ADAPTIVE COMPRESSION CODING

Panagiotis Nasiopoulos

B. Sc. (Physics) University of Thessaloniki, Greece

B.A. Sc. (Electrical Engineering) University of British Columbia

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

M.A.Sc.

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

August 1988

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical Engineering
The University of British Columbia
2356 Main Mall
Vancouver, Canada

Date: Sept. 2nd, 1988

Abstract

An adaptive image compression coding technique, ACC, is presented. This algorithm is shown to preserve edges and give better quality decompressed pictures and better compression ratios than that of the Absolute Moment Block Truncation Coding. Look-up tables are used to achieve better compression rates without affecting the visual quality of the reconstructed image. Regions with approximately uniform intensities are successfully detected by using the *range* and these regions are approximated by their average. This procedure leads to further reduction in the compression data rates. A method for preserving edges is introduced. It is shown that as more details are preserved around edges the pictorial results improve dramatically. The ragged appearance of the edges in AMBTC is reduced or eliminated, leading to images far superior than those of AMBTC.

For most of the images ACC yields Root Mean Square Error smaller than that obtained by AMBTC. Decompression time is shown to be comparable to that of AMBTC for low threshold values and becomes significantly lower as the compression rate becomes smaller. An adaptive filter is introduced which helps recover lost texture at very low compression rates (0.8 to 0.6 b/p, depending on the degree of texture in the image). This algorithm is easy to implement since no special hardware is needed.

Table of Contents

List of Tables

List of Figures

Acknowlegement

I would like to thank my supervisor, Dr. Rabab K. Ward, for the generous enthusiasm and encouragement she provided throughout this research. Her help is sincerely appreciated.

I would also like to thank my colleague Susan Pullman for her perceptive comments and help over the last two years.

On the personal side, I would like to express my appreciation to my wife, Maria, for her patience, encouragement and support.

Chapter 1

Introduction

Advances in communications technology lead to a higher demand for more advanced data compression techniques with higher efficiency in transmission and storage of images.

One could define Image Data Compression as the effort to minimize the information needed to represent an image. In digital image processing each picture cell, which is called a pixel, is quantized into a number of bits and stored. When the image is reconstructed after compression it is commonly degraded and distorted. The efficiency of the compression technique used is measured by: its data compressing ability, the distortion it causes, the complexity of its implementation and the visual quality of the image.

Image Data Compression Techniques can be classified into three major categories: predictive coding, transform coding, and interpolative and extrapolative coding.

In predictive coding, redundancy in the data is exploited. The encoded sample is predicted from previously transmitted values and only the error is quantized for transmission.

In transform coding, a representation of the signal is made first by taking linear combinations of samples in a block of data and then quantizing the selected coefficients for transmission.

In interpolative and extrapolative coding, only certain samples are sent to the receiver and the rest are obtained by interpolation or extrapolation.

Block Truncation Coding, BTC, has been proposed in 1979 [1] and is a technique which falls into the latter category. This method divides the picture into blocks of 4 X 4 pixels and the digital representation of each pixel is truncated to one bit by using a threshold and preserving the first and second moments of binary levels.

BTC has many advantages such as: simplicity, computational efficiency, and good quality of decompressed pictures. Its disadvantages are that it produces ragged edges and is not efficient in areas with relatively uniform gradient.

Thus, we shall present here an algorithm which we call Adaptive Compression Coding, ACC, which will keep the advantages of BTC but remedies its disadvantages. ACC adapts to the local statistics of the picture. For regions with edges, a different representation is used. For smooth regions (i.e. with approximately uniform intensity levels) the average of the intensities is used. For some of the regions represented by BTC, an improved version of BTC using look up tables for the BTC binary block is used.

The ACC algorithm is applied to 16x16 non-overlapping blocks, which if not "smooth" enough to be represented by their mean value they are divided into 8X8 or even 4X4 blocks. If the 4X4 block is not represented by its mean value then three options are considered: the first is the Absolute Moment Block Truncation Coding, AMBTC, the second is based upon AMBTC, and the third option is used if an edge is detected.

The *range* , is the variable used to detect the smoothness of the block (Chapter 4). For a 4X4 block if the range is greater than the threshold, then an edge is detected in that quadrant. If the range is less than the threshold then AMBTC or a version of it using look-up tables (Chapter 3) are used to represent or to approximate the binary block of the AMBTC. This procedure of using look up tables is shown to lead to lower compression rates without affecting the visual quality of the picture. If an edge is detected, a method presented in Chapter 5 is used to preserve that edge. It

is shown that this method can reduce or even eliminate the ragged appearance of the edges, improving dramatically the quality of the reconstructed image.

Finally, an adaptive filter is introduced which helps recover lost texture at very low compression rates, which can range from 0.8 to 0.6 bits/pixel, depending on the degree of texture in the image.

Chapter 2

Block Truncation Coding and AMBTC

2..1 Block Truncation Coding BTC

Block Truncation Coding (BTC) is an image compression technique, which was introduced in 1979 by Delp and Mitchell [1] and since then it has been applied to still and moving images [2],digital video, and graphics [3] . The extent of its applications is due to the small number of calculations involved and the simplicity of the implementation.

For this coding method the image is divided into 4X4 non-overlapping pixel blocks and each block is coded individually. The basic idea is to preserve the first two sample moments in each block. In order to achieve this a two-level quantizer is used.

The algorithm calculates the sample mean

$$\overline{x} = \frac{1}{4^2} \sum_{i=1}^{16} x_i \tag{2.1}$$

and the variance

$$\sigma^2 = \overline{x^2} - \overline{x}^2 \tag{2.2}$$

where

$$\overline{x^2} = \frac{1}{4^2} \sum_{i=1}^{16} x_i^2 \tag{2.3}$$

A threshold value, the mean $= \overline{x}$, is used to determine the output of the two-level quantizer, A and B.

If $x_i > \overline{x}$ then output $= 1$

If $x_i \leq \overline{x}$ then output = 0

Each block is then transferred into a binary block having 1's and 0's. Only the pixels which have intensities higher than the sample mean are represented by 1's, otherwise 0's are used.

In order to show the basic idea behind Block Truncation coding, let us illustrate the various steps involved by an example. Assume a 4x4 block has the intensity values:

10	150	103	20
100	120	193	50
30	0	111	32
9	50	3	11

then the sample mean is $\overline{x} = 62$

For every pixel with intensity $> \overline{x}$ the output level is 1, otherwise it is 0 .

Then, the binary representation will be

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

If q is the number of pixels with $x_i > \overline{x}$

then the two output levels A and B of the quantizer are found by solving the following equations:

$$m\overline{x} = (16 - q)A + qB \tag{2.4}$$

and

$$m\overline{x^2} = (16 - q)A^2 + qB^2 \tag{2.5}$$

Solving for A and B:

$$A = \overline{x} - \sigma\sqrt{\frac{q}{16 - q}} \tag{2.6}$$

$$B = \overline{x} + \sigma\sqrt{\frac{16 - q}{q}} \tag{2.7}$$

From these equations we can calculate the output levels A and B for the previous example. Then the reconstructed block will be:

17	136	136	17
136	136	136	17
17	17	136	17
17	17	17	17

If we assign 8 bits for the mean and 8 for the variance , this method gives a data rate of 8 + 8 + 16 (1's and 0's) bits/ 16 pixels = 2 bits/pixel. We can improve on that if 6 bits are used for the mean and 4 bits for σ. Then the obtained compressed data rate is 1.625 bits/pixel.

Figures 2.1 and 2.2 show the original picture and the reconstructed image using BTC.

Figure 2.1: Original picture.

Figure 2.2: Compressed image using BTC.

2..2 The Absolute Moment Block Truncation Coding AMBTC

Recently, an improvement on the above method (BTC) was introduced giving the comparable pictorial results and improving on calculation time on both transmitter and receiver. An additional advantage is the lower mean-square error achieved by this method.

The new method is called Absolute Moment Block Truncation Coding, AMBTC [5]. It preserves the first absolute central moment of each original block instead of the variance.

Calculations needed in this case are:

the mean $\overline{n}$

$$\overline{n} = \frac{1}{16}\sum_{i=1}^{16} x_i \tag{2.8}$$

and the first absolute moment α

$$\overline{\alpha} = \frac{1}{16}\sum_{i=1}^{16} |x_i - \overline{n}| \tag{2.9}$$

The sample first absolute moment can be calculated as follows:

$$m\overline{\alpha} = \sum_{for x_i \geq \overline{n}} x_i - \overline{n} \sum_{for x_i \geq \overline{n}} 1 - \sum_{for x_i < \overline{n}} x_i + \overline{n} \sum_{for x_i < \overline{n}} 1 \tag{2.10}$$

$$q = \sum_{for x_i \geq \overline{n}} 1 \tag{2.11}$$

and

$$m - q = \sum_{for x_i < \overline{n}} 1 \tag{2.12}$$

then

$$m\overline{n} = \sum_{for x_i \geq \overline{n}} x_i + \sum_{for x_i < \overline{n}} x_i \tag{2.13}$$

Inserting into (2.10)

$$\overline{\alpha} = \frac{2}{m}\left(\sum_{for x_i \geq \overline{n}} x_i - \overline{n}q \right) \tag{2.14}$$

In order to preserve the two moments, the quantizer should be assigned two values A and B, such that

$$m\overline{n} = qB + (m-q)A \tag{2.15}$$

$$m\overline{\alpha} = q(B-\overline{n}) - (m-q)(A-\overline{n}) \tag{2.16}$$

Then the receiver can reconstruct the two output levels A and B by using the equations:

$$A = \overline{n} - \frac{m\overline{\alpha}}{2(m-q)} \tag{2.17}$$

$$B = \overline{n} + \frac{m\overline{\alpha}}{2q} \tag{2.18}$$

where

$m = n^2 = 16$

q = number of pixels with intensity $\geq \overline{n}$

α = 1st absolute moment.

After comparing the equations 2.2 and 2.9 we realize that calculation time at the transmitter will be significantly reduced, since only additions are used in the case of AMBTC.

Similarly, at the receiver BTC evaluates two quotients and two square roots, while AMBTC has simplified the calculations to two quotients (see equations 2.6, 2.7, 2.17 , 2.18).

Finally, AMBTC achieves better mean square error performance when compared to BTC.

For the reasons mentioned above , AMBTC was chosen to be used in our adaptive compression technique.

2..3 Disadvantages of BTC and AMBTC

Although the overall quality of the compressed picture and the achieved compressed data rate for both BTC and AMBTC are good, in this work we are set on improving that.

One of the drawbacks of both methods is that a uniform operator for the whole image is used. Both methods do not adapt to the local statistics of the picture. This results in artifacts which are usually seen in regions around the edges. Although, edges are sharply reproduced, they tend to have a ragged appearance.

Another problem appears in areas with very low contrast, or where the gradient of the intensities is relatively uniform. In these areas the two output levels of each method are very close together. The compression representation here is not efficient.

An algorithm which would be based on BTC or AMBTC and could adapt to the local statistics of an image, would be more efficient and could preserve edges resulting in better pictorial results.

As any adaptive method, the complexity and compression rate of this algorithm will increase , something that must be considered, since the advantages of BTC (or AMBTC) are simplicity and easy implementation.

For textured regions AMBTC shall be used. For regions where the gradient of the

intensities is relatively uniform, an approximation of AMBTC using look-up tables shall be used. This is discussed in the following chapter. Smooth regions shall be represented by their average as discussed in chapter 4. Chapter 5 refers to the representation of regions which contain edges.

Chapter 3

Look-up Tables

It was shown that the Absolute Moment Block Truncation Coding method encodes every 4x4 bits block by its codeword. For every 4x4 bits block we need to store the mean and the first absolute moment and we also need 16 bits to indicate whether an intensity is above or below the mean.

If the latter 16 bits for the 4X4 block (which are only 1's and 0's) can somehow be represented by fewer bits, then a better compression ratio will be achieved. This shall be done by a look-up table. Only the address of that table will be stored. Of course, if the tables are as many as the number of different possible binary configurations for a 4X4 block, then there would be no advantage to use this method.

Using the above idea, if 256 (2^8) tables were used to represent the different edge configurations and all of the 4X4 binary blocks were approximated with one of these preset tables, then for each 4X4 pixel block we need 10 bits for the mean and the absolute moment and 8 bits for the binary representation. The compressed data rate would become:

(10 + 8) bits /16 pixels = 1.125 bits/pixel,

instead of 1.63 b/p for AMBTC.

Experiments have shown that using a reduced set of 128 tables yielded similar pictorial results to that of the 256 tables.

In this case, the compression data rate is 1.0625 bits/pixel.

The original picture, the compressed picture using BTC, and the picture reproduced

using the 128 look-up tables are shown in Fig. 3.3 , Fig.3.4, and Fig.3.5 respectively.

Figure 3.3: Original picture.

Figure 3.4: Picture using BTC.

Figure 3.5: Picture reconstructed by approximating all binary tables by one of the 128 look-up tables.

Although the data rate is reduced from 1.63 bits/pixel to 1.0625 bits/pixel, the deterioration of the picture does not justify such a move. To remedy that we shall do the following:

We first check if the original table exactly matches any entry in the look-up table. If yes, then only the address of that look-up table is stored. Otherwise, a test will be carried out to determine whether or not the look-up tables will be used. This is based on Weber's law [14]. The results of Weber's work are summarized in Fig. 3.6. If we consider a background of intensity I and if we change the intensity I in a patch within that region, until a change in the intensity is obvious, we shall call the "just noticeable difference" ΔI. It is found that ΔI is a function of I. Fig. 3.6 (a) shows that for average intensities the Weber ratio ΔI / I is constant at a value of about 0.02. Furthermore, contrast sensitivity depends on the intensity of the surroundings. Fig. 3.6 (b) shows two patches of light surrounded by light of intensity I_o. The range over which the

Weber ratio remains constant is very small. In general, the ratio $\Delta I/I$ reaches higher values than 0.02.

For our case, areas of interest where look-up tables can be used are those areas with very low contrast, or where the gradient of the intensities is relatively uniform. In these areas, as long as the two output levels of AMBTC are close enough to obey Weber's law, look-up tables can be used to approximate the original binary blocks without differences being noticed.

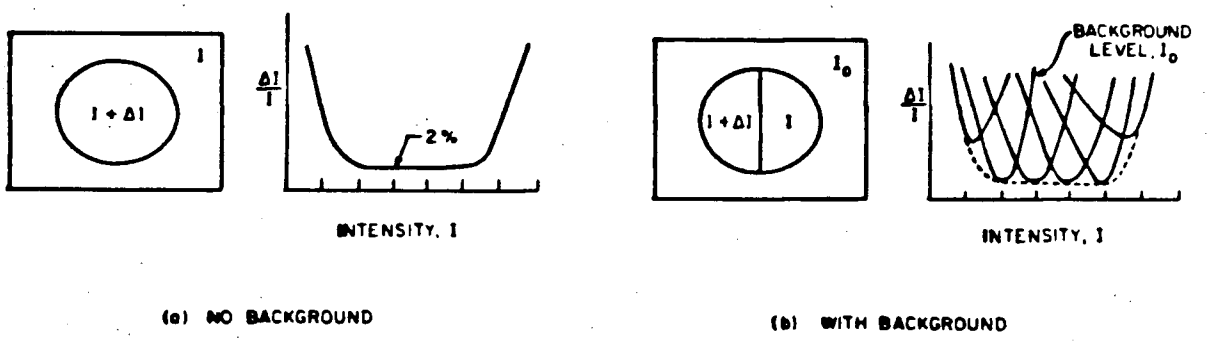


Figure 3.6: Contrast sensitivity.

Actually, our experiments verified that, in general, if the two output levels A and B of the block decompression by AMBTC differ by 20 or less the visual quality of the picture is not affected. In this case , approximation of these blocks with one of the preset tables is not going to alter the visual quality of the picture. The following example shows the use of an approximate table.

$\sigma = 9$

$\overline{x} = 179$

157	166	177	181		0	0	0	1
176	177	181	184		0	0	1	1
177	180	188	189		0	1	1	1
174	183	189	193		0	1	1	1

Table 3.1: Original block. Binary block.

The two output levels of AMBTC are

$$\text{level A} = 169$$

$$\text{level B} = 187$$

The binary block in Table 3.1 may be approximated by that shown in Table 3.2.

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

Table 3.2: Approximate table.

169	169	169	187
169	169	187	187
169	187	187	187
187	187	187	187

Table 3.3: Reconstructed block.The output levels are close and no alteration of the picture will be noticed.

Table 3.3 shows the reconstructed block. The two output levels shown in this table differ by 18 only and no visual alteration of the picture will be noticed.

The 128 look-up tables were chosen to represent different cases of edges. Most of the edges were chosen to have width of at least two pixels. This is justified because

edges are almost never represented as step functions but more like gradual steps. An elaborate explanation is given in Chapter 5.

The first 64 tables are chosen to be the inverse representation of the other 64. That means that if a pixel has the value of 1 in one table, then that pixel becomes 0 in the corresponding "symmetric" table. See table 3.4 for an example.

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

Table 3.4: Look-up table 3

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

Look-up table 68

The first 64 tables are shown in Appendix A.

Many different criteria, such as autocorrelation , were used to approximate the original binary blocks with one of the 128 look-up tables. Among them the most efficient computation proved to be the one which uses the following method:

first, a computation of the absolute difference between the original block matrix and the look-up tables is done and the look-up table with the minimum difference is found:

$$min_{tables} \sum_{i=1}^{4} \sum_{j=1}^{4} |original(i,j) - lookup(i,j)| \tag{3.19}$$

then we do an AND operation of the original matrix block and all the look-up tables, and find the maximum over all the tables:

$$max_{tables} \sum_{i=1}^{4} \sum_{j=1}^{4} [original(i,j) \times lookup(i,j)] \tag{3.20}$$

Then, the closest table is anyone which yields the minimum difference and the maximum AND result.

The following tables show results obtained by using this criterion.

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

Table 3.5: Original table. Approximate table.

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

Table 3.6: Original table. Approximate table.

If the look-up tables are used for the two previously mentioned cases, the resulting picture appears the same as the one obtained by Block Truncation Coding. The two pictures are shown in Fig. 3.7 and 3.8. In the second image 135 block tables were represented exactly by one of the look up tables and 3618 were approximated. The rest of the tables were represented by AMBTC. The compression data rate for this image is 1.49 bits/pixel.

Figure 3.7: Image reconstructed using only BTC. Compression rate = 1.625 bits/pixel.

Figure 3.8: Image reconstructed using exact and approximate tables at regions where the two output levels are very close together. Compression rate = 1.49 bits/pixel.

Chapter 4

Using Range to Detect Smoothness

As it was mentioned, Absolute Moment BTC was chosen over the Block Truncation Coding method because of the simpler equations that it provides, and the savings on computation time. However, for "smooth" regions one disadvantage of AMBTC (or BTC) is that it does not take advantage of that knowledge. For our algorithm we shall represent "smooth" regions with their average intensities.

A problem, though, arises when the algorithm tries to check the smoothness in each block. One well known method to detect smoothness is to compute the variance for each block

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} x_i^2 - \left(\frac{1}{m}\sum_{i=1}^{m} x_i\right)^2 \tag{4.21}$$

where

m is the total number of pixel in the block, and

x_i is the grey value of each pixel.

This method though has two disadvantages.

- The Absolute Moment method computes the first absolute central moment in an effort to avoid multiplications, whose computational time is several times greater than the additions. If the computation of the variance is added, nothing will be gained by using the AMBTC, and the algorithm will be slowed down considerably.

- Second, when the block is large in size, a small portion of a line will not be detected. Let us assume that a threshold value of 6400 is used to detect smoothness. Then, of the two 16X16 blocks shown below , only the first will be detected as "smooth" since the two values for the variance are 8281 and 6084 respectively.

200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20
200	200	200	200	200	20	20	20	20	20	20	20	20	20	20	20

200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200	20	20	20	20	20	20	20	20
200	200	200	200	200	200	200	200	20	20	20	20	20	20	20	20
200	200	200	200	200	200	200	200	20	20	20	20	20	20	20	20
200	200	200	200	200	200	200	200	20	20	20	20	20	20	20	20
200	200	200	200	200	200	200	200	20	20	20	20	20	20	20	20
200	200	200	200	200	200	200	200	20	20	20	20	20	20	20	20
200	200	200	200	200	200	200	200	20	20	20	20	20	20	20	20
200	200	200	200	200	200	200	200	20	20	20	20	20	20	20	20

Table 4.7: The value for the variance changes from 8281 for the first block to 6084 for the second. The range though remains the same for both cases(180).

To overcome these disadvantages, another method was tested and proved successful in measuring smoothness. For each block the lowest and highest values of the grey levels are calculated and their difference is saved. Let us call this variable *range* and use it to check the degree of smoothness in each block.

This method will easily detect small portions of lines even when the variance fails. For the two 16X16 blocks shown in table 4.7 the value for the range is the same (180). Therefore, depending on the threshold, both blocks will be categorized the same. Also, since only additions are involved in computing the *range* , the computation time will be reduced.

For instance, if the variance and the range were computed for a 16X16 block on the VAX11/750, the CPU time for computing the range will be about 30% faster than computing the variance.

Our expirements have shown that a value between 16 and 25 for the threshold used to detect smoothness would yield high quality pictures and low compression rates for most of the images.

Chapter 5

Preserving Edges

Sharp edges will be reproduced in a ragged form by AMBTC. To improve the reconstructed image we should try to preserve more details in the these regions, avoiding the use of Block Truncation Coding.

One straight forward approach is to check each 4X4 block and if a sharp edge is encountered, then save the intensities for all 16 pixels. The data rate for this case would be 8 bits/pixel.

Before we continue, let us examine the nature of sharp edges or lines in a 4X4 pixel area. The lens of the digital camera is not perfect and this imperfection is the source of spherical aberration which will blur the picture. This is equivalent to applying a two dimensional low-pass filter on the image. The result will be a smoothed picture, where a sharp edge will never be a step function but it will look more like a ramp(see fig.5.9)

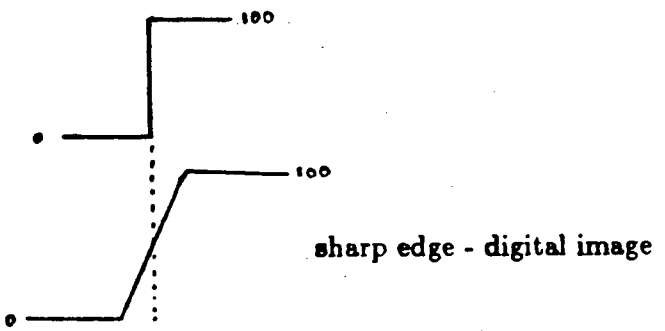
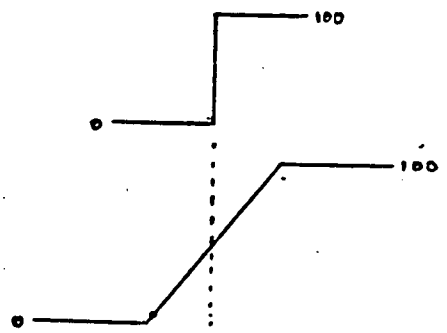


Figure 5.9: Representation of a sharp edge on a digital image. Edges are usually represented by a gradual step.

Our experiments have shown that a range value of 120 for a 4X4 block represents very well the existence of a sharp edge, which would be reproduced with artifacts by BTC.

The approach used here is to check for the presence of edges and if an edge is found we shall preserve it. This is done by checking the range value for each 4X4 block. If it is higher than the threshold value which corresponds to the presence of a sharp edge, then that block is divided into four 2X2 blocks. Afterwards, the range for each 2X2 block is compared with a threshold value which is half of the one used for the 4X4 original block. The reason for using half of the original threshold value is that a sharp edge is represented by a gradual step, and as it can be seen in fig 5.10 below, when the range for the original block is 100, then the range for each 2X2 block which has part of the edge is half of that value (50).



0 50 100 100
0 50 100 100
0 50 100 100
0 50 100 100

Figure 5.10: A sharp edge in a 4X4 block. The range is 100. For the 4 2X2 sub-blocks the range changes to 50 and 0.

Since the 16 pixels cover a very small area, the presence of an edge will usually be detected in only two of the four 2X2 blocks, unless a corner is involved. For the 4X4 blocks which have a sharp edge, the original intensities of those 2X2 blocks which contain the edge will be saved exactly as they were in the original picture. The other 2X2 blocks will be represented by their averages.

Since an edge usually passes through only two of the 2X2 blocks in a 4X4 quadrant, this will result in an average compressed data rate of

(8 blocks x 8 bits + 2 x 8 bits) / 16 pixels = 5 bits/pixel

This is a 3 bits/pixel improvement over the straight forward implementation, i.e. saving the original intensities of each pixel in the 4X4 block which contain an edge.

The following pictures, Fig. 5.11, show compressed images using the BTC method and the above mentioned method for preserving edges. A dramatic improvement in picture quality can be easily seen.

Figure 5.11: AMBTC, ACC, and Difference pictures are shown for section of F16.

Chapter 6

Implementation of Adaptive Coding

This chapter is a summary of the ACC algorithm. The algorithm is applied to non-overlapping sixteen by sixteen (16X16) pixel blocks. If the range for each block is smaller than the threshold usually having values between 16 and 25, the average intensity of the pixels is stored. Otherwise, the block is divided into four eight by eight (8X8) blocks. For each of these sub-blocks the range is compared with the threshold and, as before, if the range is smaller the average is saved, else the 8X8 sub-block is divided into four 4X4 blocks.

At this level (the 4X4 pixel block), several different tests are made, keeping in mind that we want to save as much information as we can:

1. The range for the 4X4 block is checked. If it is zero then the intensities are all the same and the mean is only needed to be stored. If the range is not zero go to 2.

2. The range is compared with a threshold value which corresponds to the existence of a sharp edge for this image (usually 120). If the range is greater than the threshold then an edge is detected. In that case the corresponding quadrant is subdivided into four two by two (2X2) blocks. For the 2X2 blocks which have part of the edge, the intensities will be saved as they are in the original picture. For the other 2X2 blocks the mean of the intensities will be stored. If the range is smaller than the threshold (an edge is not detected) go to 3.

3. The regular AMBTC representation of the 4X4 block is first found. If the binary block representation in the AMBTC matches exactly one of the look-up tables then the look-up table version of AMBTC is used. The 7 bits address of the look-up table is stored instead of the 16 bits which regular AMBTC would have used. If such a match is not found go to 4.

4. The difference of the two output levels of AMBTC are compared to a given threshold (different from that in (2) used for detecting edges). If the the difference is greater than the fixed value of 20 the regular AMBTC representation is used. If the difference is not greater than this threshold value the look-up table version of AMBTC is used. Here, the AMBTC output levels, A and B, will be very close in value and the binary block will be approximated by one of the look-up tables.

Thus, for a 4X4 block there are four posibilities: 1) the intensities are all equal, 2) there is an edge, 3) the binary block of AMBTC is represented by a look-up table, and 4) AMBTC is used.

6..4 Data Identification

For each 16X16 block a codeword of 5 bits in length is used to identify whether the block is represented by its average (first bit = 1) or if it is divided into four 8X8 blocks (first bit = 0). Then the next four bits show whether the first, second, third, or fourth 8X8 blocks are represented by their averages (corresponding bit set to 1) or whether they were subdivided into four quadrants (bit = 0). If the 16X16 block is represented by its average the codeword is followed by the average (represented by 8 bits).

For each 8X8 block which is not represented by its average a codeword of 8 bits is used to identify each of the 4X4 sub-blocks and the four different cases. Two bits are needed for each 4X4 block since for such a block four possibilities exist. This codeword is followed by the relevant values i.e. the averages of the intensities, the address of the look up table and the values for AMBTC. If an edge exists within a 4X4 quadrant we need to specify which of the 2X2 blocks does that edge passes through. For such 2X2 blocks the most significant bit, MSB, of the word used to store the information for that block is used to identify whether the block has an edge (MSB set to one) or the mean of the intensities is stored (MSB set to zero). If the mean is stored the word is an 8 bits long byte. Thus 7 bits are left to represent the mean instead of the usual 8. In this case the original value of the mean is divided by 2 before the compression. During decompression this value is multiplied by 2. If the 2X2 block contains an edge then the 4 original intensities have to be stored. In this case one of these intensities has to be represented by 7 bits so as to allow the MSB of this 32 bit word to identify the existence of an edge. In the worst case, this results in a 0.144 bits/pixel increase in the compression rate.

Chapter 7

The Limits of ACC and an Adaptive Filter to Improve on that

As the threshold value used to detect smoothness increases, more and more areas of the image are represented by their averages. As the algorithm reaches the upper limits of compression, some egdes are still reproduced fairly accurately, but degradation becomes significant and many textured regions are smoothed over and may have a blocky appearance. The reason for having that blocky appearance is that the blocks which were represented by their average have values which are not close together and their boundaries can be seen. The threshold values at which ACC reaches the upper limits and the attained compression rates differ from one picture to another and mainly depend on the degree of texture in the image. For the picture shown in fig. 7.13 the threshold used is 40 and the data rate is 0.66 b/p.

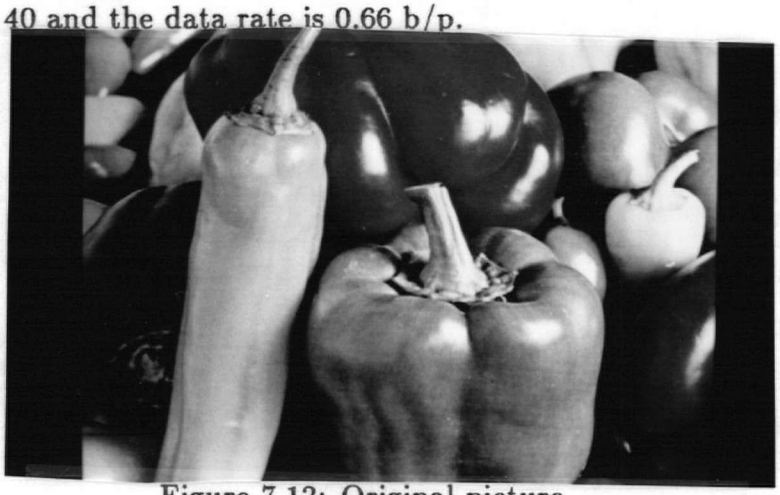


Figure 7.12: Original picture

Figure 7.13: Reconstructed "blocky" picture.

Since it might be desirable to reach very low data rates and still obtain a "realistic " picture, an adaptive filter should be applied to those areas where the "blocky" artifacts appear, in an effort to give back to the image the lost continuity and improve the limits of ACC. The adaptive filter designed for this case is based on a 3X3 averaging filter.

After the reconstuction of the image is complete, the picture is scanned (for these edges which give the picture its blocky appearance) by an 8X8 window and the range is compared with the threshold value previously used. If it is smaller, then the area is smoothed by a 3X3 average filter. This method takes care of the blocky appearance of the reconstructed image, but the smoothed regions now look flat and artificial. Fig 7.14 shows the picture smoothed by an adaptive 3x3 average filter.

In order to give back to the image the lost "texture", Gaussian random noise is added to the averaged areas. The value of the noise is kept at very low levels, so that the difference in luminance is close to the contrast level detectable by the visual system. The image obtained after the full adaptive filter is applied is shown in fig. 7.15.

Figure 7.14: Averaged adaptive filter used to improve blocky picture.

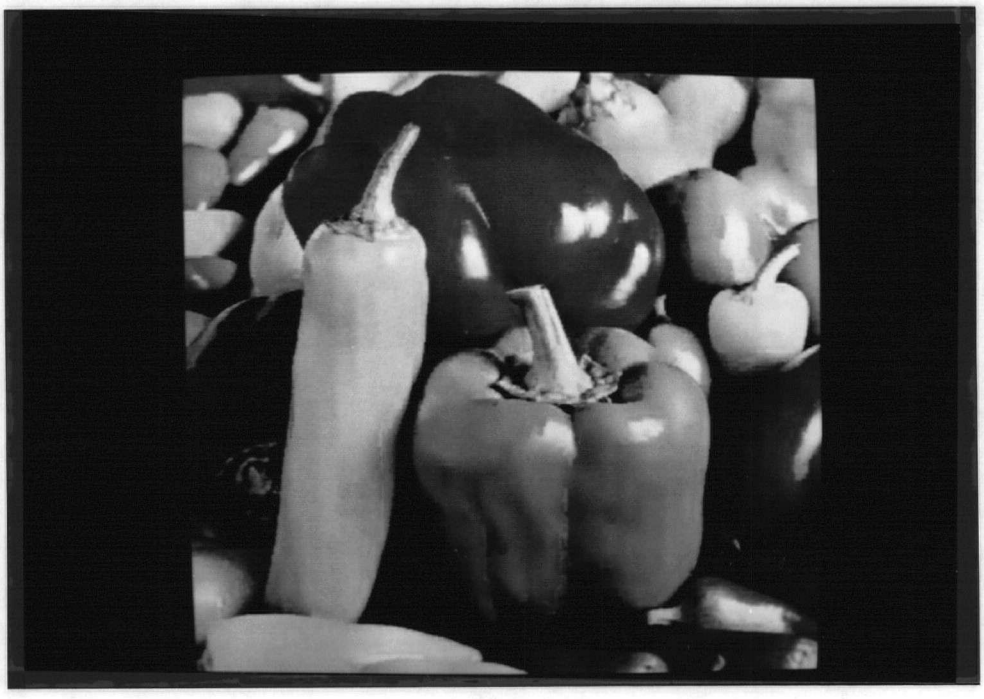


Figure 7.15: Full adaptive filter used on the blocky picture.

Chapter 8

A Comparative Study

The main objectives of the Adaptive Compression Coding algorithm presented in this thesis are two folds. First ACC must adapt to the local characteristics of the image, in order to improve the image quality. The other objective is to reduce the compression ratio achieved by Absolute Moment BTC, or BTC.

Based on a chosen threshold for the range, the algorithm can smooth unimportant parts of the picture which have very low variations (i.e. background), while preserving regions of main interest.

This characteristic makes the ACC algorithm ideal for cases such as medical pictures showing blood cells. These images have a " flat " background of the same grey level while the blood cells are darker and are the only parts of interest in the picture.

In order to assess the capabilities of the algorithm a comparison should be made based on image quality, compression ratio, and decompression speed, which is also of importance.

Six test images are used for comparison, two of which are medical pictures. They are described in the following table .

Image	Size	Bits/pixel
Girl	512x512	8
F16	512x512	8
Peppers	512x512	8
Lake	512x512	8
Chromo	240x240	8
Smear	512x512	8

Table 8.8: Test Images.

The comparison is done using three different range smoothness-thresholds for each image, in order to demonstrate how the performance of ACC depends on the threshold used to detect smoothness. The first one uses a threshold value which yields a very high quality picture. The second value gives a high quality, low data rate image and the third is a high threshold value to demonstrate the upper limit of ACC using the adaptive method discussed in chapter 7.

8..5 Visual Analysis

A visual evaluation of the images is very important. The pictures from the Adaptive Compression Coding technique appear much better than those of AMBTC. It is known that RMSE is not an especially accurate measure of visual fidelity, so besides the RMSE the difference pictures were also chosen as a mean for the visual analysis of the images.

This section is organized as follows:

The original image , the picture compressed using ACC, and the difference picture are shown. Also the difference picture for AMBTC, is shown, in order to be compered with ACC.

As it can be seen, the quality of the images obtained from using the first two threshold values is significantly higher than that of AMBTC. The difference-pictures

show that images, and especially edges are very faithfully preserved by ACC.

As the upper limit of the algorithm is approached degradation becomes obvious, but compression data rates have been reduced significantly.

For highly textured images, such as Lake, the performance of ACC is not as good as for the other images. The reason for this is that more edges must be preserved and fewer regions can be represented by their average.

Figure 8.16: Original F16 image.

Figure 8.17: Image obtained by AMBTC. Rate = 1.625 b/p.

Figure 8.18: Reconstructed image using ACC. Threshold = 16, rate = 1.13 b/p.

Figure 8.19: Reconstructed image using ACC. Threshold = 30, rate = 0.92 b/p.

Figure 8.20: Reconstructed image using ACC. Threshold = 50, rate = 0.59 b/p.

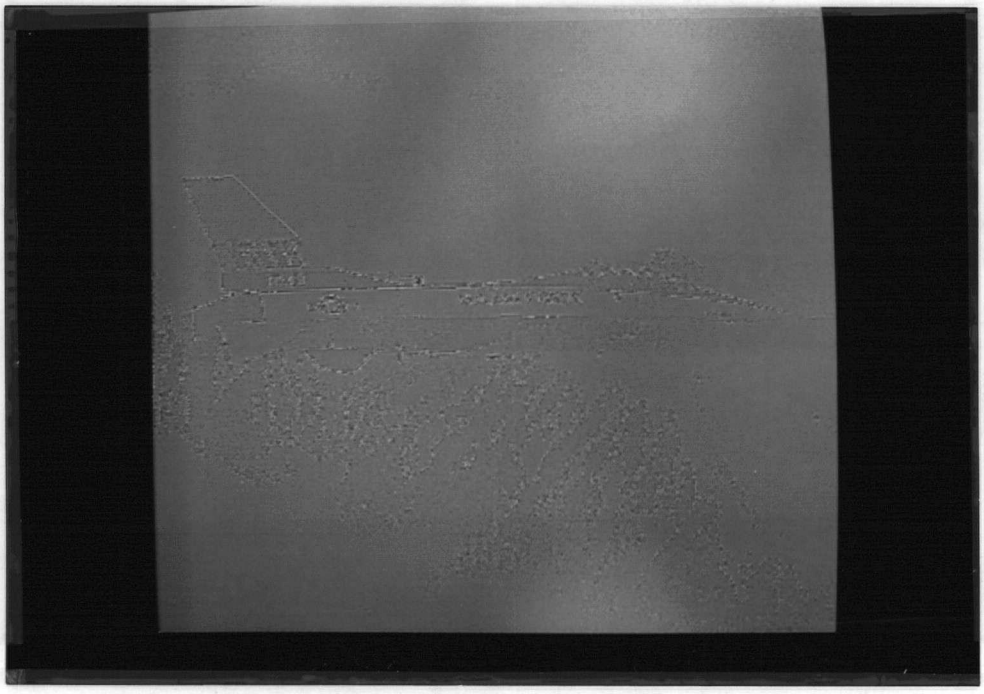


Figure 8.21: Difference picture between original and image obtained by AMBTC.

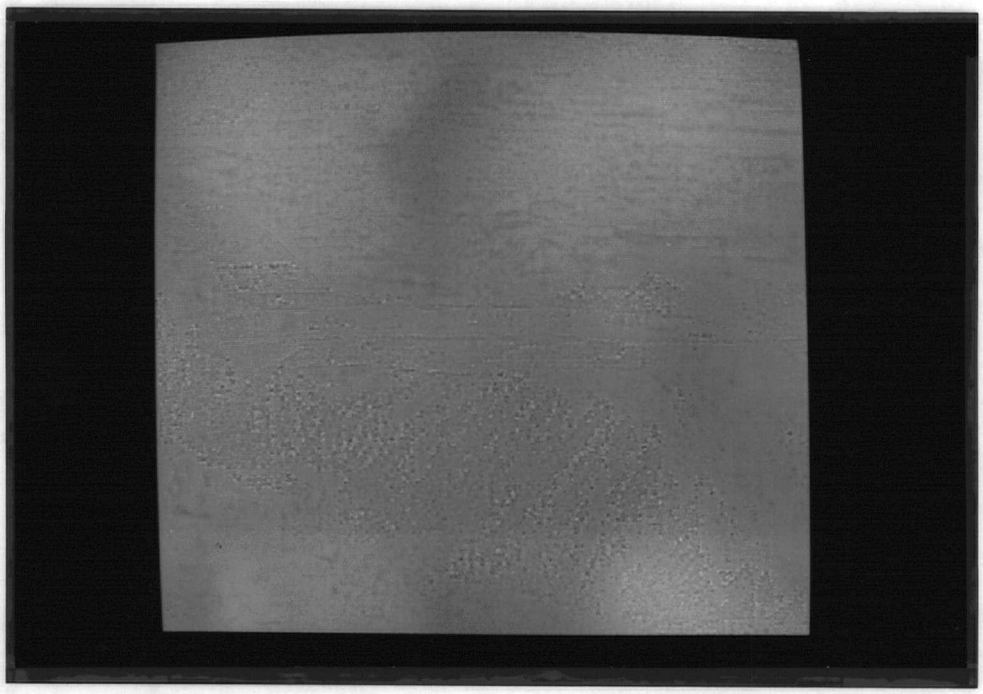


Figure 8.22: Difference picture between original and image obtained by ACC. (rate = 1.13 b/p.)

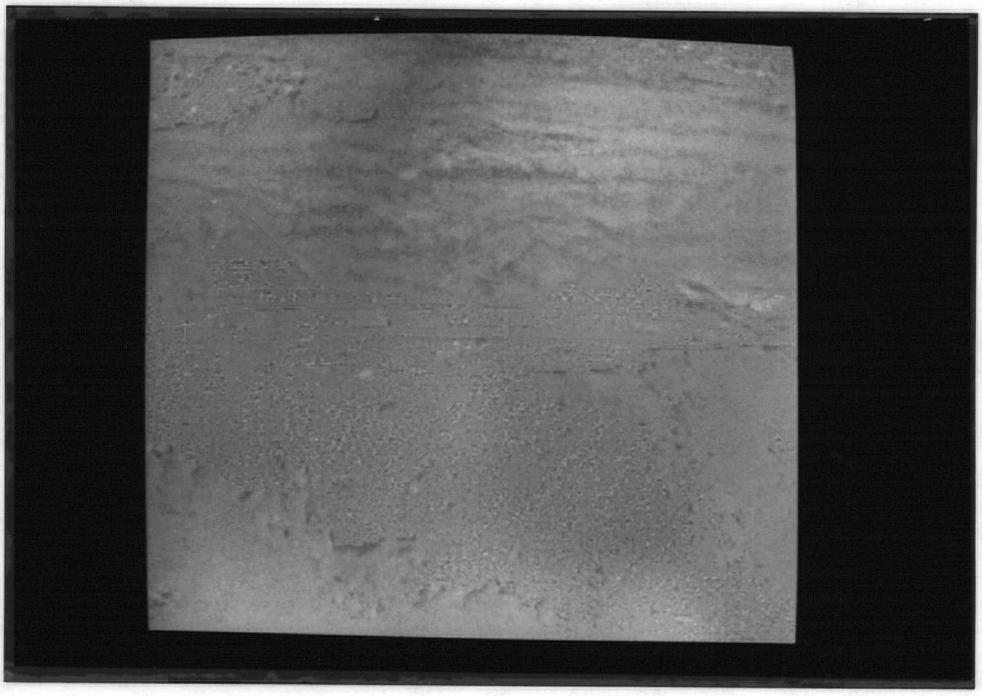


Figure 8.23: Difference picture between original and image obtained by ACC. (rate = 0.92 b/p.)

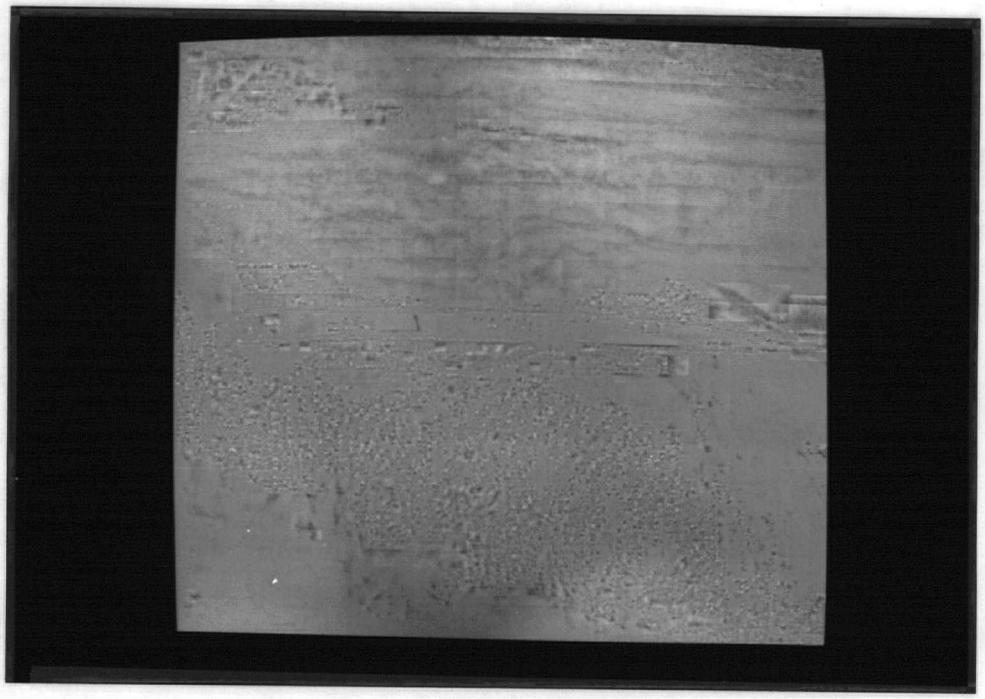


Figure 8.24: Difference picture between original and image obtained by ACC. (rate = 0.59 b/p.)

Figure 8.25: Original GIRL image.

Figure 8.26: Image obtained by AMBTC. Rate = 1.625 b/p.

Figure 8.27: Reconstructed image using ACC. Threshold = 16, rate = 1.38 b/p.

Figure 8.28: Reconstructed image using ACC. Threshold = 20, rate = 1.20 b/p.

Figure 8.29: Reconstructed image using ACC. Threshold = 30, rate = 0.93 b/p.

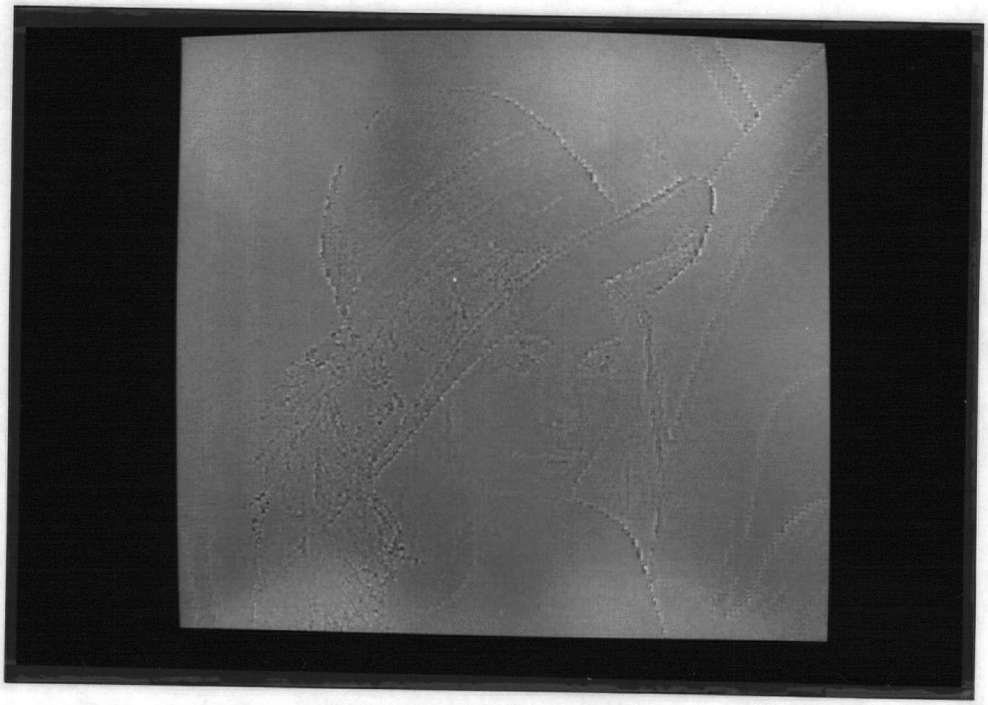


Figure 8.30: Difference picture between original and image obtained by AMBTC.

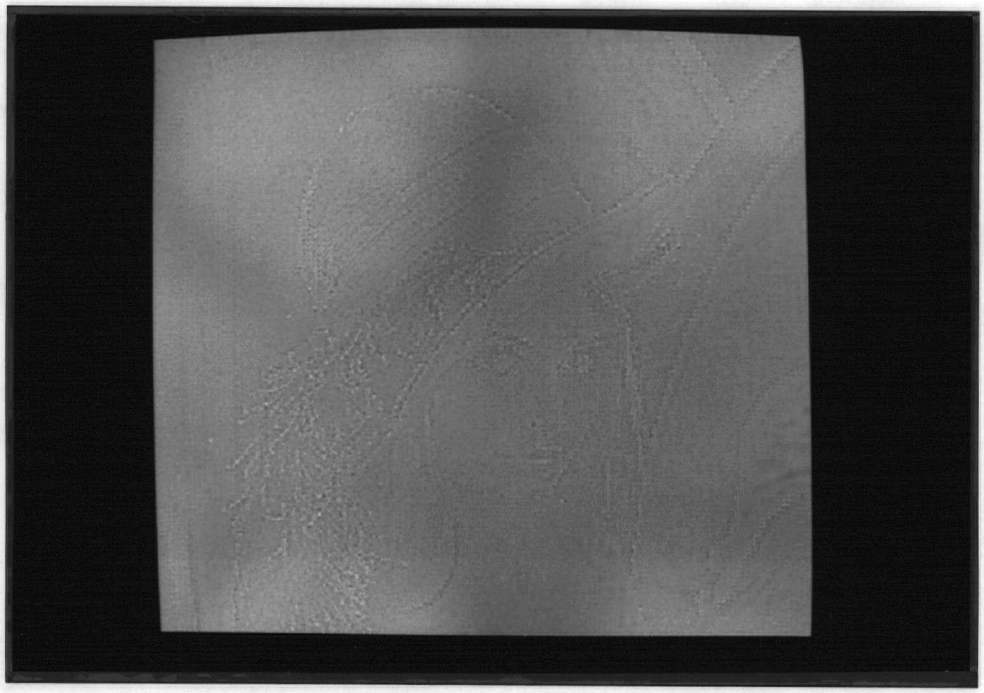


Figure 8.31: Difference picture between original and image obtained by ACC. (rate = 1.38 b/p.)

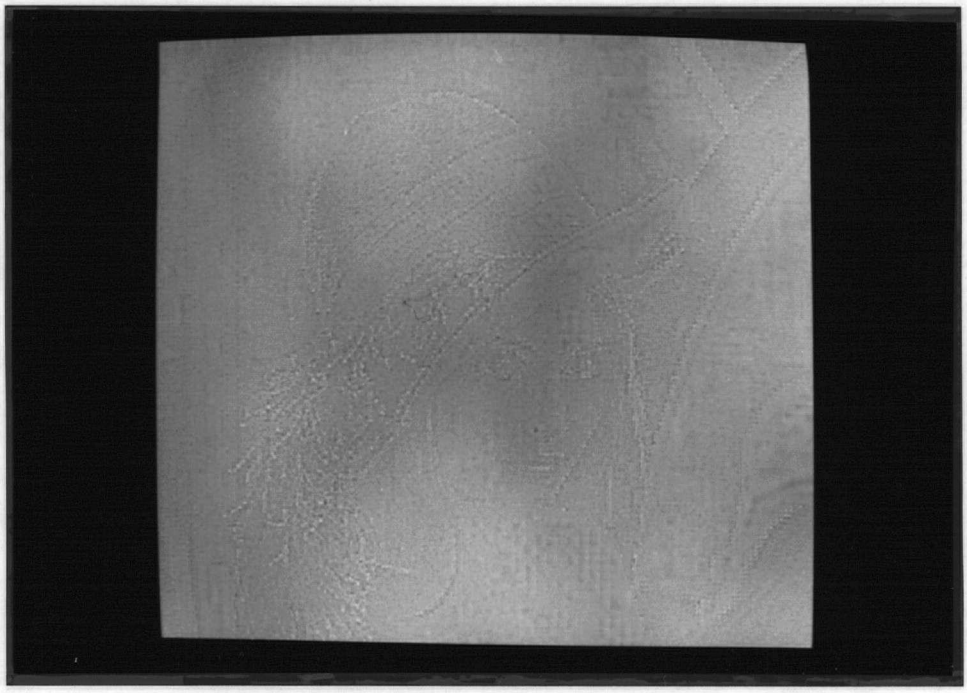


Figure 8.32: Difference picture between original and image obtained by ACC. (rate = 1.20 b/p.)

Figure 8.33: Difference picture between original and image obtained by ACC. (rate = 0.93 b/p.)

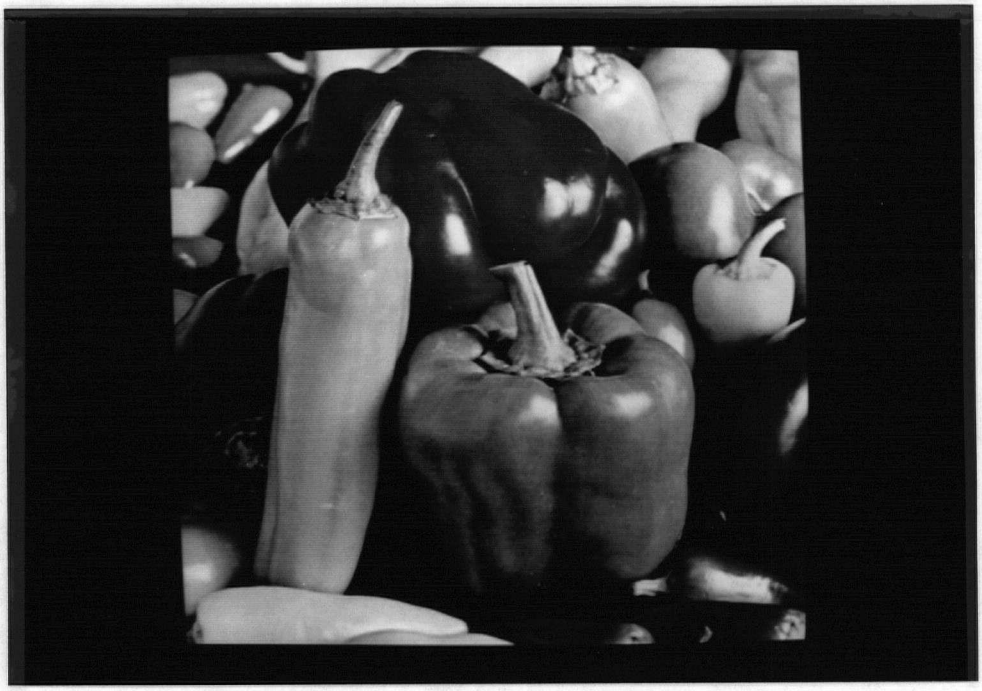


Figure 8.34: Original PEPPERS image.

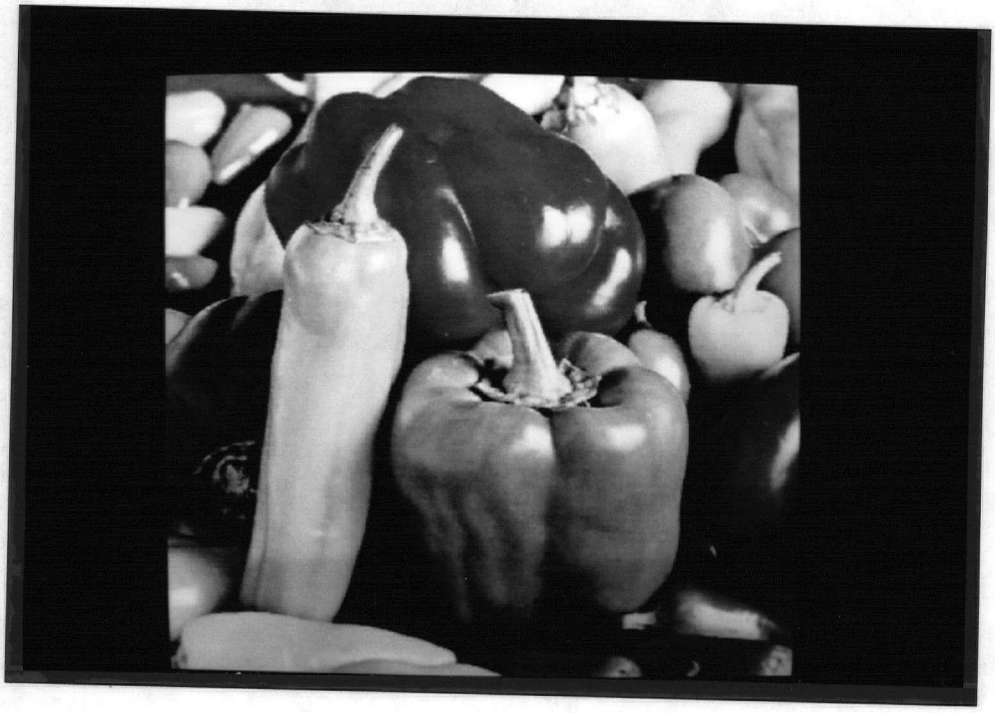


Figure 8.35: Image obtained by AMBTC. Rate = 1.625 b/p.

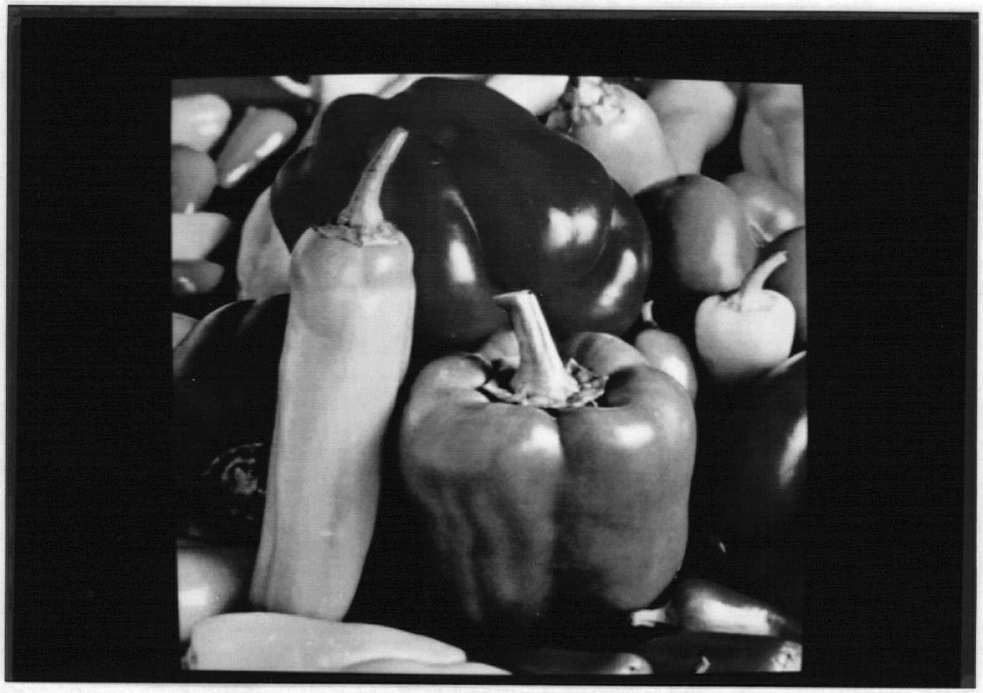


Figure 8.36: Reconstructed image using ACC. Threshold = 16, rate = 1.31 b/p.

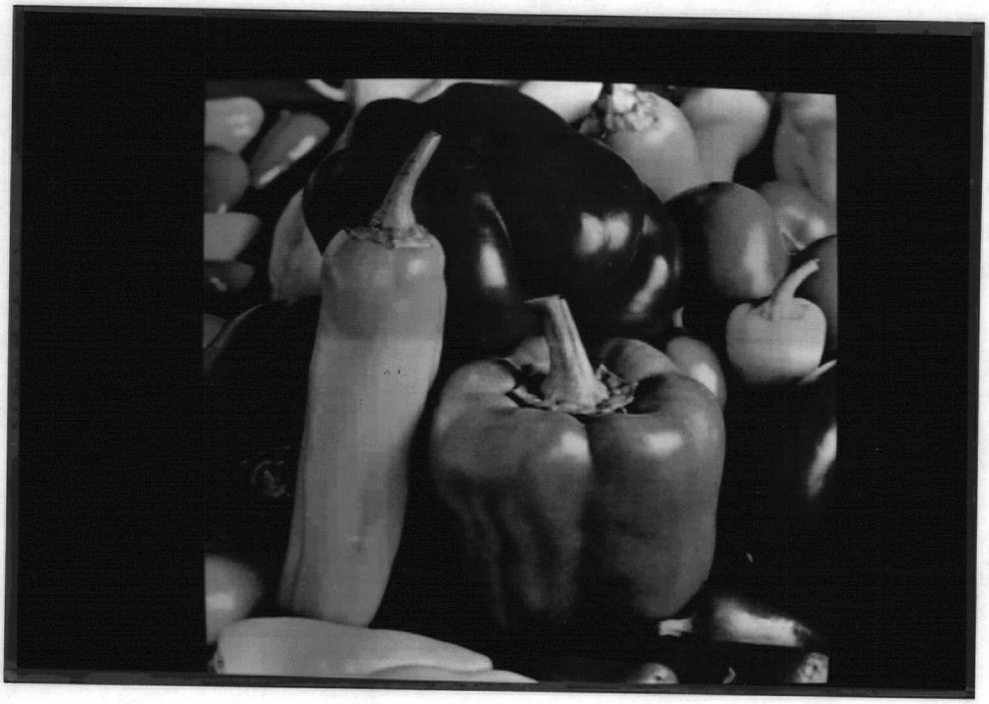


Figure 8.37: Reconstructed image using ACC. Threshold = 20, rate = 1.14 b/p.

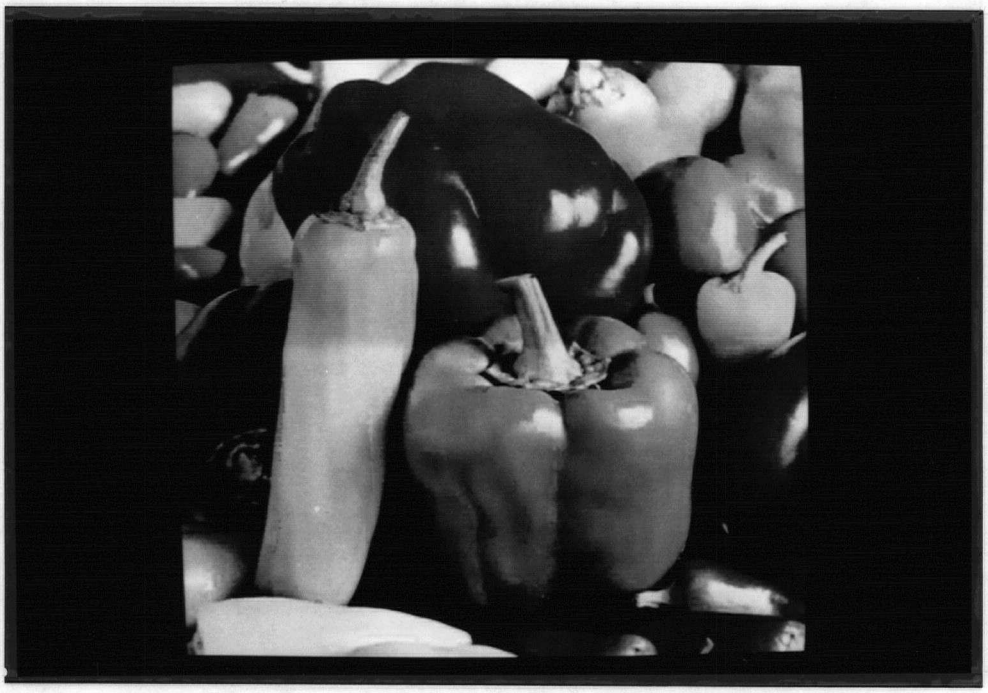


Figure 8.38: Reconstructed image using ACC. Threshold = 40, rate = 0.61 b/p.

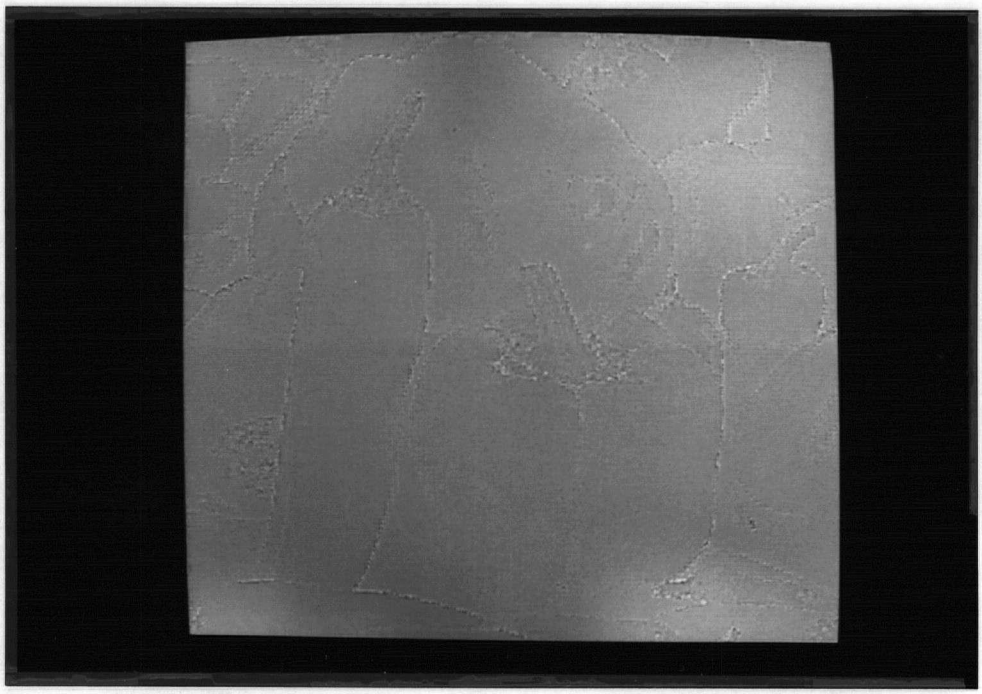


Figure 8.39: Difference picture between original and image obtained by AMBTC.

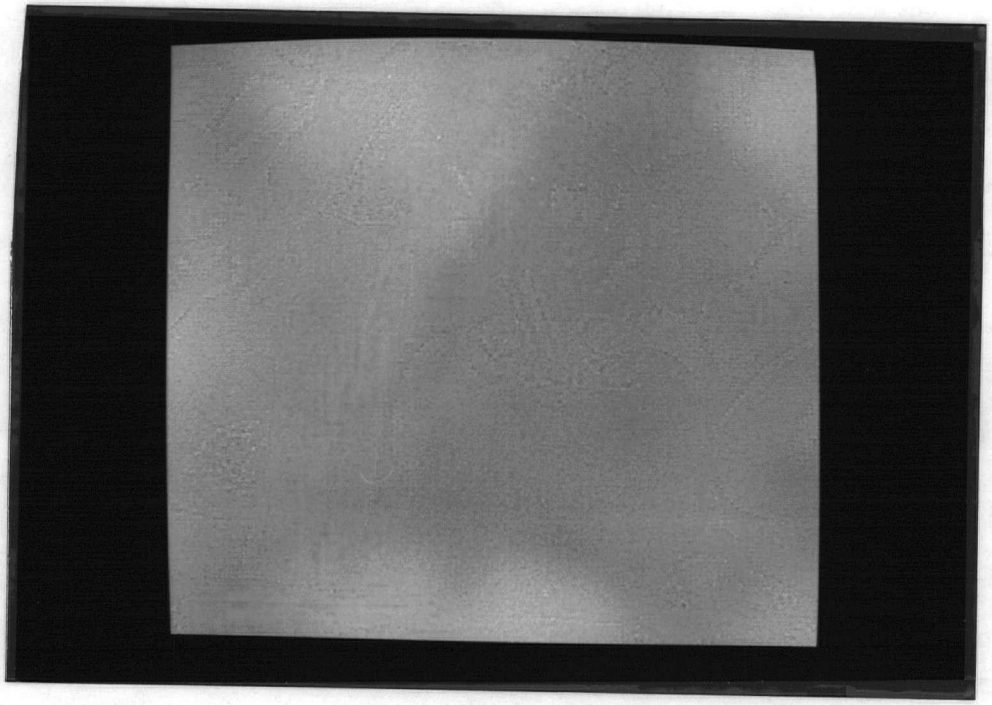


Figure 8.40: Difference picture between original and image obtained by ACC. (rate = 1.31 b/p.)

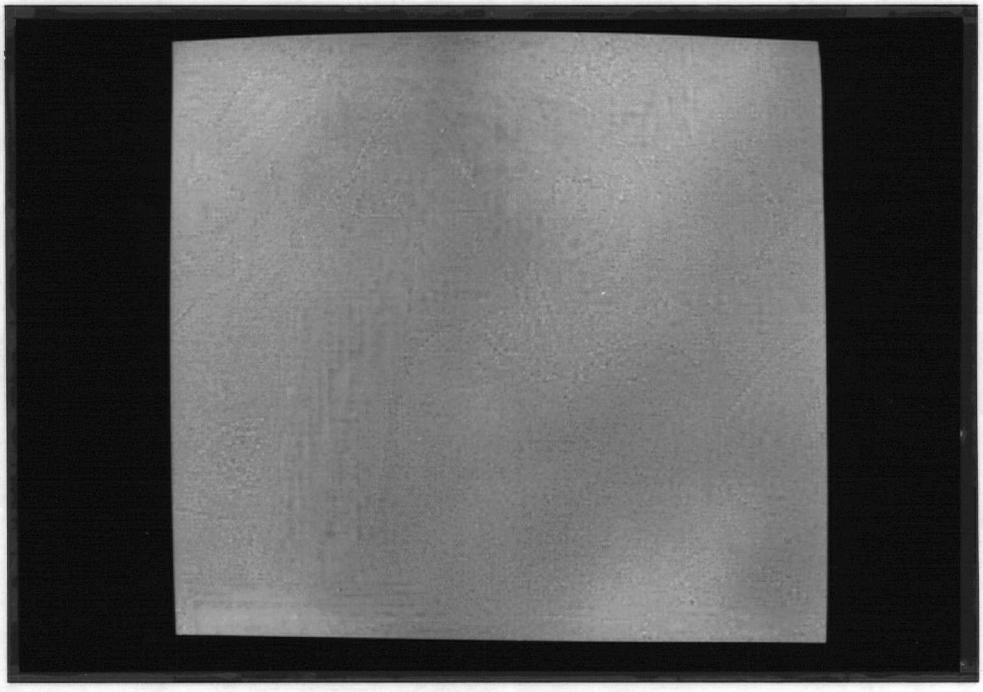


Figure 8.41: Difference picture between original and image obtained by ACC. (rate = 1.14 b/p.)

Figure 8.42: Difference picture between original and image obtained by ACC. (rate = 0.61 b/p.)

Figure 8.43: Original LAKE image.

Figure 8.44: Image obtained by AMBTC. Rate = 1.625 b/p.

Figure 8.45: Reconstructed image using ACC. Threshold = 16, rate = 1.54 b/p.

Figure 8.46: Reconstructed image using ACC. Threshold = 25, rate = 1.34 b/p.

Figure 8.47: Reconstructed image using ACC. Threshold = 50, rate = 0.81 b/p.

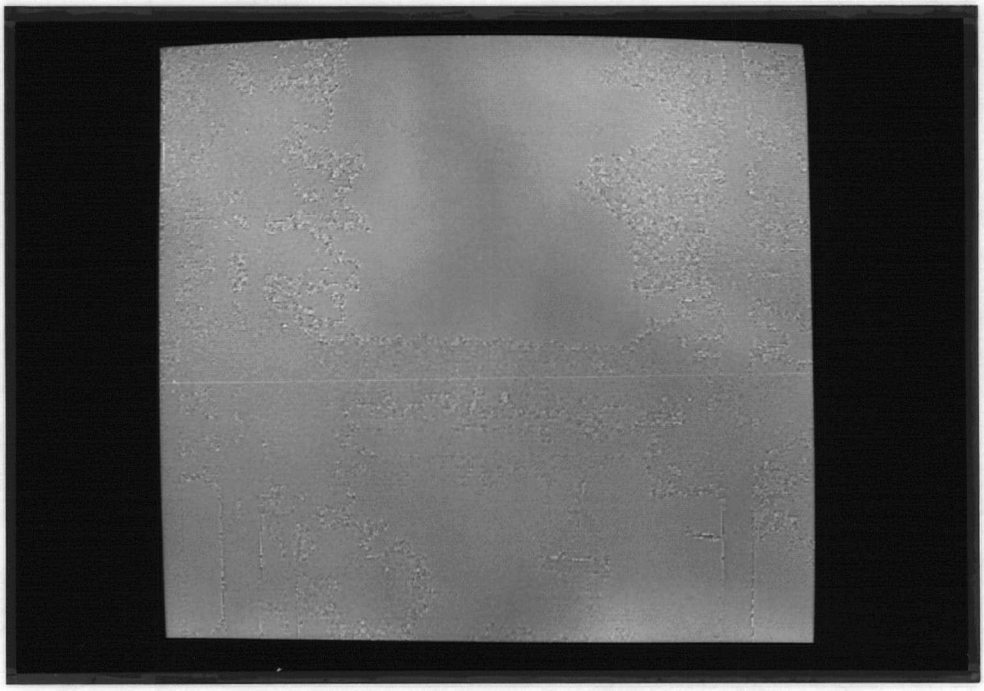


Figure 8.48: Difference picture between original and image obtained by AMBTC.

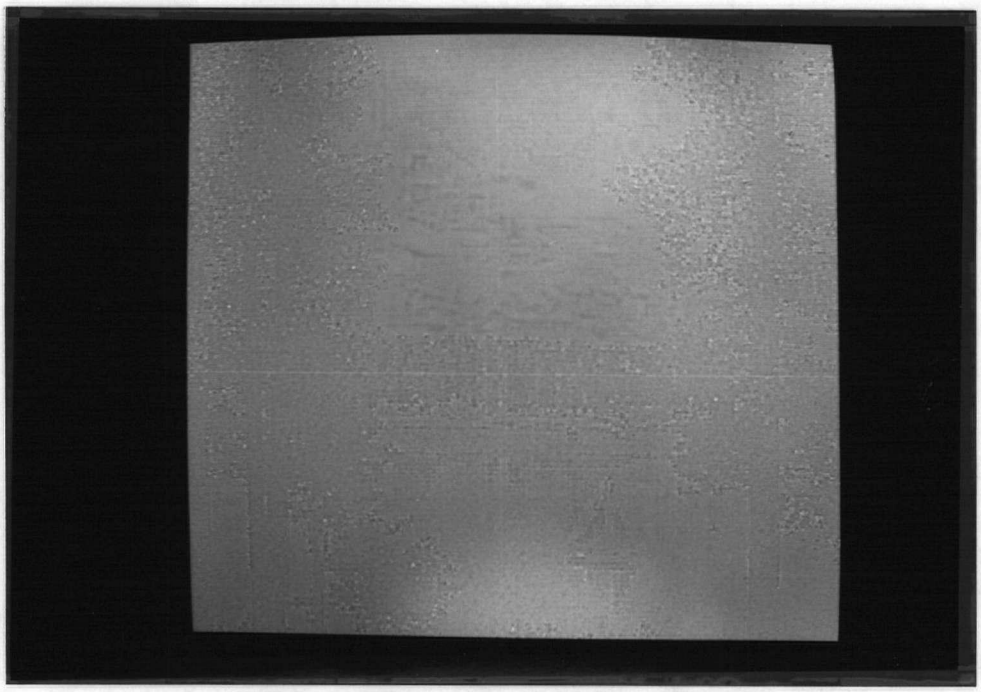


Figure 8.49: Difference picture between original and image obtained by ACC. (rate = 1.54 b/p.)

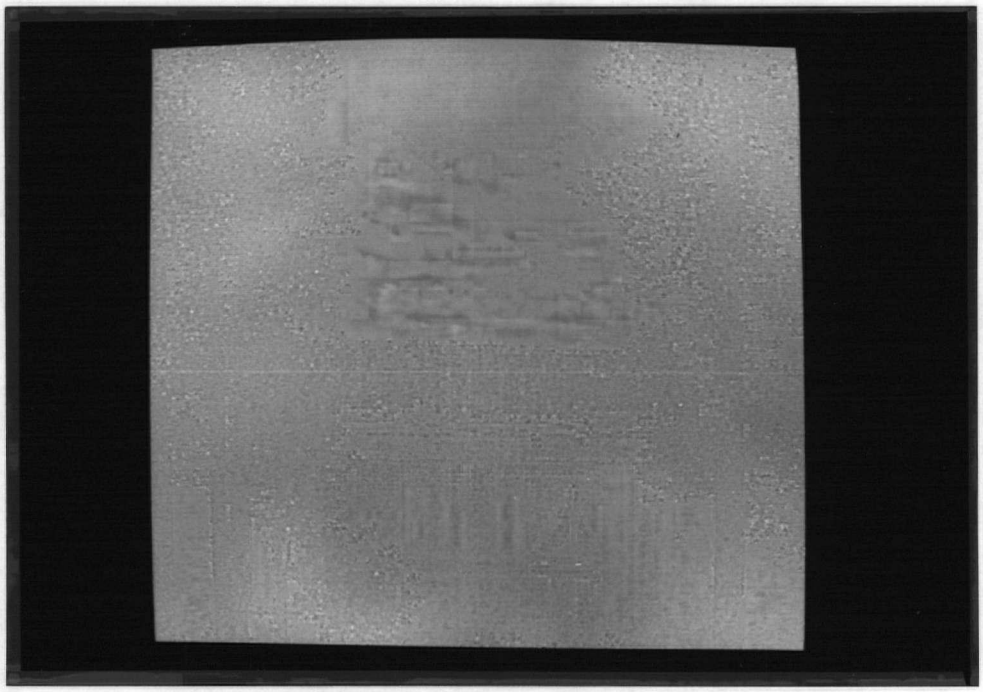


Figure 8.50: Difference picture between original and image obtained by ACC. (rate = 1.34 b/p.)

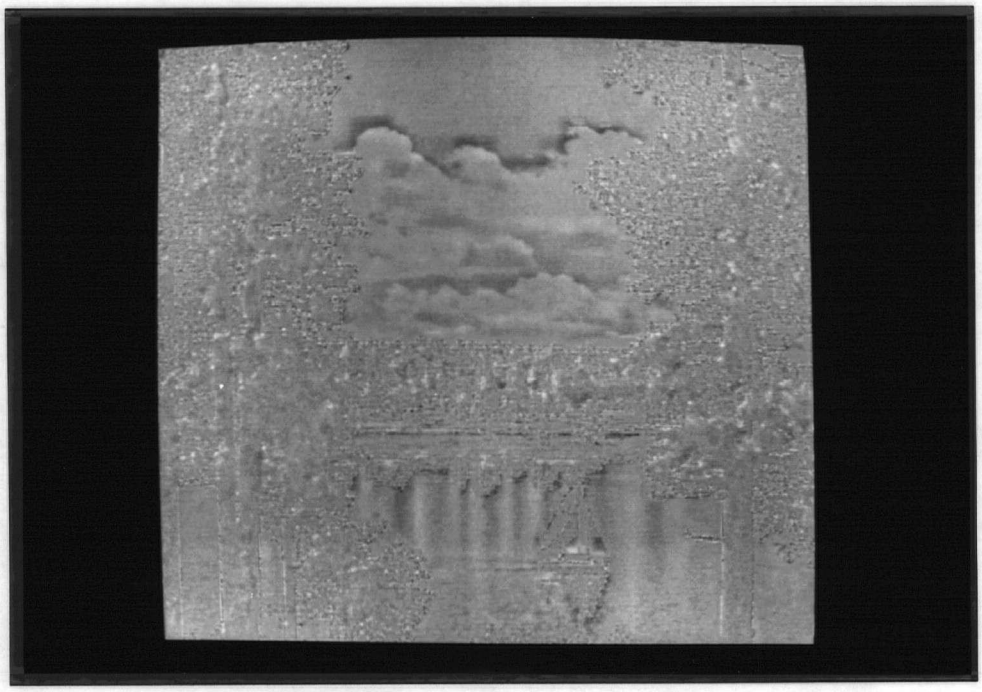


Figure 8.51: Difference picture between original and image obtained by ACC. (rate = 0.81 b/p.)

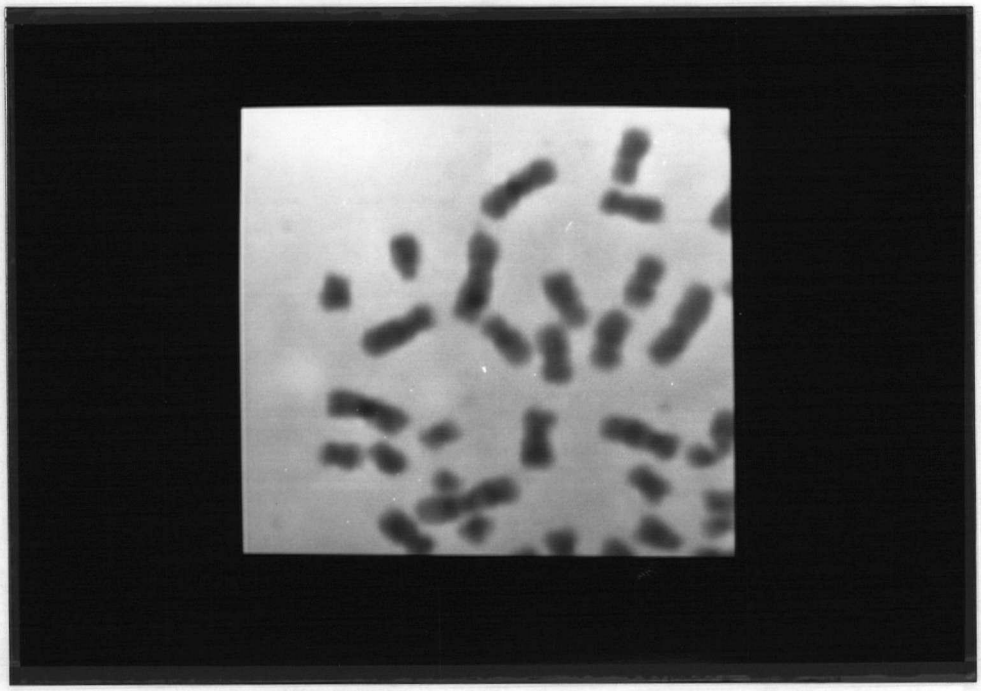


Figure 8.52: Original CHROMO image.

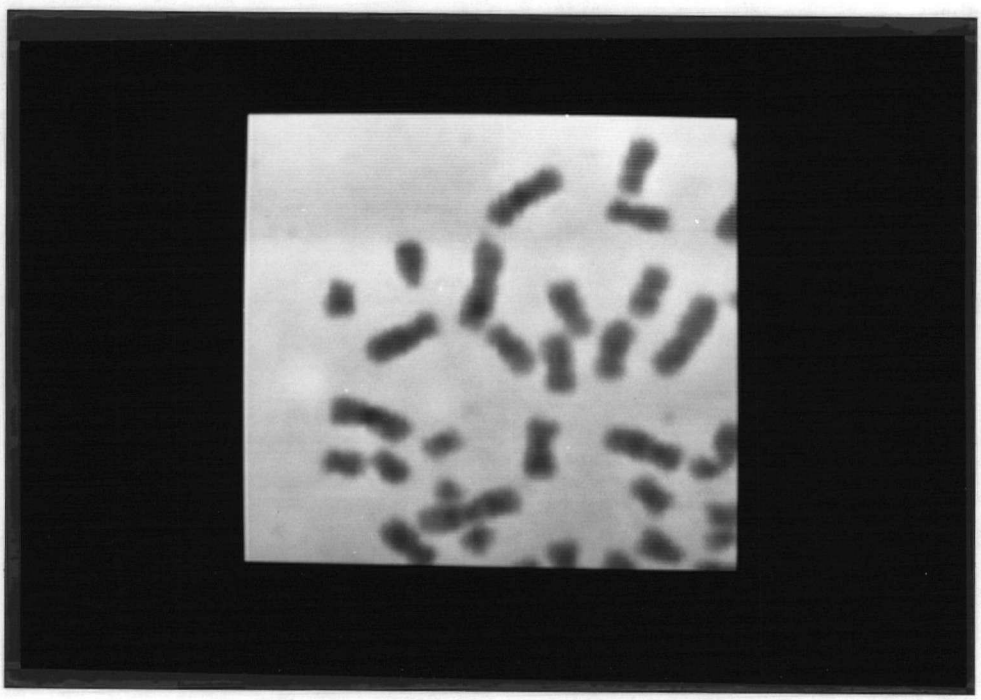


Figure 8.53: Image obtained by AMBTC. Rate = 1.625 b/p.

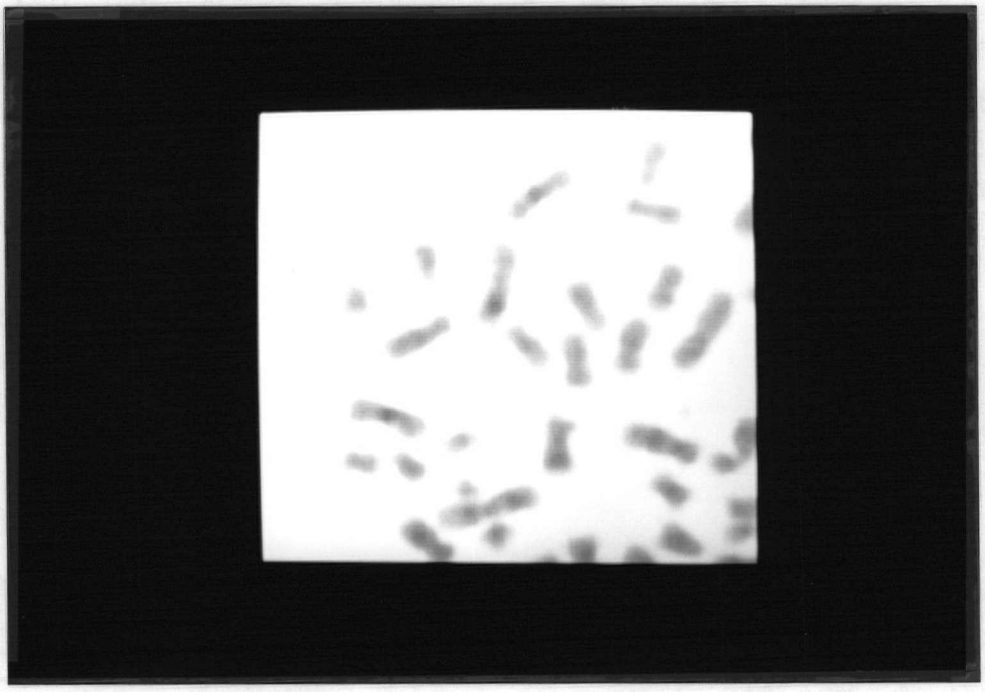


Figure 8.54: Reconstructed image using ACC. Threshold = 16, rate = 1.43 b/p.

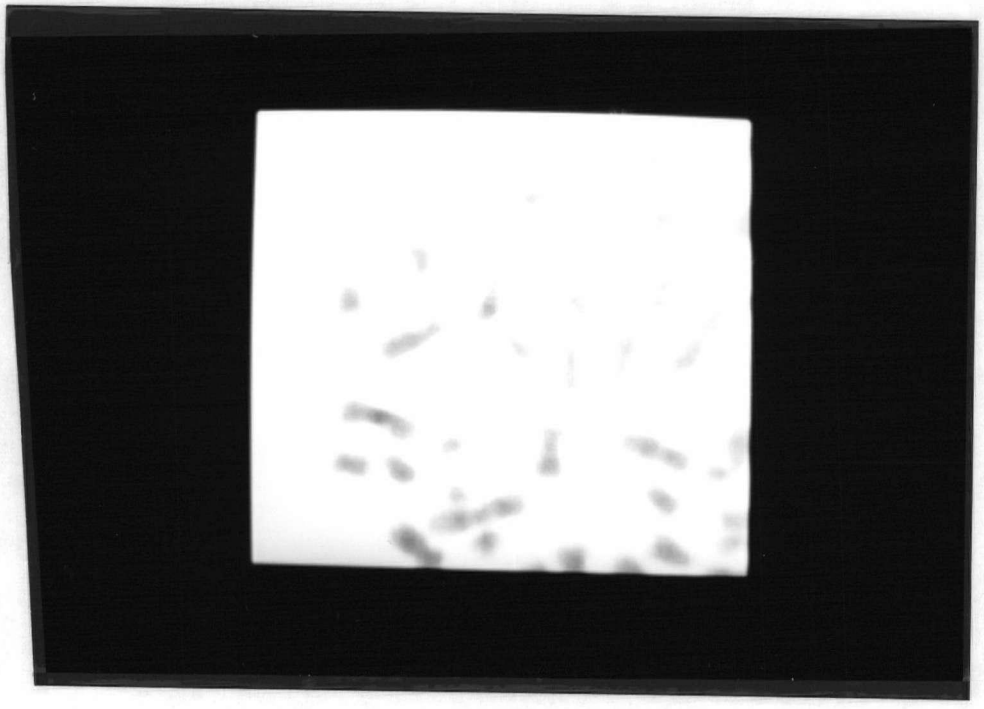


Figure 8.55: Reconstructed image using ACC. Threshold = 20, rate = 0.66 b/p.

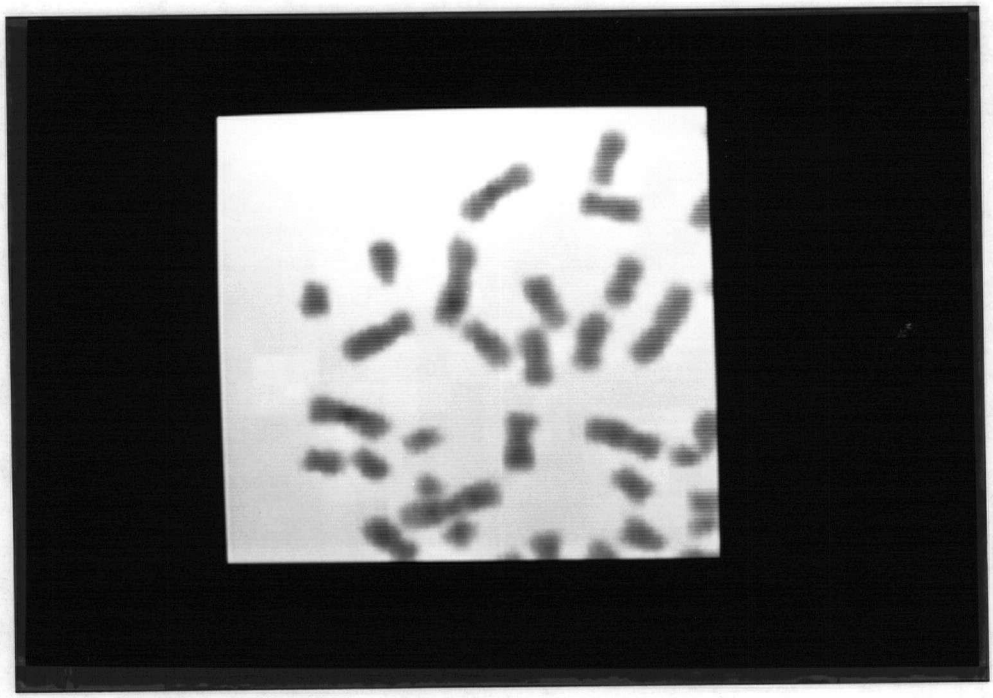


Figure 8.56: Reconstructed image using ACC. Threshold = 30, rate = 0.58 b/p.

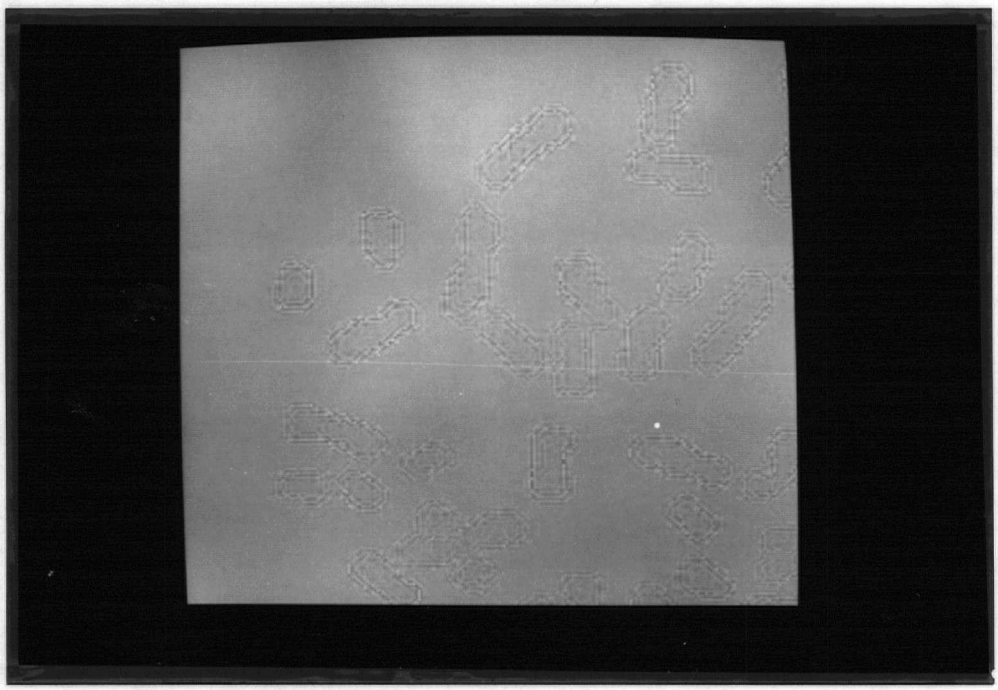


Figure 8.57: Difference picture between original and image obtained by AMBTC.

Figure 8.58: Difference picture between original and image obtained by ACC. (rate = 1.43 b/p.)

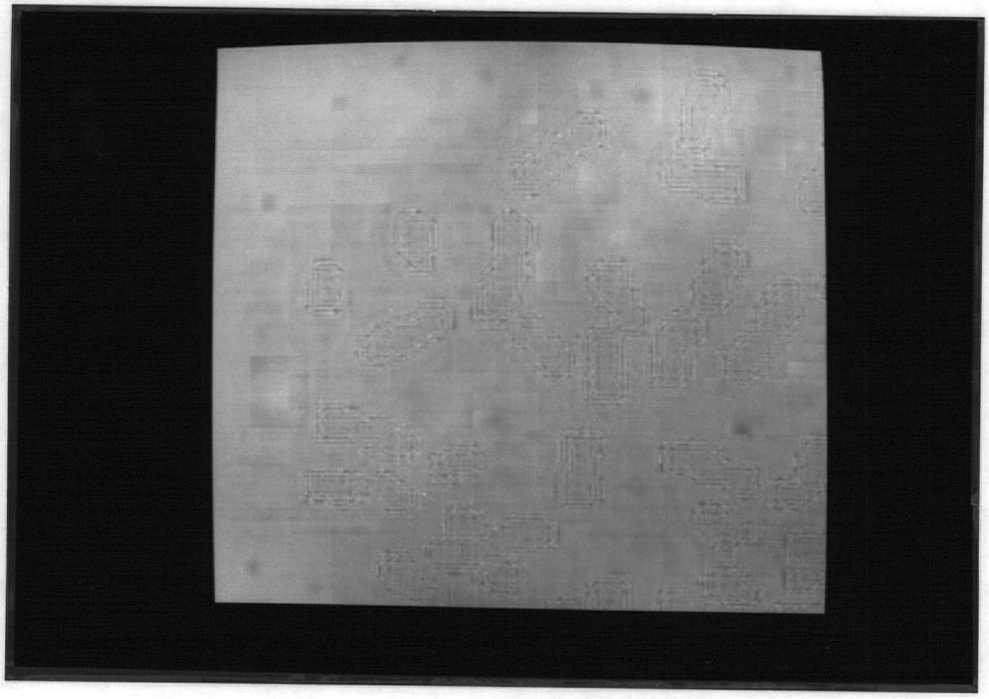


Figure 8.59: Difference picture between original and image obtained by ACC. (rate = 0.66 b/p.)

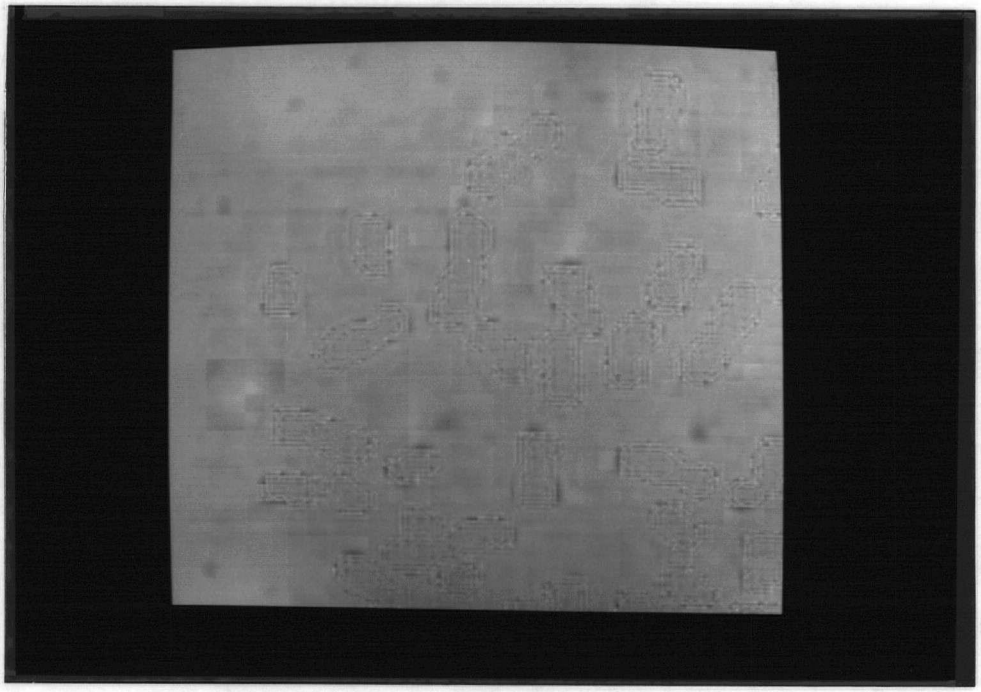
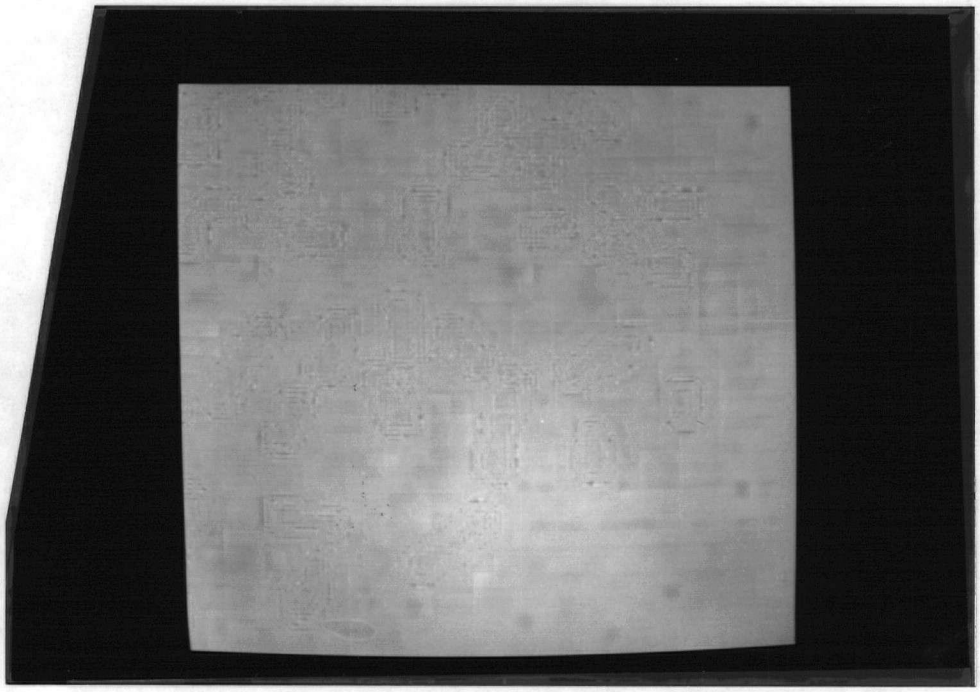


Figure 8.60: Difference picture between original and image obtained by ACC. (rate = 0.58 b/p.)



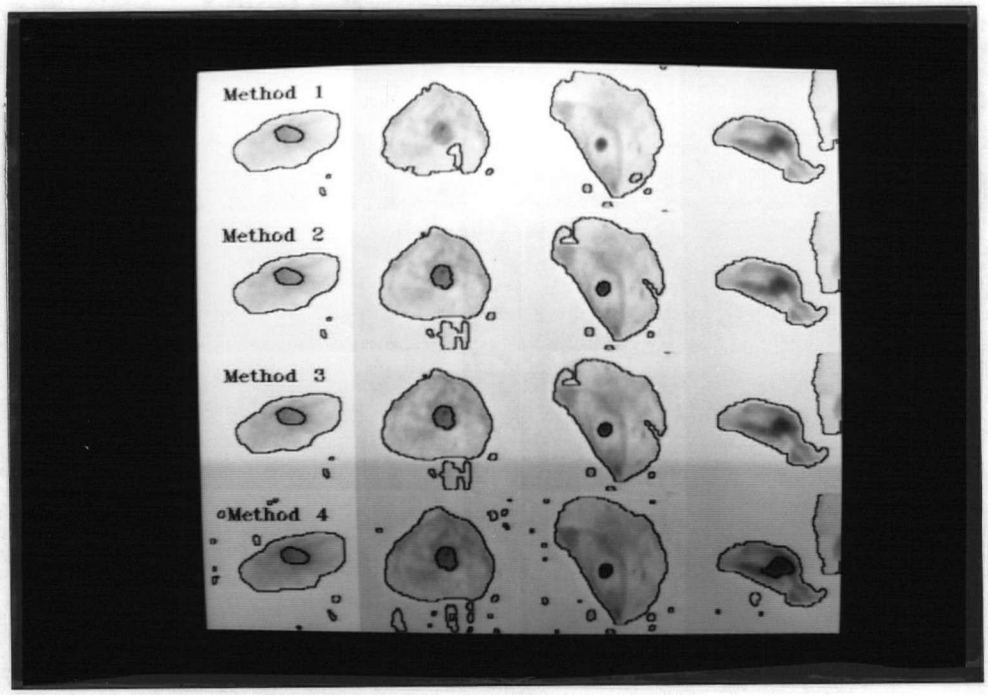


Figure 8.61: Original SMEAR image.

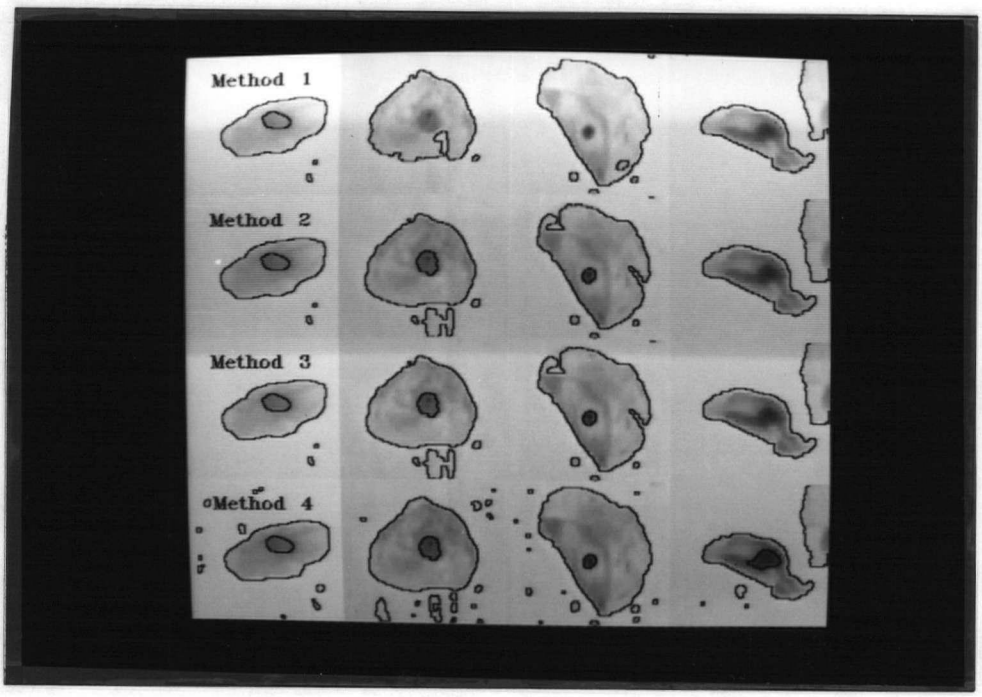


Figure 8.62: Image obtained by AMBTC. Rate = 1.625 b/p.

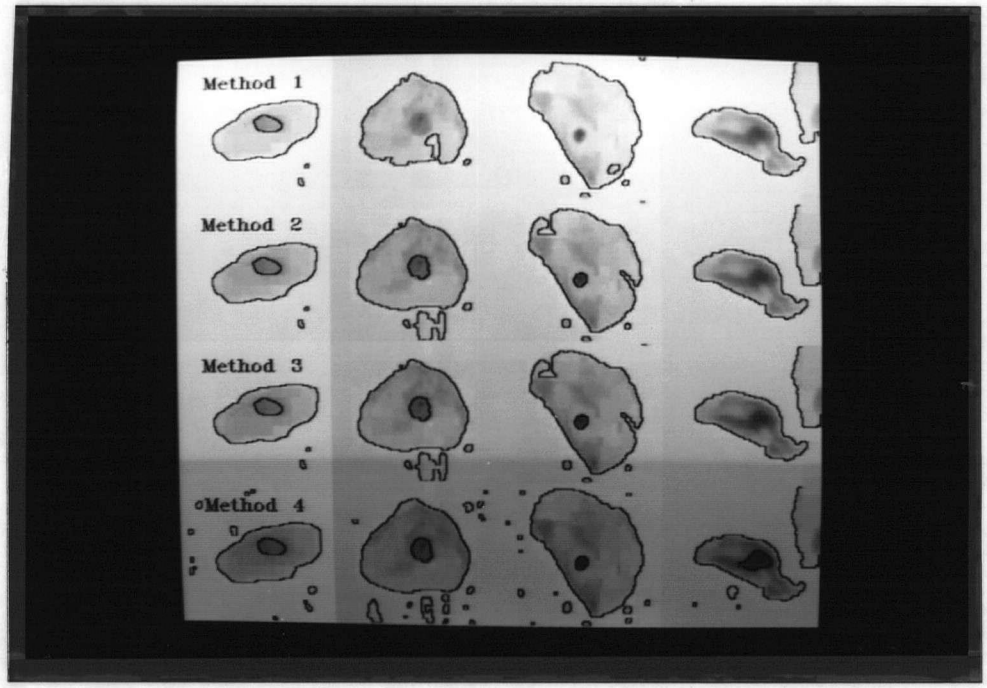


Figure 8.63: Reconstructed image using ACC. Threshold = 20, rate = 1.00 b/p.

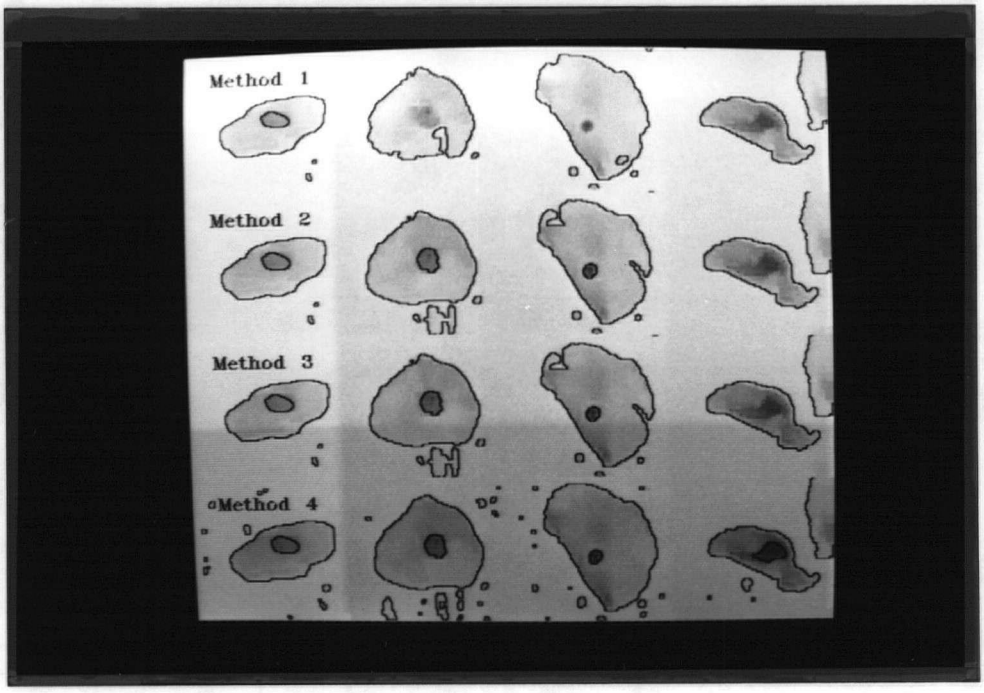


Figure 8.64: Reconstructed image using ACC. Threshold = 30, rate = 0.49 b/p.

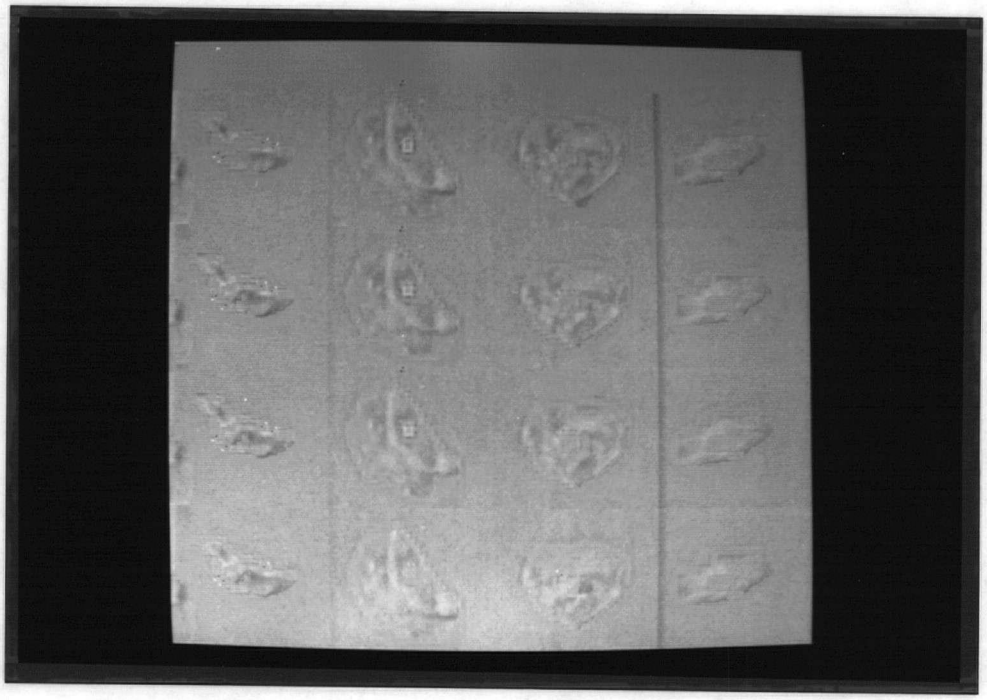


Figure 8.65: Difference picture between original and image obtained by AMBTC.

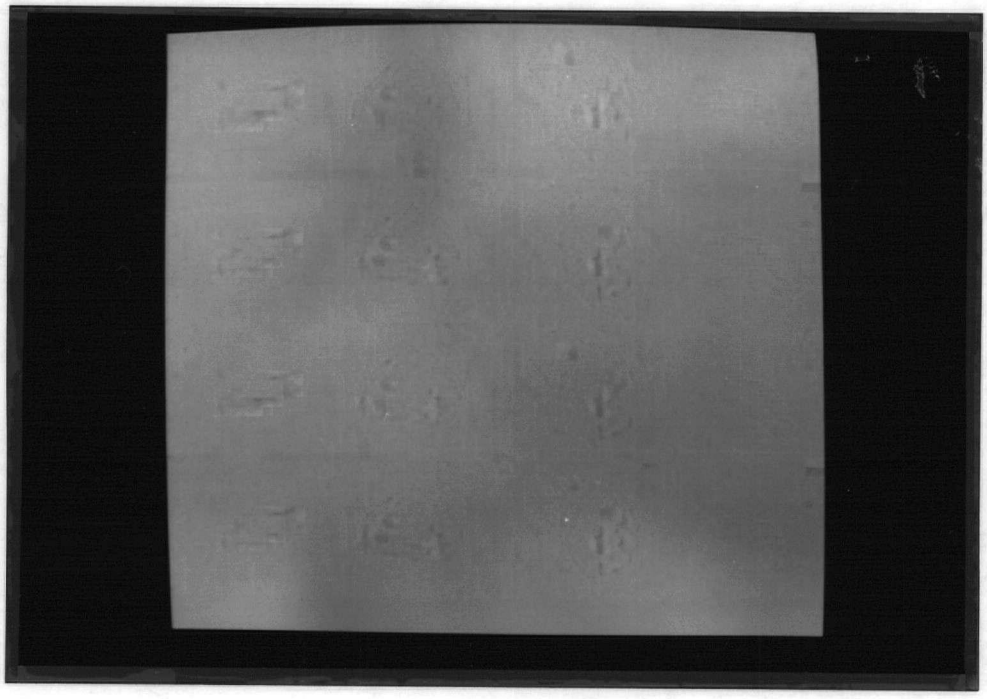


Figure 8.66: Difference picture between original and image obtained by ACC. (rate = 1.00 b/p.)

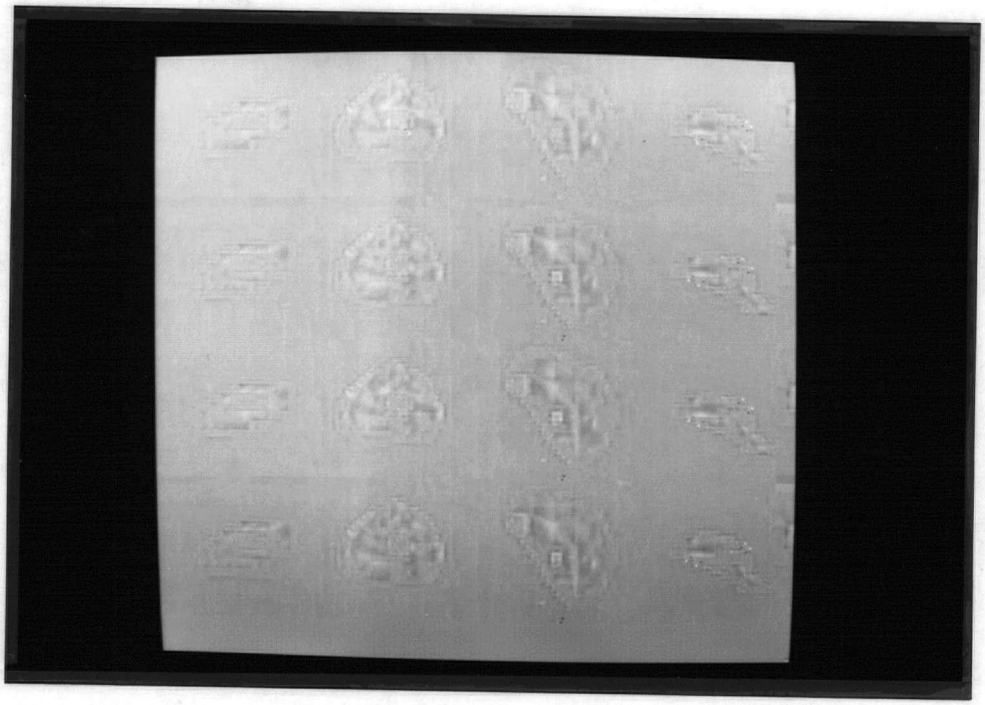


Figure 8.67: Difference picture between original and image obtained by ACC. (rate = 0.49 b/p.)

8..6 RMSE and Decompression Time Analysis

Another measure of reconstructed image fidelity is the Root Mean-Square Error defined by

$$e = \sqrt{\frac{1}{NxN}\sum_{i=1}^{N}\sum_{j=1}^{N}(x_{i,j} - x^*_{i,j})^2} \tag{8.22}$$

where $x_{i,j}$ and $x^*_{i,j}$ represent the original and reconstructed images respectively.

For high quality images ACC yields lower RMS error than that of AMBTC, for all images. At the upper limits of the method , smoothing becomes significant and the RMSE obtained is larger than that of AMBTC.

Whatever the application of a compression method is, decompression time is of importance. For the cases of the Adaptive Compression Coding, decompression time depends on the threshold value used. When that value gives rates close to those of AMBTC the processing times obtained are almost the same . As the threshold increases and the compressed data rates become lower the decompression time becomes significantly faster . The RMSE and Decompresion time are shown in the following table .

Image	Range	Rate	RMSE	Decomp. Time
F16BTC		1.63	7.63	29.80s
F1616	16	1.13	5.75	23.61s
F1630	30	0.92	6.18	18.81s
F1650	50	0.59	9.5	17.12s
GirlBTC		1.63	6.61	29.5s
Girl16	16	1.38	5.89	34.12s
Girl20	20	1.20	6.05	29.51s
Girl30	30	0.93	7.5	23.13s
PeppersBTC		1.63	5.69	30.25s
Peppers16	16	1.31	4.72	32.50s
Peppers20	20	1.14	4.89	28.79s
Peppers40	40	0.61	8.67	18.01s
LakeBTC		1.63	8.52	30.26s
Lake16	16	1.54	7.22	33.37s
Lake25	25	1.34	7.51	29.22s
Lake50	50	0.81	10.85	23.10s
ChromoBTC		1.63	3.13	6.33s
Chromo16	16	1.43	2.12	3.01s
Chromo20	20	0.66	2.89	3.33s
Chromo30	30	0.58	3.61	3.55s
SmearBTC		1.63	2.10	28.75s
Smear20	20	1.00	1.45	10.81s
Smear30	30	0.49	3.80	14.09s

Table 8.9: Compressed Data Rates , Root Mean-Square Errors and Decompression Times.

Chapter 9

Conclusions

An adaptive image compression technique, ACC, was presented. This algorithm was developed around Absolute Moment BTC . The latter version was chosen instead of BTC because of its efficiency and the error reduction that it provides. Both AMBTC and BTC are known to have many advantages. For each 4X4 pixel block the compression data of each method uses a 16 bit binary block.

For our method, look up tables were used to replace the binary block of the Block Truncation method in two different cases. This approach leads to lower compression data rates without affecting the quality of the reconstructed image. Smooth regions of the image were approximated by their average and smoothness was successfully detected by using the *range* .

A method for preserving edges was introduced. It was shown that as more details are preserved around edges the pictorial results improve dramatically. It was also shown that this method can reduce or even eliminate the ragged appearance of the edges, resulting in images far superior than those of AMBTC.

For most of the images ACC yielded Root Mean-Square Errors smaller than those obtained by AMBTC. Decompression time was comparable to that of AMBTC for low threshold values. The decompression time becomes significantly lower as the compression rate becomes smaller.

The efficiency of the algorithm depends on the degree of texture in the image.

The threshold for detecting edges was fixed at 120 and the threshold used in the

look up tables for detecting textures was fixed at 20. Although the method uses range for the threshold to detect smoothness, it can easily be seen that a value between 16 and 25 for that threshold will yield high quality and low data rates for most of the images, with RMSE and decompression times lower than those of AMBTC.

The upper limits of ACC on compression rates are also discussed. An adaptive filter was used to improve the visual quality of the decompressed picture and the upper limits of ACC.

Another advantage of the algorithm is its easy implementation , since no special hardware is needed.

References

[1] E. J. Delp and R. Mitchell, "Image Compression Using Block Truncation Coding",*IEEE Transactions on Communications* , Vol. COM-27, No. 9, pp. 1335 - 1342, September 1979.

[2] D. J. Healy and O.R. Mitchell, " Digital video bandwidth compressing block truncation coding", *IEEE Transactions on Communications* , Vol. COM-29, pp. 1809 - 1817, Dec. 1981.

[3] E. J. Delp and O. R. Mitchell , " Multilevel Graphics Representation Using Block Truncation Coding", *Proc. IEEE* , Vol. 68, No. 7, pp. 868 - 873, July 1980.

[4] A. N. Netravali and J. O. Limb, "Picture Coding: A Review", *Proceedings of the IEEE* , Vol. 68, No. 3, pp. 366 - 403, March 1981.

[5] M. D. Lema and O. R. Mitchell, "Absolute Moment Block Truncation Coding and its Application to Colour Images", *IEEE Transactions on Communications* , Vol. COM-32, No. 10, pp. 1148 - 1157, October 1984.

[6] J. L. Mannos and D. J. Sakrison, "The Effects of a Visual Fidelity Criterion on the Encoding of Images", *IEEE Transactions on Information Theory* , Vol. IT-20, No. 4, pp. 525 -536, July 1974.

[7] A. K. Jain, "Image Data Compression: A Review", *Proceedings of the IEEE* , Vol. 69, No. 3 , pp. 349 - 389, 1981.

[8] Z. L. Budrikis, "Visual Fidelity Criterion and Modeling", *Proceedings of the IEEE* , Vol. 60, No. 7, pp. 771 - 778, July 1972.

[9] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-Generation Image-Coding Techniques", Proceedings of the IEEE , Vol. 73, No. 4, pp. 549 - 574, 1985.

[10] D. R. Halverson, N. C. Griswold, and G. L. Wise, " A Generalized Block Truncation Coding Algorithm for Image Compression", *IEEE Transactions on Acoustics, Speech, and Signal Processing* , Vol. ASSP-32, No. 3, pp. 664 - 668, June 1984.

[11] G. R. Arce and N. C. Gallagher,Jr., "BTC Image Coding Using Median Filter Roots", *IEEE Transactions on Communications* , Vol. COM-31, No. 6, pp. 784 -793, June 1983.

[12] A. Rosenfeld and C. Kak, "Digital Picture Processing", *Second Edition*, Volume 1, pp. 176 - 181 , Orlando, Florida: Academic Press, 1982.

[13] D. J. Vaisey and A. Gersho, "Variable Block-Size Image Coding", *in Proc. IEEE ICASSP* , pp. 1051 - 1054, May 1987.

[14] W. K. Pratt, *Digital Image Processing* , Wiley, New York, 1978.

[15] D. Marr, *Vision* , Freeman, San Francisco, 1982.

[16] B. K. P. Horn, *Robot Vision* , MIT Press, Cambridge, 1986.

[17] R. A. Nobakht and S. A. Rajala, " An Image Coding Technique Using A Human Visual System Model and Image Analysis Technique", in *Proc. IEEE ICASSP* , vol. 69, no. 5, 529-541, 1981.

[18] R. M. Gray, " Vector Quantization ", *IEEE ASSP Magazine* , pp. 4-29 April, 1984.

[19] R. M. Haralick and L. G. Shapiro, "Image Segmentation Techniques", *Image Processing* , vol. 29, pp. 100-132, 1985.

[20] D. H. Kelly, "Effects of sharp edges on the visisbility of sinusoidal gratings", *J. Opt. Soc. Amer.* , vol. 60, pp. 98-103, Jan. 1970.

Appendix A

Appendix

The 128 look-up tables are chosen to represent different cases of edges. Most of the edges are chosen to have width of at least two pixels. This is justified because edges are almost never represented as step functions but more like gradual steps.

Only 64 of the 128 tables are shown. The other 64 tables are chosen to be the inverse representation of the first 64.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
0 1 1 0
0 1 1 0
0 1 1 0
0 1 1 0
```

```
1 1 0 0
1 1 0 0
1 1 0 0
1 1 0 0
```

```
0 0 0 0
1 1 1 1
1 1 1 1
0 0 0 0
```

```
0 0 0 0
0 0 0 0
1 1 1 1
1 1 1 1
```

```
1 1 1 1
1 1 1 1
1 1 0 0
1 1 0 0
```

```
0 0 0 0
0 1 1 1
0 1 1 1
0 1 1 0
```

```
1 1 1 1
1 1 1 1
0 0 1 1
0 0 1 1
```

```
0 0 0 0
1 1 1 0
1 1 1 0
0 1 1 0
```

```
1 1 0 0
1 1 0 0
1 1 1 1
1 1 1 1
```

```
0 1 1 0
0 1 1 1
0 1 1 1
0 0 0 0
```

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

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

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

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

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

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

```
0 0 0 0
0 0 0 0
1 0 0 0
0 1 0 0
```

```
1 1 1 1
1 0 0 0
1 0 0 0
1 0 0 0
```

```
0 0 0 0
0 1 1 1
0 1 0 0
0 1 0 0
```

```
0 0 0 0
0 0 0 0
0 0 1 1
0 0 1 0
```

```
0 0 0 1
0 0 0 1
0 0 0 1
1 1 1 1
```

```
0 0 1 0
0 0 1 0
1 1 1 0
0 0 0 0
```

```
0 1 0 0
1 1 0 0
0 0 0 0
0 0 0 0
```

```
1 0 0 0
1 0 0 0
1 0 0 0
1 1 1 1
```

```
0 1 0 0
0 1 0 0
0 1 1 1
0 0 0 0
```

```
0 0 1 0
0 0 1 1
0 0 0 0
0 0 0 0
```

```
1 1 1 1
0 0 0 1
0 0 0 1
0 0 0 1
```

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