NASA Technical Paper 2722

September 1987

Experiments in Encoding Multilevel Images as Quadtrees

Donald L. Lansing

NASA

NASA
Technical
Paper
2722

1987

Experiments in Encoding Multilevel Images as Quadtrees

Donald L. Lansing

Langley Research Center
Hampton, Virginia

National Aeronautics
and Space Administration

**Scientific and Technical
Information Office**

Nomenclature

byte	unit of storage consisting of 8 bits, used to hold integer values in range 0 to 255
FLAG	one element of array of words used to store 32 node flags
Gray-Level	gray level value
gray level	unsigned, 1-byte value representing index into color table containing gray levels or pseudocolors of image
IND	single bit indicating whether quadrant has uniform gray level
IVAL	one byte value returned during image decoding, = gray level of uniform quadrant
K	suffix indicating a multiple of 1024
L	length of one side
LEVEL	one element of array of words used to store gray levels of four quadrants
multilevel image	digital image containing up to 256 gray levels or pseudocolors
NINT (x)	function returning value of integer nearest to floating point number x
node flag	single bit used to indicate whether quadtree node represents uniform quadrant
pixel	smallest individual element of digital image; characterized by its gray level
pseudocolor	arbitrary assignment of color to pixels in image in order to highlight or accentuate some feature or property
QUAD ( )	procedure for coding image in quadtree data structure
standard image	digital image, 512 pixels on a side, having up to 256 gray levels
word	unit of storage, equivalent to 4-byte, 32-bit integer data type
x, y	coordinates
μ	average value of all gray levels in quadrant
σ	standard deviation of all gray levels in quadrant
σ/μ	segmentation parameter, controls spread of gray levels about their mean value

Procedures:

EQUAL (quad)	determine if current quadrant has uniform gray level
ENCODE (0-1,Gray-Level)	pack bit to mark node type and gray level if node is uniform
DECODE (IND,IVAL)	unpack bit IND indicating node type and gray level IVAL if uniform
VISIT (quad)	process data for quadrant; that is, display quadrant or calculate its histogram

Introduction

With the explosive growth in the use of raster graphics and the expanding application of image processing methods in many engineering and scientific disciplines has come a concomitant increase in the number of files containing images and graphics in digital form. A number of file formats are currently in use because of the variety of mainframes, graphics software packages, and display devices employed for this type of work. Clearly, any scheme which will reduce the amount of storage required by image and graphics files, reduce the time required to access and process an image, or accelerate file transmission between internal systems or research centers is potentially of considerable interest. This paper is a first step toward developing data on image storage requirements for several encoding methods and exploring the use of quadtree representations for multilevel images.

This paper presents the results of encoding a variety of images with up to 256 gray levels using three schemes: full-raster, runlength, and quadtree. Full-raster and runlength encoding are standard methods used at the Langley Research Center for storing and transmitting raster images. There is, however, no previous experience here using a quadtree format.

An image quadtree is a hierarchial representation of a 2^n by 2^n binary array in the form of a tree. Considerable literature is available on the use of quadtrees to store and manipulate binary images. Reference 1 is an excellent survey. The application of quadtrees to multilevel images is relatively undeveloped although their use for analyzing geographic data has been mentioned in the literature. The potential advantage of quadtree encoding for reducing storage is that an entire area with a uniform gray level may be encoded in a single leaf node. Whereas runlength encoding achieves compression by taking advantage of coherence along scan lines, quadtree encoding may achieve compression by taking advantage of coherence over areas.

This paper describes a pointerless quadtree encoding scheme. Such space saving representations are of interest (refs. 2 and 3) for reducing the overhead associated with maintaining the tree and thereby also reducing the storage of the encoded image. Data are presented on the size of the quadtree required to encode selected images and on the relative storage requirements of the three encoding schemes. In addition, a parametric segmentation method based on the statistical variation of gray levels within a quadrant formed during the quadtree subdivision process is described. Such a method may be used to maintain the storage required by an encoded image within prescribed limits (but with loss of detail and the introduction of visible artifacts) or to represent an image to various levels of detail for feature identification. Several sets of black and white and pseudocolor segmented images obtained by varying the segmentation parameter are shown.

The author wishes to acknowledge discussions with Terence Abbott of the Flight Management Division and Keith Miller of Virginia Commonwealth University during which the recursive encoding to eliminate storing x, y, L was suggested. Mary-Anne K. Posenau of the Analysis and Computation Division recommended the economical procedure for packing 1-bit node flags. The assistance of William von Ofenheim and Mary-Anne K. Posenau of the Analysis and Computation Division for making these images available on the mainframe containing the applications software is gratefully acknowledged. William von Ofenheim also improved the original program by incorporating a commercially available library of graphics subroutines to drive the display device.

Encoding Schemes

Three encoding schemes were used: full-raster, runlength, and quadtree. This section describes each of these schemes and gives relevant information on their performance and storage requirements. Throughout the paper, the term "standard image" refers to an image size of 512 by 512 = 262,144 = 256K pixels having up to 256 gray levels (8 bits per pixel). It is convenient to express storage requirements in units of "words." Here a "word" refers to a FORTRAN integer data type of 4 bytes or 32 bits. The term "gray level" refers to one of 256 possible indices in a look-up table which maps the index into a brightness, producing a gray scale image. These indices could also be mapped through a color look-up table to produce a pseudocolor image.

Full-Raster

In this scheme, the gray level of each pixel in the image is stored in some systematic order. The typical convention is row by row, left to right from the top of the image to the bottom. A standard image requires 64K words of storage when saved in this manner. Since this fixed amount of storage is independent of the content of the image, it represents both the best and worst possible cases.

Runlength

The runlength encoding scheme scans the image top to bottom and left to right and encodes both

the number of consecutive pixels with the same gray level and the gray level. The scheme employed in this study used constant-length encoding; that is, 1 byte was required to encode the number of pixels and 1 byte was required to encode the gray level. Thus, 2 bytes are required to store from 1 to 256 consecutive pixels having equal gray levels. Runlength encoding compresses an image by removing redundancy along scan lines. For a standard image of uniform gray level, 1 word of storage is required per image row; thus, 512 words would be required for the complete image. In the worst case, that is, when there are no consecutive equal gray levels and each pixel must be encoded separately, 512 by 512 by 2 bytes = 128K words are needed. This is twice the amount of storage needed by full-raster encoding.

Quadtree

A quadtree is a tree data structure in which each node is a leaf node or an interior node with four children. Quadtrees have been used to represent and manipulate binary images (ref. 1). The leaf nodes of the quadtree correspond to some identifiable feature in the image which must be saved for later retrieval and further processing. The features are identified by means of a recursive subdivision process whose intermediate incomplete stages correspond to the interior nodes of the tree.

The quadtree encoding scheme used here recursively subdivides a 2^n by 2^n pixel image into four square quadrants. The length of the sides of the quadrants is halved at each subdivision step. If the gray level within an entire quadrant is constant, information to redraw the quadrant is stored in a leaf node of the encoding tree. If the gray level within the quadrant is not constant, the quadrant is subdivided further. The subdivision process may proceed down to the level of individual pixels. For a standard image, this occurs at the ninth subdivision. After the nth subdivision, the length of each side of a quadrant is $2^{(9-n)}$. The potential advantage of quadtree encoding is that an entire quadrant may be encoded in a manner which uses less space than is required by full-raster or runlength encoding.

This recursive subdivision process and the resulting quadtree are illustrated in figure 1 for a simple 8 by 8 black and white image. After the first subdivision, two 4 by 4 quadrants, labeled a and p, with uniform gray levels and two nonuniform quadrants, "northeast" and "southwest," have been identified. For quadrants a and p, the information to reconstruct them is encoded in two leaf nodes and the subdivision process terminated. The northeast and southwest quadrants are nonuniform and must be subdivided further. When the northeast quadrant is subdivided, four new 2 by 2 uniform quadrants, labeled b, c, d, and e, are found. Adding this information to the quadtree produces one new interior node and four new leaf nodes. Finally, subdividing the southwest quadrant locates two uniform quadrants, k and j, and two nonuniform quadrants which must be subdivided once more. This final subdivision produces quadrants of length unity, that is, individual pixels. The gray levels of the pixels are encoded and the subdivision terminated. The southwest quadrant adds 3 interior nodes and 10 leaf nodes to the quadtree. The final tree, shown at the bottom of figure 1, has 4 layers, 5 interior nodes, and 16 leaf nodes. The 5 interior nodes represent part of the overhead of using a tree data structure. The leaf nodes contain sufficient information to redraw the quadrants. The tree is a complete representation of the original image.

Quadtree encoding compresses an image by removing redundancy over areas. If an image can be decomposed into a "few" quadrants of equal gray level, the savings over other encoding methods may be substantial in spite of the memory overhead needed to maintain the tree data structure. However, a quadtree representation is based on a purely geometric subdivision process which may not take advantage of image content. Hence, a sizable tree may be required to encode a uniform area which is poorly or inappropriately placed vis-à-vis the subdivision grid. One purpose of this paper is to obtain a quantitative indication of the trade-offs between the two counteracting processes of data compression versus tree memory overhead.

Recursive encoding/decoding algorithm. The main issue in this paper is image storage. Since no consideration is given to additional processing or manipulation, only enough data need to be encoded to permit redrawing the image. Thus, pointers need not be saved; this reduces the overhead of the tree structure somewhat. Redrawing the picture requires the ability to redraw each encoded quadrant, which means being able to place the quadrant in the image and to determine its size and gray level. Thus, (x, y) coordinates of the lower left corner of the quadrant, the length (L) of one side, and a gray level in the range of 0–255 must be available. Because of the systematic geometric fashion in which the quadtree algorithm subdivides the image, it is possible to reconstruct the coordinates and size of a quadrant implicitly from the depth of a node and its relationship to its siblings. Thus, x, y, and L need not be encoded; this results in a sizable reduction in storage. In addition to encoding the gray level of a quadrant, all that is

required is a single bit to indicate whether a node is an interior node which must be further subdivided before decoding or is a leaf node for which a gray level is saved. In the current algorithm, each interior node of the tree is flagged with a 1 to indicate that its gray level is not uniform and that it must be further subdivided before encoding. Each leaf node is flagged with a 0 to indicate it has a uniform gray level and is to be coded without further subdivision. A gray level, one 8-bit byte, must be coded for each leaf node. Introduce the term FLAG to indicate a word consisting of 32 1-bit node flags, 0's and 1's, and LEVEL to indicate a word consisting of 4 1-byte gray levels. These items are encoded separately.

Pseudocode for the quadtree encoding and decoding schemes is shown in figure 2. Encoding begins with an entire image whose side length, L, is 512 pixels and whose lower left-hand corner is located at (0,0). The procedure EQUAL (quad) determines whether the current quadrant, the whole image at this stage, has a uniform gray level. If so, encode one FLAG bit, 0, and one LEVEL byte, Gray-Level, the current gray level, and terminate. If not, encode one FLAG bit, 1, and no gray level and make the first call to the procedure QUAD (x, y, L) which begins the recursive subdivision and encoding procedure.

A call to procedure QUAD (x, y, L) implies that the current quadrant has a nonuniform gray level and must be further subdivided. The four new quadrants, designated as northwest to southwest, are examined in turn with procedure EQUAL to determine if their gray level is uniform. If so, a FLAG bit and gray level are packed; if not, another call is made to procedure QUAD. At each recursive call, four new nodes are added to the tree, representing the four new quadrants. A return from QUAD (x, y, L) indicates that four sibling nodes have been coded and it is time to ascend one level in the tree.

The encoded form of the binary image in figure 1 is shown in figure 3. Each node is marked with its appropriate node flag to indicate whether it is uniform, 0, or not, 1. The gray level indices of black and white are designated by B and W, respectively, rather than by numbers for clarity. One FLAG containing a sequence of 21 bits is required to contain the node flags. The letter designating the quadrant corresponding to each uniform quadrant is shown below the leaf nodes. Four LEVEL's, each containing a 4-byte integer representation of four gray levels, are needed to encode the image information. (For clarity, the notation only indicates the gray level in each of the 4 bytes of the LEVEL's and not the actual integer value of their packed form.) Only 5 words are required to store the image as a quadtree, whereas 16 words are required for full-raster encoding and 14 for runlength encoding.

Changing the encoding algorithm to decode the tree is straightforward as shown in figure 2. The procedure DECODE unpacks one FLAG bit, IND, and, if it is a 0, one gray level IVAL. The decoding algorithm systematically reconstructs x, y, and L. If the current node is a leaf node representing a quadrant with a uniform gray level, all information is now available to process it (redraw, construct histogram, or reconvert to full-raster format) in the procedure VISIT ( ). If the node is an interior node, that is, a quadrant whose gray level is nonuniform, it must be further subdivided.

Algorithm performance. The expected performance of the algorithm can be predicted theoretically.

A complete quadtree with n layers ($n = 0$ corresponding to the root node) contains $(4^{n+1} - 1)/3$ nodes of which 4^n nodes are leaf nodes. Thus, approximately, for nearly full trees with many layers,

$$\frac{\text{Total nodes}}{\text{Leaf nodes}} = \frac{4}{3}$$

Said differently, there is, on the average, one interior node for every three leaf nodes. This ratio also gives an upper bound on the fraction of interior nodes to leaf nodes. Each 32-bit FLAG will, on the average, consist of 8 1's indicating interior nodes and 24 0's indicating leaf nodes. For each leaf node there is a corresponding 1-byte gray level. Thus, for each FLAG there are $24/4 = 6$ LEVEL's. For every six LEVEL's, which contain the image information needed to describe the picture, there is one FLAG needed to code the quadtree. Thus, the overall storage efficiency of the algorithm is

$$\frac{\text{gray level storage}}{\text{Total storage}} = \frac{\text{LEVEL's}}{(\text{FLAG's} + \text{LEVEL's})}$$

$$= \frac{6}{7} = 86 \text{ percent}$$

These theoretical performance estimates were achieved for all images encoded subsequently.

The worst case performance occurs when all 256K pixels must be encoded. This encoding requires 64K LEVEL's and an additional 1/6 or slightly under 11K FLAG's for a total of about 75K words. This is a considerable improvement over the worst case performance of runlength encoding but somewhat worse than full-raster by the amount of overhead required to store the tree. The best case performance, achieved for an image with a constant gray level, requires two words: one FLAG and one LEVEL.

Example Standard Image

To conclude this discussion of encoding methods, consider how a multilevel image of standard size and ideal content would be encoded by each of the three schemes. Let the image consist of a 4 by 4 checkerboard of 16 equal-size squares, each square having 128 pixels on a side as shown in figure 4. Let each square have a uniform gray level, the numbers 0–15 in figure 4(a), different from all the others. Full-raster encoding requires 64K words, 1 byte per pixel, 4 pixels per word. Runlength encoding requires on each scan line a set of 2 bytes (length and gray level) for each square or 2 words per line. The entire image can be stored in 1024 words. The quadtree representation consists of a tree with 21 nodes on three layers: root node (entire image), 4 nodes on the second layer each representing one quadrant of the original image (each of these quadrants contains four of the checkerboard squares), and 16 nodes on the third layer, each node representing one of the checkerboard squares. One FLAG is required to encode the tree (16 0's and 5 1's) and 4 LEVEL's are required to encode the gray levels. (The notation for the levels only indicates which gray level is contained in each byte.) Thus, only 5 words are required to save the entire image, an appreciable reduction in storage. It is this potential for decreased storage by encoding uniform areas as a single byte that makes quadtree encoding potentially attractive as a compression and storage technique. If the 16 squares all have the same gray level, the storage requirements for full-raster, runlength, and quadtree encodings are 64K, 512, and 2 words, respectively.

Segmentation

The purpose of segmentation is to partition an image into identifiable regions or components (ref. 4). Segmentation is one method of feature identification and extraction of interest in automatic scene analysis and pattern recognition. The quadtree encoding scheme provides the framework into which a segmentation method can be easily incorporated. This section of the paper describes a segmentation method which makes use of the statistical variation of gray levels within the quadrants formed during the quadtree subdivision process.

The image is segmented by relaxing the requirement of identifying a "uniform" quadrant as one whose pixels all have exactly the same gray level and replacing it with a criterion which makes use of the statistical properties of the gray level distribution. The method employed calculates the mean μ and standard deviation σ of the gray levels of all the pixels in a quadrant. If σ/μ is less than or equal to a prescribed value, the quadrant is declared "uniform." All the pixels in the quadrant are assigned a gray level equal to the mean value and the quadrant is not subdivided further. The ratio σ/μ is called the "coefficient of variation." It expresses the magnitude of dispersion of a random variable relative to its expected value. In the present context, σ/μ measures the extent to which the gray levels in a quadrant are allowed to vary about their mean value.

With this criterion, the image is segmented or partitioned into square areas, called "segments," with the property that each has a constant gray level. This segmentation method adaptively low pass filters areas of the image corresponding to the systematic geometric regions into which it is subdivided by the quadtree encoding process. Thus, the ratio σ/μ acts as a parameter which controls the fidelity with which the segmented image represents the original image. Large values of σ/μ permit extensive averaging so that the segmented image retains only those coherent features of the original image which extend over considerable areas. Detail is removed from the original image, and the segmented image has a "blocky" appearance. As σ/μ is decreased, the permitted amount of variation of the gray levels from the mean is reduced and more detail from the original image is retained in the segmented image. The limiting case $\frac{\sigma}{\mu} = 0$ corresponds to requiring that all pixels within the current quadrant have exactly equal gray levels. A coarse representation of a scene may be useful (1) in pattern recognition and classification when it is desirable to compare the gross features of a scene with a pattern prior to comparing minute features (refs. 5 and 6) or (2) as a means of controlling the amount of storage allowed for an image, but with some sacrifice in image detail. Some examples will be shown subsequently of the appearance which this segmentation method induces on various types of images.

Code Validation

The software to implement the quadtree encoding scheme and segmentation method was validated in several ways. For very simple images, such as the one shown in figure 4, the pointers were constructed allowing the tree structure to be checked in full detail and the results of segmentation were easily predictable. Histograms obtained while decoding the quadtree format (with $\frac{\sigma}{\mu} = 0$) were checked against histograms obtained from the full-raster format. These always agreed completely. Numerous quadtree encoded images were redrawn and visually compared with the originals. In several instances, the

difference image between the encoded image (with $\frac{\sigma}{\mu} = 0$) and the original image was formed. The difference image should, of course, be zero. All these tests were passed.

Procedure

The three encoding schemes were applied to a series of images of increasing complexity and detail in order to generate data on relative storage requirements and show some results of using the segmentation method. The images were all standard images, that is, 512 by 512 pixels with up to 256 gray levels. All images were initially obtained or created in the full-raster format from which they were run-length or quadtree encoded. Some of the test images were generated from mathematical equations. Another is a representation of airflow in a supersonic nozzle calculated from a computational fluid dynamics code. The most complex image is a Landsat image of Washington, D.C.

The application code was executed on a time-shared minicomputer with virtual memory. The software was written in FORTRAN with recursion simulated by means of a data stack as described in chapter 11 of reference 7. The program was able to quadtree or runlength encode a standard image in the worst case.

Images were displayed interactively on a commercial color graphics workstation for program development and verification and on a film recorder for quality hard copy. The photographs in this report were obtained from this recorder. Each of these devices could display up to 256 gray levels.

A quadtree encoded image can be displayed on the workstation in two ways. The most natural method is to redraw each quadrant as a panel by using the graphics library subroutines. Each panel is drawn during decoding in the same order as it was encoded. There is some fascination in watching the recursive nature of the encoding being unraveled and revealed graphically in real time. An encoded image can also be converted to full-raster format and displayed as a raster of pixels, which is the method used to obtain the photographs in this report. This latter method was usually faster than drawing individual panels.

Test Images

In this section, a brief description is given of each of the test images. These images were chosen to represent a range of "complexity" and "detail," that is, number of gray levels and amount of coherence. The range of gray levels in each image was always linearly stretched to use the full 256 gray levels available on the display devices so that the extremes of "white" and "black" are always present and there is no darkening of the images due to gray level compression. These images are grouped together in related sets at the back of this report. Changes in brightness among these photographs are due in part to the developing, printing, and reproduction process. Hence, observed changes in brightness between different image sets or images in the same set may not be due entirely to segmentation or gray level compression.

Concentric Circles

Two patterns of concentric circles were generated mathematically: one set with the center in the middle of the image, several of these are shown in figure 5, and a second set with the center at the lower left-hand corner of the image. The annular region between consecutive circles has a uniform gray level. The number of gray levels was increased by decreasing the width of the rings. Figure 5 shows images having 10, 50, and 100 gray levels. As the number of gray levels increases, it becomes increasingly difficult to identify distinct rings in the images.

Sinewave Hills

Another mathematically generated family of images consists of a pair of hills and valleys whose elevation $Z(i,j)$ was obtained from the equation

$$Z(i,j) = \sin\left(\frac{2\pi i}{511}\right) \sin\left(\frac{2\pi j}{511}\right) \quad (0 \leq i,\ j \leq 511)$$

The lower left-hand corner of the image corresponds to $i = j = 0$. The gray level assigned to a point (i,j) is determined from the expression

$$\text{NINT}\,(127.5\,[Z(i,j) + 1])$$

Figure 6 shows three selected images from this set for 11, 61, and 121 gray levels. It was a matter of convenience to choose an odd number of gray levels, allowing an equal number of levels for the hills and valleys and one level with the average gray level for mean elevation $Z = 0$. As for the circles, the image appears more continuous in brightness as the number of gray levels increases.

Diffuser

Figure 7 shows a 256 gray level representation of an image showing the Mach number distribution in a supersonic diffuser. This image is a representative example of displays of calculations from computational fluid dynamics codes. Four versions of this image, each having a different number of gray levels,

were produced by compressing the gray level range by factors of 1/2, 1/4, and 1/8 as the full-raster file was read into the application program. The reduced number of gray levels was always spaced out over the full 256 levels available. Consequently, there is no darkening of the images due to gray level compression. As the number of gray levels is reduced, distinct bands of constant gray level appear in the photographs. These bands, called "quantization bands," frequently appear in coarsely quantized images.

Washington, D.C.

The most complex image used is a Landsat image showing southwest Washington, D.C., some of the northern Virginia suburbs (south to the Wilson Bridge and Alexandria), and portions of the Potomac River. (See fig. 8.) The original image contained 96 gray levels. Several versions of this image were also produced by compressing the gray level range by factors of 1/2, 1/4, and 1/8. Many features with linear coherence (not necessarily along horizontal scanlines) can be identified: highways, bridges, roads, and runways. The Potomac River and several lakes and man-made basins exhibit appreciable area coherence. When compressed to only 12 gray levels, the image takes on a grainy appearance. This loss of detail from the image is due to the large increments of intensity between adjacent gray level indices resulting from spacing out the 12 indices over 256 levels. At 24 gray levels and above, little degradation in the quality of the image is visible to the unaided eye.

Results

Tables 1 to 4 list the storage requirements for encoding the test images. The number of segments is the number of squares with uniform gray levels into which an image is decomposed; this is also the number of leaf nodes on the quadtree. The number of words is the total number of 4-byte words required to store the image, that is, the sum of the number of FLAG's and LEVEL's required. For the quadtree scheme, further details on storage can be closely approximated by using the theoretical performance estimates developed previously; that is,

$$\text{Total number of nodes in tree} \approx \tfrac{4}{3} \times \text{Number of segments}$$

$$\text{Number of FLAG's} \approx \tfrac{1}{7} \times \text{Number of words}$$

$$\text{Number of LEVEL's} \approx \tfrac{6}{7} \times \text{Number of words} \approx \tfrac{1}{4} \times \text{Number of segments}$$

By comparison, the storage requirement for full-raster encoding is 64K words regardless of image content or number of gray levels. Examples of segmented images are shown for the diffuser and Washington, D.C., as these are more complex and realistic images than the circles and hills.

Concentric Circles and Sinewave Hills

Both the circles and hills images have been quadtree encoded by using the criterion of exact equality of all pixels in a square. The results are shown in tables 1 and 2. The number of segments into which these images are decomposed quickly reaches 50 000 for a moderate number of gray levels and then rises rapidly as the number of gray levels increases. At 250 gray levels, the complete quadtree for concentric circles with center at the image center contains about 320 000 nodes, emphasizing that very large trees can result from multilevel images. Runlength encoding always uses less storage than the full-raster format. It is seen that when an image contains many gray levels, quadtree encoding may exceed 64K words needed to store the gray level of each pixel. Runlength encoding always uses the least storage, from 50 to 80 percent of that used by the quadtree. The differences are large enough to strongly favor runlength encoding for these images.

Diffuser

The diffuser and Washington, D.C., images are more representative of computer graphics and satellite images encountered in practice than the highly structured, mathematically generated circles and hills images. The segmentation method was applied to these more realistic images. The results are described in the following discussion.

Results of encoding the diffuser image are presented in table 3. Photographs of the segmented image for selected values of σ/μ are shown in figures 9 and 10 for 256 and 64 gray levels, respectively.

The diffuser was quadtree encoded without loss of information, $\frac{\sigma}{\mu} = 0.0$, in 15 677 to 4365 words, depending upon the number of gray levels in the original image. This is appreciably less storage than is required by full-raster encoding. The number of segments generated ranged from 53 746 to 14 962. Runlength encoding of this image for $\frac{\sigma}{\mu} = 0.0$ was always more efficient than quadtree encoding. Large values of σ/μ result in large areas being declared uniform and averaged. The storage requirements decrease as σ/μ increases; however, increasing amounts of detail are removed.

Also shown in table 3 (as the "runlength" results for $\frac{\sigma}{\mu} > 0$) are the results of runlength encoding the segmented image. For the larger values of σ/μ, that is, for images containing large uniform

segments, quadtree encoding is more efficient than runlength encoding. However, as σ/μ decreases, a crossover value occurs below which runlength encoding becomes more efficient.

As seen in figures 9 and 10, the horizontal centerline of these images happens to coincide with the midline of the diffuser so that the segmentation is symmetric above and below this line. The quadtree-based segmentation restructures the image into squares with uniform gray levels. As σ/μ decreases, the spread of gray levels considered "equal" is tightened about the mean value and the number of segments increases. For $\frac{\sigma}{\mu} = 0.005$ and 256 gray levels, the "block" structure of the segmented image is discernible only upon close inspection.

For comparison, figure 11 shows several of the photographs from figures 9 and 10 in pseudocolor. Each of the gray levels has been assigned a color taken from the "rainbow" scale shown at the top of each photograph. The correspondence between black and white in the gray shade and pseudocolor images has been preserved. The block nature of the segmentation is more evident in these color photographs than in the previous black and white photographs. This is to be expected since the eye is known to be more sensitive to gradations of color than to shades of gray.

Washington, D.C. Table 4 summarizes the results of segmenting and encoding the image of Washington, D.C. Two series of quadtree-encoded black and white images are shown in figures 12 and 13 for 96 and 24 gray levels, respectively. Pseudocolor photographs of a few of these images are given in figure 14. Runlength results for $\frac{\sigma}{\mu} = 0.0$ are for encoding the original image, whereas for $\frac{\sigma}{\mu} > 0.0$, they are for encoding the segmented image.

It was not possible to quadtree encode Washington, D.C., with $\frac{\sigma}{\mu} = 0.0$ using less storage than full-raster encoding. At about $\frac{\sigma}{\mu} = 0.05$, more than 64K words are needed. However, recall that quadtree encoding a standard image cannot exceed an upper limit of about 75K words. As σ/μ was decreased, the number of segments increased. The images with 96 and 48 gray levels approached the theoretical predicted maximum storage. At $\frac{\sigma}{\mu} = 0.0$, runlength encoding required more storage than full-raster at 24, 48, or 96 gray levels. For $\frac{\sigma}{\mu} > 0.0$, runlength encoding the segmented images requires more storage than the quadtree.

For $\frac{\sigma}{\mu} > 0.10$, the segmented images show large areas of uniform gray level. This is adequate to preserve large homogeneous regions such as the Potomac River, several lakes and basins, the mall, and the Ellipse and some of the more evident linear features such as highways and runways at National Airport. Fine detail is completely eliminated. At $\frac{\sigma}{\mu} = 0.10$, this detail begins to emerge in the photographs. Although 100 000 segments or more are generated, numerous blocks of uniform gray levels are still easily identified. Close to 200 000 segments are required to attain an image which to the unaided eye has a high degree of resemblance to the original image without the effects of segmentation being obvious. This number of segments requires nearly 64K words of storage and represents only a small saving over full-raster encoding but is still better than runlength encoding. An interesting trend is evident from table 4. For fixed σ/μ, more segments and storage are required as the number of gray levels decreases.

Concluding Remarks

Results have been presented of image encoding by using full-raster, runlength, and quadtree encoding schemes. The runlength method was a constant length code requiring 1 byte for the number of pixels and 1 byte for the gray level. The quadtree method, which recursively subdivides an image into quadrants of uniform gray level, incorporated a parametric segmentation method which identified uniform quadrants on the basis of the statistical distribution of gray levels. Tables are presented showing the storage requirements for the three encoding schemes when applied to a series of multilevel images containing increasingly complex scenes. Specific examples of black and white and pseudocolor segmented images are given.

A recursive method for encoding and decoding the quadtree has been described. The position and size of a uniform quadrant are systematically constructed during the course of coding and need not be saved; thus, the amount of storage required to save the quadtree representation of an image is reduced. The resulting encoding scheme has only 14 percent overhead and can store a 512 by 512 image with 256 gray levels in a maximum of 75K words.

Quadtrees containing 100 000 to 200 000 segments were generated for moderately complex images having 100–200 gray levels. The amount of storage required is highly dependent upon the individual scene. Both runlength and quadtree encoding generally required less storage than full-raster encoding. The worst case performance of quadtree encoding is considerably better than that of runlength encoding and hence would be preferred for highly complex scenes.

NASA Langley Research Center
Hampton, Virginia 23665-5225
July 22, 1987

References

1. Samet, Hanan: The Quadtree and Related Hierarchical Data Structures. *ACM Comput. Surv.*, vol. 16, no. 2, June 1984, pp. 187–260.
2. Lauzon, Jean Paul; Mark, David M.; Kikuchi, Lawrence; and Guevara, J. Armando: Two-Dimensional Run-Encoding for Quadtree Representation. *Comput. Vis., Graph., and Image Process.*, vol. 30, no. 1, Apr. 1985, pp. 56–69.
3. Gargantini, Irene: An Effective Way To Represent Quadtrees. *Commun. ACM,* vol. 25, no. 12, Dec. 1982, pp. 905–910.
4. Gonzalez, Rafael C.; and Wintz, Paul: *Digital Image Processing.* Addison-Wesley Publ. Co., 1977.
5. Hall, Gene: Large-Kernel Convolutions in Image Processing. *Digit. Design,* vol. 16, no. 9, Aug. 1986, pp. 46–48.
6. Williams, Lance: Pyramidal Parametrics. *Comput. Graph.,* vol. 17, no. 3, July 1983, pp. 1–11.
7. Wagener, Jerrold L.: *FORTRAN 77—Principles of Programming.* John Wiley & Sons, Inc., c.1980.

Table 1. Storage Space for Encoding Concentric Circle Images

[Full-raster encoding requires 64K words]

Number of gray levels	Centers at image center			Centers at lower left corner		
	Quadtree		Runlength	Quadtree		Runlength
	Number of segments	Number of words	Number of words	Number of segments	Number of words	Number of words
20	32 896	9 595	4 441	17 062	4 977	2 481
40	63 460	18 510	9 129	34 123	9 953	4 836
60	88 936	25 940	13 813	49 729	14 506	7 170
80	113 476	33 098	18 480	64 516	18 818	9 508
100	134 008	39 086	23 134	77 221	22 524	11 844
120	149 152	43 503	27 818	89 518	26 110	14 183
140	163 024	47 549	32 487	102 469	29 888	16 524
160	176 788	51 564	37 167			
180	190 840	55 662	41 829	125 728	36 671	21 209
200	204 808	59 736	46 509			
210	212 116	61 868				
220	218 668	63 779	51 181	142 234	41 486	25 898
240	233 560	68 122	55 857	149 608	43 636	28 236
250	240 700	70 205	58 202			

Table 2. Storage Space for Encoding Sinewave Images

[Full-raster encoding requires 64K words]

Number of gray levels	Quadtree		Runlength
	Number of segments	Number of words	Number of words
21	47 908	13 974	6 539
41	85 618	24 973	13 035
61	117 340	34 225	19 535
81	137 638	40 145	26 035
101	161 311	47 050	32 549
121	183 124	53 412	39 049
141	203 590	59 381	45 557
151	212 131	61 872	48 793
161	221 980	64 745	52 053
181	235 084	68 567	58 135
201	241 174	70 343	63 228

Table 3. Storage Space for Encoding Supersonic Diffuser Images

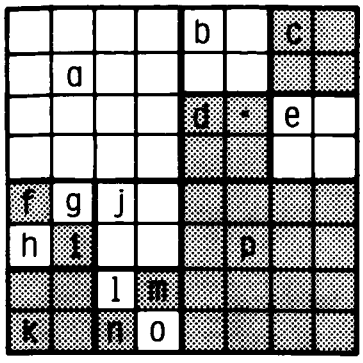
[Full-raster encoding requires 64K words]

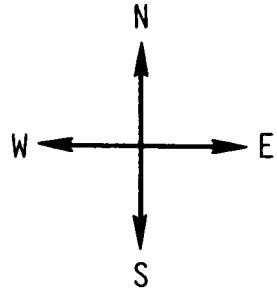
σ/μ	Quadtree		Runlength	Quadtree		Runlength
	Number of segments	Number of words	Number of words	Number of segments	Number of words	Number of words
	256 Gray Levels			64 Gray Levels		
0.100	4 483	1 308	3 125	4 258	1 243	2 314
0.050	4 987	1 455	3 586	4 741	1 384	2 630
0.020	6 154	1 796	4 814	6 361	1 857	3 565
0.010	8 224	2 399	6 978	10 732	3 131	4 197
0.005	14 071	4 105	10 580	21 787	6 355	4 197
0.0	53 746	15 677	13 818	24 181	7 054	4 197
	128 Gray Levels			32 Gray Levels		
0.100	4 462	1 302	2 777	4 249	1 241	1 787
0.050	4 867	1 420	3 197	4 744	1 384	2 024
0.020	6 088	1 776	4 383	6 949	2 028	2 394
0.010	8 920	2 602	6 315	12 844	3 747	2 405
0.005	18 691	5 452	7 768	14 656	4 275	2 406
0.0	38 050	11 099	7 785	14 962	4 365	2 406

Table 4. Storage Space for Encoding Washington, D.C., Image

[Full-raster encoding requires 64K words]

σ/μ	Quadtree		Runlength	Quadtree		Runlength
	Number of segments	Number of words	Number of words	Number of segments	Number of words	Number of words
	96 Gray Levels			24 Gray Levels		
0.20	16 297	4 755	14 821	21 499	6 271	14 951
0.15	49 360	14 397	38 014	63 544	18 534	36 426
0.10	119 467	34 845	71 766	140 398	40 950	64 343
0.08	157 462	45 927	86 198	174 244	50 822	73 799
0.06	195 208	56 936	98 865	204 187	59 555	81 386
0.05	213 571	62 292	104 626	219 856	64 125	
0.0	259 483	75 683	116 237	244 399	71 284	88 359
	48 Gray Levels			12 Gray Levels		
0.20	17 332	5 056	14 486	33 193	9 683	17 194
0.15	52 342	15 267	36 992	80 359	23 439	35 055
0.10	123 433	36 003	68 475	171 379	49 986	55 339
0.08	162 742	47 462	81 776	203 896	59 470	60 272
0.06	200 707	58 540	92 870	212 218	61 898	61 411
0.05	216 700	63 205	97 260	213 673	62 323	
0.0	251 587	73 380	105 236	214 312	62 508	61 704





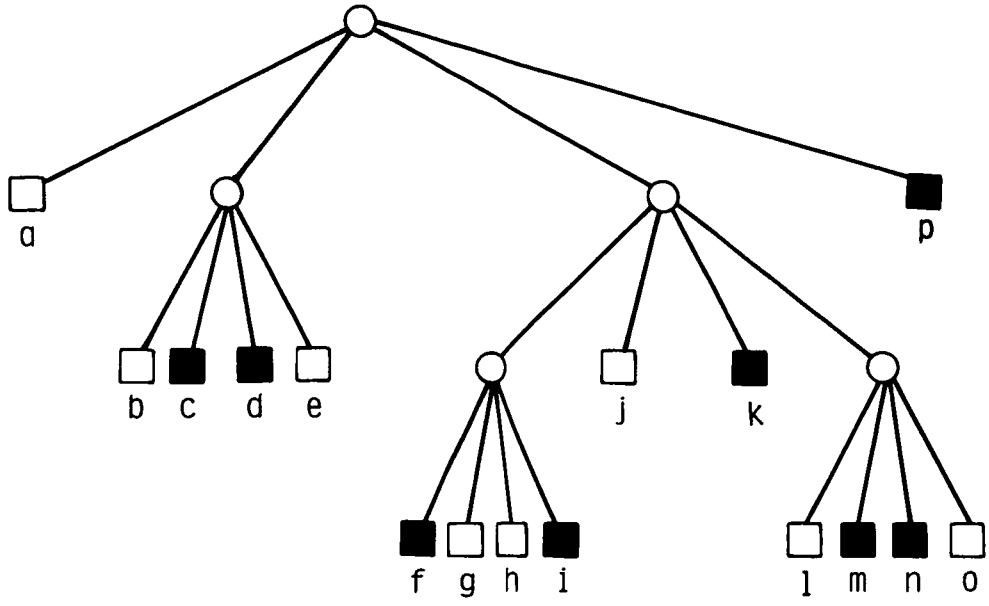


Figure 1. Simple binary image and its quadtree representation.

```
********** BEGIN QUADTREE ENCODING **********

if (EQUAL [image]) then ENCODE (0,Gray-Level)          **check entire image
else ENCODE (1), QUAD (0,0,512)

********** QUADTREE ENCODING COMPLETE **********

procedure QUAD (x,y,L)
if (EQUAL [NW quad]) then ENCODE (0,Gray-Level)        **check NW quadrant
else ENCODE (1), QUAD (x,y+L/2, L/2)

if (EQUAL [NE quad]) then ENCODE (0,Gray-Level)        **check NE quadrant
else ENCODE (1), QUAD (x+L/2, y+L/2, L/2)

if (EQUAL [SW quad]) then ENCODE (0,Gray-Level)        **check SW quadrant
else ENCODE (1), QUAD (x,y,L/2)

if (EQUAL [SE quad]) then ENCODE (0,Gray-Level)        **check SE quadrant
else ENCODE (1), QUAD (x+L/2,y,L/2)

return
```

****

To convert this encoding algorithm to the decoding algorithm, replace the five if-then-else statements with

```
DECODE (IND,IVAL)
if (IND = 0) then                                      **quadrant is uniform
    Gray-Level = IVAL, VISIT (quad)
else                                                   **quadrant is nonuniform
    QUAD (-,-,L/2)
```

in which the dashes are to be replaced with the appropriate values of x and y.

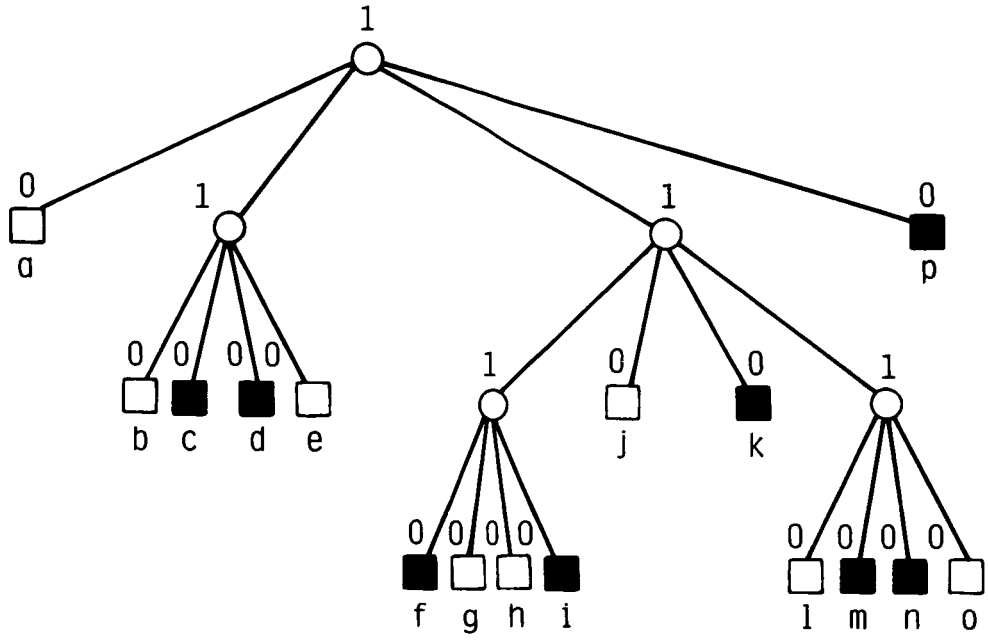
****

OTHER PROCEDURES USED

EQUAL (quad)	determine if current quadrant has a uniform gray level
ENCODE (0-1,Gray-Level)	pack bit to mark node type and gray level if node is uniform
DECODE (IND,IVAL)	unpack bit IND indicating node type and gray level IVAL if uniform
VISIT (quad)	process data for quadrant; i.e., display quadrant or calculate its histogram

****

Figure 2. Quadtree encoding/decoding algorithm.



One FLAG containing the bit sequence :

```
1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0
  a   b c d e     f g h i j k   l m n o p
```

Four LEVELS, each containing four gray levels :

W = white, B = black

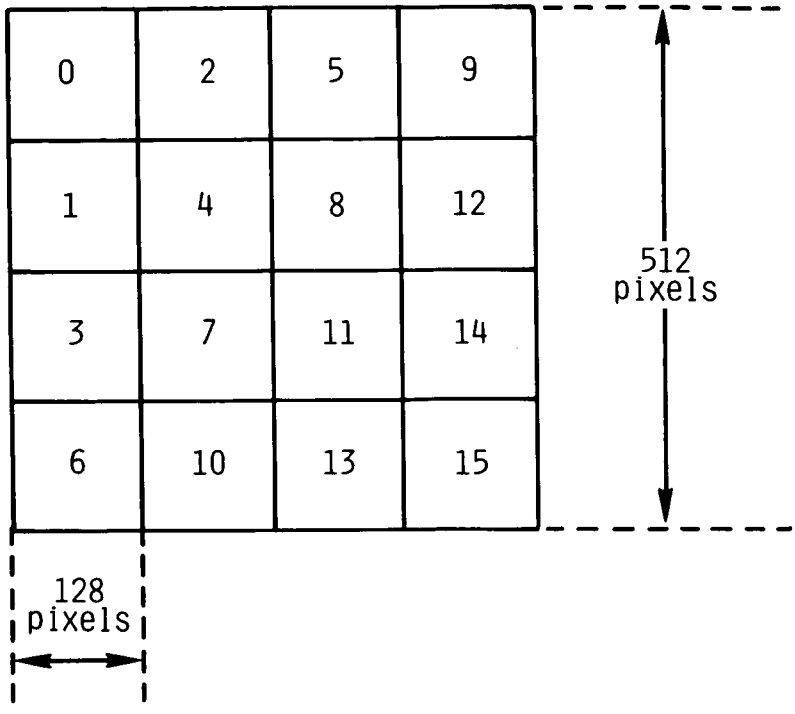
(W,W,B,B) corresponding to quadrants a,b,c,d

(W,B,W,W) corresponding to quadrants e,f,g,h

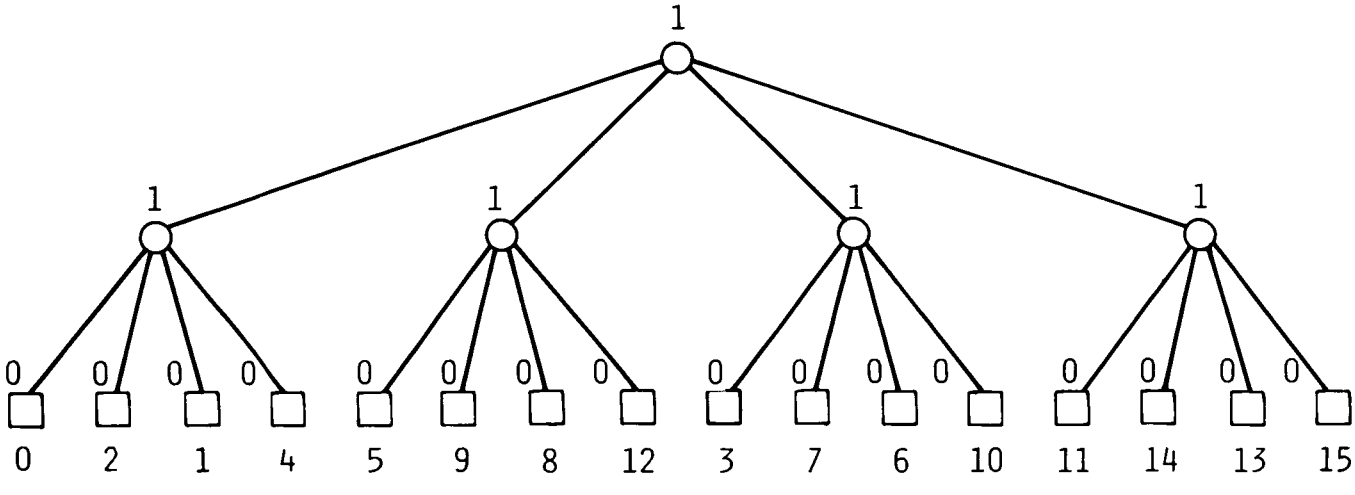
(B,W,B,W) corresponding to quadrants i,j,k,l

(B,B,W,B) corresponding to quadrants m,n,o,p

Figure 3. Encoded form of quadtree for image in figure 1.



(a) 4 by 4 checkerboard pattern.

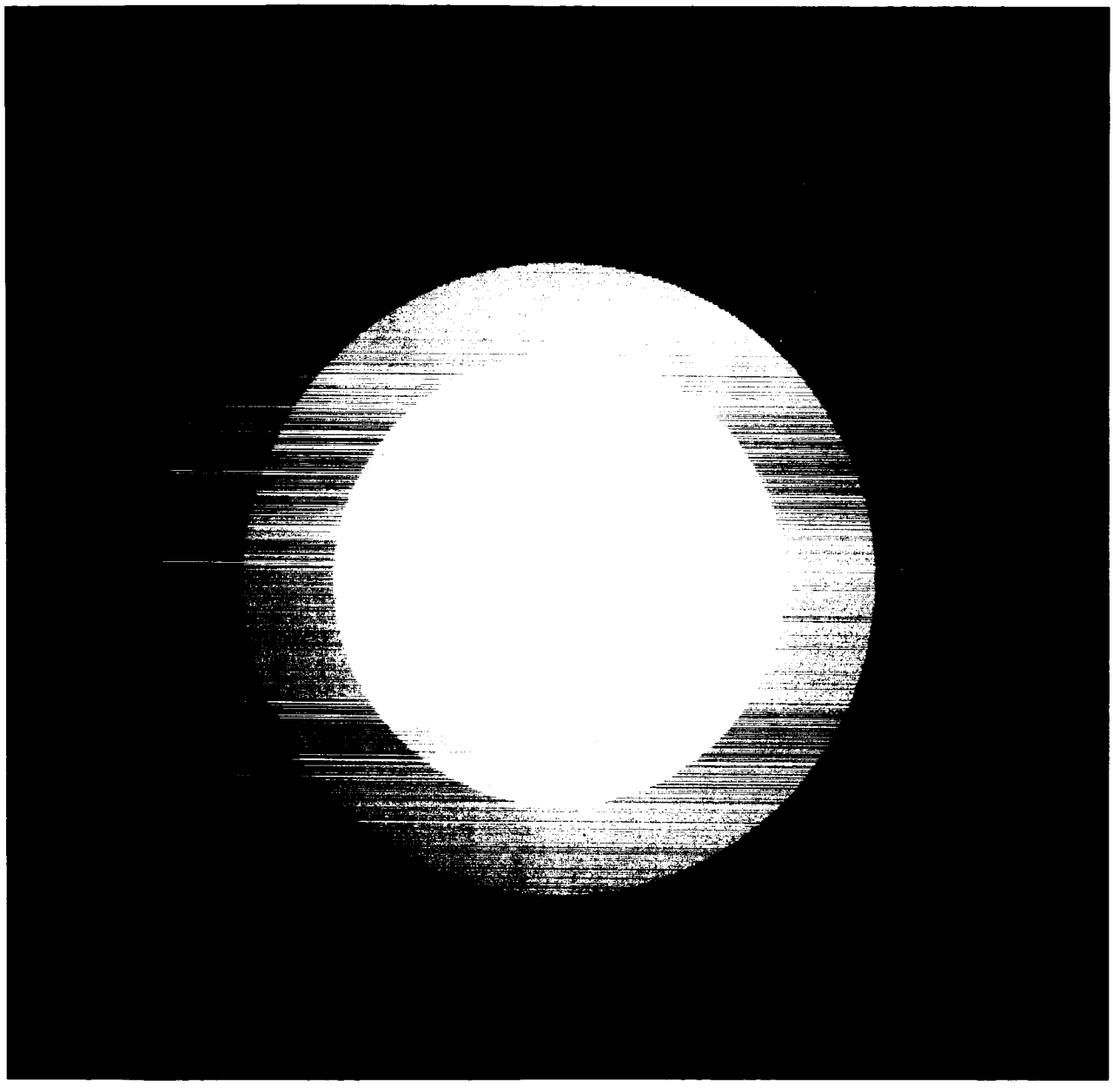


(b) Quadtree representation.

```
One flag : 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0
Four levels : (0,2,1,4) (5,9,8,12) (3,7,6,10) (11,14,13,15)
```

(c) Encoded form.

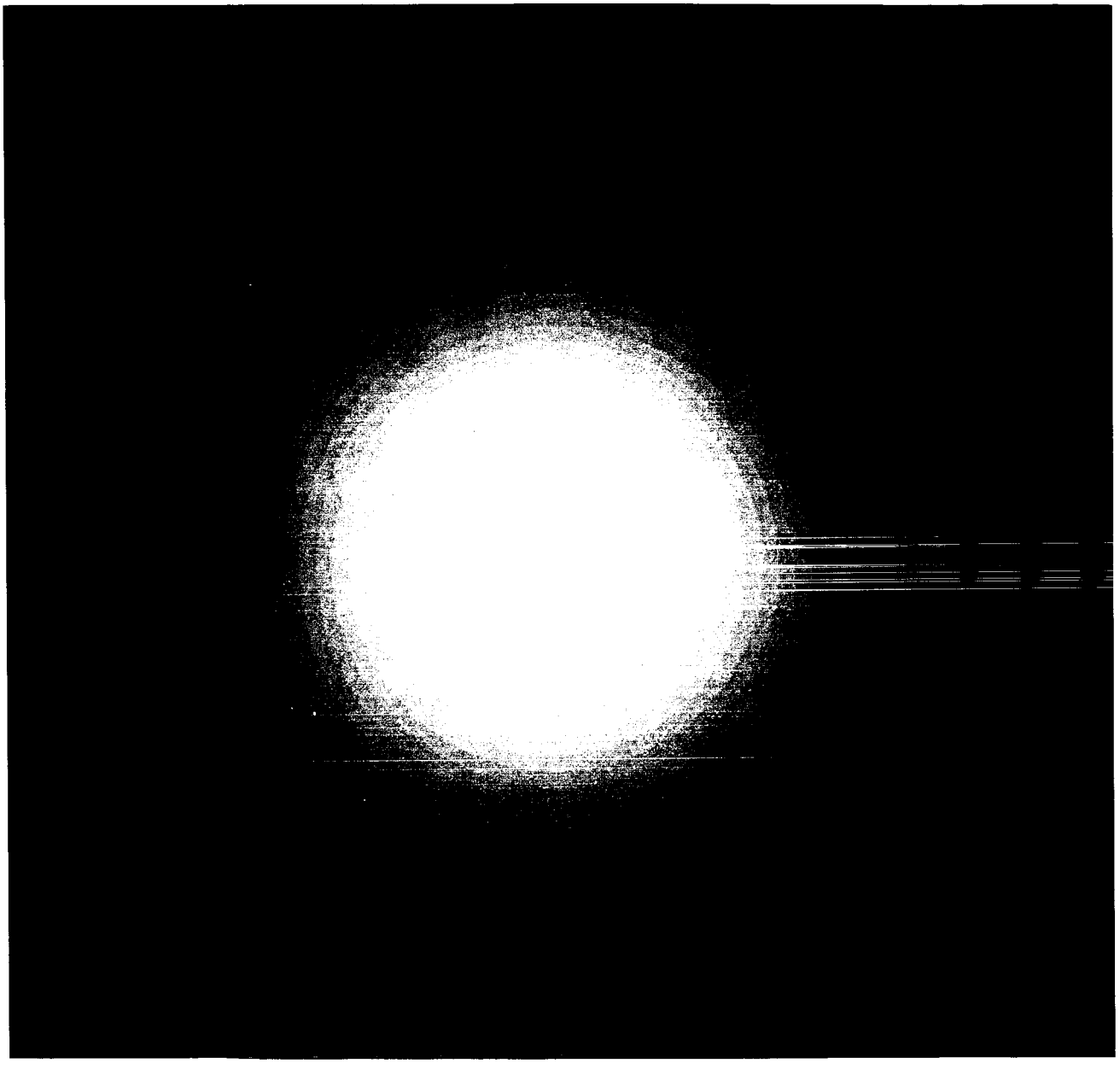
Figure 4. Example standard image.



L-86-4952

(a) 10 gray levels.

Figure 5. Test images of concentric circles.

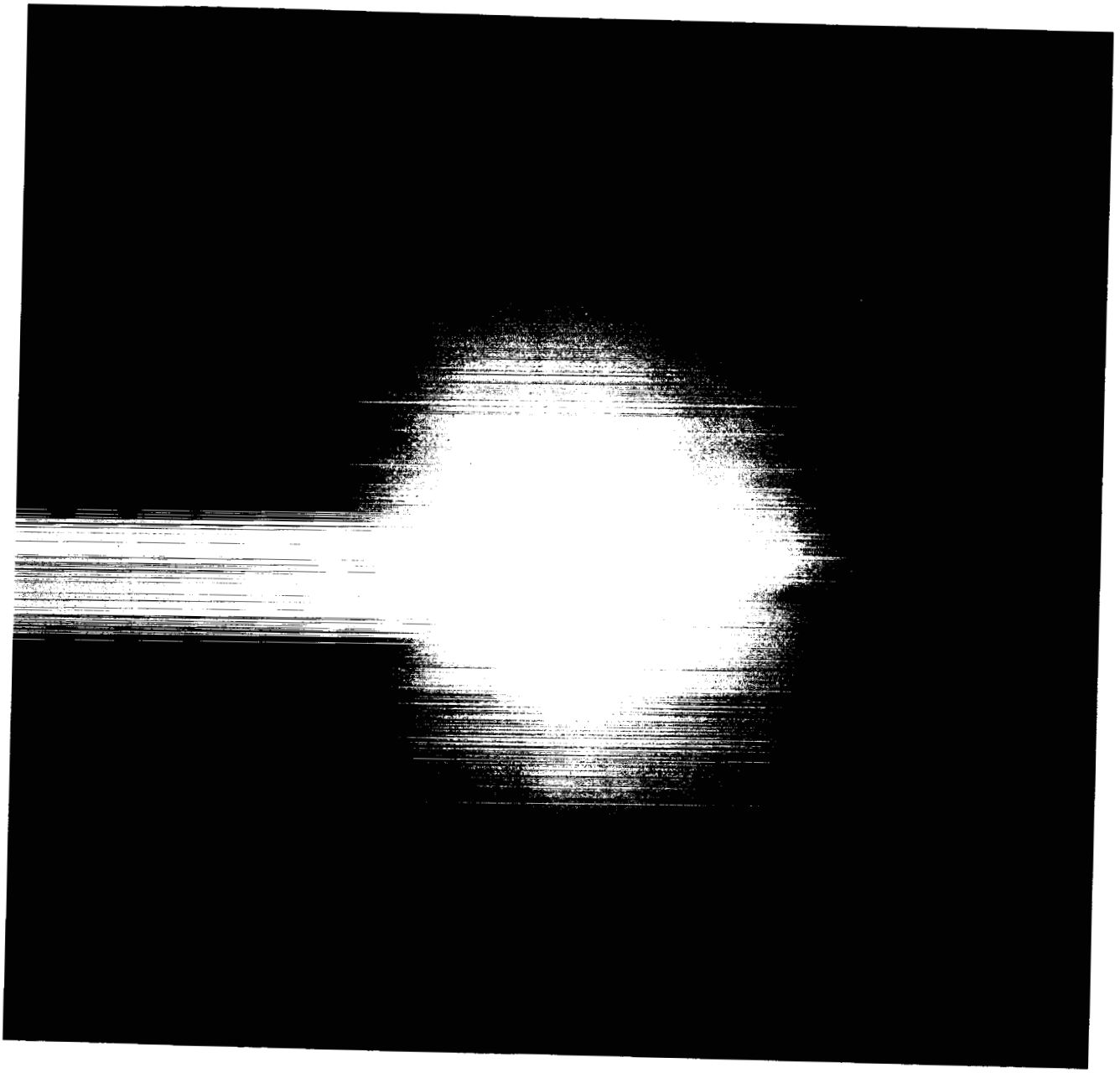


L-86-4949

(b) 50 gray levels.

Figure 5. Continued.

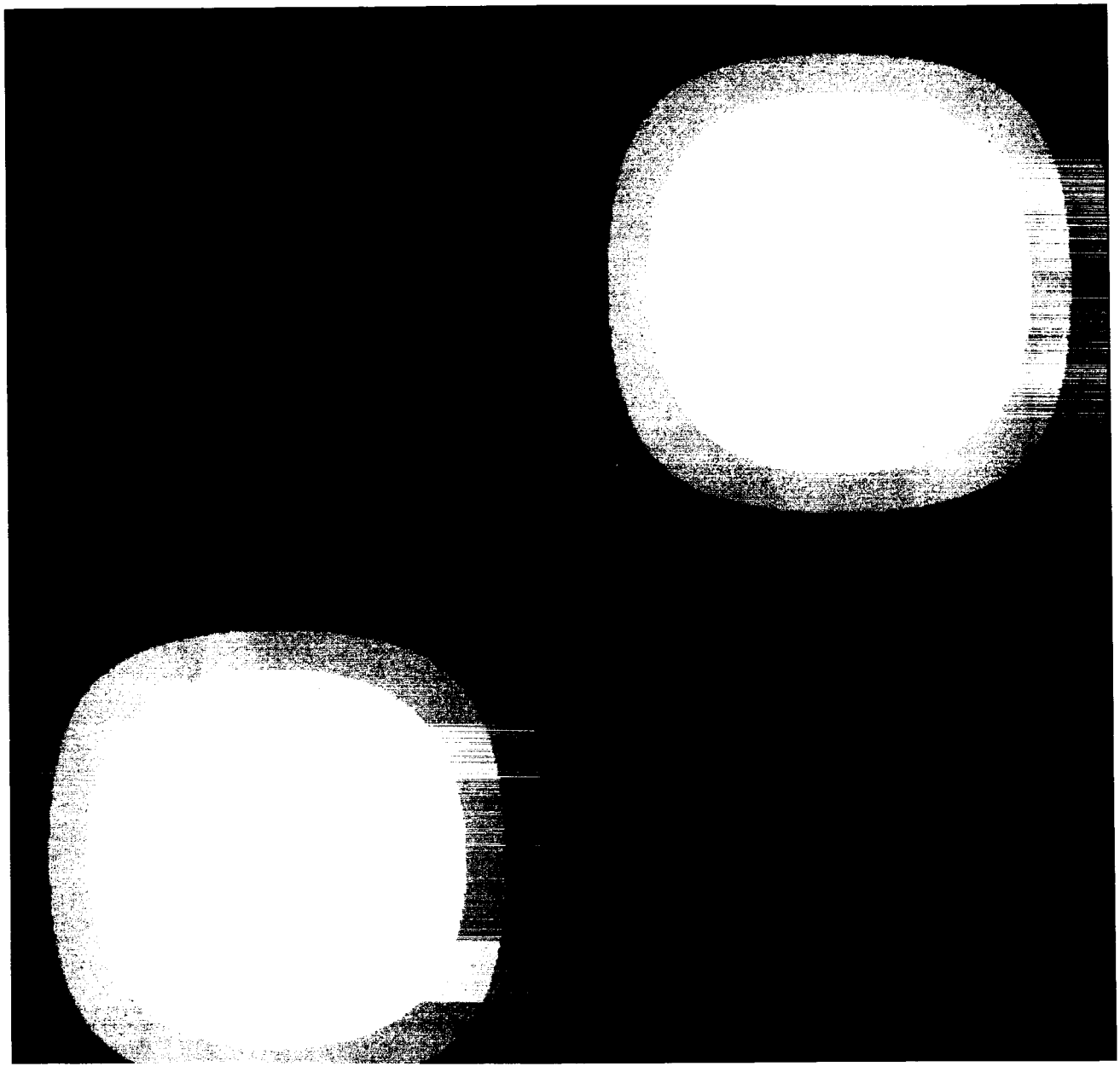
ORIGINAL PAGE IS



L-86-4950

(c) 100 gray levels.

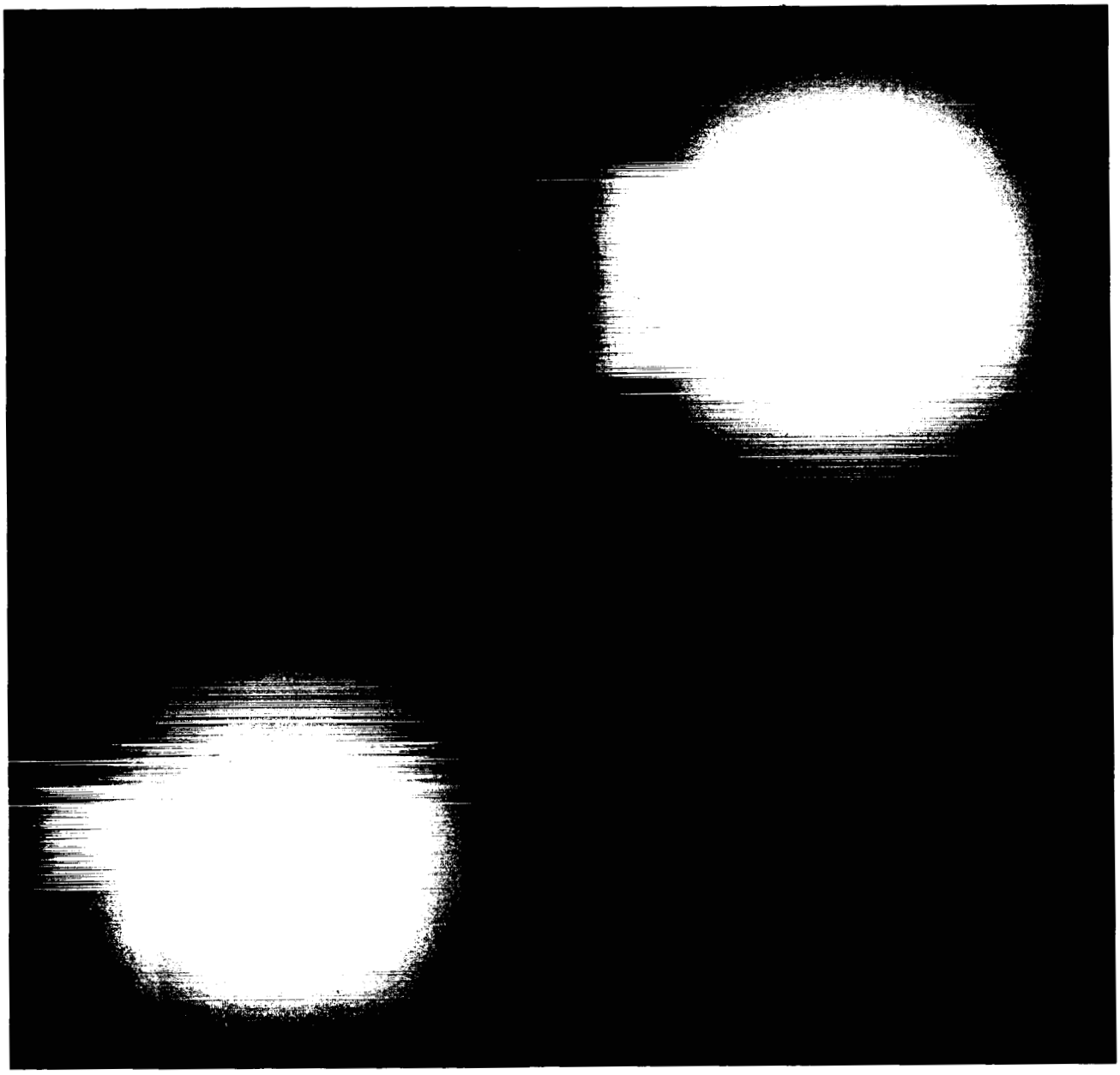
Figure 5. Concluded.



L-87-5667

(a) 11 gray levels.

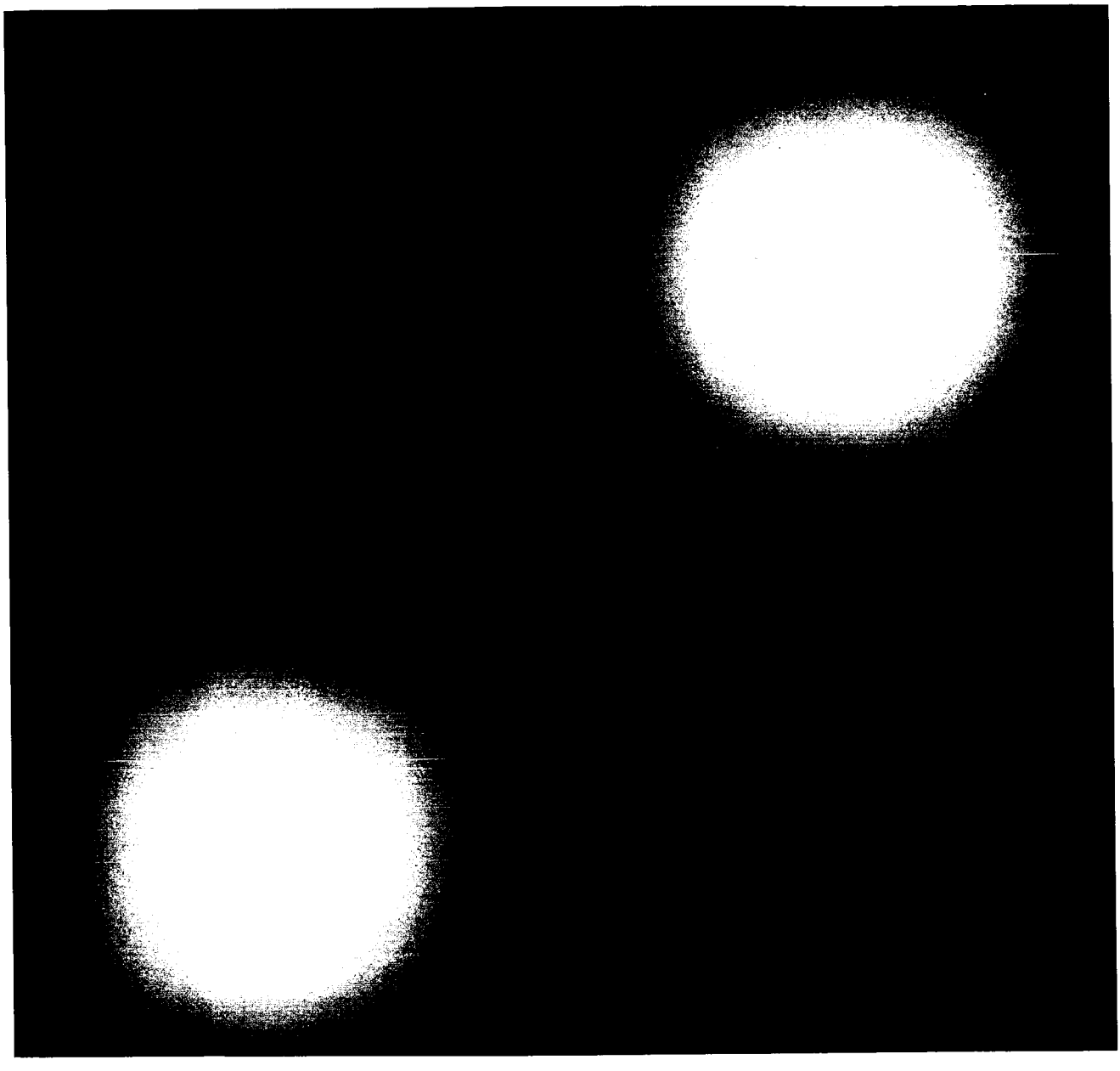
Figure 6. Test images of sinewave hills.



L-87-5668

(b) 61 gray levels.

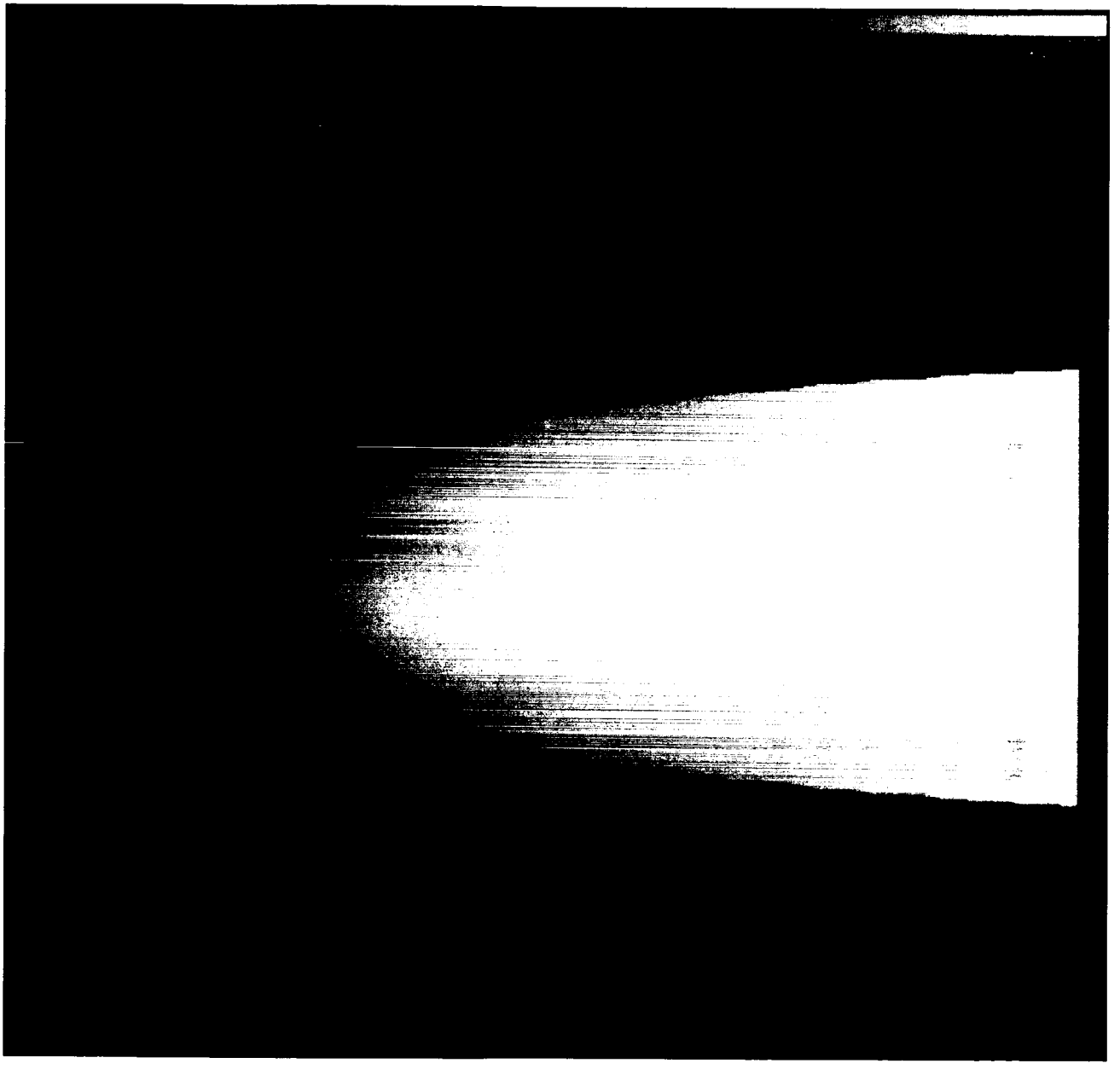
Figure 6. Continued.



L-87-5669

(c) 121 gray levels.

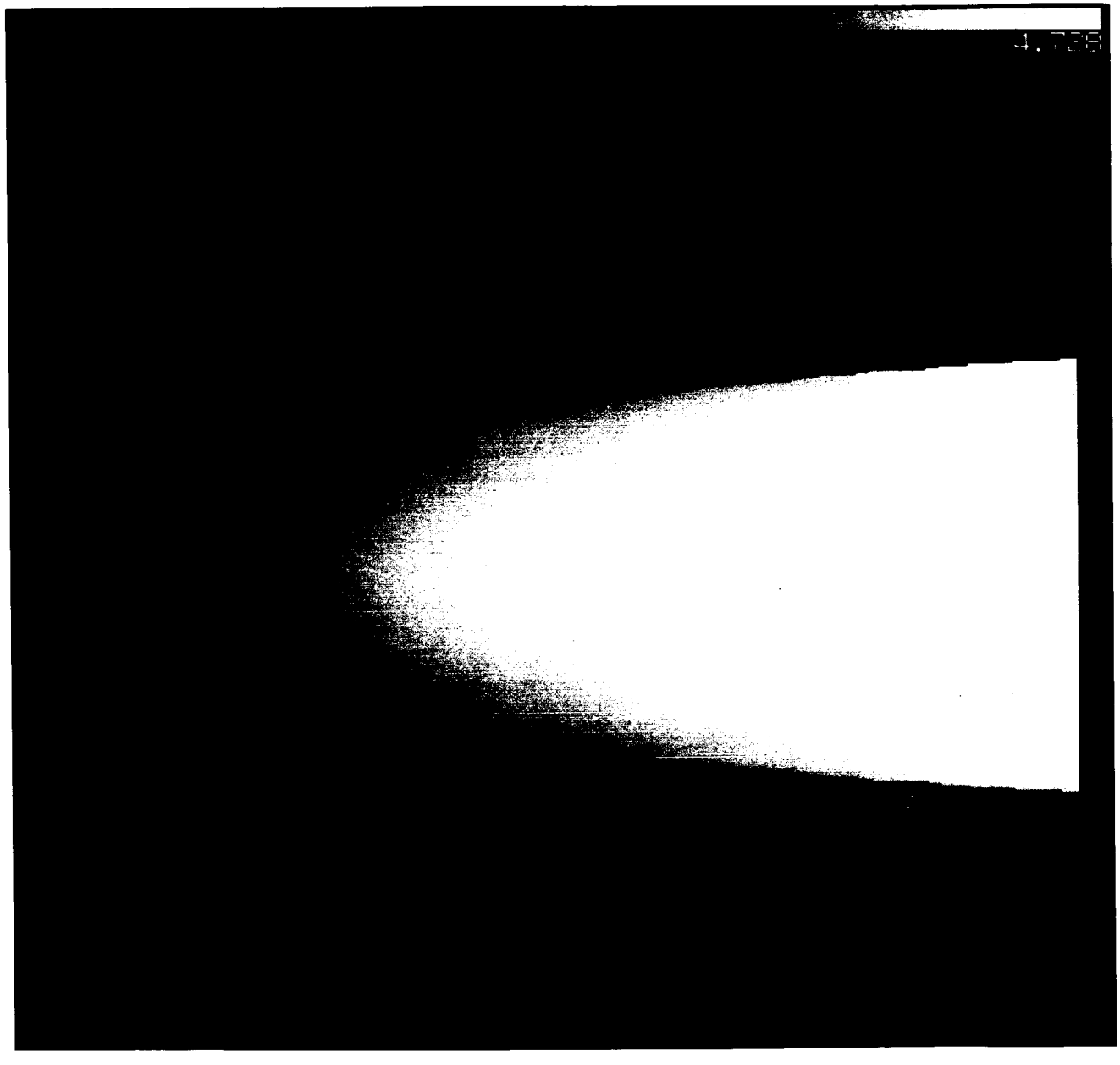
Figure 6. Concluded.



L-87-5676

(a) 256 gray levels.

Figure 7. Test images of diffuser.

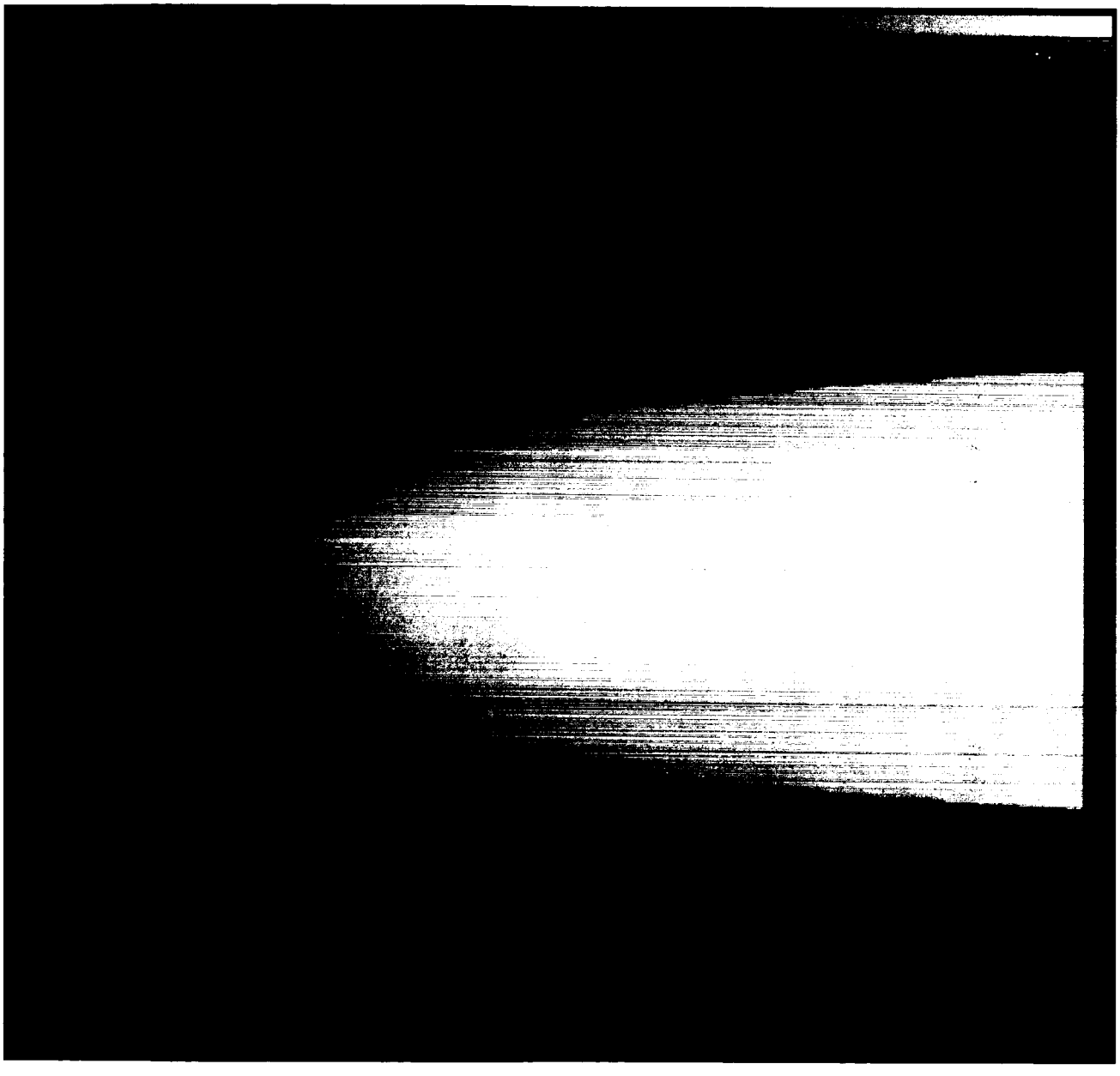


L-87-5671

(b) 128 gray levels.

Figure 7. Continued.

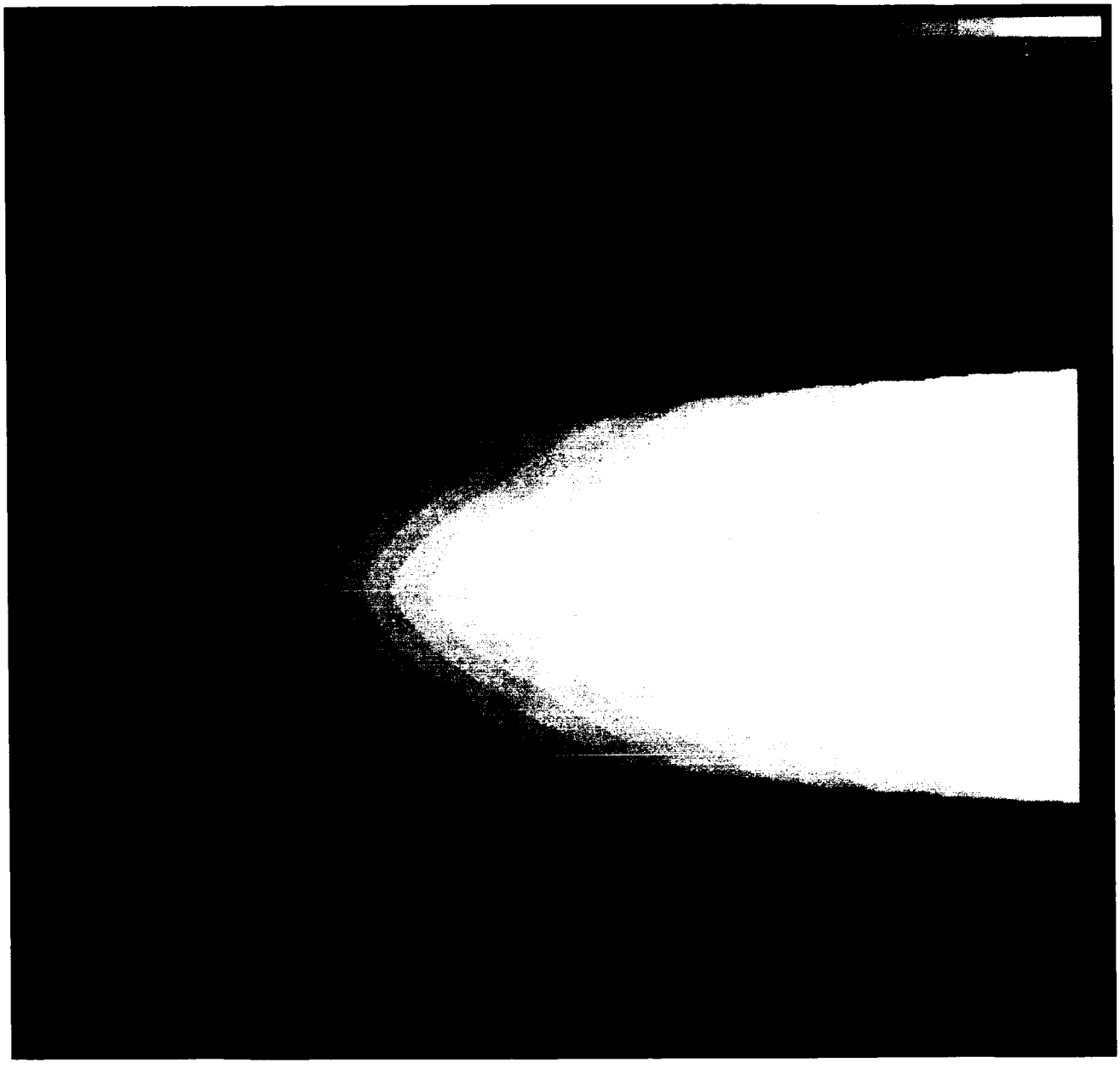
ORIGINAL PAGE IS
OF POOR QUALITY



L-86-4948

(c) 64 gray levels.

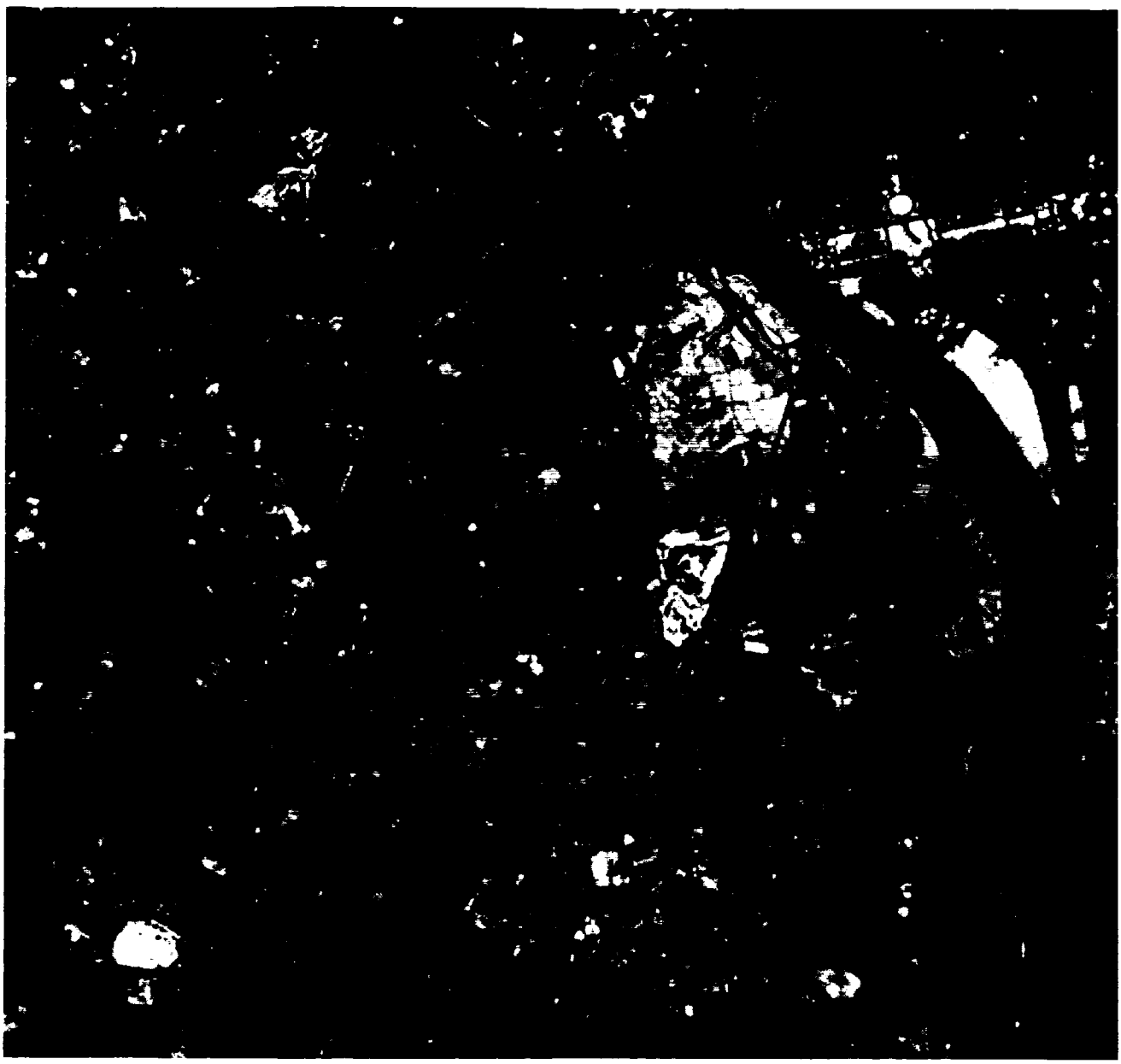
Figure 7. Continued.



L-87-5672

(d) 32 gray levels.

Figure 7. Concluded.



L-87-5673

(a) 96 gray levels.

Figure 8. Test images of Washington, D.C.

L-87-5674

(b) 48 gray levels.

Figure 8. Continued.

L-86-5352

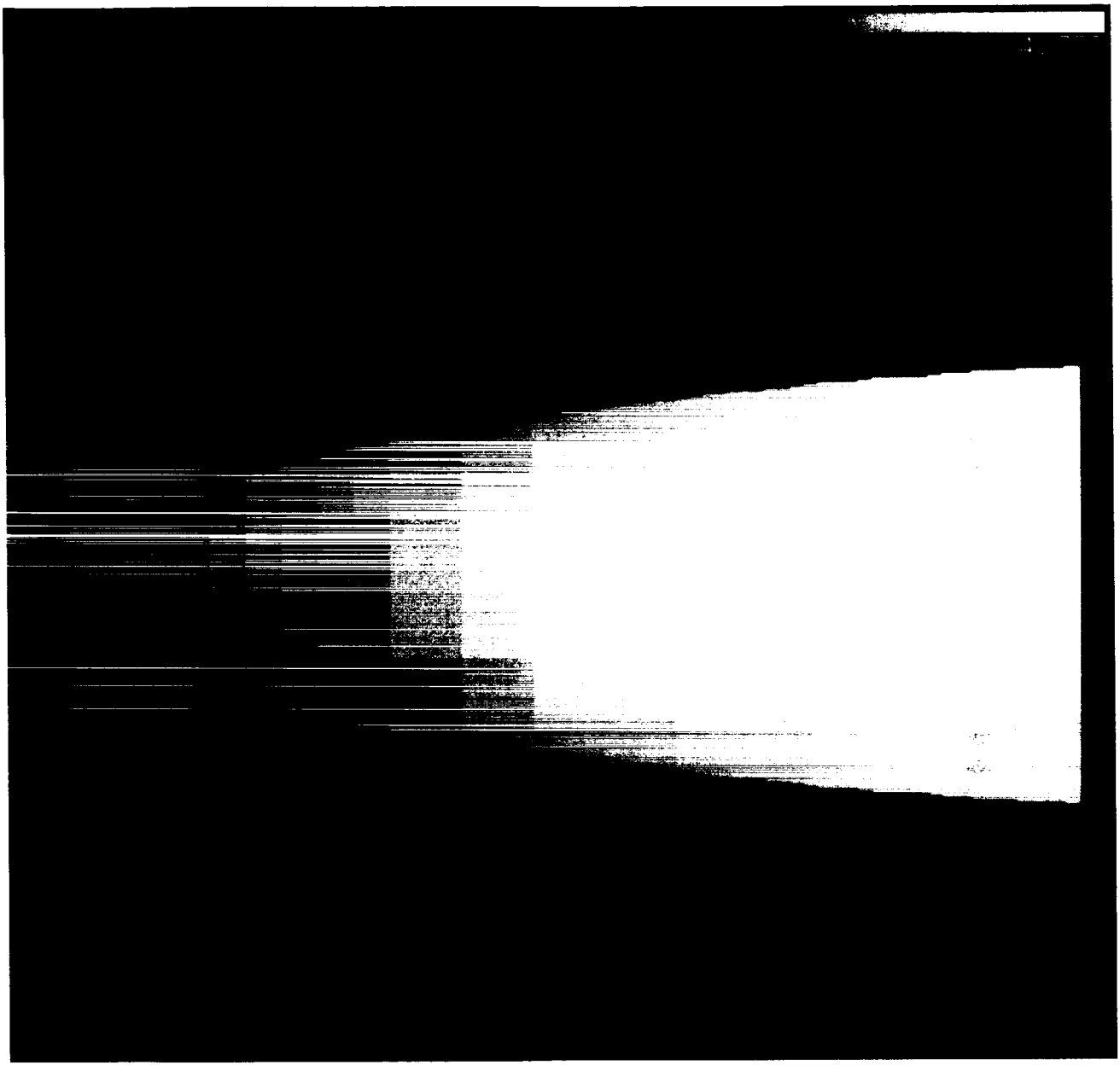
(c) 24 gray levels.

Figure 8. Continued.

L-87-5675

(d) 12 gray levels.

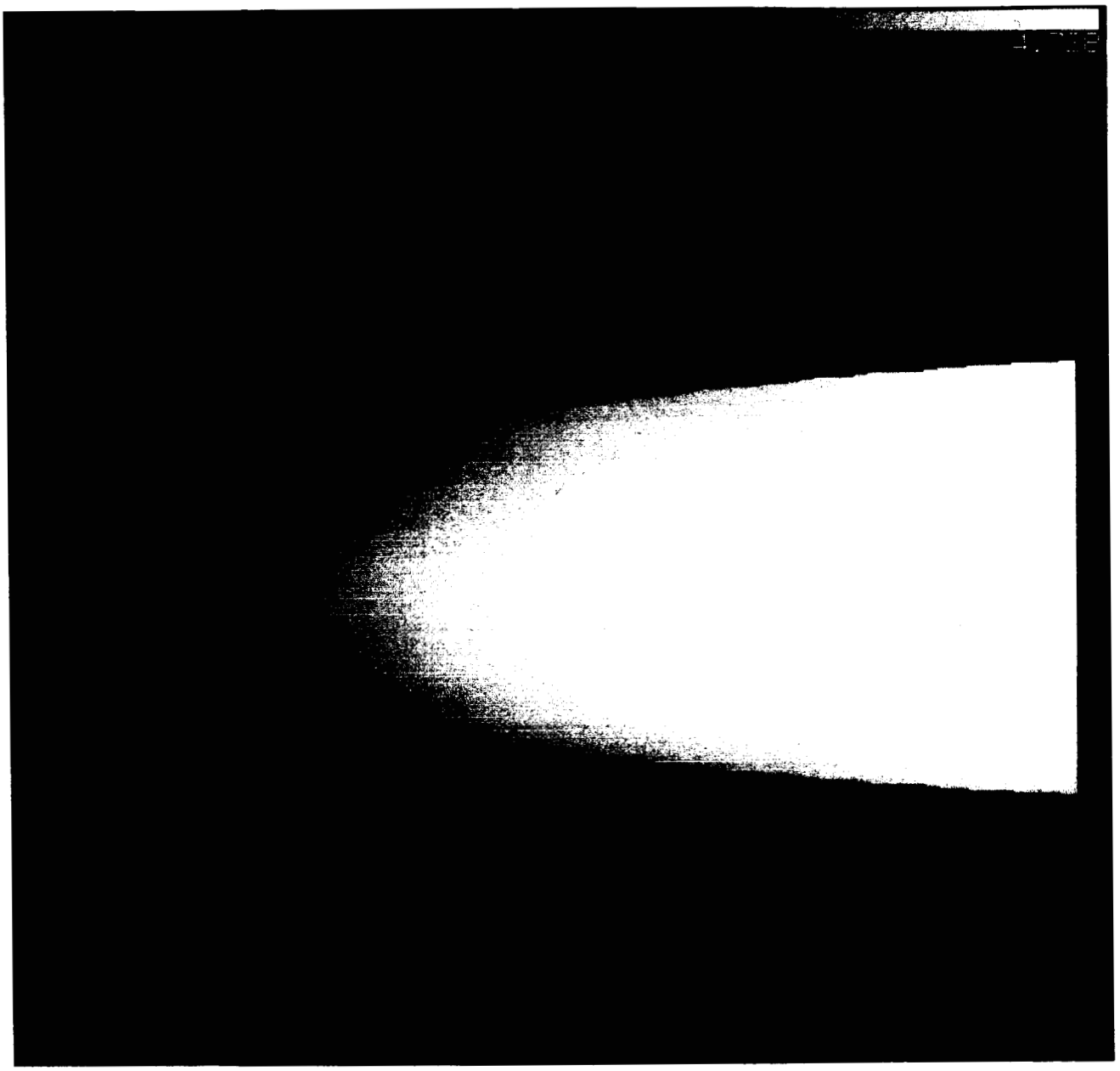
Figure 8. Concluded.



L-87-5677

(b) 256 gray levels; $\frac{\sigma}{\mu} = 0.05$.

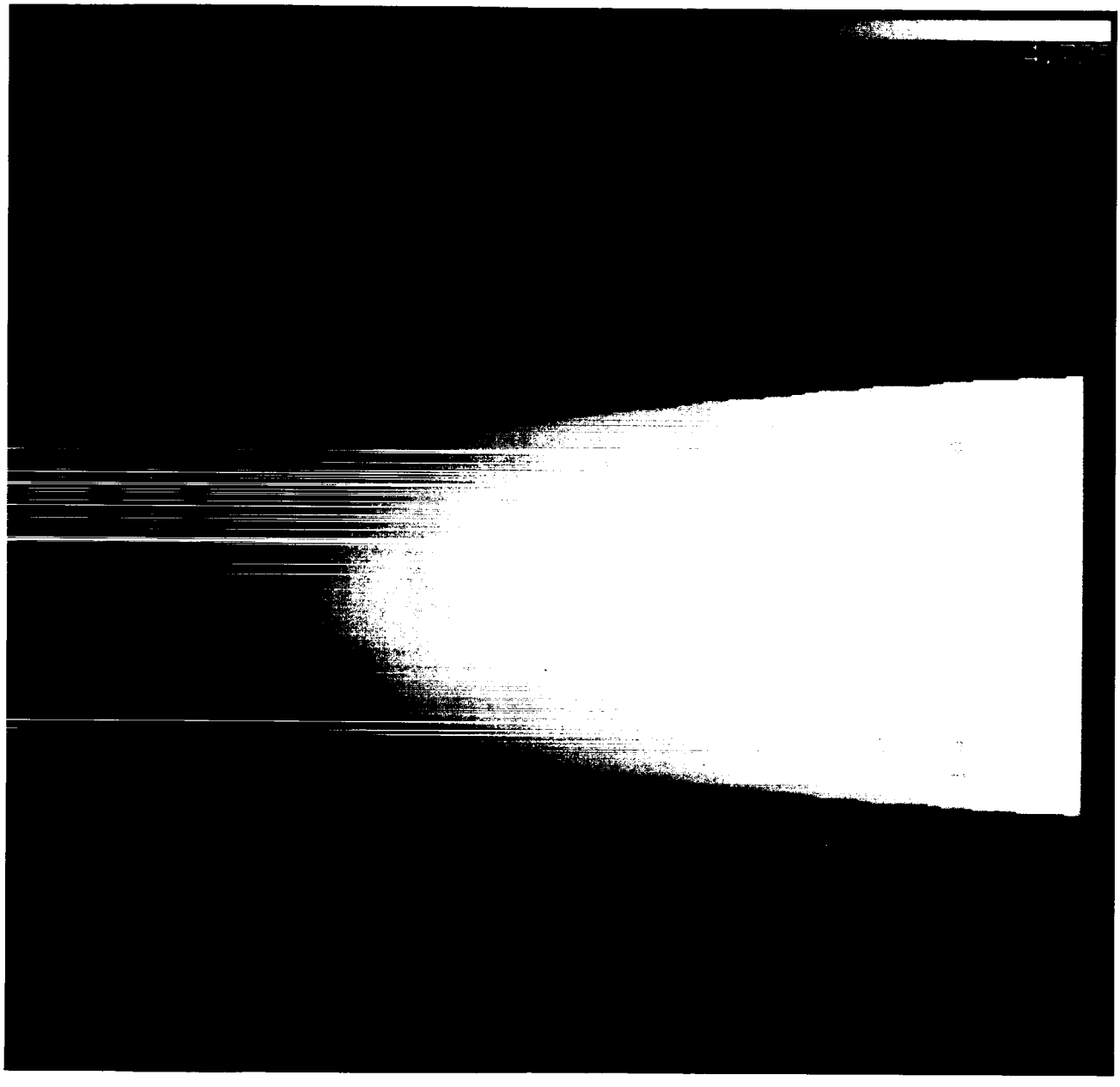
Figure 9. Continued.



L-87-5670

(a) 256 gray levels, unsegmented, for comparison.

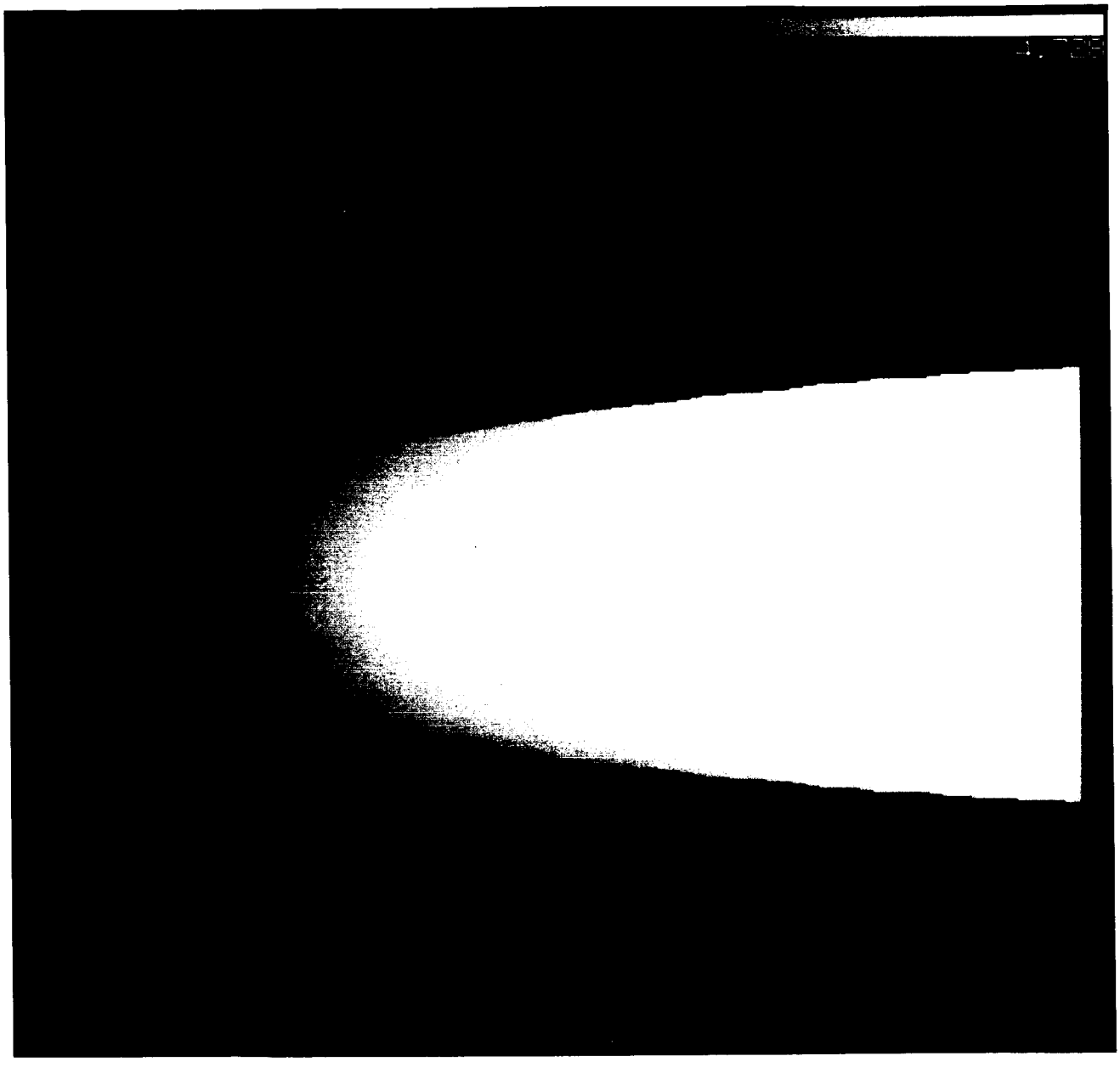
Figure 9. Segmented images with 256 gray levels of diffuser.



L-87-5678

(c) 256 gray levels; $\frac{\sigma}{\mu} = 0.01$.

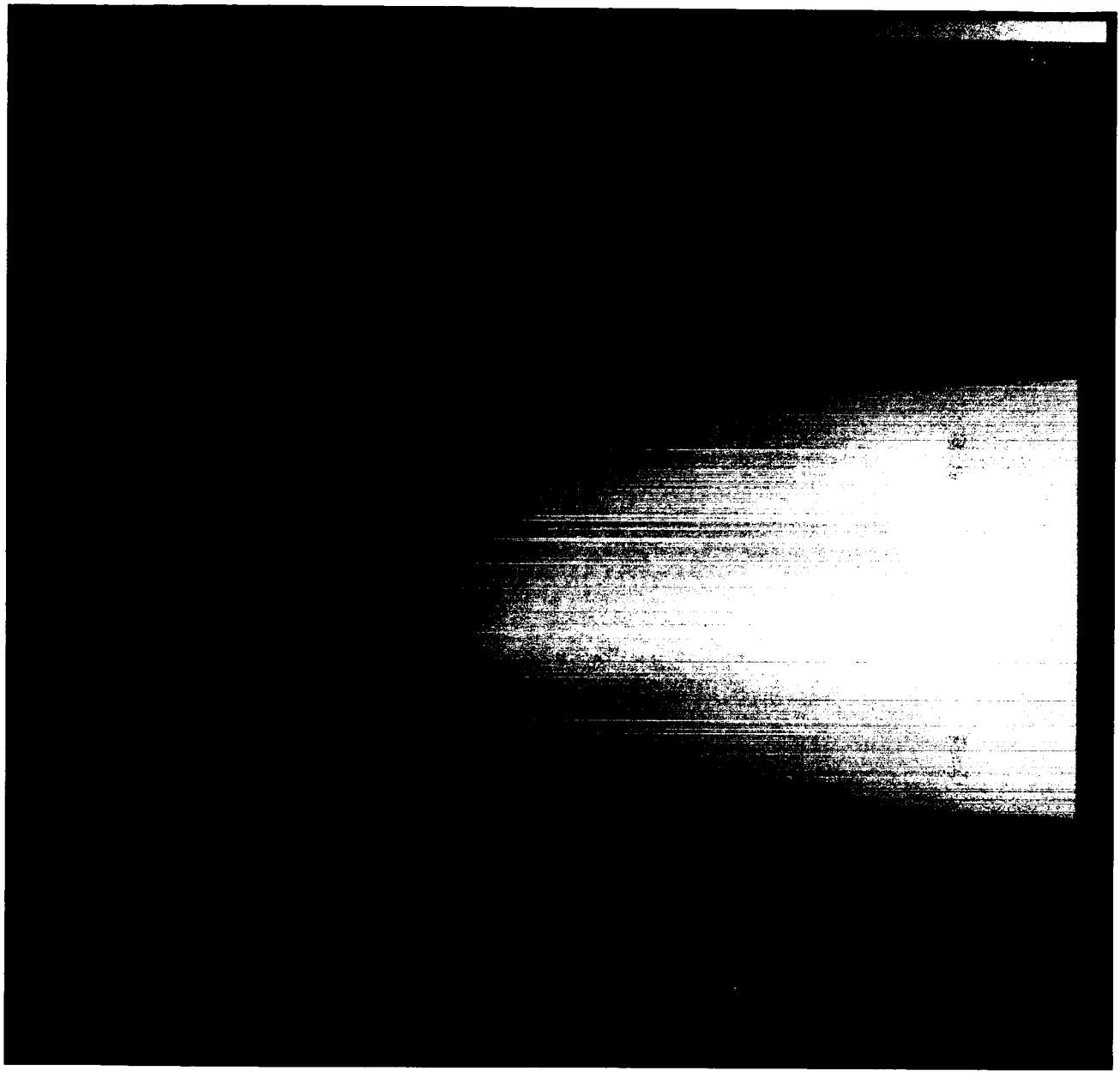
Figure 9. Continued.



L-87-5679

(d) 256 gray levels; $\frac{\sigma}{\mu} = 0.005$.

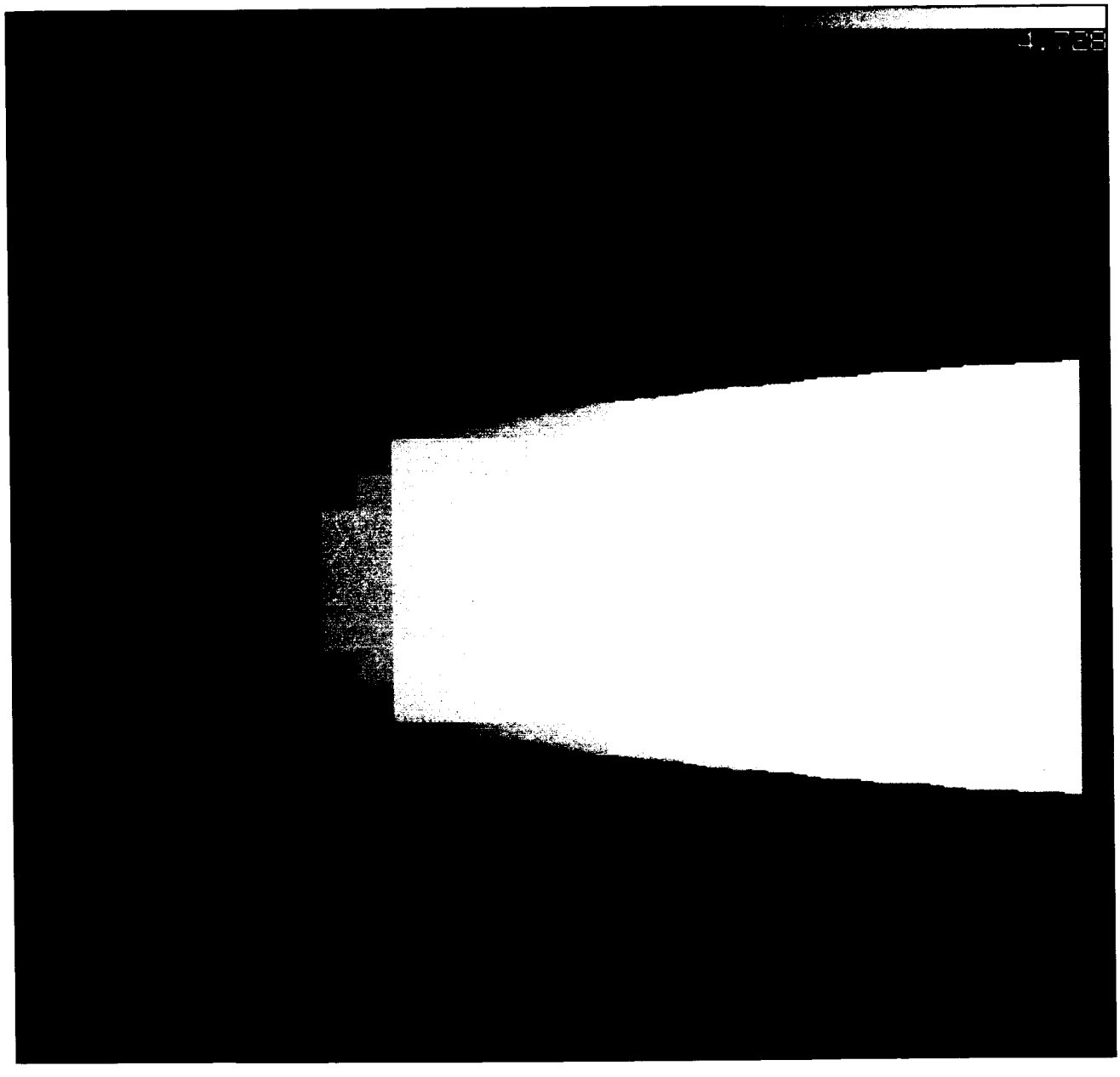
Figure 9. Concluded.



L-86-4948

(a) 64 gray levels, unsegmented, for comparison.

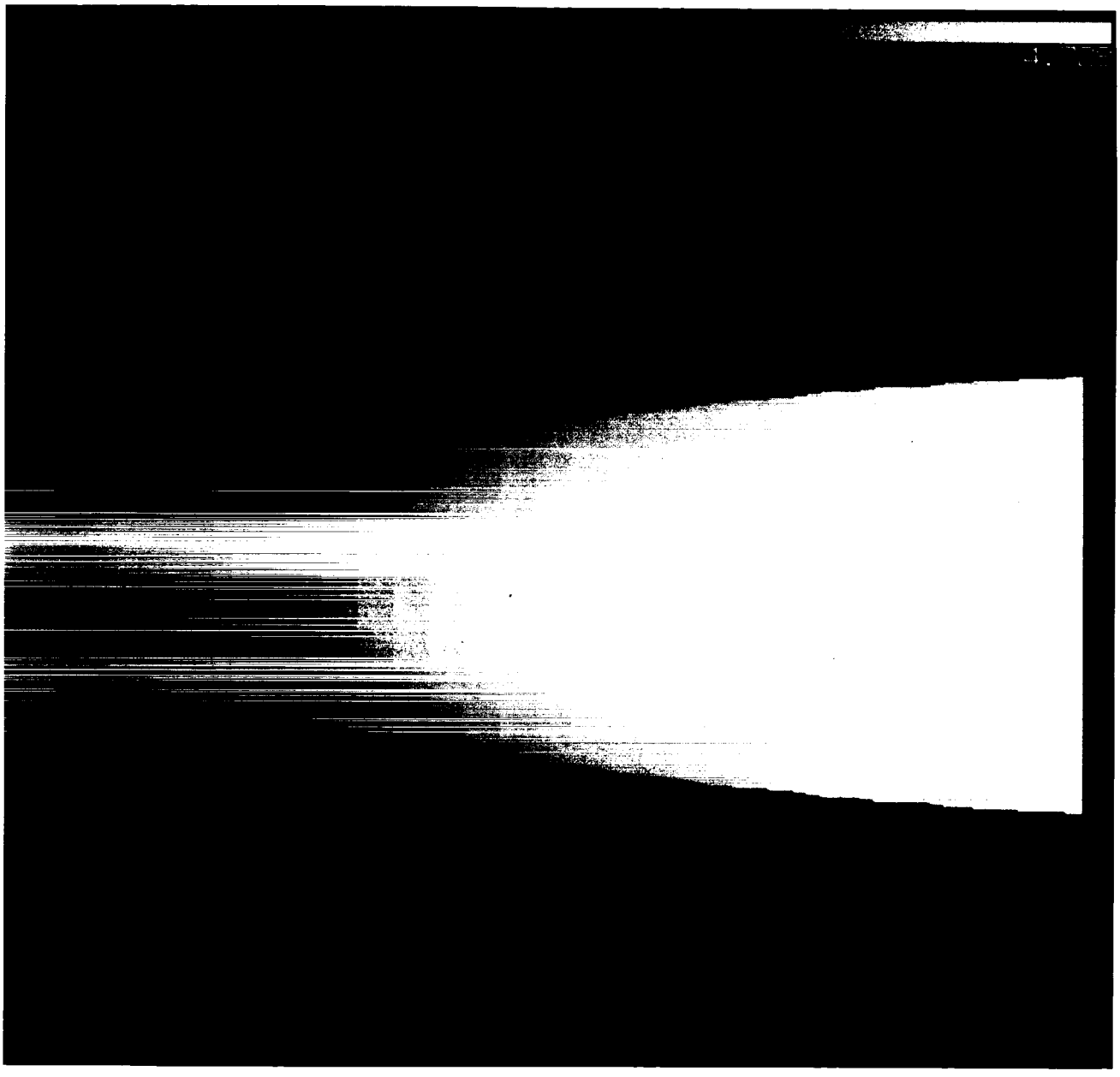
Figure 10. Segmented images with 64 gray levels of diffuser.



L-86-4945

(b) 64 gray levels; $\frac{\sigma}{\mu} = 0.10$.

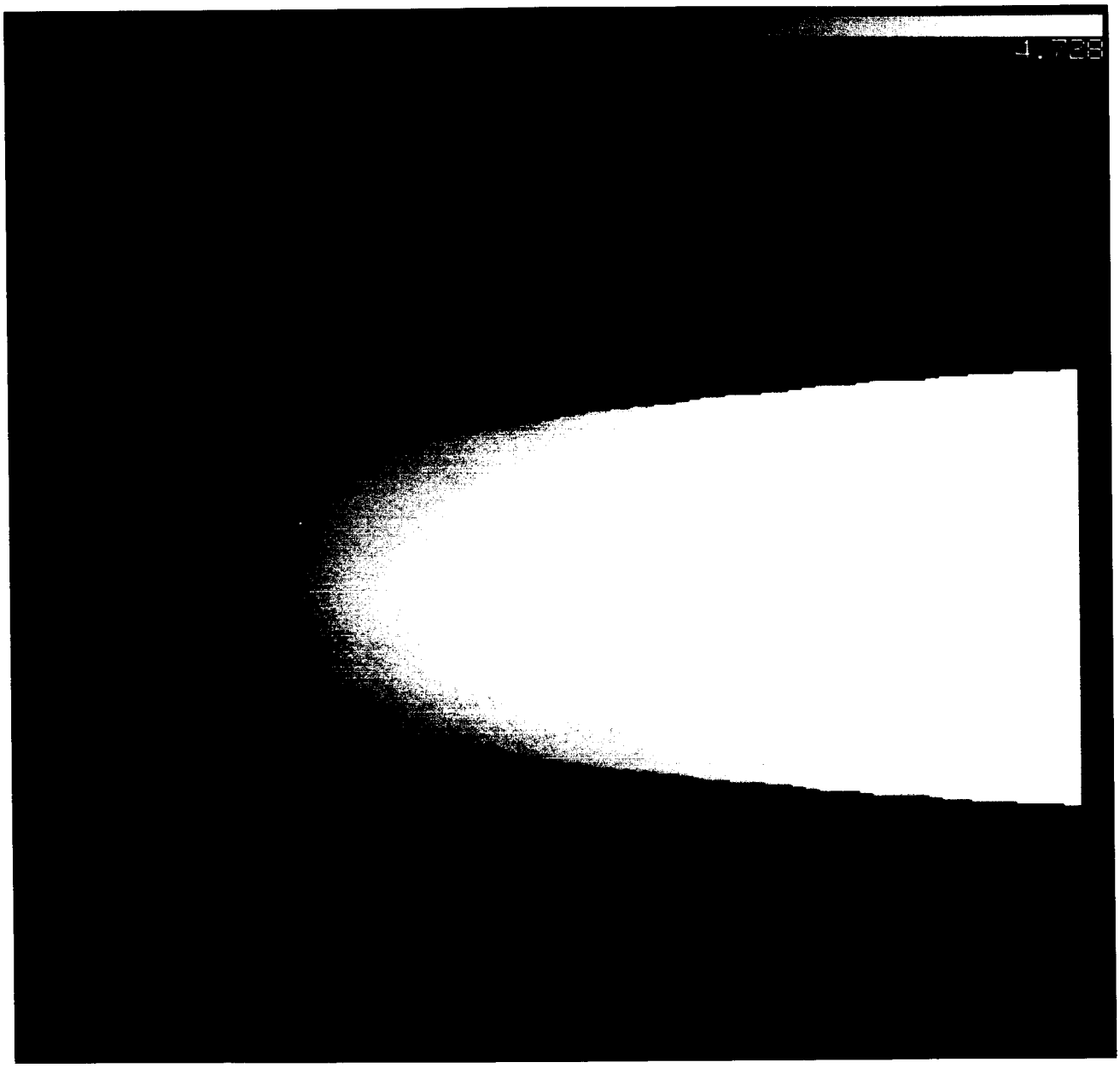
Figure 10. Continued.



L-86-4947

(c) 64 gray levels; $\frac{\sigma}{\mu} = 0.02$.

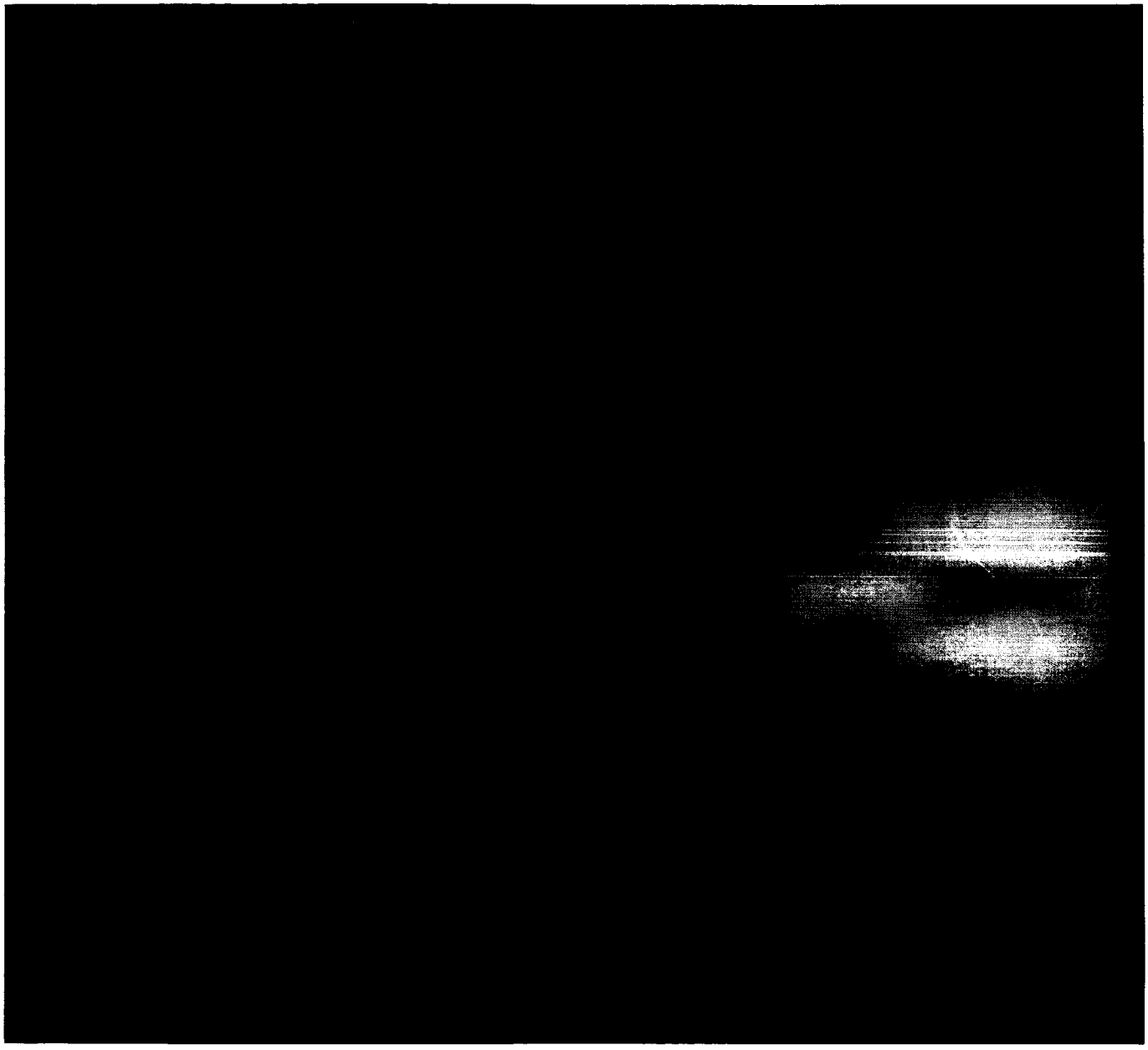
Figure 10. Continued.



L-86-4946

(d) 64 gray levels; $\frac{\sigma}{\mu} = 0.01$.

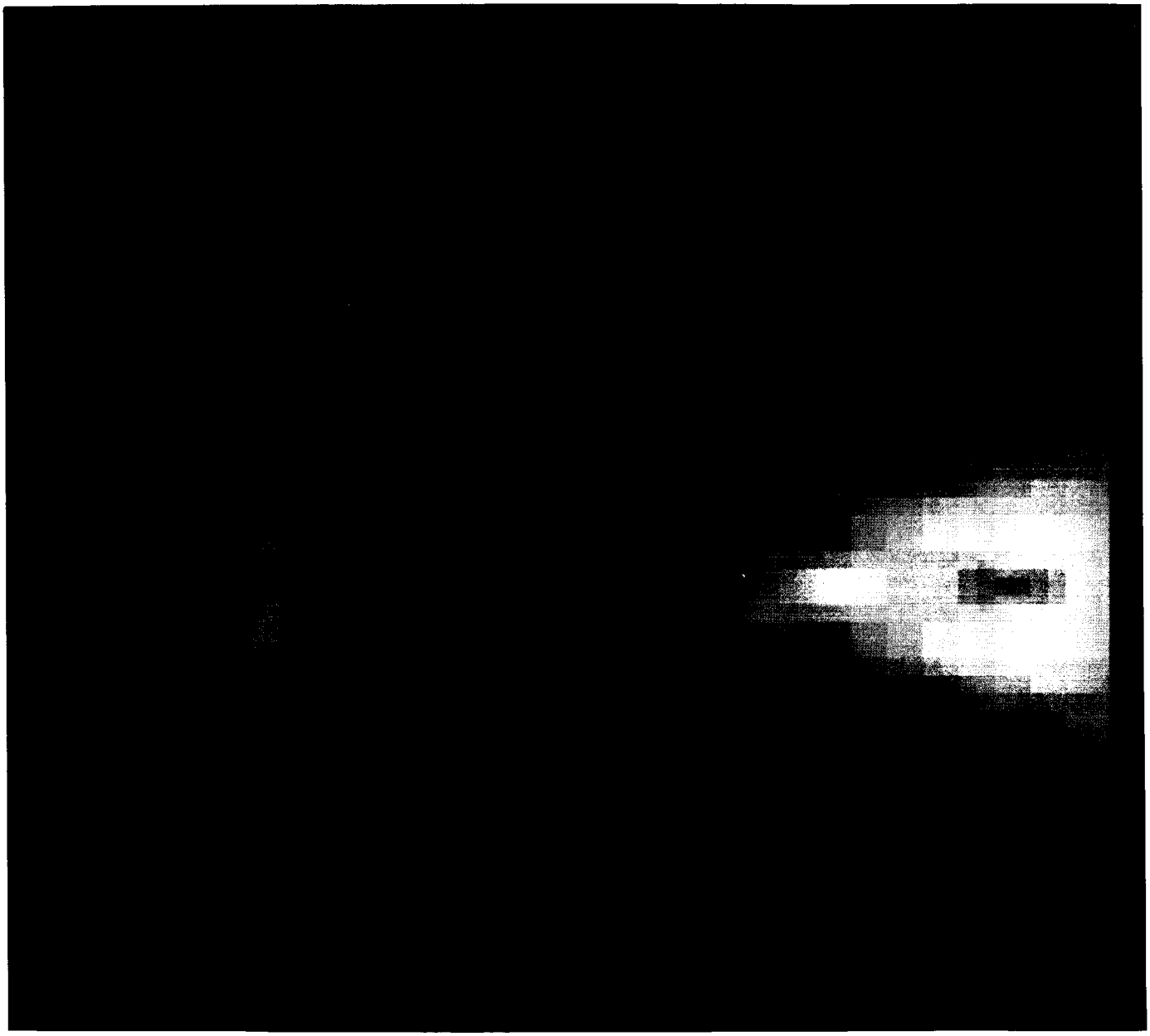
Figure 10. Concluded.



L-86-4976

(a) 256 colors; unsegmented.

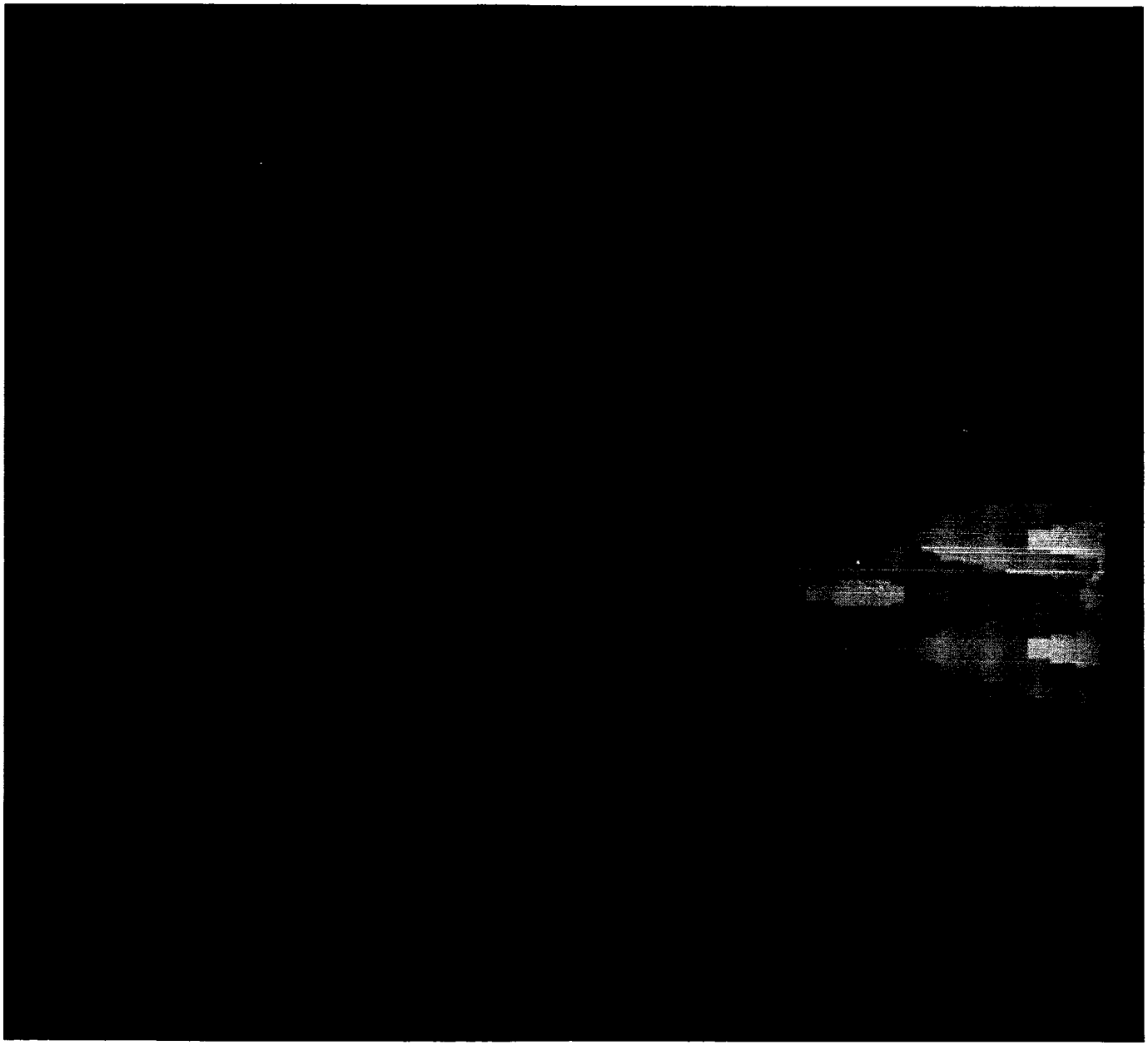
Figure 11. Color images of diffuser.



L-86-4974

(b) 256 colors; $\frac{\sigma}{\mu} = 0.005$.

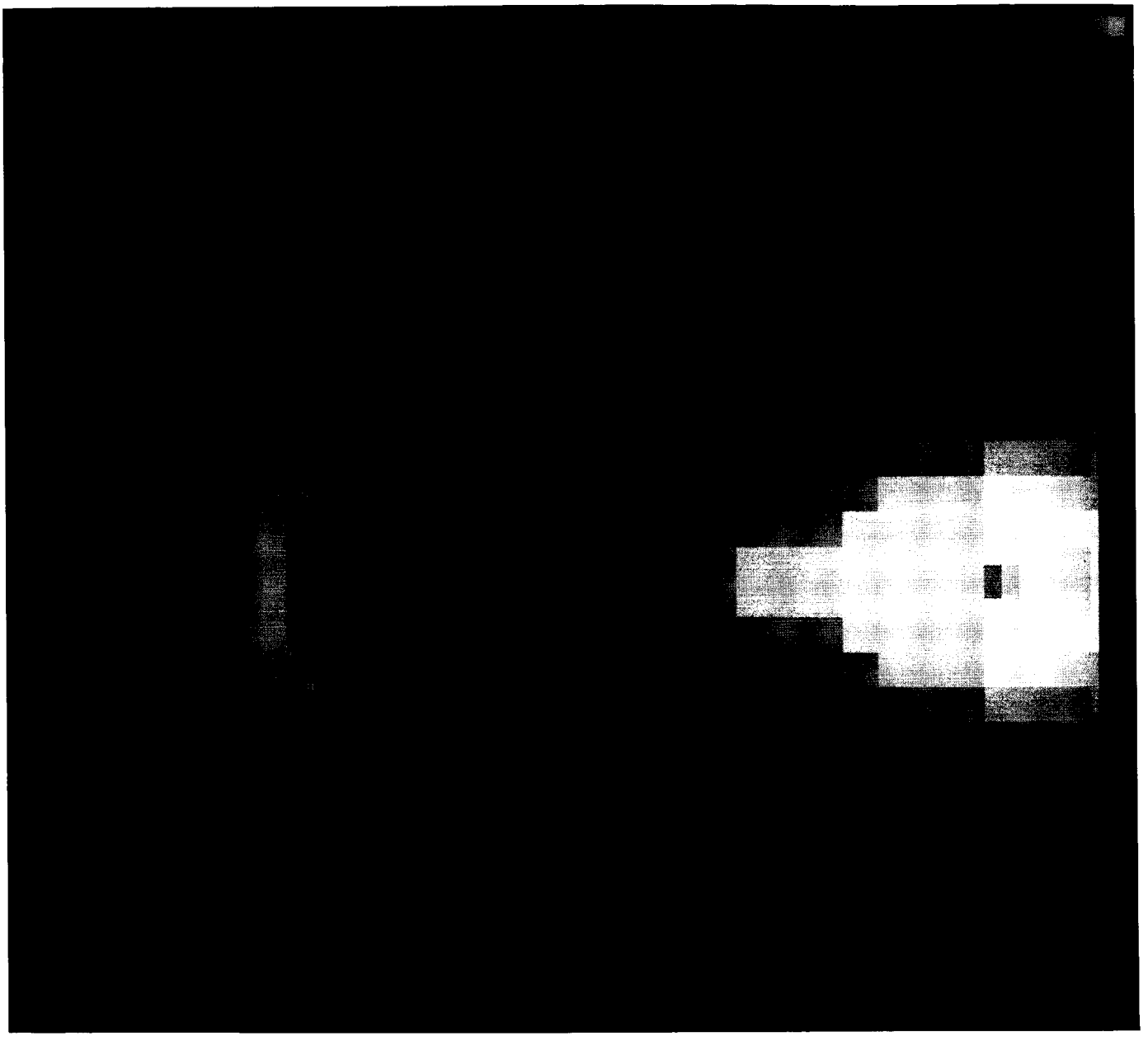
Figure 11. Continued.



L-86-4975

(c) 64 colors; unsegmented.

Figure 11. Continued.



L-86-4973

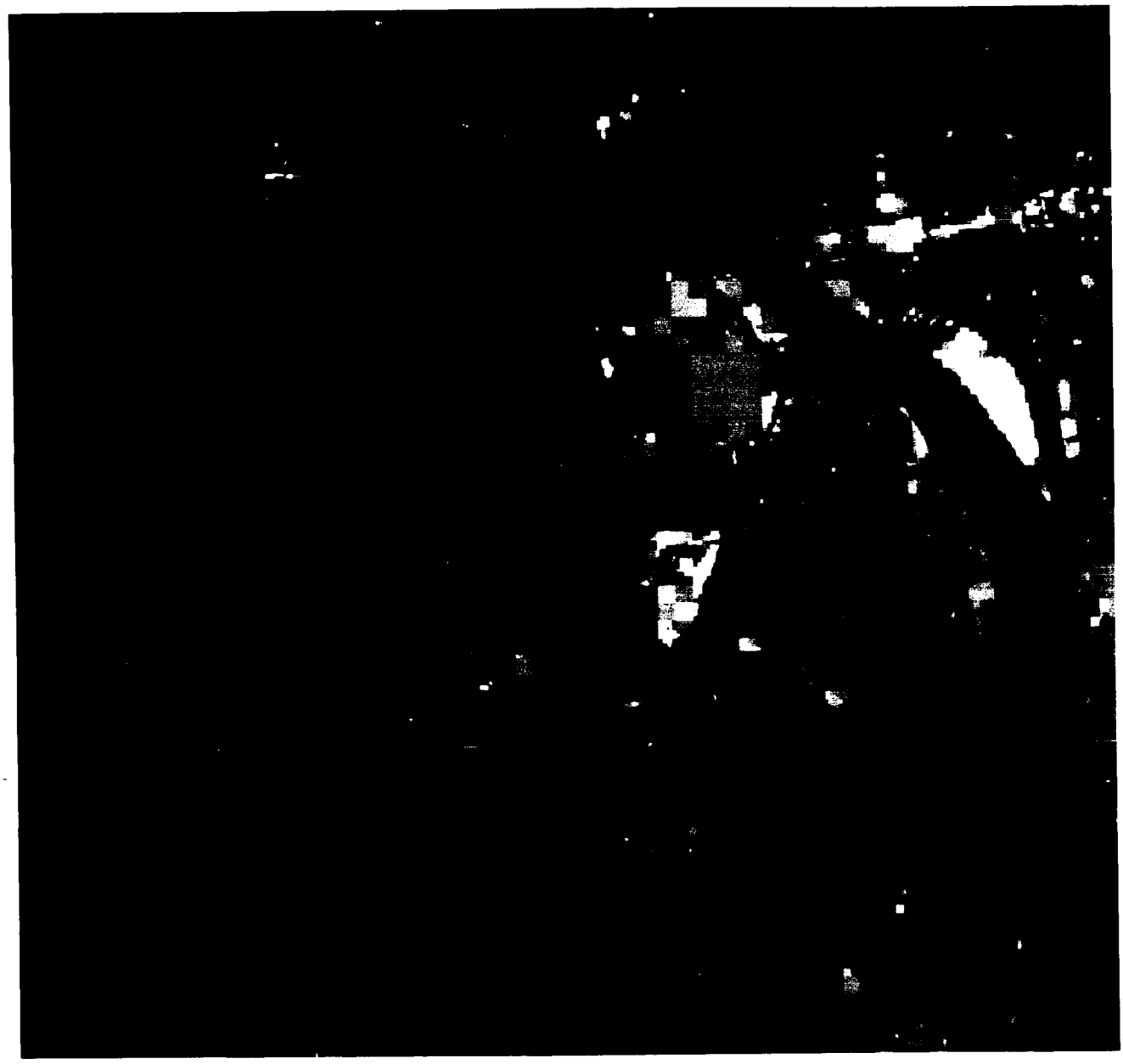
(d) 64 colors; $\frac{\sigma}{\mu} = 0.01$.

Figure 11. Concluded.

L-87-5673

(a) 96 gray levels, unsegmented, for comparison.

Figure 12. Segmented images with 96 gray levels of Washington, D.C.



L-87-5680

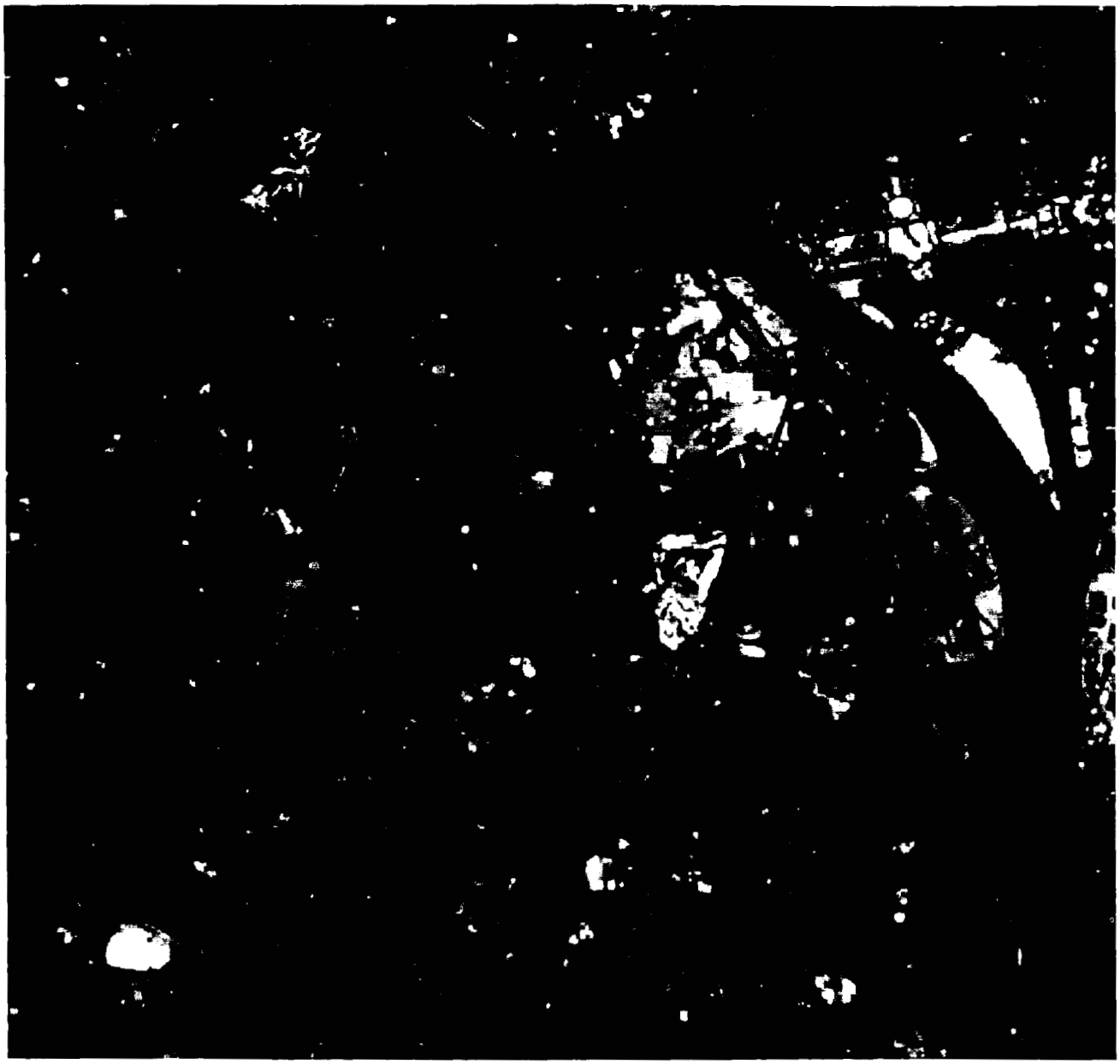
(b) 96 gray levels; $\frac{\sigma}{\mu} = 0.20$.

Figure 12. Continued.

L-87-5681

(c) 96 gray levels; $\frac{\sigma}{\mu} = 0.15$.

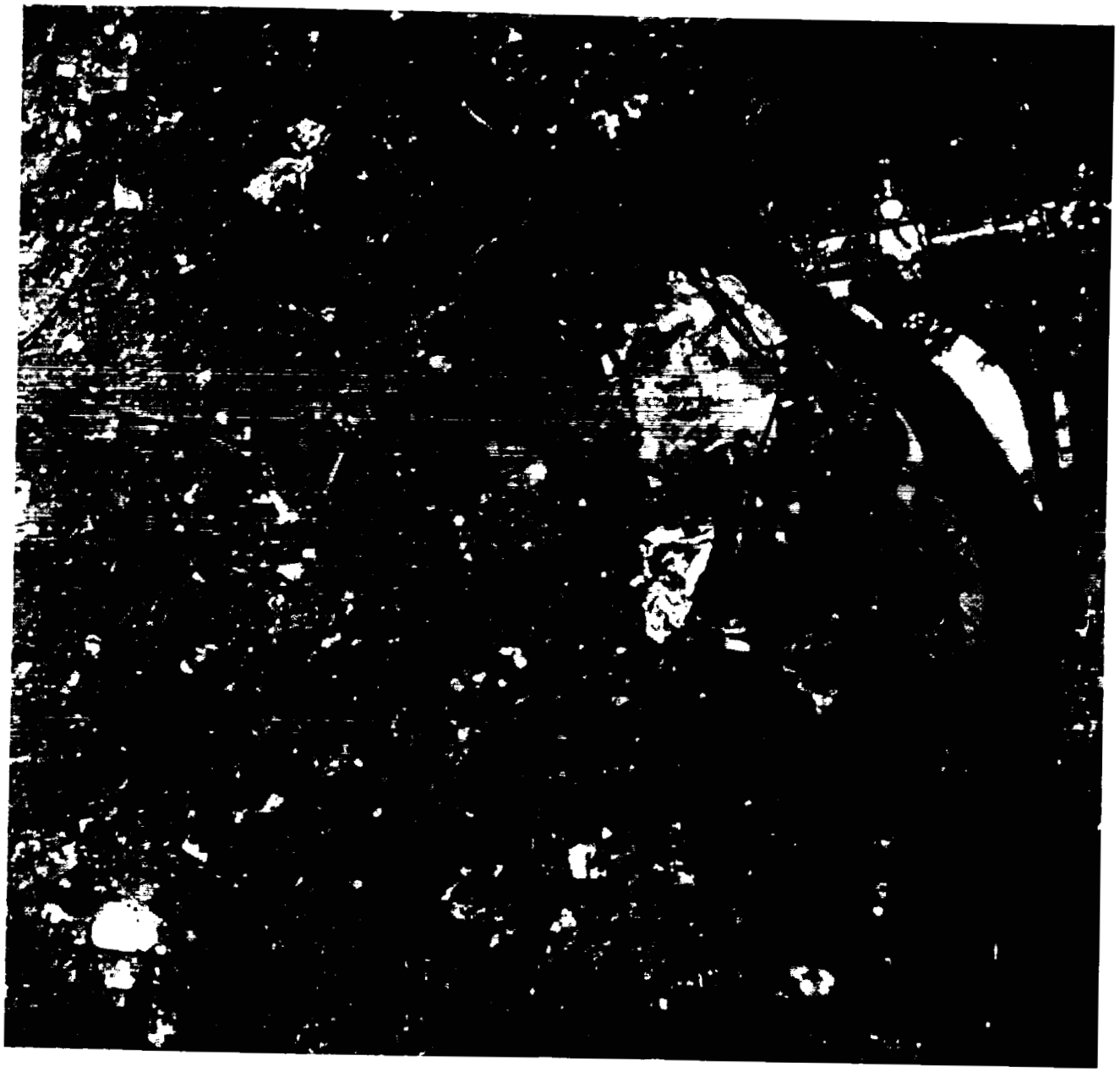
Figure 12. Continued.



L-87-5682

(d) 96 gray levels; $\frac{\sigma}{\mu} = 0.10$.

Figure 12. Continued.



L-87-5683

(e) 96 gray levels; $\frac{\sigma}{\mu} = 0.08$.

Figure 12. Continued.

L-87-5684

(f) 96 gray levels; $\frac{\sigma}{\mu} = 0.06$.

Figure 12. Concluded.

L-86-5352

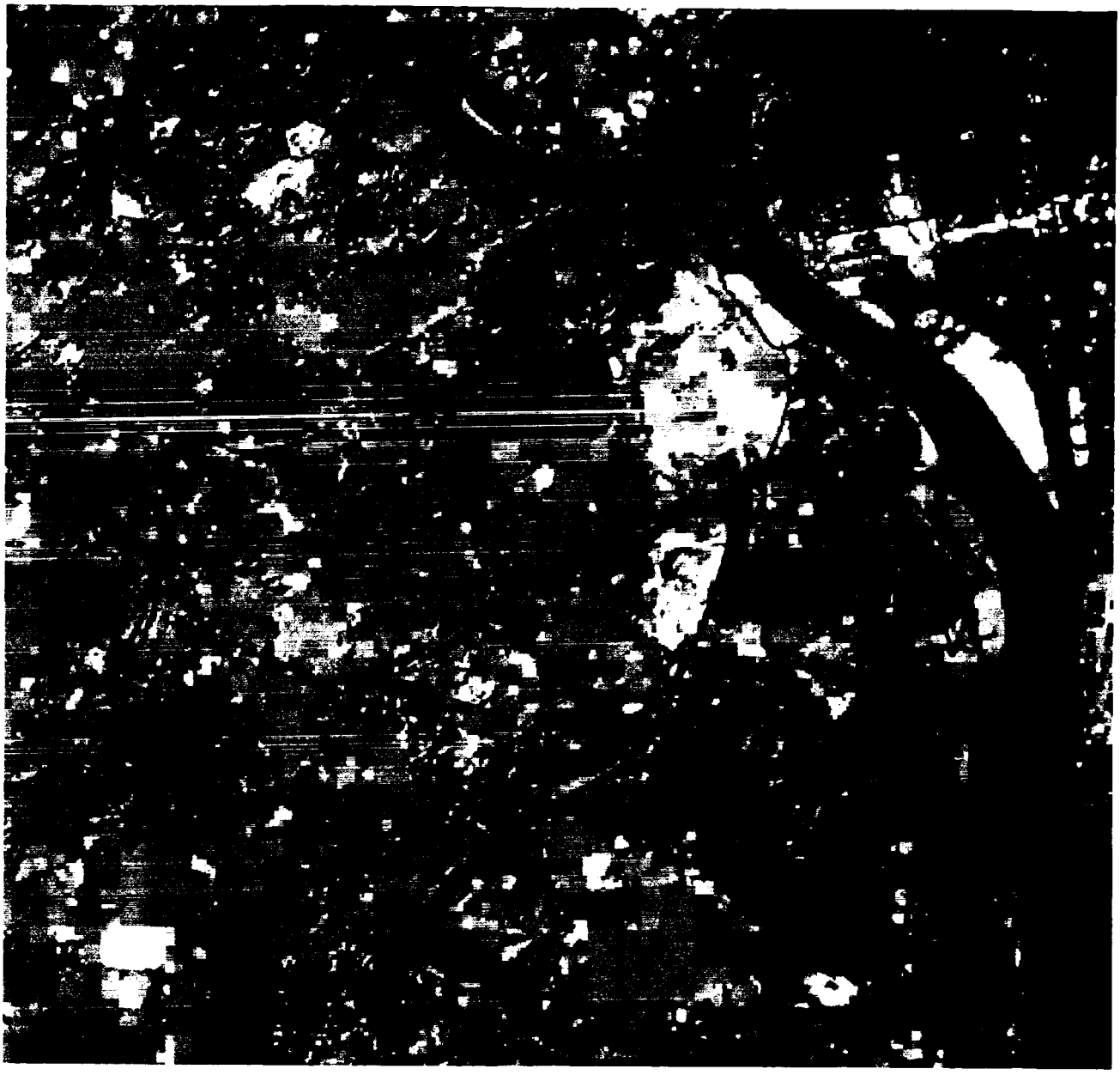
(a) 24 gray levels, unsegmented, for comparison.

Figure 13. Segmented images with 24 gray levels of Washington, D.C.

L-87-5685

(b) 24 gray levels; $\frac{\sigma}{\mu} = 0.20$.

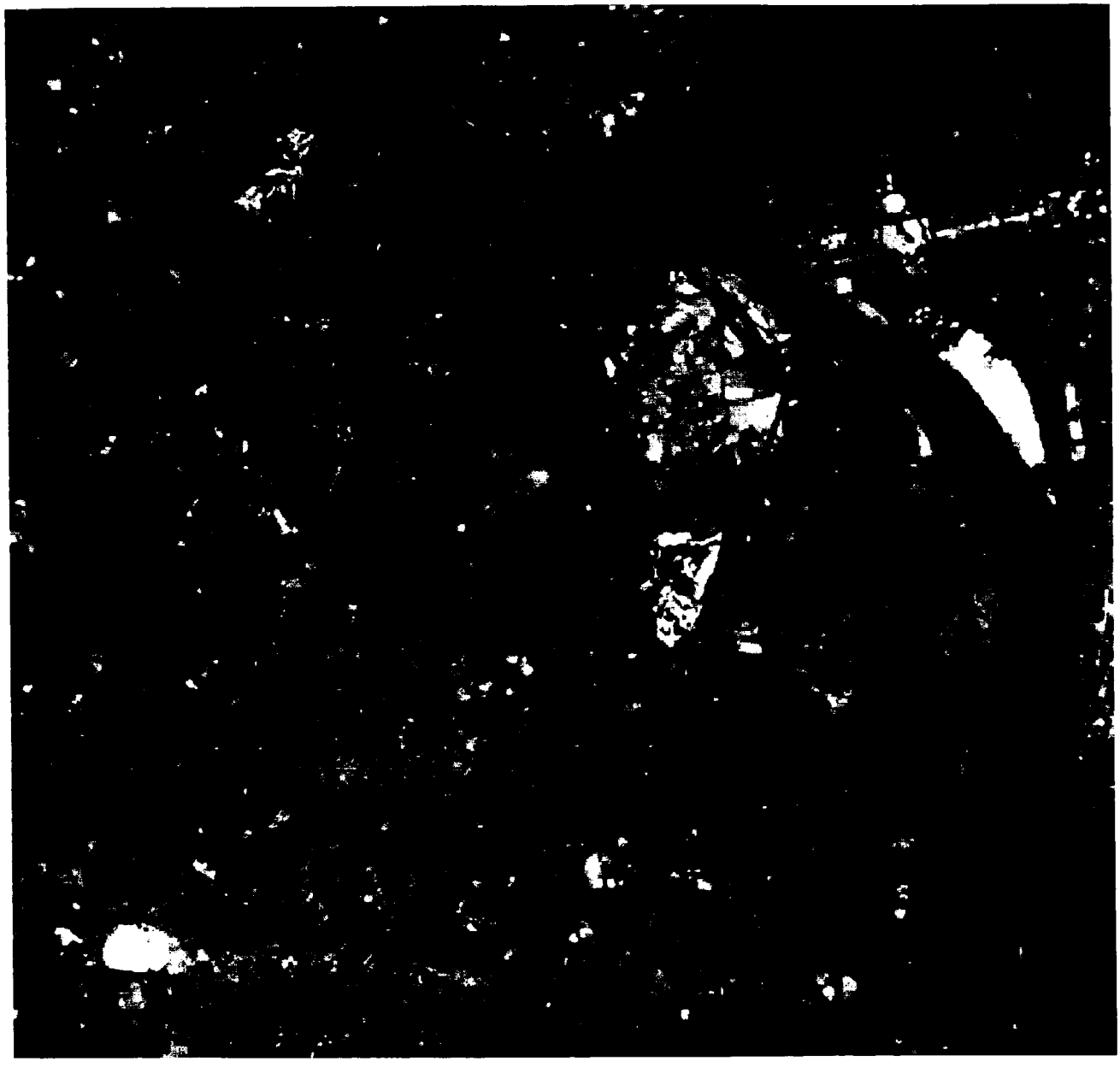
Figure 13. Continued.



L-87-5686

(c) 24 gray levels; $\frac{\sigma}{\mu} = 0.15$.

Figure 13. Continued.



L-87-5687

(d) 24 gray levels; $\frac{\sigma}{\mu} = 0.10$.

Figure 13. Continued.

L-87-5688

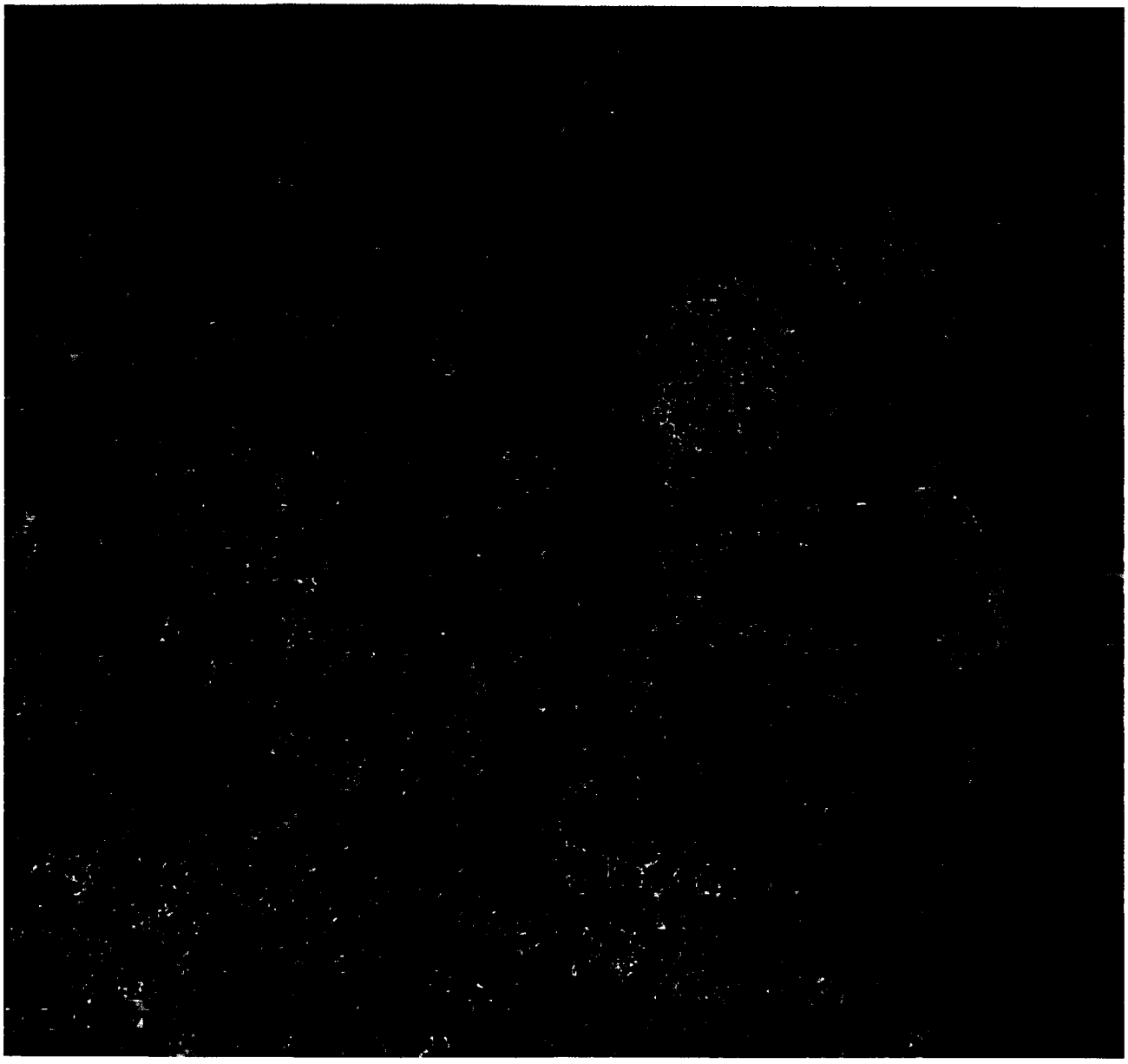
(e) 24 gray levels; $\frac{\sigma}{\mu} = 0.08$.

Figure 13. Continued.

L-87-5689

(f) 24 gray levels; $\frac{\sigma}{\mu} = 0.06$.

Figure 13. Concluded.

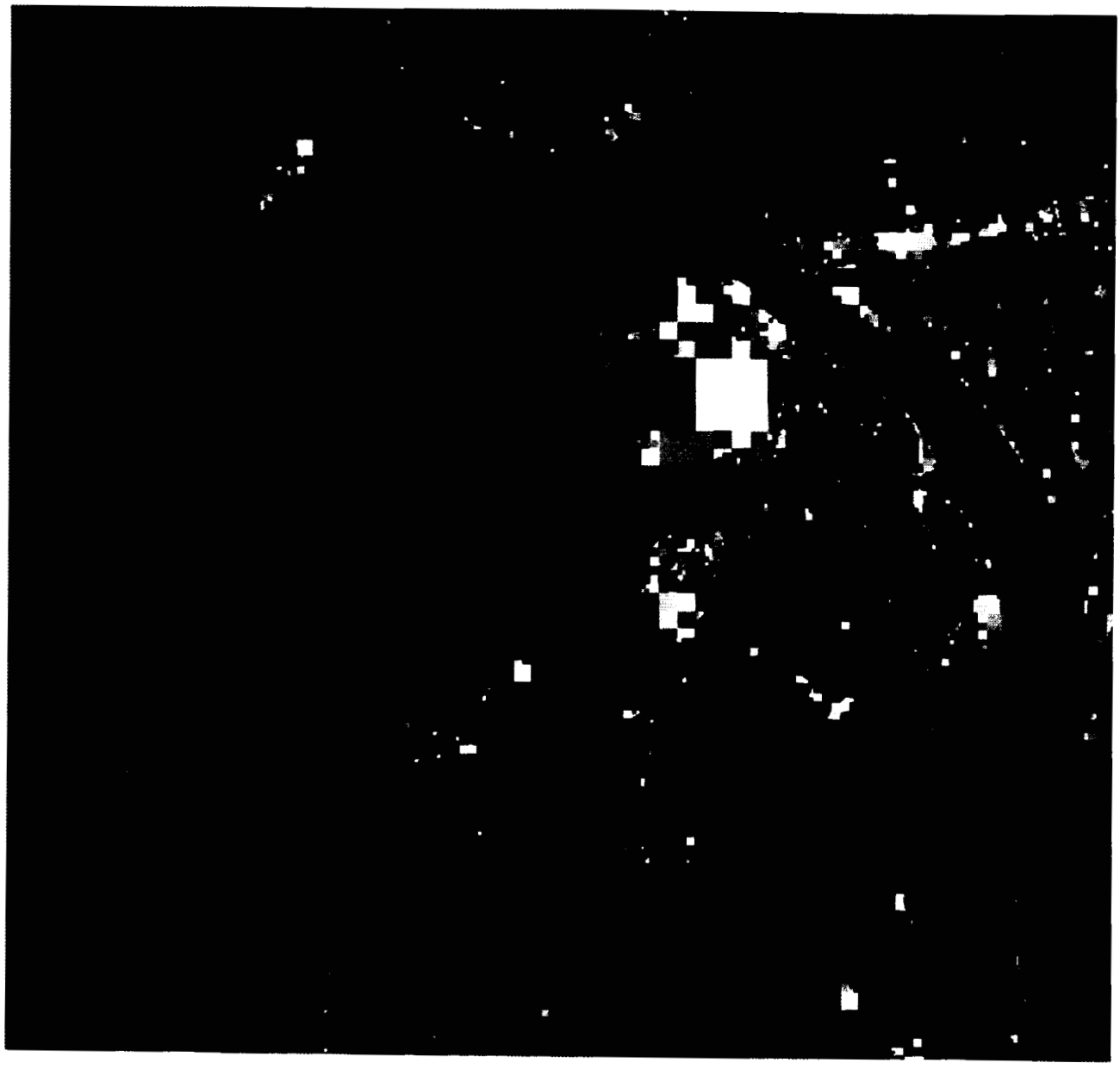


L-86-6290

(a) 96 colors; unsegmented.

Figure 14. Color images of Washington, D.C.

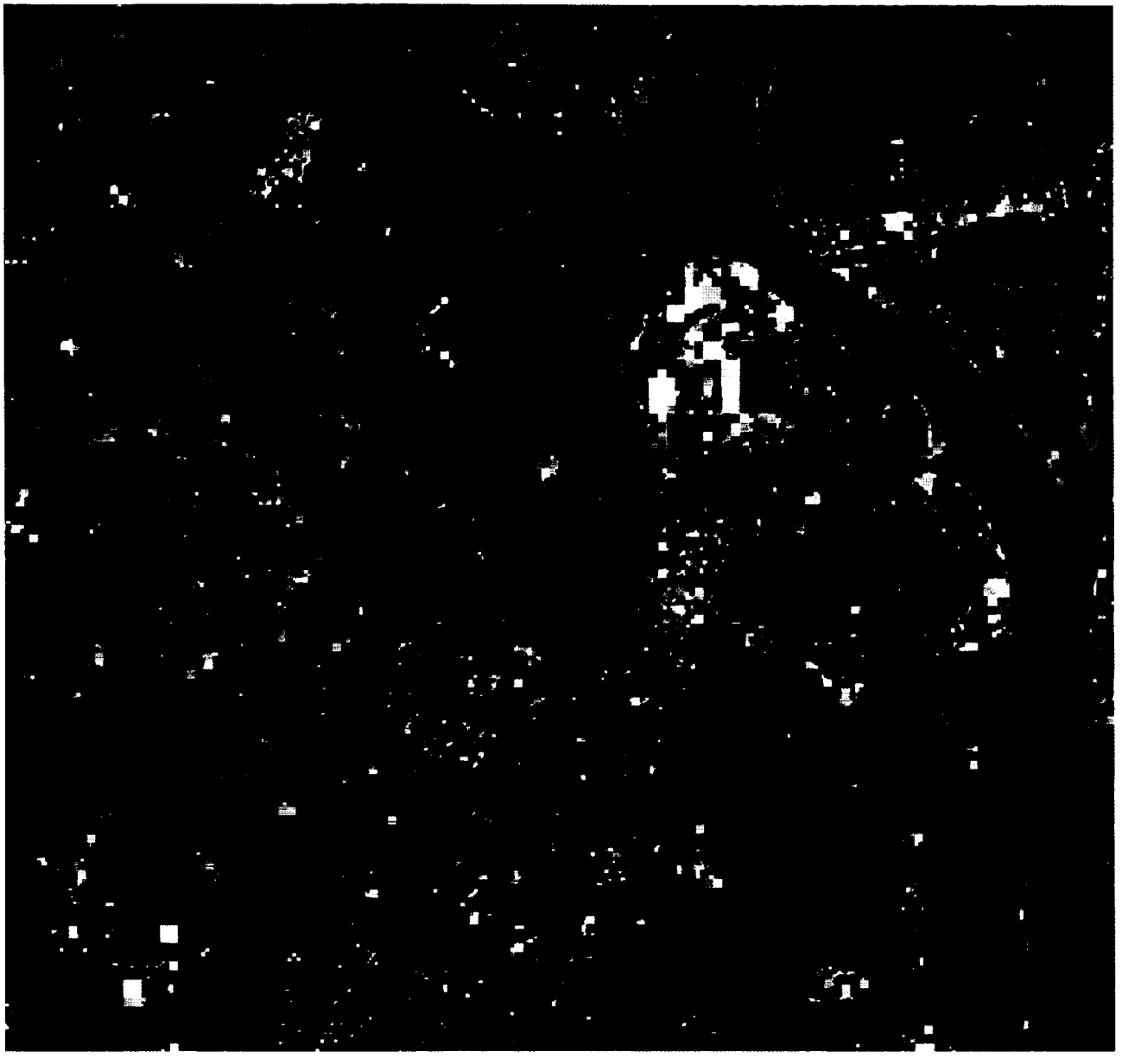
ORIGINAL PAGE
COLOR PHOTOGRAPH



L-86-6292

(b) 96 colors; $\frac{\sigma}{\mu} = 0.20$.

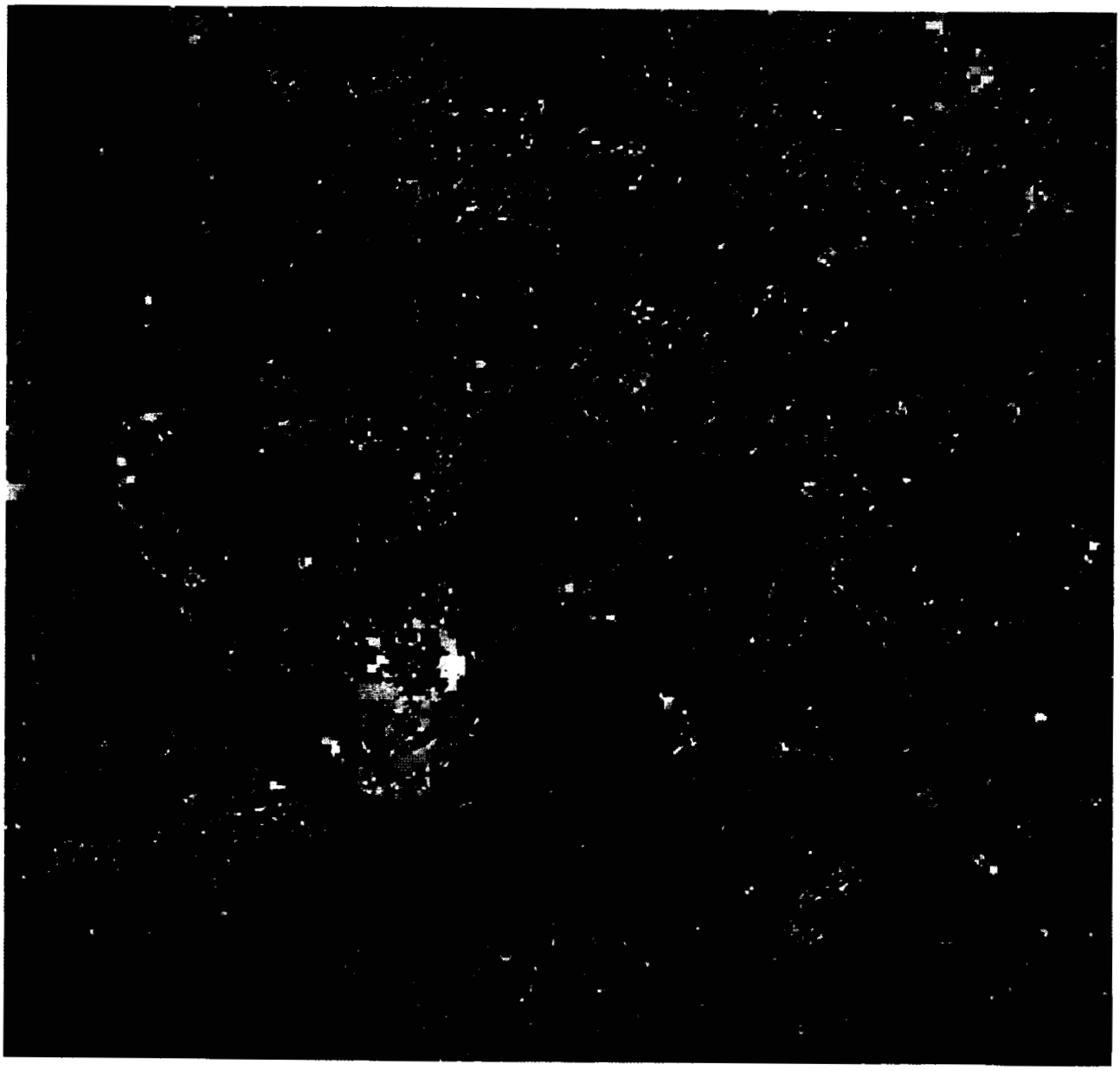
Figure 14. Continued.



L-86-6293

(c) 96 colors; $\frac{\sigma}{\mu} = 0.15$.

Figure 14. Continued.



L-86-6291

(d) 96 colors; $\frac{\sigma}{\mu} = 0.10$.

Figure 14. Concluded.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TP-2722		

4. Title and Subtitle	5. Report Date
Experiments in Encoding Multilevel Images as Quadtrees	September 1987
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Donald L. Lansing	L-16292
9. Performing Organization Name and Address	10. Work Unit No.
NASA Langley Research Center Hampton, VA 23665-5225	505-60-01-01
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, DC 20546-0001	Technical Paper
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This paper investigates image storage requirements for several encoding methods and explores the use of quadtrees with multigray level or multicolor images. The results of encoding a variety of images having up to 256 gray levels using three schemes—full-raster, runlength, and quadtree—are presented. Although there is considerable literature on the use of quadtrees to store and manipulate binary images, their application to multilevel images is relatively undeveloped. The potential advantage of quadtree encoding is that an entire area with a uniform gray level may be encoded as a unit. This paper describes a pointerless quadtree encoding scheme. Data are presented on the size of the quadtree required to encode selected images and on the relative storage requirements of the three encoding schemes. A segmentation scheme based on the statistical variation of gray levels within a quadtree quadrant is described. This parametric scheme may be used to control the storage required by an encoded image and to preprocess a scene for feature identification. Several sets of black and white and pseudocolor images obtained by varying the segmentation parameter are shown.

17. Key Words (Suggested by Authors(s))	18. Distribution Statement
Image encoding File compression Quadtrees Image processing Image Data structures	Unclassified—Unlimited Subject Category 64

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	59	A04

NASA FORM 1626 OCT 86 NASA-Langley, 1987

For sale by the National Technical Information Service, Springfield, Virginia 22161-2171