap Method

1) Works very effectively in reducing blocking effects.
2) Simple to implement in coder.
3) Does not directly degrade image.
4) Very little additional computation per subimage in decoding.
5) Very little extra hardware required.

The first advantage is evident from Chapter IV. The overlap method was very successful in reducing or eliminating blocking effects. The method is also simple to

implement in the segmentation process of the coder. Since our images were 256x256 pixels and our subimages were 16x16 pixels, exactly 289 (17x17) equally sized subimages resulted upon segmentation with one pixel overlap. Normally, 256 (16x16) subimages would be present with no overlap. The overlap method does not directly degrade the basic image quality when applied, or in other words, the same spatial frequencies are preserved if the coder control inputs are the same. To emphasize the point, if an infinite number of bits is available to code an image, zero error should occur and does when using the overlap method. Finally, when the image is decoded, very little additional computation per subimage is required. All that needs to be done is to average each of the overlapped pixels. In fact, this also means that very little additional hardware would be needed in the coder-decoder system. Now, the disadvantages will be presented.

Disadvantages of the Overlap Method

1) Increases number of pixels to code, resulting in more subimages.
   a) Increases coded bits.
   b) Increases computation in coder.
2) Can be difficult to implement for other image sizes and/or more complicated segmentation procedures.

The first disadvantage of the overlap method is the

most obvious and the most harmful. Due to the overlap, more pixels have to be processed to code the entire image. In our study, this produced 13 percent more subimages to code (289 versus 256). This leads to approximately a 13 percent increase in coded bits if the same image quality is to be retained. This is undesirable if coding efficiency is the objective. Secondly, this increase in subimages also means an increase in computation time to code an image. This may be a significant problem for some real time applications. Finally, the overlap procedure can be more difficult to implement if the image size is not convenient, or if a more complicated segmentation procedure is necessary.

V.2 Summary of the Filtering Method

In this section, the advantages and disadvantages of the filtering method will be listed. Again, the advantages will be discussed first.

Advantages of the Filtering Method

1) Works very effectively in reducing blocking effects.
2) Simple and efficient to implement.
3) Does not change the coding process.
   a) The number of coded bits remains the same.
   b) Computation in the coder remains the same.
4) More easily switched on and off if so desired.

The filtering method also works extremely well in reducing the blocking effect problem. It is simple and efficient to implement provided the reduction filter is not overly complicated. A strong advantage is that it does not change the coding process, since the filtering method is applied entirely at the decoder. As a result, the number of coded bits and coding computations does not increase. This is an important factor when considering coding efficiency. Finally, since the filtering method is an enhancement procedure that is applied after the complete reconstruction of the coded image, it can readily be switched on and off if the user so desires. Now, the disadvantages will be examined.

Disadvantages of the Filtering Method

1) Can directly degrade an image.
2) Creates filter computations in decoder after image reconstruction.
3) Additional hardware in decoder.

The biggest disadvantage of the filtering method is that it can directly degrade the quality of the image. The reduction filter not only smooths the discontinuous blocking effects, but it can also blur sharp edges in the image. Fortunately, since it was decided to filter only near the discontinuities, those areas are all that is affected. This

is enough, however, to create an "inverse" blocking effect in very high quality images. If an image is abundant with sharp edges, a noticeable grid pattern in the image will be slightly blurred by the reduction filter. Admittedly, this problem can be somewhat alleviated by switching the reduction filter into the decoder only when blocking effects are present, which indicates low bit rate coding. The switch could reasonably be triggered by the bit assignment scheme. To clarify the nature of this degradation problem, if an infinite number of bits were available to code with, using the filtering method would, in general, create error in the decoded image. The second disadvantage of the filtering method is that it requires additional computation in decoding. This is due to the application of the reduction filter. This also means that extra hardware would probably be necessary to implement the system. If a more complicated reduction filter is designed, the last two disadvantages would escalate.

V.3 Conclusions

Both reduction methods have their advantages and disadvantages. Since blocking effects occur only in low bit rate schemes, a decision on which method is better will be made with applications of this nature in mind. After all,

if blocking effects never occur, there is no reason to try and reduce them.

It has been found that the filtering method and overlap method do equally well when the blocking effect is primarily due to high frequency error. The overlap method, however, works best when blocking effects are present and sharp edges have been somewhat preserved in the image, which is the case when quantization error is predominant. This might lead us to choose the overlap method as the overall best reduction method. However, the overlap method has a high cost associated with it for low bit rate systems, namely, an increase in coded bits. When this cost is accounted for by adjusting the control inputs in the coder to "equalize" coding efficiencies, the filtering method is superior for low bit rate applications. This was apparent from the combinational error study in Chapter IV.

At this point, any ideas of improving or expanding the two methods should be considered, as this could conceivably alter the conclusions. First, we shall deal with the overlap method. What if more than one pixel was overlapped? For instance, a two pixel overlap scheme could be developed with some intelligent recombination procedure at the decoder. Even though this is certainly possible, it is believed that this will not help the situation at hand for two reasons. First, overlapping by more than one pixel will

increase the number of subimages, thus decreasing the coding efficiency even further. Second, a significant advantage will probably not be gained in solving the blocking effect problem, since the one pixel overlap method has already done so well. Therefore, it does not seem likely that overlapping by more than one pixel would be rewarding.

Now, let us examine the filtering method. Can any improvements be made? The answer is almost certainly yes. The reduction filter that was used was not necessarily optimal for dealing with blocking effects. It was an extremely simple lowpass filter, which, of course, was one of the reasons it was chosen. A more extensive study could be administered to develop a filter that still effectively reduced the blocking effect, but left the basic image quality untouched. Perhaps a notch filter would even be suitable. The point is that there seems to be room for improvement in the filtering method, solidifying it as the choice for reducing blocking effects in low bit rate coding systems.

In short, both reduction methods worked extremely well to reduce blocking effects, but the filtering method was somewhat superior. In a low bit rate system that segments the image into subimages, using a reduction method is highly recommended. For the DCT coding system with no reduction method, blocking effect became noticeable on the screen at

about 0.7 bits/pixel (only intraframe coding) for Walt. When the filtering method was applied, remnants of the blocking effect did not again become noticeable until about 0.3 bits/pixel. This extra amount of bit rate reduction achieved while avoiding unpleasant blocking effects is very significant, and could be useful in low bit rate applications. The following chapter illustrates the performance of the DCT coder equipped with the filtering method, when it was used with video-conferencing as the application in mind.

CHAPTER VI - APPLICATION TO VIDEO-CONFERENCING

In this chapter, the coding efficiency of the DCT coder will be extended by using its interframe capabilities. Throughout this chapter, the DCT coder will be equipped with the filtering method to reduce blocking effects. A three frame sequence of Walt will be the test material for this chapter. The interframe threshold, T1, will be set in such a way so as to eliminate the coding of subimages that exhibit little or no change from the previous frame. As before, the image size is 256x256 pixels and the subimage size is 16x16 pixels. The original three frame sequence of Walt will be shown along with the sequence coded at two separate bit rates, referred to as sequence examples 1 and 2. Of course, a previous frame was used to code the first frame in each case since the coder is now set up for interframe coding.

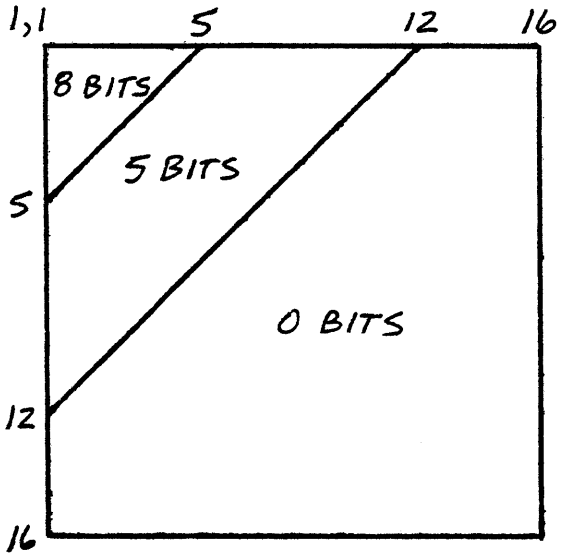
Sequence example 1 was coded with fairly high image quality in mind. Upon visual inspection, it looks very close to the original sequence. Figure VI-1 shows the original sequence, whereas figure VI-2 shows sequence example 1. The bit assignment map for this example is displayed in figure VI-3 for the DCT coefficients. Table VI-1 summarizes the numerical results for example 1.

Frame 1 Frame 2 Frame 3

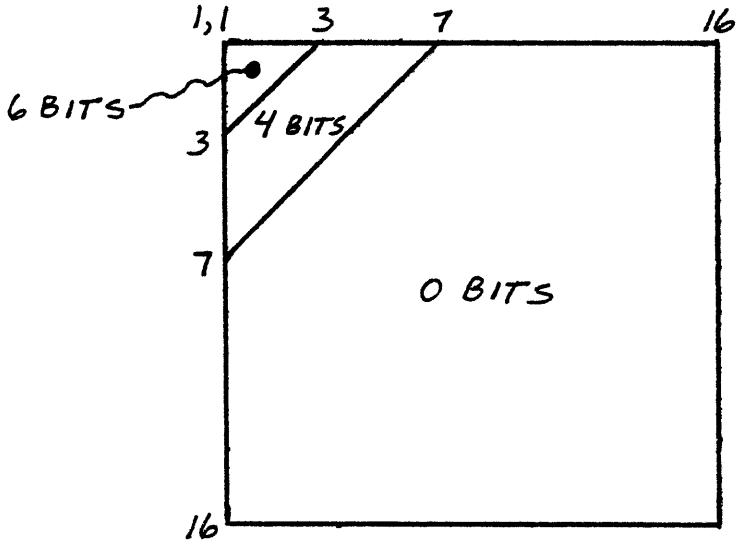
Figure VI-1: Original Walt Sequence.
8 Bits/Pixel.

Frame 1 Frame 2 Frame 3

Figure VI-2: Sequence Example 1.
0.534 Bits/Pixel.



Example 1



Example 2

Figure VI-3: The subimage bit maps for the two sequence examples. The 1,1 coefficient is always coded to 8 bits.

Table VI-1 - Sequence Example 1		
Frame Number	NMSE	Bits/Pixel
1	0.00082	.554
2	0.00055	.532
3	0.00071	.516

The average NMSE was 0.00069 with .534 average bits/pixel. The coded sequence looks quite good when compared to the original.

Sequence example 2 was coded to illustrate the potential of the coder with respect to coding efficiency. Figure VI-4 shows the coded example, which displays the image sequence for a low bit rate set-up. The quality is not as good as the original, but it is still acceptable to a casual viewer. The bit map that was used for this example is also shown in figure VI-3. Table VI-2 displays the numerical results.

Table VI-2 - Sequence Example 2		
Frame Number	NMSE	Bits/Pixel
1	0.00206	.245
2	0.00176	.235
3	0.00184	.233

The average NMSE was 0.00187 with .238 average bits/pixel. This low bit rate is quite impressive when one examines the quality maintained in the coded sequence. Blocking effects

Frame 1 Frame 2 Frame 3

Figure VI-4: Sequence Example 2.
0.238 BITS/Pixel.

have been suppressed quite well by the filtering method.

This chapter was included to illustrate the success of the DCT coder equipped with the filtering method to reduce blocking effects. The above results reveal that good quality can be maintained at coding efficiencies as low as .238 bits/pixel when the image sequence is a person talking. This type of image sequence is the kind that one would expect to find in the environment of video-conferencing. The image quality that the coder retains would most likely be very acceptable in this type of environment. Let us assess the bit rates that could be expected for a video-conferencing system. A coding efficiency of .238 bits/pixel leads to about 15,600 bits/frame. Now, assume that 15 frames per second are to be transmitted. This leads to a bit rate of about 234 Kbits/second, whereas the original image sequence (8 bits/pixel) would have required 7.86 Mbits/second.

CHAPTER VII - SUMMARY

This thesis has presented two possible solutions for reducing blocking effects that can occur due to segmentation procedures in low bit rate image coding systems. After an extensive study, it was found that the filtering method was superior, although both reduction methods work well. The results were based on a system that uses the Discrete Cosine Transform. Finally, the coding system was applied to the specific application of video-conferencing to test its coding potential. In conclusion, note that all of the obtained results were based on the assumption of error free coding and decoding.

BIBLIOGRAPHY

[1] W.H.Chen. "Scene Adaptive Coder", IEEE 1981 Inter. Conf. on Comm., 22.5/1-6, Vol.2,1981.

[2] W.K.Pratt. DIGITAL IMAGE PROCESSING. New York: John Wiley & Sons,1978.

[3] H.C.Andrews. "Two Dimensional Transforms" in Topics in Applied Physics: PICTURE PROCESSING AND DIGITAL FILTERING, Vol. 6, T.S.Huang, Ed., Springer-Verlag, New York,1975.

[4] W.H.Chen,H.Smith,and S.C.Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform", IEEE Trans. Commun., Vol. COM-25, pp.1004-1009,Sept.,1977.

[5] D.J.Fink, Ed., TELEVISION ENGINEERING HANDBOOK, McGraw-Hill, New York,1957.

[6] F.G.Stremler. INTRODUCTION TO COMMUNICATION SYSTEMS. Menlo Park, California, Addison-Wesley,1977.

[7] A.V.Oppenheim & R.W.Schafer. DIGITAL SIGNAL PROCESSING. Prentice-Hall, Englewood Cliffs, N.J.,1975.

[8] L.R.Rabiner & B.Gold. THEORY AND APPLICATION OF DIGITAL SIGNAL PROCESSING. Prentice-Hall, Englewood Cliffs, N.J.,1975.

[9] A.V.Oppenheim. APPLICATIONS OF DIGITAL SIGNAL PROCESSING. Prentice-Hall, Englewood Cliffs, N.J.,1978.

[10] H.Kitajima. "A Symmetric Cosine Transform" in IEEE Trans. on Comp., Vol. C-29, No. 4, April,1980.

[11] A.Rosenfeld & A.C.Kak. DIGITAL PICTURE PROCESSING. Academic Press, New York,1976.

[12] K.N.Ngan. "Adaptive Transform Coding of Video Signals" in IEEE Proc., Vol. 129, Pt.F,No.1,February,1982.

[13] M.Ghanbari & D.E.Pearson. "Fast Cosine Transform Implementation for Television Signals" in IEEE Proc., Vol. 129, Pt. F, No. 1, February,1982.

[14] A.Z.Meire & E.Yudilevich, "A Pinned Sine Transform Image Coder", IEEE Trans. Commun., Vol.Com-29, No.12, 1728-1735, December,1981.

APPENDIX A - TWO DIMENSIONAL FAST DCT AND INVERSE DCT

A two dimensional fast Discrete Cosine Transform program was written in fortran, along with its inverse counterpart. The program is based on Chen's [4] one dimensional algorithm, and works for any square, power of two image. The computational savings of the fast DCT (FDCT) over the direct DCT (DDCT) are staggering. A two dimensional DDCT leads to $54N^3+2N+4$ multiplications, where the image size is NxN pixels. On the other hand, the FDCT uses only $2N^2LOG_2N-3N^2+8N$ multiplications. Table A-1 shows a comparison of multiplications between the FDCT and the DDCT for various image sizes.

Table A-1 - Number of Multiplications

DCT Size	DDCT	FDCT	Savings Factor
4x4	3,468	48	72
8x8	27,668	256	108
16x16	221,220	1,408	157
32x32	1,769,540	7,424	238
64x64	14,155,908	37,376	379
128x128	113,246,468	181,248	625
256x256	905,970,180	854,016	1,061

In writing the FDCT and inverse FDCT (FIDCT) programs, effort was made to make them as efficient as possible. Both are set up as subroutines with two inputs, namely, the image

intensity array and a variable, LN, which indicates the size of the image array. The programs will work on any square numerical array that is some power of two in size. However, both programs are currently set up to process up to a 64x64 FDCT and FIDCT. The limiting factor is the dimensioning of arrays. Any larger size image can be processed if dimensioning is properly taken care of in the programs.

The FDCT and FIDCT fortran subroutines are included on the following ten pages. Comments are included in the programs themselves.

fdct.f

```
      subroutine fdct(pict,ln)
c
c     This computes the DCT using a fast algorithm as pro-
c     posed by Chen (IEEE Trans. on Comm., vol. COM 25, no.9,
c     Sept77,p1004). The scale factor 2/n is left off to save
c     multiplications. It is instead included in the IDCT.
c     The array size is n*n, where n=2**ln, and is located in
c     pict(point 1,1 being the origin). The range of integer ln
c     is 2 through 6. This can be extended to larger values but
c     dimensioning in the program must be increased.
c
      dimension coco(5,63),sico(5,63),pict(64,64),temp(64)
      n=2**ln
      if(ln.lt.2)go to 100
      if(ln.gt.6)go to 100
c     The above line is only necessary for dimensioning purposes.
      lns1=ln-1
      lns2=ln-2
c
c     Calculation of necessary sine and cosine coefficients.
c     Arrays are used to save multiplications in the program.
c     Note that the arrays get only partially filled.
c
      halfpi=2.0*atan(1.0)
      k1=4
      do 1 i1=1,lns1
      do 2 i2=1,k1,2
      x=halfpi*i2/k1
      coco(i1,i2)=cos(x)
    2 sico(i1,i2)=sin(x)
    1 k1=k1+k1
      pifour=cos(halfpi/2.0)
c
c     The actual DCT algorithm begins here. Do loop 3 controls
c     each column of the image going through the DCT. Do loop
c     17 transposes the image. Do loop 20 controls the 2-D
c     trasform. The DCT algorithm is broken down into three
c     consecutive parts, namely; (1)The initial plus/minus
c     butterfly staircase-see Chen-do loop 5. (2)Calculation
c     of pifour butterflies and plus/minus-sine/cosine butter-
c     flies up to but not including the last column bit-reversed
c     sine/cosine butterflies-do loop 6. (3)Calculation of the
c     last column sine/cosine, bit-reversed butterflies-do loop 14.
c
      do 20 i20=1,2
      do 3 i3=1,n
      do 4 i4=1,n
    4 temp(i4)=pict(i3,i4)
c
c     Initial plus/minus butterfly staircase.
c
      nn=n
      m=n
```

fdct.f

```
      md2=m/2
      do 5 i5=1,lns1
      do 19 i19=1,md2
      ftemp=temp(i19)+temp(m)
      temp(m)=temp(i19)-temp(m)
      temp(i19)=ftemp
   19  m=m-1
    5  md2=md2/2
c
c     Calculation of first two DCT coefficients (bit-reversed).
c
      ftemp=(temp(1)+temp(2))*pifour
      temp(2)=(temp(1)-temp(2))*pifour
      temp(1)=ftemp
c
c     Calculation of pifour butterflies (do loop 7) and plus/
c     minus-sine/cosine butterflies (do loop 8). Do loop 6
c     controls both.
c
      if(n.lt.5)go to 50
      do 6 i6=1,lns2
      nx=nn/8
      nn5=5*nx+1
      nn7=7*nx
      do 7 i7=1,nx
      ftemp=(temp(nn7)-temp(nn5))*pifour
      temp(nn7)=(temp(nn7)+temp(nn5))*pifour
      temp(nn5)=ftemp
      nn5=nn5+1
    7  nn7=nn7-1
      jsccol=1
      jcount=jsccol
      nxx2=nx+nx
      nxx4=nxx2+nxx2
      k3=3
      k4=4
      nn6=3*nn/4
      do 8 i8=1,lns2
      j1=nn/2+1
      j3=j1+nxx2
      j2=j3-1
      j4=j2+nxx2
      do 9 i9=1,jcount
      do 10 i10=1,nx
      ftemp=temp(j1)+temp(j2)
      temp(j2)=temp(j1)-temp(j2)
      temp(j1)=ftemp
      ftemp=temp(j4)-temp(j3)
      temp(j4)=temp(j4)+temp(j3)
      temp(j3)=ftemp
      j1=j1+1
      j2=j2-1
      j3=j3+1
   10  j4=j4-1
      j1=j1+nxx4-nx
```

fdct.f

```
      j2=j2+nxx4+nx
      j3=j3+nxx4-nx
    9 j4=j4+nxx4+nx
      if(nxx2.lt.3)go to 49
      nxx4=nxx2
      nxx2=nx
      nx=nx/2
      j5=nn/2+nx+1
      j7=j5+nx
      j6=nn-nx
      j8=j6-nx
      do 11 i11=k3,k4
      l=i11-1
      nbr=0
      l2=1
   27 l1=l2
      l2=l2+l2
      if(mod(l,l2)-l1)29,28,28
   28 nbr=nbr+k4/l2
   29 if(l2-k4)27,30,30
   30 if(j5.gt.nn6)go to 40
      do 12 i12=1,nx
      pict(i3,j5)=temp(j6)*sico(jsccol,nbr)-temp(j5)*coco(jsccol,nbr)
      j5=j5+1
      j6=j6-1
      pict(i3,j7)=-temp(j7)*sico(jsccol,nbr)-temp(j8)*coco(jsccol,nbr)
      j7=j7+1
   12 j8=j8-1
      go to 45
   40 do 13 i13=1,nx
      temp(j5)=temp(j5)*coco(jsccol,nbr)-temp(j6)*sico(jsccol,nbr)
      temp(j6)=pict(i3,j6)
      j5=j5+1
      j6=j6-1
      temp(j7)=temp(j7)*sico(jsccol,nbr)+temp(j8)*coco(jsccol,nbr)
      temp(j8)=pict(i3,j8)
      j7=j7+1
   13 j8=j8-1
   45 j5=j5+nxx4-nx
      j6=j6-nxx4+nx
      j7=j7+nxx4-nx
   11 j8=j8-nxx4+nx
      k3=k4+1
      k4=k4+k4
      jsccol=jsccol+1
    8 jcount=jcount+jcount
   49 continue
    6 nn=nn/2
c
c     Calculation of last column, bit-reversed, sine/cosine butterflies.
c
   50 jsccol=1
      k3=3
      k75=k3
      k4=4
```

fdct.f

```
      do 14 i14=1,lns1
      i51=k4
      do 15 i15=k3,k4
      l=i15-1
      nbr=0
      l2=1
   31  l1=l2
      l2=l2+l2
      if(mod(l,l2)-l1)33,32,32
   32  nbr=nbr+k4/l2
   33  if(l2-k4)31,34,34
   34  if(i15.gt.k75)go to 60
      pict(i3,i15)=temp(i15)*sico(jsccol,nbr)+temp(i51)*coco(jsccol,
     * nbr)
      go to 65
   60  temp(i15)=temp(i15)*coco(jsccol,nbr)-temp(i51)*sico(jsccol,nbr)
      temp(i51)=pict(i3,i51)
   65  i51=i51-1
   15  continue
      jsccol=jsccol+1
      k75=k75+k75
      k3=k4+1
   14  k4=k4+k4
c
c     Now, bit-reverse the sequence for each column in the image.
c
      do 16 i16=1,n
      l=i16-1
      nbr=0
      l2=1
   41  l1=l2
      l2=l2+l2
      if(mod(l,l2)-l1)43,42,42
   42  nbr=nbr+n/l2
   43  if(l2-n)41,44,44
   44  nbr=nbr+1
   16  pict(i3,i16)=temp(nbr)
    3  continue
c
c     Transpose the image array.
c
      do 17 i17=1,n
      do 18 i18=1,n
      if(i17.le.i18)go to 17
      ftemp=pict(i17,i18)
      pict(i17,i18)=pict(i18,i17)
   18  pict(i18,i17)=ftemp
   17  continue
   20  continue
      go to 90
      print89
   89  format(/,2x,'Discrete Cosine Transform completed')
      go to 90
  100  print101
  101  format(/,2x,'Error... ln not in range of 2 thru 6.')
```

fdct.f

```
   90  return
       end
```

fidct.f

```
      subroutine fidct(pict,ln)
c
c     This computes the IDCT using a fast algorithm as proposed
c     by Chen (IEEE Trans. on Comm., Vol. COM 25, no.9, Sept77,
c     p.1004). An extra scale factor of 2/n (4/n**2 all together)
c     is used with the IDCT to compensate for not using it in the
c     forward DCT. This is done to save multiplications in the
c     DCT. The array size is n*n, where n=2**ln, and is located
c     in pict (point 1,1 being the origin). The range of integer
c     ln is 2 through 6. This can be extended to larger values,
c     but dimensioning in the program must be increased.
c
      dimension coco(5,63),sico(5,63),pict(64,64),temp(64)
      n=2**ln
      if(ln.lt.2)go to 100
      if(ln.gt.6)go to 100
c     The above line is only necessary for dimensioning purposes.
      lns1=ln-1
      lns2=ln-2
      const1=n*n/4.0
c
c     normalize the DCT-IDCT pair by multiplying by 4/n**2.
c
      do 21 i21=1,n
      do 22 i22=1,n
   22 pict(i21,i22)=pict(i21,i22)/const1
   21 continue
c
c     Calculation of necessary sine and cosine coefficients.
c     This look-up table is used to save multiplications in the
c     program. Note that the arrays are only partially filled.
c
      halfpi=2.0*atan(1.0)
      k1=4
      do 1 i1=1,lns1
      do 2 i2=1,k1,2
      x=halfpi*i2/k1
      coco(i1,i2)=cos(x)
    2 sico(i1,i2)=sin(x)
    1 k1=k1+k1
      pifour=cos(halfpi/2.0)
c
c     The actual IDCT algorithm begins here. Do loop 3 controls
c     each column of the image going through the IDCT. Do loop
c     17 transposes the image. Do loop 20 controls the 2-D
c     transform. The IDCT algorithm is broken down into three
c     consecutive parts, namely; (1)Calculation of the sine/
c     cosine, bit-reversed butterflies - see Chen - do loop 5.
c     (2)Calculation of pifour butterflies and plus/minus-sine/
c     cosine butterflies-do loop 7. (3)Calculation of the plus/
c     minus butterfly staircase-do loop 14.
c
      do 20 i20=1,2
```

fidct.f

```
      do 3 i3=1,n
c
c     Bit-reverse each column of the transformed image.
c
      do 4 i4=1,n
      l=i4-1
      nbr=0
      l2=1
 71   l1=l2
      l2=l2+l2
      if(mod(l,l2)-l1)73,72,72
 72   nbr=nbr+n/l2
 73   if(l2-n)71,74,74
 74   nbr=nbr+1
  4   temp(i4)=pict(i3,nbr)
      ftemp=(temp(1)+temp(2))*pifour
      temp(2)=(temp(1)-temp(2))*pifour
      temp(1)=ftemp
c
c     Calculation of bit-reversed, sine/cosine butterflies.
c
      jsccol=1
      k1=jsccol
      kt=3
      kb=4
      m=kb
      do 5 i5=1,lns1
      do 6 i6=1,k1
      l=kt-1
      nbrt=0
      l2=1
 31   l1=l2
      l2=l2+l2
      if(mod(l,l2)-l1)33,32,32
 32   nbrt=nbrt+m/l2
 33   if(l2-m)31,34,34
 34   l=kb-1
      nbrb=0
      l2=1
 41   l1=l2
      l2=l2+l2
      if(mod(l,l2)-l1)43,42,42
 42   nbrb=nbrb+m/l2
 43   if(l2-m)41,44,44
 44   ftemp=sico(jsccol,nbrt)*temp(kt)-sico(jsccol,nbrb)*temp(kb)
      temp(kb)=coco(jsccol,nbrt)*temp(kt)+coco(jsccol,nbrb)*temp(kb)
      temp(kt)=ftemp
      kt=kt+1
  6   kb=kb-1
      jsccol=jsccol+1
      k1=k1+k1
      kt=m+1
      m=m+m
  5   kb=m
c
```

fidct.f

```
c     Calculation of plus/minus-sine/cosine butterflies (do loop 8)
c     and pifour butterflies (do loop 13). Do loop 7 controls both.
c
      if(n.lt.5)go to 80
      nn=8
      nnh=4
      nnq=2
      nne=1
      do 7 i7=1,lns2
      nx=1
      nxx2=2
      nxx4=4
      kt=nne
      kb=nnq-1
      nnq1=nnq
      jsccol=i7-1
      do 8 i8=1,i7
      j4=nn
      j2=nn-nxx2
      j3=j2+1
      j1=j3-nxx2
      do 9 i9=1,nnq1,2
      do 10 i10=1,nx
      ftemp=temp(j1)+temp(j2)
      temp(j2)=temp(j1)-temp(j2)
      temp(j1)=ftemp
      ftemp=temp(j4)-temp(j3)
      temp(j4)=temp(j3)+temp(j4)
      temp(j3)=ftemp
      j1=j1+1
      j2=j2-1
      j3=j3+1
   10 j4=j4-1
      j1=j1-nxx4-nx
      j2=j2-nxx4+nx
      j3=j3-nxx4-nx
    9 j4=j4-nxx4+nx
      if(i8.eq.i7)go to 70
      j6=nn-nx
      j5=nnh+nx+1
      j8=j6-nx
      j7=j5+nx
      do 11 i11=1,nnq1,4
      nbrt=0
      l2=1
   51 l1=l2
      l2=l2+l2
      if(mod(kt,l2)-l1)53,52,52
   52 nbrt=nbrt+nnq1/l2
   53 if(l2-nnq1)51,54,54
   54 nbrb=0
      l2=1
   61 l1=l2
      l2=l2+l2
      if(mod(kb,l2)-l1)63,62,62
```

fidct.f

```
 62  nbrb=nbrb+nnq1/12
 63  if(12-nnq1)61,64,64
 64  do 12 i12=1,nx
     ftemp=temp(j6)*coco(jsccol,nbrb)-temp(j5)*coco(jsccol,nbrt)
     temp(j6)=temp(j5)*sico(jsccol,nbrt)+temp(j6)*sico(jsccol,nbrb)
     temp(j5)=ftemp
     ftemp=-temp(j7)*sico(jsccol,nbrt)-temp(j8)*sico(jsccol,nbrb)
     temp(j8)=temp(j8)*coco(jsccol,nbrb)-temp(j7)*coco(jsccol,nbrt)
     temp(j7)=ftemp
     j5=j5+1
     j6=j6-1
     j7=j7+1
 12  j8=j8-1
     kt=kt+1
     kb=kb-1
     j5=j5+nxx4-nx
     j6=j6-nxx4+nx
     j7=j7+nxx4-nx
 11  j8=j8-nxx4+nx
     nx=nxx2
     nxx2=nxx4
     nxx4=nxx4+nxx4
     jsccol=jsccol-1
     kt=nnq1/4
     nnq1=kt+kt
  8  kb=nnq1-1
 70  nn5=nnh+nne+1
     nn7=nn-nne
     do 13 i13=1,nne
     ftemp=(temp(nn7)-temp(nn5))*pifour
     temp(nn7)=(temp(nn5)+temp(nn7))*pifour
     temp(nn5)=ftemp
     nn5=nn5+1
 13  nn7=nn7-1
     nne=nnq
     nnq=nnh
     nnh=nn
  7  nn=nn+nn
c
c    Calculation of the plus/minus butterfly staircase.
c
 80  md2=2
     m=4
     do 14 i14=1,lns1
     do 15 i15=1,md2
     ftemp=temp(i15)+temp(m)
     temp(m)=temp(i15)-temp(m)
     temp(i15)=ftemp
 15  m=m-1
     md2=md2+md2
 14  m=md2+md2
c
c    Now, put the column back in the array pict.
c
     do 16 i16=1,n
```

fidct.f

```
 16   pict(i3,i16)=temp(i16)
  3   continue
c
c     Transpose the image array.
c
      do 17 i17=1,n
      do 18 i18=1,n
      if(i17.le.i18)go to 17
      ftemp=pict(i17,i18)
      pict(i17,i18)=pict(i18,i17)
 18   pict(i18,i17)=ftemp
 17   continue
 20   continue
      go to 90
      print89
 89   format(/,2x,'Inverse Discrete Cosine Transform completed')
      go to 90
100   print101
101   format(/,2x,"Error....In not in range of 2 thru 6.")
 90   return
      end
```

APPENDIX B - SELECTION OF A ZONAL FILTER

A decision had to be made on what type of zone to use for filtering the DCT coefficients. A linear region was chosen based on a brief study using three different zonal shapes on three separate images, namely, Walt, Clock and Village. The images were selected because they represent a good mix of different characteristics encountered in typical images. The size of Walt and Clock is 256x256 pixels, whereas , for the zonal filter study, the size of Village is 128x256 pixels.

Each image was first segmented into equal blocks of size 16x16 pixels. A two dimensional DCT was then performed on each segment. At this point, the DCT coefficients were passed through a zonal filter. The result of the study was based on calculations of the percent of total energy that remained in the DCT coefficients saved by the zonal filter for each segment in the image. An average percent of total energy saved was then acquired for the entire image, which depends on the number of DCT coefficients saved. Both the shape and the size of the zonal filter were varied in the study. The size of the filter is a measure of the number of DCT coefficients it retains.

Three separate zonal shapes were studied based on three simple curves, namely, a hyperbola, a straight line and a

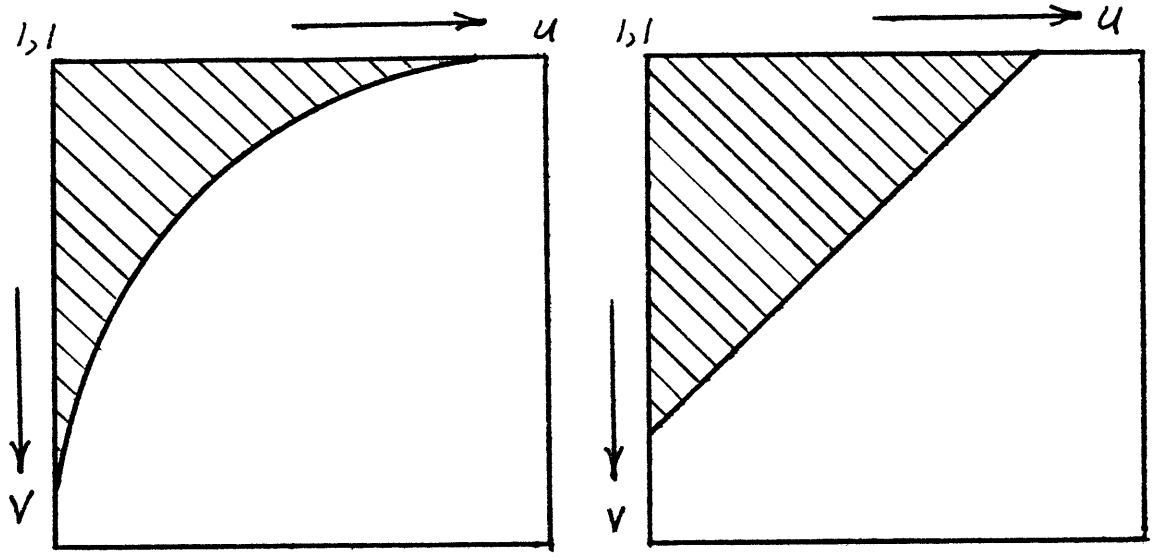
circle. Sample zonal shapes are shown in figure B-1. The size of the filter was controlled by varying the radius of each region. The zonal filter operated by truncating any DCT coefficients outside the shaded region to zero, whereas the coefficients inside the shaded region were left untouched.

Each of the three zonal filter shapes were applied to all three images. The size of the filter was varied to save anywhere from 14 to all 256 DCT coefficients in each segment. It was assumed that at least 14 coefficients around the origin would need to be saved in any practical application. Because of this, the energy calculations were modified slightly to emphasize the differences in the three filter shapes. The 14 high energy DCT coefficients shown in figure B-2 were always retained by all three filter shapes, so that the "total" energy was then calculated from the remaining 242 coefficients. Thus, the average percent energy (APES) saved was calculated by:

$$APES=\frac{1}{N}\sum_{i=1}^{N}\left[\frac{\sum_{j=1}^{Z}(C_{ij})^2}{\sum_{k=1}^{242}(C_{ik})^2}\right]x100$$

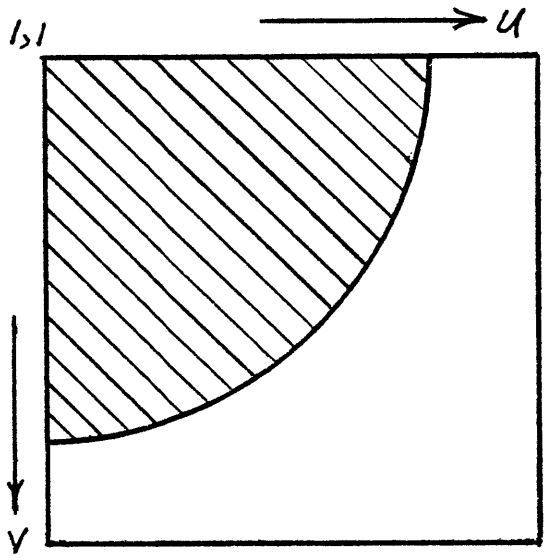
where N is the number of segments in the image, Z is the number of coefficients saved (excluding 14), and C_{XY} is the value of the Yth DCT coefficient in the Xth segment.

The selection of filter shape should be made based on



Hyperbolic
Rad=u·v

Linear
Rad=u+v



Circular
Rad=$\sqrt{u^2+v^2}$

Figure B-1: The three subimage zonal regions or shapes.

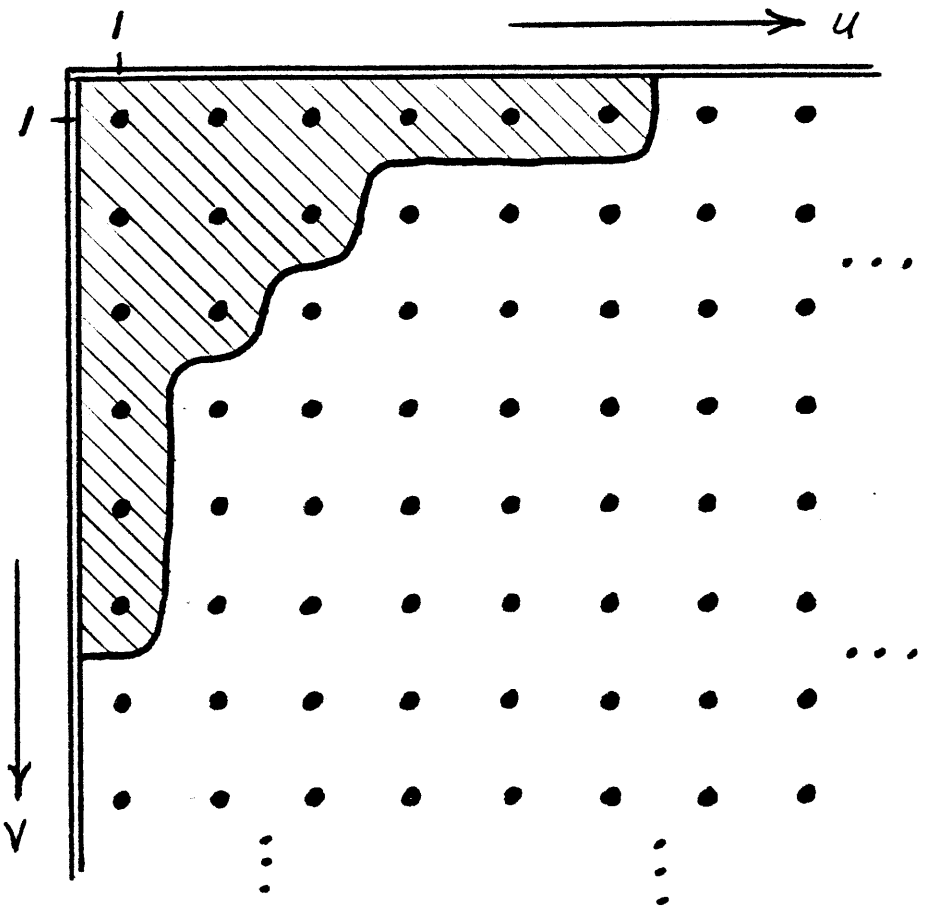


Figure B-2: The 14 DCT coefficients in each subimage that are assumed to always be saved by all three zonal filter shapes. Subimage size=16x16 pixels.

the one that retains the most image energy, given a fixed number of DCT coefficients saved. In other words, the purpose of the filter is to preserve as many higher energy DCT coefficients as possible. Unfortunately, there is no universal filter shape that will be best for all situations. The best that can be done is to use a filter shape that works well for a variety of typical images. Intuitively, one might feel that the circular filter would be appropriate, since it treats internal frequency combinations the same as the main frequency axes. On the other hand, if an image contains a lot of vertical and horizontal edges, then a hyperbolic filter would be better, since much of the image energy will fall on the two frequency axes due to the nature of the DCT.

The graphs in figures B-3,B-4 and B-5 show the results of the study for Walt, Clock and Village, respectively. Each graph presents a plot of average percent image energy saved versus the number of DCT coefficients saved for each filter shape. The range of interest in DCT coefficients saved is from 0 to 140 (excluding 14). Past this range, all three filter shapes perform similarly. For the image Walt, it is clear that the circular and linear shape perform about the same and are superior to the hyperbola shape. For the image Clock, the hyperbolic shape surpasses both the linear and circular shape, which is predictable due to the

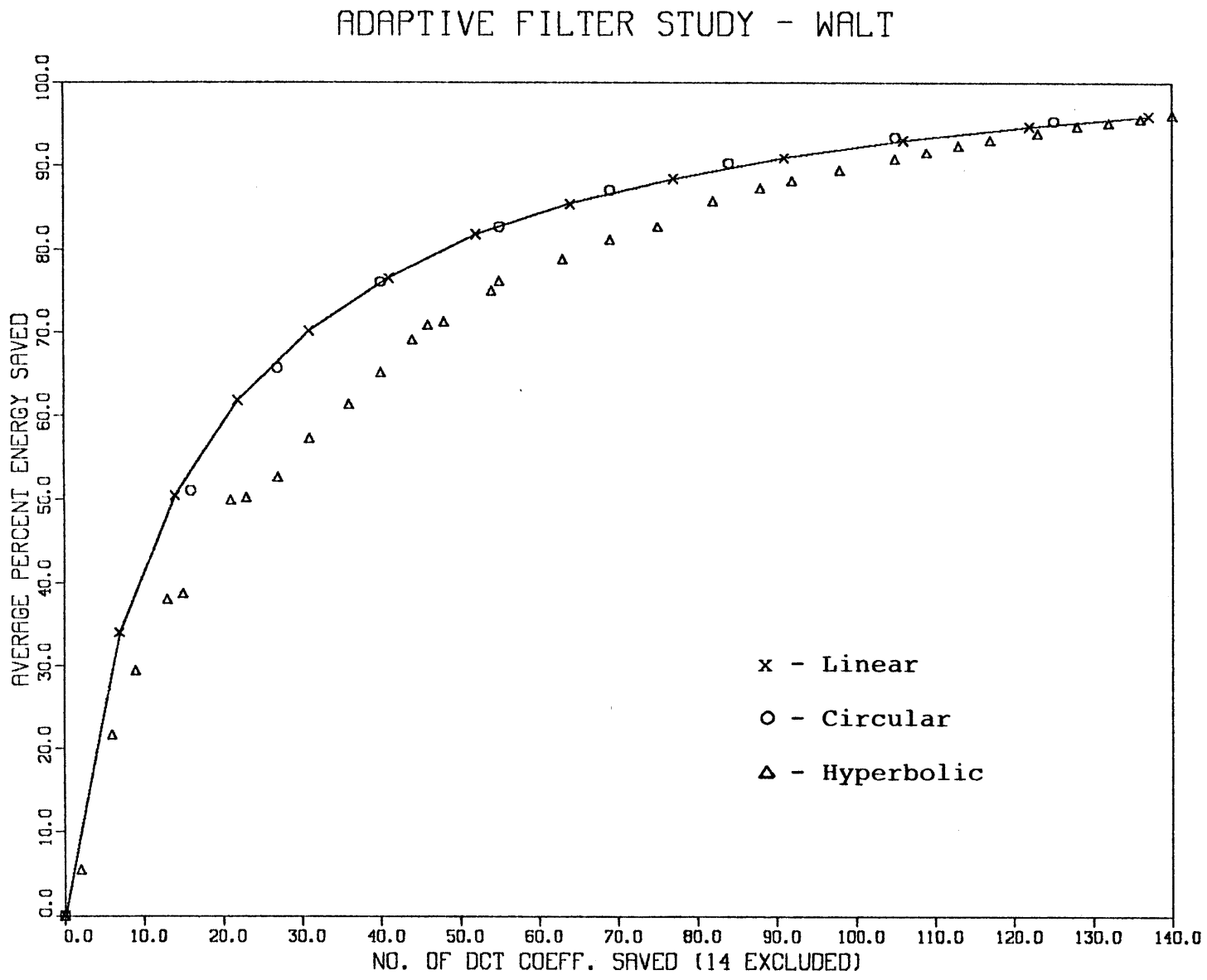

ADAPTIVE FILTER STUDY - WALT
AVERAGE PERCENT ENERGY SAVED
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0
110.0
120.0
130.0
140.0
NO. OF DCT COEFF. SAVED (14 EXCLUDED)
x - Linear
o - Circular
Δ - Hyperbolic

Figure B-3

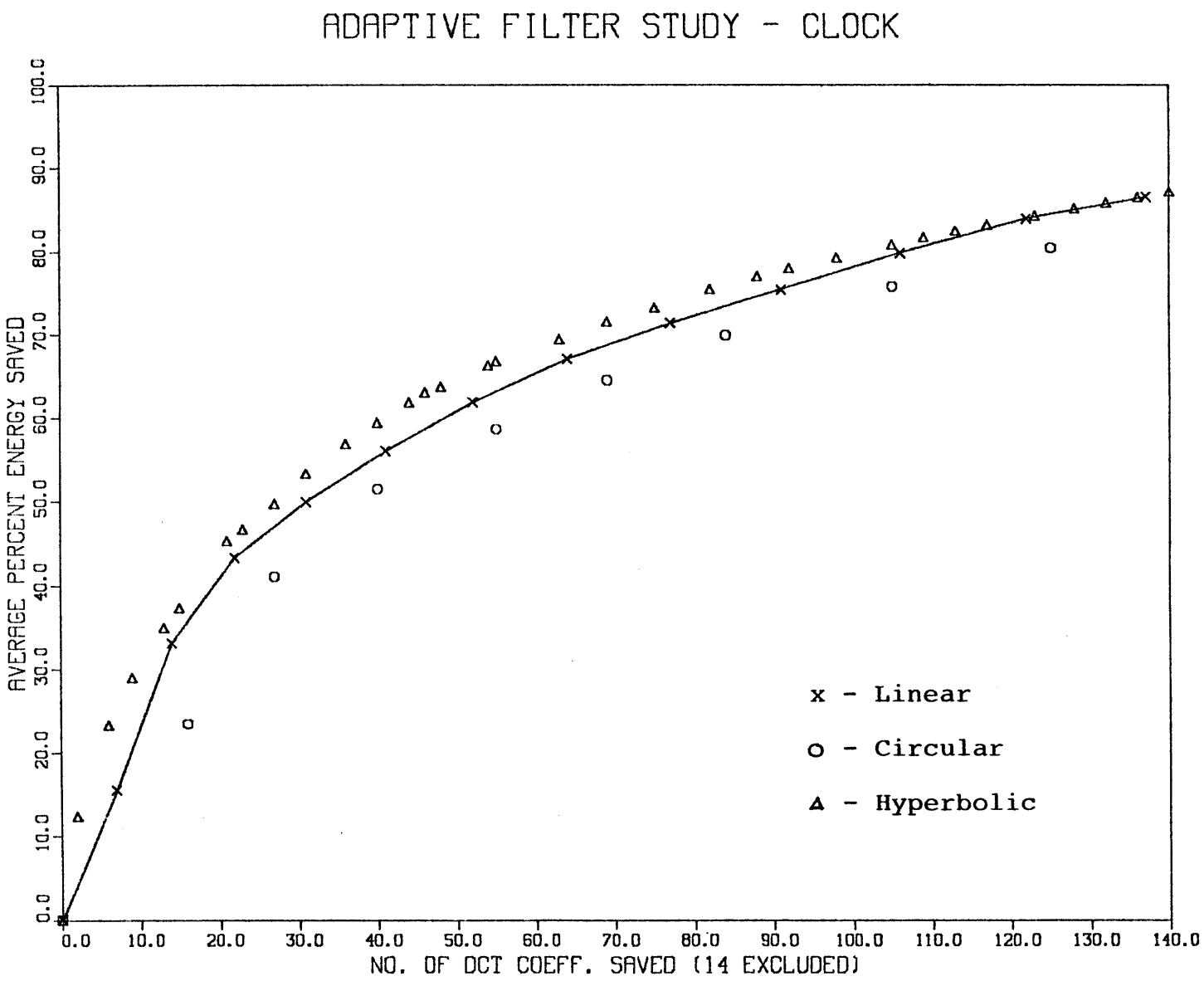

ADAPTIVE FILTER STUDY - CLOCK
AVERAGE PERCENT ENERGY SAVED
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0
110.0
120.0
130.0
140.0
NO. OF DCT COEFF. SAVED (14 EXCLUDED)
x - Linear
o - Circular
Δ - Hyperbolic

Figure B-4

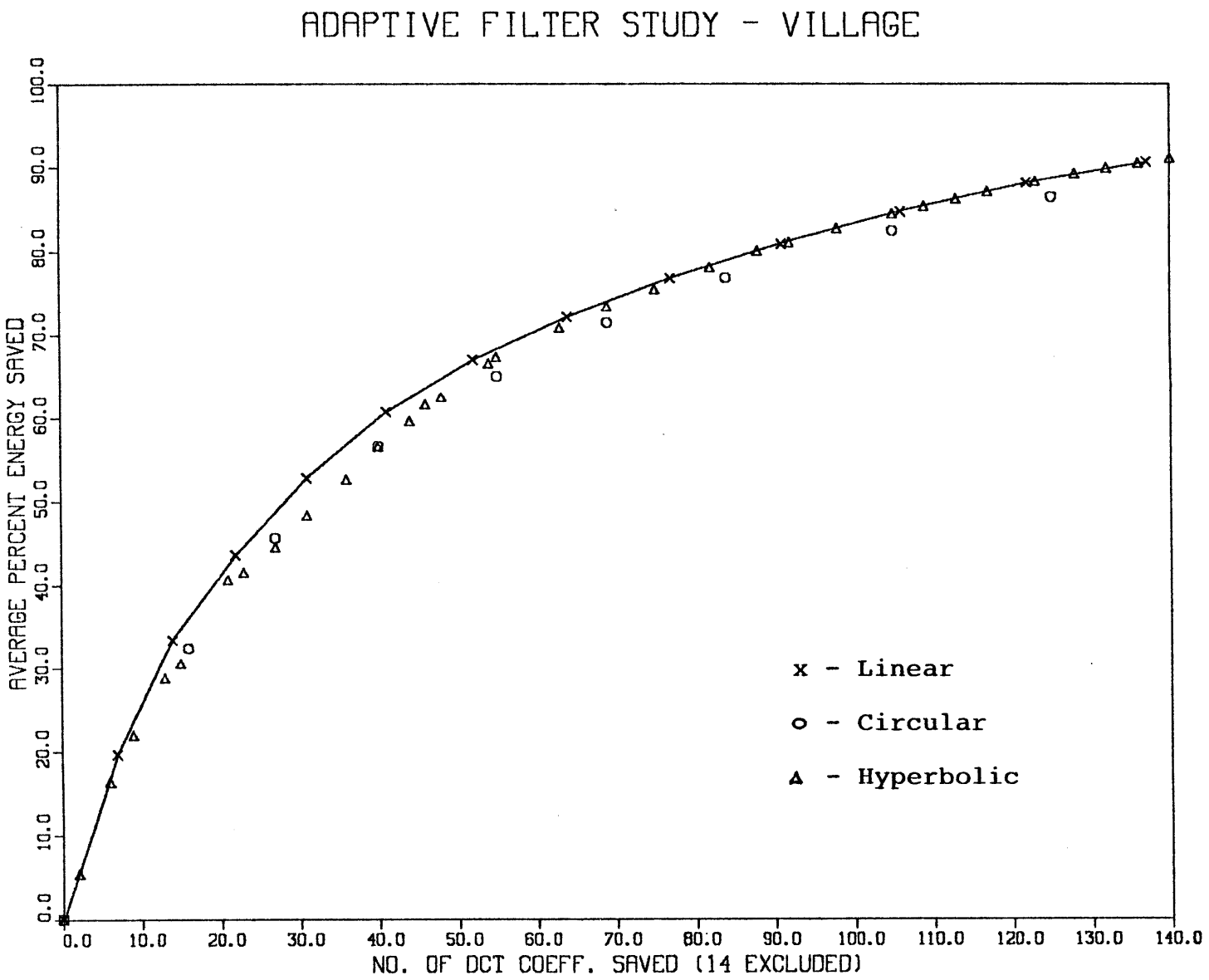

ADAPTIVE FILTER STUDY - VILLAGE
AVERAGE PERCENT ENERGY SAVED
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0
110.0
120.0
130.0
140.0
NO. OF DCT COEFF. SAVED (14 EXCLUDED)
x - Linear
o - Circular
Δ - Hyperbolic

Figure B-5

observable presence of vertical and horizontal edges in the image. The circular shape is definitely the worst, while the linear shape approaches the hyperbolic performance for this image. In the image Village, which exhibits a lot of high frequency content, the linear shape is slightly superior to both the circular and hyperbolic shapes.

From the above results, it was decided to use the linear shaped region for the zonal filter because it worked very well for all three images tested. It seems to represent a combination of the hyperbolic and circular shapes, and thus its performance is acceptable for a wider variety of images.