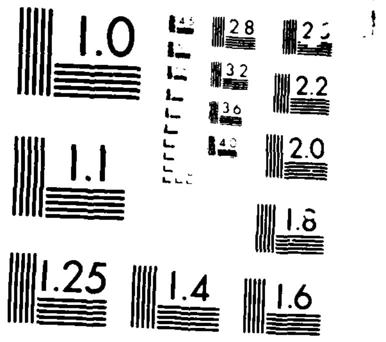
AD-A169 147 SPECIAL ASSOCIATIVE PREPROCESSING STRUCTURES FOR QUALITATIVE FEATURE EXTRACTION(U) ARMY MILITARY PERSONNEL CENTER ALEXANDRIA VA M C BIBBY 11 JUN 86 1/2

UNCLASSIFIED F/G 9/2 NL



MICROC

(2)

AD-A169 147

Special Associative Preprocessing Structures for Qualitative Feature Extraction

Captain Michael C. Bibby
HQDA, MILPEREN (DAPC-OPA-E)
200 Stovall Street
Alexandria, VA 22332

DTIC
S ELECTE D
JUN 26 1986
D

Final Report 11 June 1986

Approved for public release;
distribution is unlimited

DTIC FILE COPY

A thesis submitted to
Oregon State University
in partial fulfillment of
the requirements for the
degree of
Master of Science

8 5

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. ADA 169147	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* ASSOCIATIVE PREPROCESSING STRUCTURES FOR QUALITATIVE FEATURE EXTRACTION		5. TYPE OF REPORT & PERIOD COVERED FINAL REPORT, 11 JUN 86
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) CPT MICHAEL E BIBBY		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS STUDENT HQDA, MILPERCEN (DAPC OPA-E) 200 STOVALL STREET ALEXANDRIA, VA 22332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS HQDA, MILPERCEN ATTN DAPC-OPA-E 200 STOVALL STREET ALEXANDRIA, VA 22332		12. REPORT DATE 11 JUN 86
		13. NUMBER OF PAGES 161
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLAS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

MASTER'S THESIS, OREGON STATE UNIVERSITY

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

ASSOCIATIVE MEMORY; ASSOCIATIVE PROCESSING; VISION SYSTEMS
CELLULAR LOGIC ARRAYS; PIXEL PROCESSOR; PARALLEL STRUCTURE
COMPUTER VISION; REAL TIME PROCESSING; QUALITATIVE FEATURES
TEXTURE, LINE TREND

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

EXISTING PATTERN RECOGNITION AND CLASSIFICATION ALGORITHMS IN COMPUTER VISION REQUIRE VAST AMOUNTS OF COMPUTATIONS ON INPUT DATA. AS A RESULT, MEMORY ACCESS TIME IS A CRITICAL PARAMETER IN SYSTEM PERFORMANCE. TREMENDOUS PARALLELISM IN STRUCTURE AND ALGORITHM IS REQUIRED FOR THE SYSTEM TO OPERATE IN REAL-TIME. A PREPROCESSING STRUCTURE FOR QUALITATIVE FEATURE EXTRACTION WHICH MEETS THESE SYSTEM REQUIREMENTS IS PRESENTED.

IN GENERAL, THE STRUCTURE ARCHITECTURE CONSISTS OF A CELLULAR ARRAY OF PIXEL PROCESSORS EACH CONTAINING AN A MEMORY PARALLEL [illegible])

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

BLOCK 20 (CONT)

ASSOCIATIVE MEMORY ELEMENT. AS SUCH, MEMORY ACCESS TIME IS MINIMAL AND PARALLELISM IS MAXIMIZED. BY VARYING THIS BASIC STRUCTURE WITH REGARD TO INTERCONNECTION AND ADDITIONAL LOGIC, SPECIFIC STRUCTURES RESULT WHICH ARE CAPABLE OF EXTRACTING MEASURES OF SPECIFIC QUALITATIVE FEATURES

TWO SPECIFIC STRUCTURES ARE DESCRIBED WHICH EXTRACT, RESPECTIVELY, THE QUALITATIVE FEATURES OF TEXTURE REGULARITY AND LINE TREND. APPLICATIONS OF THESE STRUCTURES ARE PRESENTED. LOW-LEVEL SIMULATION AND PERFORMANCE ESTIMATES INDICATE THESE APPLICATIONS ARE VIABLE AND AMENABLE TO REAL-TIME OPERATION. SUGGESTIONS FOR THE DEVELOPMENT OF STRUCTURES WHICH EXTRACT OTHER FEATURES OR MULTIPLE FEATURES ARE DESCRIBED.

APPROVED:

[illegible]

Associate Professor of Electrical and Computer Engineering
in charge of major

[illegible]

Head of Department of Electrical and Computer Engineering

John C Ringle

Dean of Graduate School

Date thesis is presented June 11, 1986

Typed by Michael Bibby for Michael C. Bibby

AN ABSTRACT OF THE THESIS OF

Michael C. Bibby for the degree of Master of Science in Electrical and Computer Engineering presented on June 11, 1986 .

Title: Special Associative Preprocessing Structures for Qualitative Feature Extraction

Abstract Approved: V. Michael Powers
V. Michael Powers

Existing pattern recognition and classification algorithms in computer vision require vast amounts of computations on input data. As a result, memory access time is a critical parameter in system performance. Tremendous parallelism in structure and algorithm is required for the system to operate in real-time. A preprocessing structure for qualitative feature extraction which meets these system requirements is presented.

In general, the structure architecture consists of a cellular array of pixel-processors each containing an inherently parallel associative memory element. As such, memory access time is minimal and parallelism is maximized. By varying this basic structure with regard to interconnection and additional logic, specific structures result which are capable of extracting measures of specific qualitative features.

Two specific structures are described which extract, respectively, the qualitative features of texture regularity and line trend. Applications of these structures are presented. Low-level simulation and performance estimates indicate these applications are viable and amenable to real-time operation. Suggestions for the development of structures which extract other features or multiple features are described.

Special Associative Preprocessing Structures for Qualitative Feature Extraction

by

Michael C. Bibby

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed June 11, 1986

Commencement June 1987

3

Accesion For

NTIS CRA&I ✓
DTIC TAB ☐
Unannounced ☐
Justification

By
Distribution /

Availability Codes

Dist | Avail and / or Special

A-1

ACKNOWLEDGEMENT

I would like to express my appreciation to all the people who have encouraged and supported me during the preparation of this thesis. In particular, many thanks to Professor V. Michael Powers, my major professor, for his guidance, advice, and patience during the last two years. My success in this endevor can be largely attributed to his assistance.

Many thanks to other members of my Masters Committee: Professor W. J. Kolodziej and Professor B. E. Petersen, for their time and help during my study at the University. Thanks, as well, to Professor G. C. Alexander for his support in the formulation of my program.

I thank the United States Army and the United States Military Academy for their confidence in my success by funding my schooling.

My special thanks to my wife Cheryl. Without her constant loving support, encouragement, and understanding, I would not have reached this stage of my education. To my children, Michael, Matthew, and Lisa, who have missed their father, a special thanks as well.

Finally, I would like to thank my parents, Merl and Vi, for their encouragement and support during our stay in Oregon.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Special Associative Preprocessing Structures for Qualitative Feature Extraction

Chapter 1 INTRODUCTION

1.1 Computer Vision: An Overview

Man has dreamed of constructing intelligent automata for ages. Since the development of the Turing machine around 1950, the dream has been pursued primarily by workers in the field of artificial intelligence. Their goal has been to endow computers with information-processing capabilities comparable to those of biological organisms (Ballard and Brown, 1982). One of the basic goals of artificial intelligence is to enable machines to deal with sensory input.

"Computer vision largely deals with the analysis of pictures in order to achieve results similar to those obtained by man" (Levine, 1985). From (Ballard and Brown, 1982): "Computer vision is the construction of explicit, meaningful descriptions of physical objects from images." These definitions are exemplified in a simplified machine paradigm for computer vision as shown in Figure 1.1. Extracted from (Levine, 1985), the diagram consists of two computational stages. The first is concerned with "low-level" techniques and is the subject of this thesis. The second stage, picture interpretation, constitutes

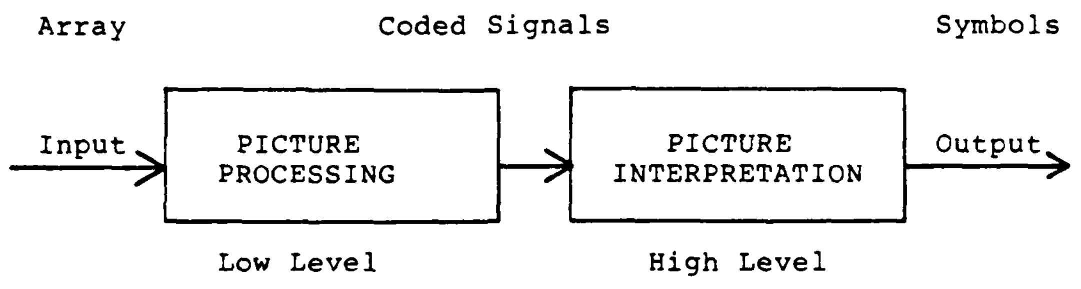


Figure 1.1 A Machine Paradigm for Computer Vision.

"high-level" processing. Here, input from the first stage is processed to produce a symbolic output which describes the input picture. Feedback paths which probably exist from the high-level to the low-level stages are ignored.

There exist several layers of analysis in the picture processing stage:

(1) Sensor Representation: Within this level are contained the functions of image sampling, quantization, and coding. More simply phrased, the concern is how the machine represents an image.

(2) Preprocessing: The types of activities in this level may include noise removal, restoration, enhancement, or other operations with regard to the image data.

(3) Feature Extraction: Coding of the picture data into quantities representative of qualitative features occurs in this level. Here, we are concerned with useful descriptive qualities of the input. For example, some common features that may be considered and symbolically

represented are color, texture, and shape.

These levels emphasize the nature of computer vision as being that of picture analysis. Synthesis of pictures, on the other hand, is commonly referred to as "computer graphics."

There are many applications of picture processing. Currently, the three most active areas, with respect to research, are biomedical image processing, remote sensing, and industrial automation (Levine, 1985). Table 1.1, extracted in part, from (Ballard and Brown, 1982), displays a more complete listing of computer vision applications.

Table 1.1 Examples of Computer Vision Applications

Domain	Tasks
Robotics	Identify or describe objects in scene. Industrial tasks.
Remote Sensing	Improve images, resource analyses, weather prediction.
Military	Spying, missile guidance, tactical analysis.
Astronomy	Improved images, chemical composition analysis.
Biomedical	Diagnosis of abnormalities, operative and treatment planning, pathology, cytology, karyotyping.
Chemistry	Analysis of molecular compositions.
Physics	Find new particles, identify tracks.

Recent developments in systems for computer vision are reviewed in some detail within the next two chapters.

1.2 Purpose

The bibliography of this thesis represents a sampling of the research which has been conducted in computer vision. Reading only a few of these references should convince the reader that modeling human vision with a computer system is, indeed, a difficult task. Even so-called simple approaches to this task require a background in mathematics, statistics, physics, optics, and electronics for basic understanding.

One premise of this paper is that a computer vision system should be highly parallel in structure and algorithm in order to conduct the massive amounts of computations required for real-time pattern recognition and classification. This premise is supported by literature reporting the development of highly parallel structures, such as cellular logic arrays, for computer vision implementations. It would seem that these implementations would also contain memory elements possessing inherent parallelism (associative memories). However, with only a few exceptions, this has not occurred.

The purpose of the research summarized in this thesis is to design and examine a style of architecture for computer vision which is (1) simple, compared to current implementations, and (2) highly parallel with respect to memory as well as structure. As a first step in our design we define the problem and establish a set of goals and objectives.

1.3 Goals and Objectives

Part of defining any problem requires setting limitations and making assumptions. Recalling Figure 1.1, we intend to design a structure which will accomplish preprocessing and feature extraction of the input picture. In accomplishing this "low-level" processing we will assume that the input to our structure has been "represented" by image sampling, quantization, and coding. In order to concentrate on concepts rather than details, we present our design at a structural level.

With this problem definition in mind we may classify our expectations for this structure into three categories of goals and objectives:

(1) Simplicity: A goal of our structure will be simplicity in architecture. Simplicity in hardware will limit cost. The structure should be "small" in scale so that an implementation may be possible on one VLSI chip. Additionally, the on-chip design should be "regular" such that component layout and interconnections are relatively simple. The design should minimize overhead in both time and hardware caused by too general of a design.

A final goal, considered under this general heading of simplicity, is in regard to chip function. The chip should simplify the input data (decrease the bandwidth) for high-level processing.

(2) Parallelism: Parallelism is a primary goal in that an increase in computational speed is thereby achieved. The basic structure must be parallel in architecture, utilize inherently parallel associative memory, and allow for pipelining where repetitive, sequential computations are required.

(3) Flexibility: Another primary goal of the structure is that it be versatile. The architecture should be able to perform a variety of operations upon the input picture data, thus enabling the detection or extraction of a variety of features. The structure should be applicable, as well, to performing standard computer vision preprocessing such as noise removal or enhancement.

The chip design should be such that the size may be expanded as technology develops. Until this time, however, the chips should have the capability to be "stacked" (placed together in a manner to allow for preprocessing of larger input pictures). In this regard, there should not develop a problem in the handling of border information.

At this point, we have introduced the subject of computer vision and set the stage for the design of an associative picture processing (or preprocessing) structure for qualitative feature extraction. The actual design will evolve during the next several chapters.

1.4 Chapter Outline

Together, Chapter 2 and Chapter 3 are basically a review of literature. In the former chapter the principles and concepts of associative memory and associative processing are reviewed. A basic model of an associative memory is presented. Various organizations for implementation of this type memory are discussed. A survey of current associative array processors and their application to computer vision completes Chapter 2. Chapter 3 reviews the principles and concepts of cellular logic arrays and cellular logic processing. Different array organizations are discussed with a summary of operations which can be performed and are applicable to computer vision. This chapter ends with a section which surveys cellular logic computers and their use in computer vision.

In Chapter 4, we marry the concepts of associative processing and cellular logic processing to develop an associative preprocessing structure. Initially, a general design of this structure is presented. Called a FES (for Feature Extracting Structure), the purpose of the structure is to preprocess input image data and extract a qualitative feature of the image. These features may be used in higher level pattern recognition and classification algorithms.

In Section 4.3 specific FES's are considered which will produce, as output, a specific feature of the input picture such as a "shape" or "texture" attribute. We present specific structures and algorithms for the extraction of the texture feature of "regularity" and the shape feature of "line trend".

In the final section of Chapter 4, multi-feature FES's are hypothesized. This structure appears to be capable of simultaneously producing a set of qualitative features. This implies that the FES approach is even more valuable in computer vision applications.

Applications of the FES are presented in varying levels of development in Chapter 5. In our first application, we use a FES adapted for extracting the texture feature of "regularity" as an aid to focusing for vision or photographic systems. By way of verification, we have modeled the focusing process and conducted experiments. Analysis of the experimental results leads to a conclusion that the FES is useful in focusing. Programs and experimental data for this focusing application are contained in the Appendix.

A second example application involves the use of a FES adapted for detecting the shape feature of "line trend". The data produced here may be used as an aid to picture orientation in a vision system. Although a verification is not conducted at a level which is comparable to the modeling and experiments performed for the first application, we conclude this application is viable as well.

In the last section of Chapter 5, we consider other seemirgly practical applications for the FES. Presented as a collection of promising ideas, this section highlights the potential of the FES.

In Chapter 6 we summarize the material presented in this thesis. The important concepts and conclusions are reviewed and we analyze the FES with respect to the goals and objectives set forth in this chapter. The performance of the FES is estimated with regard to cost, size and speed. Based upon these estimates we conclude that the FES is capable of real-time picture processing. In the final section, suggestions are made for future research in associative preprocessing structures for qualitative feature extraction.

The Bibliography represents more than the sum of the references within each chapter. Here, nearly all of the "experts" in the field of computer vision have been represented by some of their most recent works.

Chapter 2 ASSOCIATIVE MEMORY

2.1 Introduction

It has been at least 25 years since the conception of the idea of associative processing. In the past, implementation problems and high hardware costs of associative memories have limited associative processing to small and highly specialized systems. However, advances in computer technology and the development of new architectural concepts have made the design of larger, more flexible associative processing systems possible (Yau and Fung, 1977).

One premise of this thesis is that associative processing is an important concept applicable to computer vision systems. Such systems require tremendous speed and parallelism to operate in a real-time environment. Due to the inherent parallelism of associative memories, it is realistic to expect that future vision systems will possess associative memory as an integral component. With the rapid development of large-scale integrated-circuit (LSI) technology, implementation costs of these memories have been greatly reduced. It is anticipated that associative memories will be used extensively in a variety of future processing systems (Foster, 1976). For example, associative memory has an established role in memory management systems. A new scheme which uses sizeable associative memories to increase memory management performance is detailed in (Thakkar, 1986).

Before proceeding further, some basic definitions are required. An _associative memory_ is defined as a storage device which stores data in cells. These cells can be accessed or loaded on the basis of their contents. An _associative processor_ utilizes an associative memory as an essential component. Data transformation operations, both arithmetic and logical, can be performed over many sets of arguments with a single instruction. Finally, an _associative array processor_ is a single-instruction, multiple-data (SIMD) structure comprised of multiple associative processors (Hwang and Briggs, 1984).

The value of using associative memory instead of random access memory (RAM) can best be seen by comparison. RAM requires a word "address" to access stored data. This address must be provided, either directly or indirectly, by the user. In all cases, accesses are done in a sequential manner and only one word of memory may be accessed at a time. In associative memory words are accessed based upon their contents. As such, no user supplied address is required. Words are accessed in parallel, allowing for multiple accesses using a single instruction. Associative memory is best used in non-numeric data processing. In fact, when conducting floating-point operations, the use of RAM achieves greater performance.

The most significant advantage of associative memory use lies in its ability to perform parallel search and comparison operations at a rate not possible using RAM. This is shown via an example in Section 2.2.1. The price paid for this remarkable speed advantage is that of hardware cost. Recent estimates show associative memory to be approximately 1.4 to 1.6 times more expensive than comparable RAM (Foster, 1976).

2.2 Associative Memories

Having introduced the concept of associative memory, we now proceed to examine its structure in some detail. A basic model is presented and the operation of this structure is shown by an illustrative example. Associative memory organizations are discussed followed by an example associative memory implementation.

2.2.1 A Basic Model

The structure of a "basic" associative memory is given in Figure 2.1. In this model, an associative memory is shown to be an array of bit-cells. The entire array, then, consists of n words each having m bits. Each cell in the array contains a one-bit memory element and assaciated comparison logic gates. This logic-in-memory is what enables parallel read or write operations to occur in the memory array.

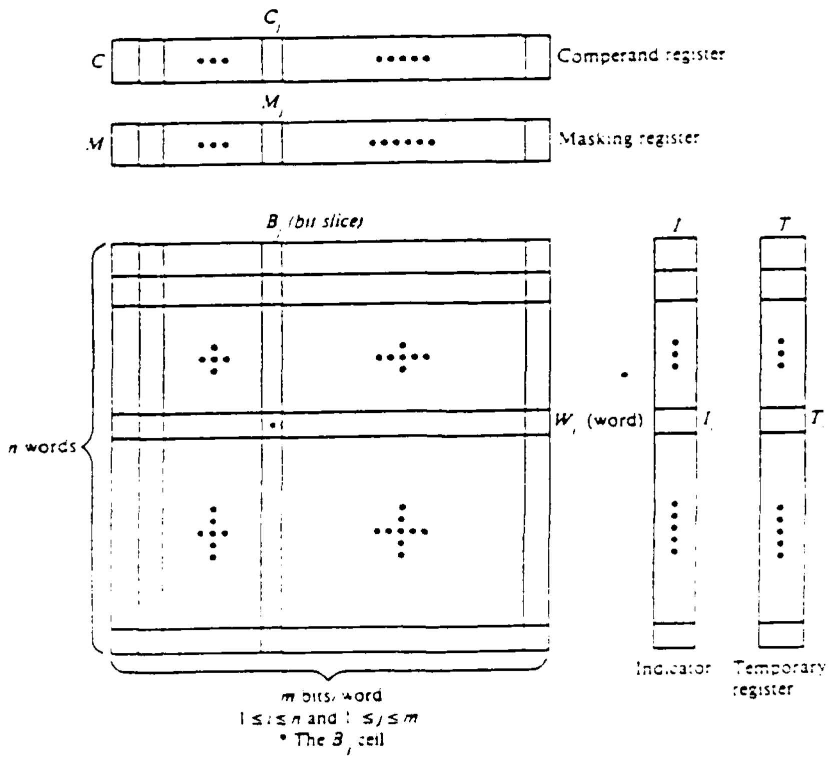


Figure 2.1 Associative Memory Model (Foster,1976)

In a parallel search/comparison operation, comparisons and masking are involved in the execution. The comparand register is used to hold the "key" operand being compared or searched for. The masking register is used to enable or disable the bit-slices required for the operation. A bit-slice is a column of bit-cells of all words at the same position. A bit position disabled by the pattern in the masking register is said to be "masked". The indicator register (and one or more temporary registers) are utilized for holding the current (or previous) match patterns. With this structure in mind we consider the example depicted in Figure 2.2.

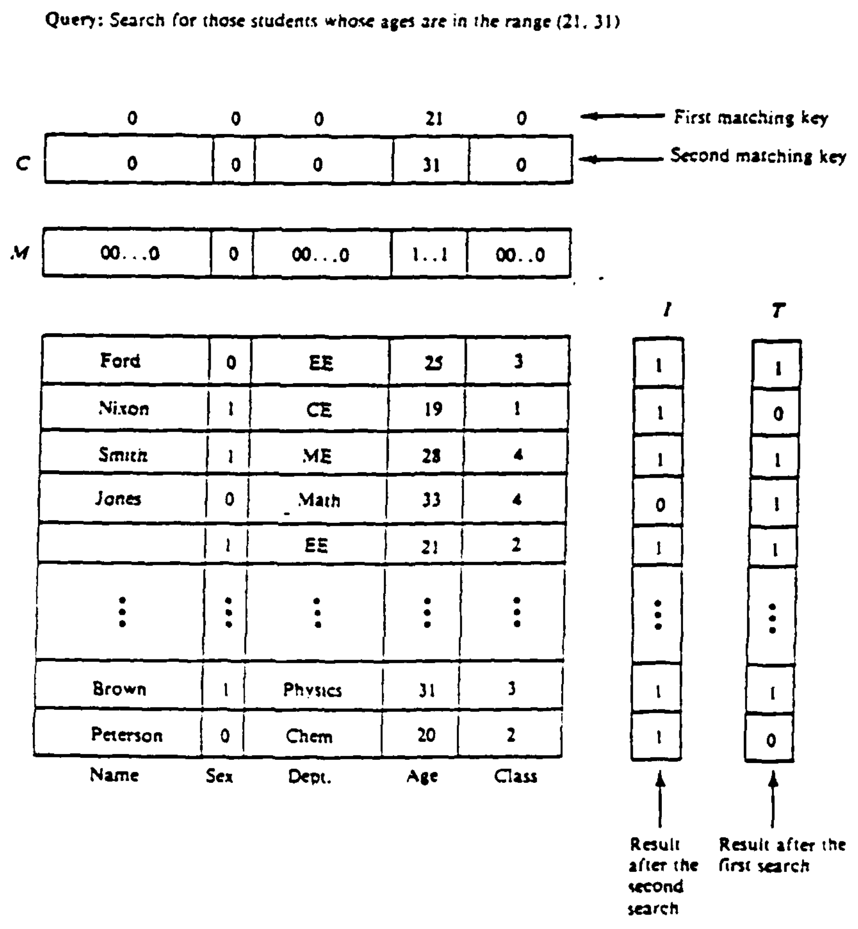


Figure 2.2 An Example Search Operation Using Associative Memory. (Foster, 1976)

Imagine accessing an unsorted "student file" and requesting the names of all students between 21 and 31 years of age. If this were done using RAM it would require an access to every entry within the file. Figure 2.2 depicts the same search but utilizing associative memory. The masking register is set such that only the age field will be involved in the comparison process. The first age, 21, was latched into the comparand register, memory accessed and results appear in the indicator register. (The comparison logic, avoided here for simplicity, will be described later.) After transferring this data to a temporary register the same process is performed using the second age, 31. Finally, the temporary and indicator registers are "anded" to provide the requested information. Note that only two associative memory accesses are required for the entire search operation.

2.2.2 Organizations

Associative memories can be classified into four categories based upon their organization: fully parallel, bit-serial, word-serial and block oriented. Some extensions or modifications to these basic organizations, however, have been implemented or proposed. The primary emphasis here will be devoted to the fully parallel and bit-serial organizations.

The word-serial system does not utilize parallel word operations. Considering an associative memory as a "black box" device, this organization qualifies as an associative memory due to its high speed. This device essentially represents a hardware implementation of a simple program loop used for a linear search. (Yau and Fung, 1977) The organization is efficient as it requires only a single instruction to conduct a comparative search and because data rates of circulating memories are much higher than those of RAM.

Block oriented organizations are utilized for applications such as information storage and retrieval with regard to mass storage. Proposals of this type involve the use of high-speed drums or disks with associated "logic-per-track" devices (Yau and Fung, 1977).

In a fully parallel organization the comparison process is performed in parallel by word and by bit. All bit-slices not masked are simultaneously involved in the operation. Logically complex word match tags are used. This organization is shown in Figure 2.3 (a).

Figure 2.3(b) depicts the bit-serial organization. Here only one bit-slice (not masked) at a time is involved in the comparison. The figure is misleading in that it shows the bit-parallel organization to be seemingly less complex than the bit-serial organization. Actually, the word match networks in (a) contain much more logic than the word logic blocks in (b). Often referred to as a compromise between the fully parallel and word-serial organizations, the bit-serial organization is implemented in most existing associative array processors (Foster, 1976).

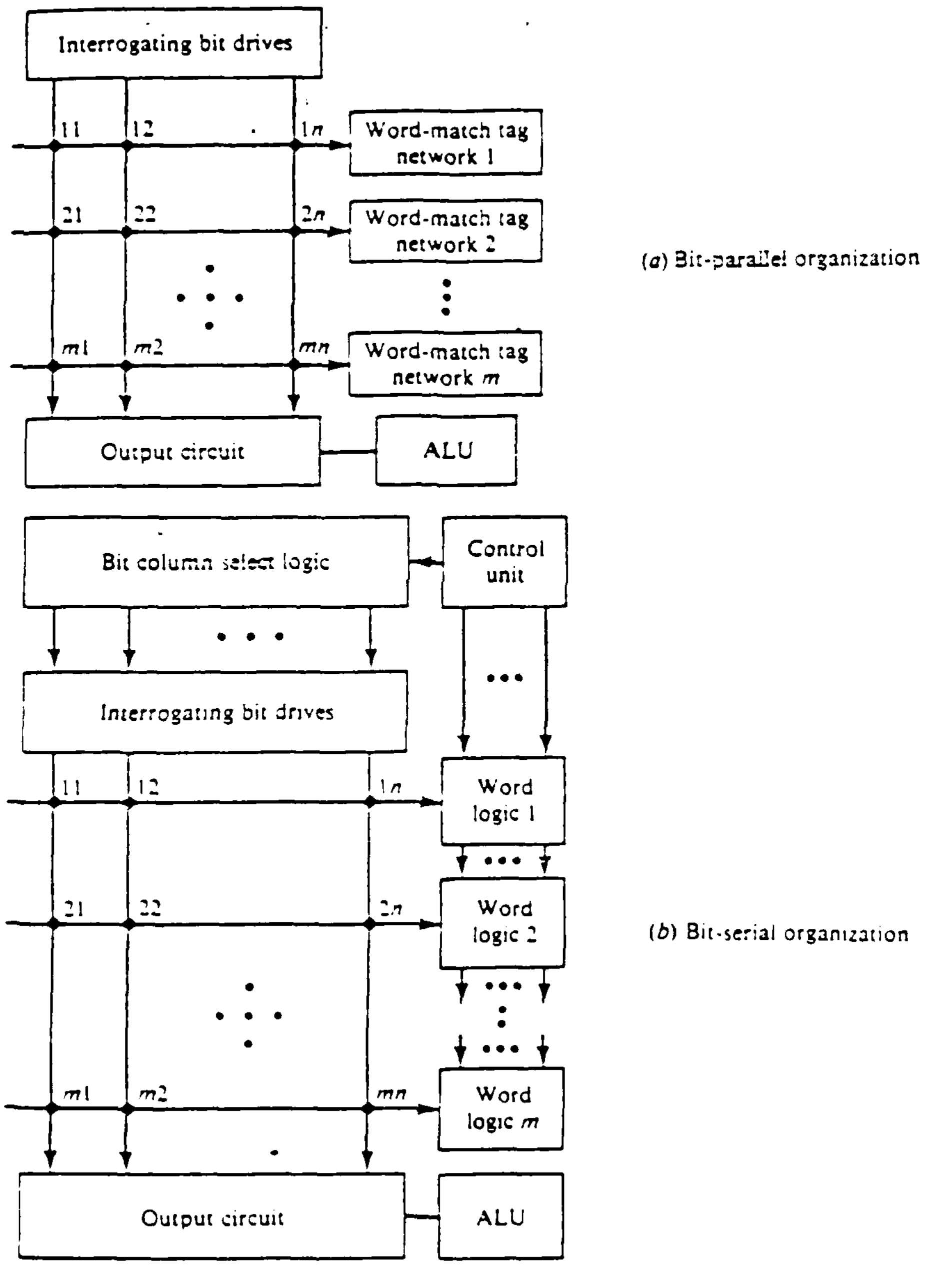


Figure 2.3 Associative Memory Organizations (Foster, 1976)

2.2.3 An Implementation

In this section an example implementation of an associative memory is presented. By redrawing the basic model of Figure 2.1 with regard to function, the block diagram of Figure 2.4 results. We now examine each block in turn.

The comparand, mask and output registers need be of the same size as the word size in the memory array. The

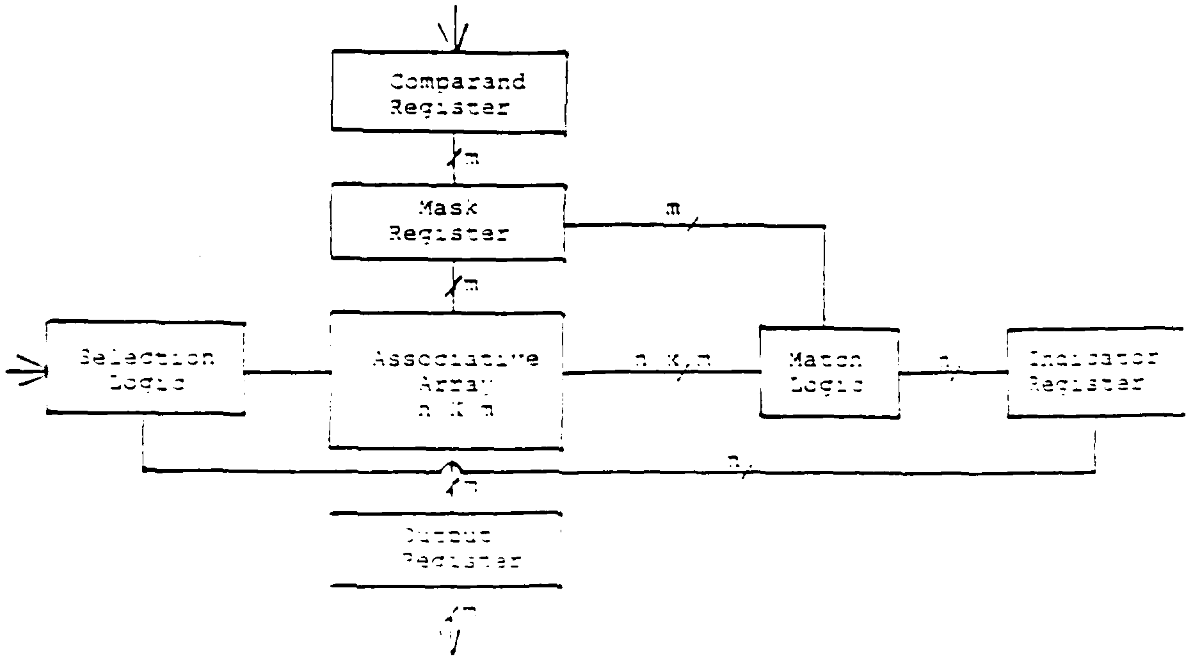


Figure 2.4 A Functional Diagram of an Associative Memory

comparand and masking registers are connected such that their "anded" value enters the memory array, thus, masking occurs. Additionally, to insure that masked bit-slices do not affect the result the masking register is connected to the match logic as well. (For example, to preclude matches of a comparand "0" when the mask is "0". See match logic which follows.) Upon completion of an operation data may be latched into the output register as directed by the selection logic and match results.

Briefly, the selection logic may consist of multiplexers, encoders, decoders and other logic devices. This logic allows for external accessing and loading of the memory array. It is important to realize that actually getting the desired data from an associative memory is not any faster than RAM. (Recall that the associative memory advantage is in comparison and search operations.)

The memory array is composed of an array of basic memory cells, one of which is shown in Figure 2.5. Each cell is capable of holding one bit of information.

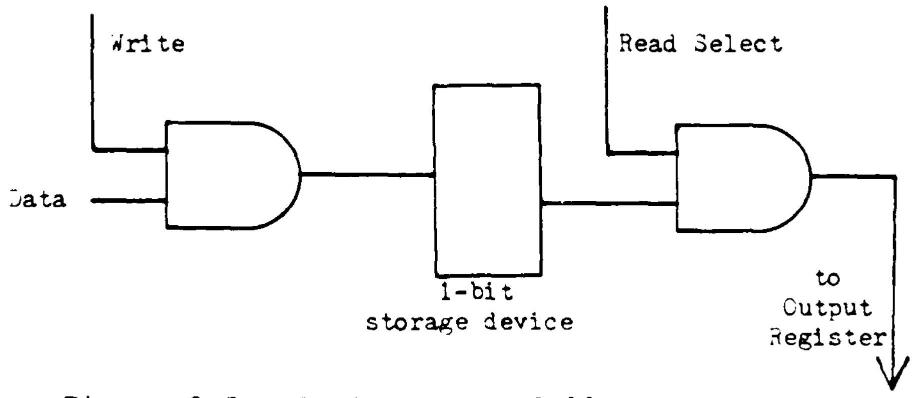


Figure 2.5 Basic Memory Cell

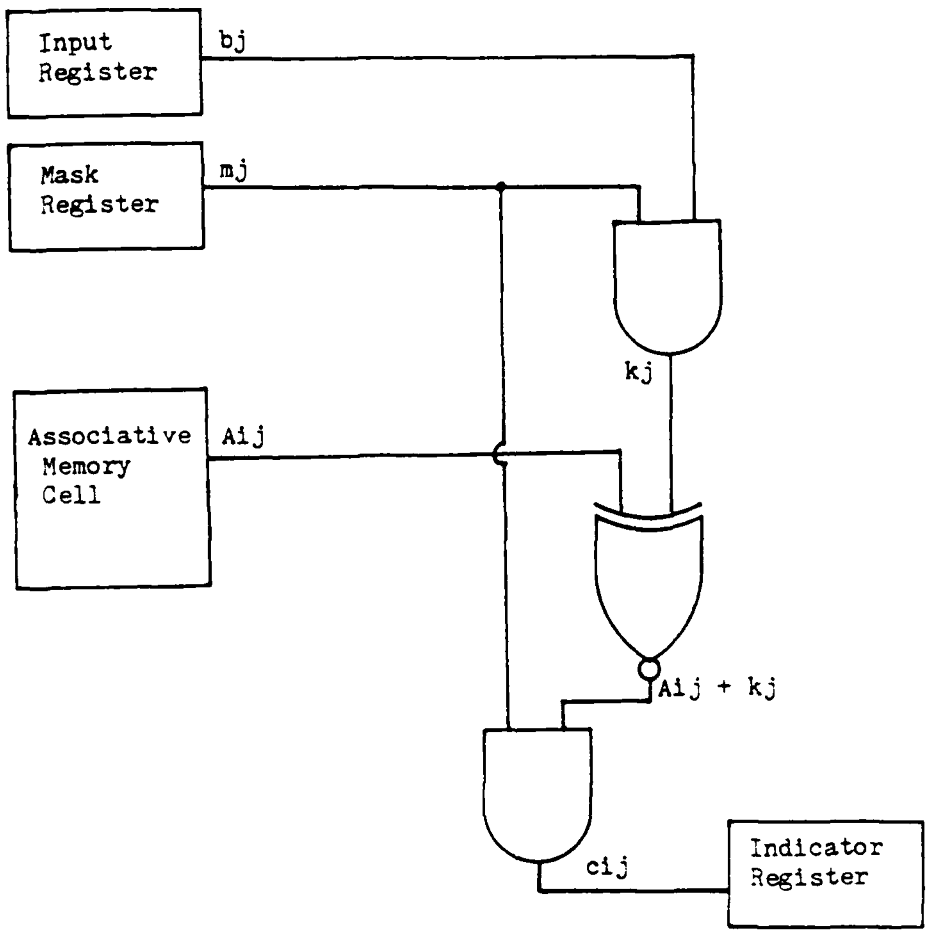


Figure 2.6   Structure of a 1-bit Match Logic

The match logic is shown in Figure 2.6 for a one bit, one word memory implementation. The output of the match logic feeds the indicator (or match) register. The indicator register must therefore be of length equal to the number of words in the memory array. A result of one in the indicator register would indicate a match of the associated word in memory. A zero result indicates no match.

For implementations of larger associative memories, the match logic is much more complicated than depicted in Figure 2.6. For example, a large memory was indicated in Figure 2.2, the "student file" illustration. The match logic required here must be able to perform the comparisons of "less than" and "greater than" as well as equality.

2.3 Associative Array Processors

In this section a few examples of existing associative array processors are reported. The intent is to show that these processors actually exist and could be or have been used in computer vision applications. The information presented has been extracted from (Foster, 1976) and (Yau and Fung, 1977).

The Parallel Element Processing Ensemble (PEPE) utilizes an almost fully parallel associative memory organization. "Almost" refers to the fact that rather than all bit-slices being compared simultaneously, only a group of the unmasked bit-slices are compared at the same time. PEPE was designed as a special-purpose computer to perform real time radar tracking in an antiballistic missile environment. Other than having a different sensor system (radar instead of visible light) PEPE closely resembles a computer vision system. Many of the data operations conducted on radar images by PEPE mirror computer vision operations.

The STARAN, developed by Goodyear Aerospace, utilizes the bit-serial associative memory organization. STARAN possesses high-speed I/O capabilities and can interface easily with conventional computers. Termed a general purpose associative array processor, some of STARAN's many practical applications have been matrix computations, air traffic control, signal processing and computer vision.

The OMEN computers, developed by Sanders Associates, possess the bit-serial organization. The associative memory array of 64 words by 16 bits is augmented by processing elements (PE). The PE's are connected such that simultaneous arithmetic operations can be performed on several bit-slices at one time.

The Associative Linear Array Processor (ALAP) is an exciting development because it has been developed on a single LSI wafer. Another recent development by Honeywell is the Extended Content Addressed Memory System (ECAM). Honeywell has developed a superchip technique which allows for the production of associative memory arrays up to the billion-bit range. The potential of these two systems for computer vision applications is enormous.

Chapter 3 CELLULAR LOGIC ARRAYS

3.1 Introduction

In the previous chapter, we discussed associative array processing and mentioned applications to computer vision. Each computer system reviewed utilized associative memory. However, each system varied in its architecture. In this chapter we will review the literature regarding a specific computer architecture: cellular logic array computers. The references listed in the Bibliography, when combined, fairly represent the state of the art with regard to computer vision. Based upon these sources, it seems apparent that the most popular and promising computer architecture for computer vision is a cellular logic structure.

This structure is not a new idea. S. M. Unger suggested using a two-dimensional array of processing elements as a natural computer architecture for image processing and recognition over 25 years ago. (Rosenfeld, 1983) In Section 3.2 we review the principles of image processing using the cellular logic structure. Emphasis is on the two-dimensional processor array where each processor is responsible for one element of the image (a pixel) and neighboring processors are responsible for neighboring pixels. By using hardwired communications between neighboring processors, local operations can be performed or local features can be detected in a highly parallel manner. The section basically summarizes the concepts presented in a special computer vision issue of Computer (January, 1983) In particular, the article by Azriel Rosenfeld is summarized in detail.

In the last section of this chapter, we review current implementations of cellular logic processing. In this review, the diversity of the basic cellular array architecture is most interesting. Recent IC technology has had profound effects in these implementations and has expanded the number of practical applications for cellular arrays with regard to computer vision. (Preston, 1983)

3.2 Cellular Arrays

A cellular array is a two-dimensional array of processors. Each processor can communicate directly with its neighbors in the array as shown in Figure 3.1. For simplicity, the figure only depicts connections between each processor and its four vertical and horizontal neighbors. (Note that "border" processors have less than four neighbors.) We assume that each processor is capable of receiving distinguishable messages from its neighbors and that it can send different messages to each neighbor.

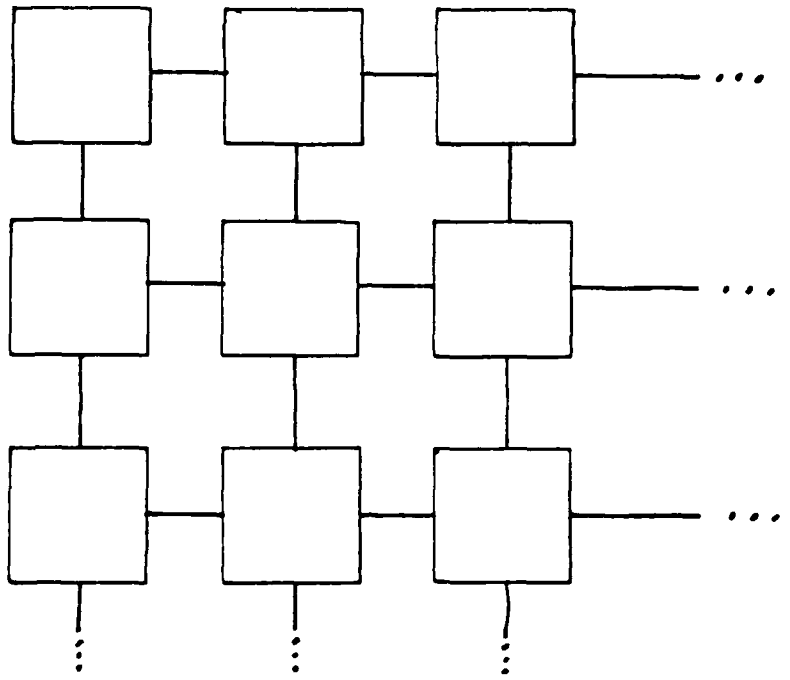


Figure 3.1 A Rectangular, Two-Dimensional Cellular Array

For image processing, each cell (processor) gets the value of an element of the picture (a pixel) as input. If the array is smaller than the image, the image may be processed a "block" at a time with special consideration given to block boundaries. Alternatively, providing each cell has the memory capability, each cell could receive a block of pixels as input. It should be noted, however, that all cellular arrays constructed to date possess very little memory. Therefore, we only consider the single pixel input alternative.

The principle advantage of this arrangement is that the processors can operate in parallel within their neighborhood. Local operations can be performed on the entire image in an amount of time independent of image size. Example basic operations and algorithms are presented in Section 3.2.2.

Cellular logic arrays need not be rectangular in form as shown in Figure 3.1. In fact, two-dimensional arrays are quite expensive and the largest in production are only about 100 x 100 in size. Due to this, other organizations have been proposed.

3.2.1 Organizations

Other than the two-dimensional cellular array discussed previously, there are structured cellular arrays which require much less hardware but at a cost of decreased performance. One dimensional cellular arrays, or "cellular strings," have been used to process two-dimensional images by scanning row-by-row and operating in parallel on each row. Local operations can be performed by storing several rows to obtain the needed values of neighbors. This procedure requires $O(n)$ time for an $n \times n$ image at a cost of $O(n)$ which is attractive compared to the $O(n^2)$ hardware required by the two-dimensional organization. Note that both of these organizations can be easily laid out on a chip as no connections need cross.

Another alternative organization is the cellular pyramid. Here each cell is connected to two or more "brothers" in its own string, a "father" in the string above, and to its "sons" in the string below. For a binary tree this results in a cellular triangle as shown in Figure 3.2. Quadtree and Octree decompositions have also been implemented. These structures use less hardware than the two-dimensional structure, but can perform "counting" operations in O(log n) steps. Due to connections needed for local operations, however, chip layout poses some problems. These structures have been used effectively for shape analysis and pattern recognition applications. (Chandhuri, 1985)

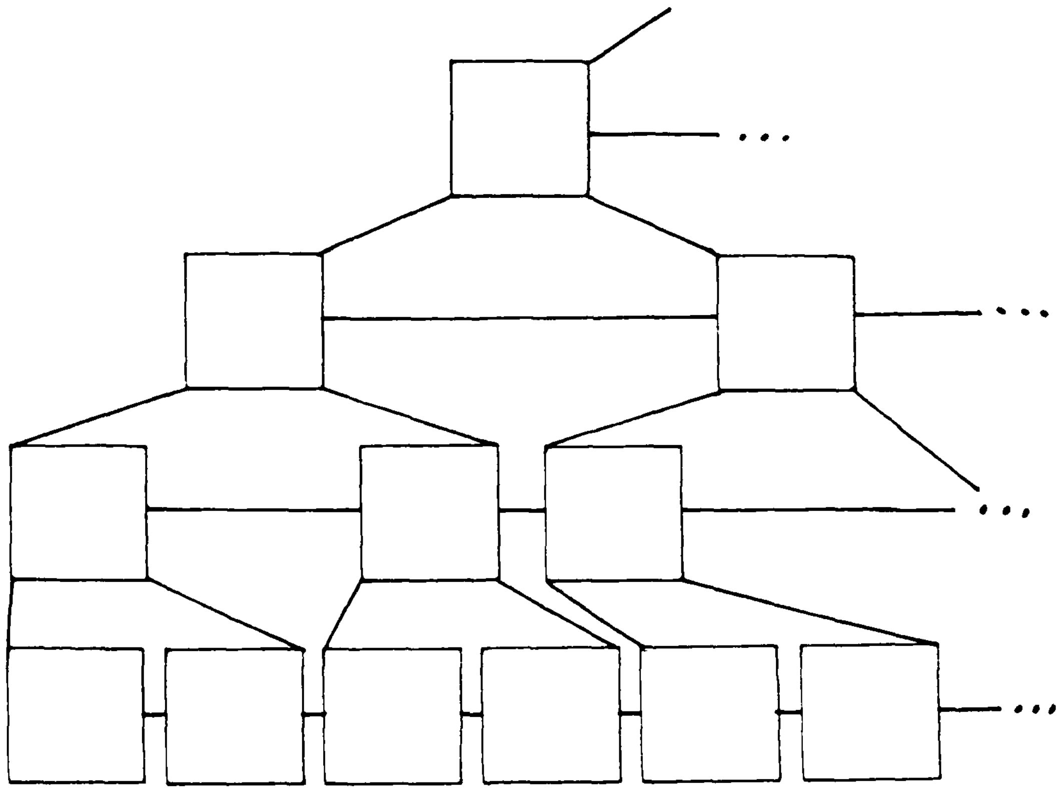


Figure 3.2 A Cellular Triangle

A final organization is termed cellular graphs. In this structure cells are considered "nodes" of a graph. We allow the "arcs" which connect the nodes to be arbitrary. This type of "reconfigurable" structure has proved to be useful in segmentation operations. The organization is most effective when it has the ability to modify itself (during processing based on the input) from the initial configuration. This structure would be best used for image processing at the region level.

3.2.2 Operations

As mentioned before, the principal advantage of using cellular arrays is in local operations where each processor can operate in parallel. At this point, we consider operations using the rectangular, two-dimensional cellular logic array. As a very simple example, suppose we wanted to average each pixel with its neighbors (a "smoothing" operation). Ignoring the special treatment required for border pixels, this could be accomplished by having each processor, in parallel, execute the following sequence of instructions:

1. Add own value into a previously initialized register.
2. Read and add all neighbor values to register.
3. Normalize. (Divide by total number of pixels in

neighborhood.)

4. Replace own value by register contents.

Given a repertoire of such instruction sequences, a wide variety of local image operations can be performed. Each cell could store these as microprograms or the instructions could be "broadcast" to each cell as required.

In contrast, consider a conventional computer, having only a single processor, attempting the same type of operation. Here the averaging process can proceed only one pixel at a time. Thus, the time required is directly proportional to the image size. For an "n x n" image, using a rectangular cellular array increases the amount of hardware required by a factor of n^2 but also decreases the time required by a similar factor.

A wide variety of image processing algorithms have been developed for cellular arrays. A representative set of such algorithms are briefly described here:

(1) Local Operations: These are operations like the averaging example previously described. The local image property depends upon only the input values of a small set of the pixel's neighbors (and possibly the value of the pixel itself.) Cellular arrays can compute local properties in parallel where the amount of time required is independent of the image size. Local properties are widely used in image processing for smoothing, deblurring, edge detection, texture analysis, etc.

(2) Value Counting (histogramming): Here we desire to count the number of times a given value occurs in an image. For example, we can shift and sum these counts left in each row and store the results in the leftmost cells. Next, we shift the sums in the leftmost cells upward to a final summation in the uppermost left cell. The total time is proportional to the width and height of the image. This method can be used to construct the grey-level histogram of an image in O(n) time.

(3) Moments and Transforms: The moment value of an image or the coefficients of the Fourier (or other) transform may be computed using cellular arrays. Here we multiply the image pixelwise by the appropriate basis matrix and sum the results (Fourier Transform). Each summing step and "broadcasting" of basis matrix can be accomplished in O(n) time. The multiplication is done in a single parallel step (Rosenfeld, 1983).

(4) Connected Components: Given an array of 0's and 1's, we can define local "shrinking" or "expanding" operations. In the former, we collapse each component of 1's to a smaller size and in the latter the operation is reversed. Time, of course, is dependent upon the component size and the degree of change required.

(5) Region Properties: Properties of a region such as its area or perimeter can be easily computed using cellular arrays. From these properties region shape features such as compactness, elongation, thickness, etc.

can be determined.

3.3 Cellular Logic Computers

Cellular logic computers, under development since the 1950's, are now in use for image processing in hundreds of laboratories world-wide. (Preston, 1983) In this section we will briefly review several of these computers. According to Preston, cellular logic computers can be divided into four major architectural types, as follows:

(1) Single-element Subarray Machines: Having only one processing element, these machines appear to represent an obsolete architecture. Examples of this type of architecture are the Cellscan, GLOPR and BIP computers. These machines were the first successful cellular logic computers constructed in the 1960's.

(2) Multiple-element Subarray Machines: Computers of this type are characterized by possessing multiple processors which operate under the control of a control processor. Examples of computers having this type of architecture are the diff3, PHP, PHP II and PICAP II. The upper-bound for these systems is on the order of 500 million elements per second and is mainly limited by memory data rates.

(3) Pipelined Architectures: This type of architecture involves multiple processors as in (2). Here,

however, each processing stage feeds the next in a linear chain of processors. An example of a current pipelined cellular logic computer is the Cytocomputer. This computer uses eight pipelined levels in processing input data. An architecture of this type is fully utilized only when the number of program steps is equal to the number of processing elements.

(4) Full-array Processors: This category can be distinguished from the others by the large number of processing elements utilized. The processors are tightly connected to each other and operate in concert. The MPP contains 128 x 128 processing elements. Other examples of this architecture, such as the Illiac-IV, STARAN, and CLIP4, contain a smaller number of processing elements. In image processing using these architectures, several significant problems exist. For iterative operations the "border problem" is severe, yielding large amounts of overhead. Problems also occur in chip design when many processors are integrated on a single chip. Specifically, the number of pins per chip becomes unmanageable as the number of processing elements is increased.

Preston's conclusion is that despite advances in VLSI design, the multiple-element subarray processors or pipelined processors offer the greatest flexibility, economy, speed and programming advantage compared with the other architectures. With sub-nanosecond RAM's on the horizon for use in these machines, he predicts abandonment of full-array systems. Some current image analysis computers having the architectures discussed are summarized in Table 3.1. (Hwang and Fu, 1983)

Table 3.1 Summary of Cellular Logic Computers

Machine	Architecture	Applications
Illiac-IV	SIMD, 64 PE's in 8x8 mesh	Landsat images, radar signal processing
STARAN	SIMD, 256 PE's in each of 32 arrays	Image magnification and convolution, cartography
CLIP4	SIMD, 96x96 PE's	Image processing and feature extraction
MPP	SIMD, 128x128 PE's	Landsat image processing
Cytocomputer	Pipelined, 88/25 processor stages	Biomedical image processing
GOP	Pipelined, 4 arithmetic	Image processing and pattern classification
Systolic Processor	Systolic pipeline	Image processing, FFT
PICAP II	MIMD/SIMD	Image processing, Computer graphics
FLIP	MIMD, 16 processors	General image processing and pattern recognition
ZMOB	MIMD, 256 microprocessors	Artificial intelligence and image processing

Chapter 4 STRUCTURES FOR FEATURE EXTRACTION

4.1 Introduction

In Section 4.2 we present the basic structure of our Feature Extraction Structure (FES). Using this basic structure, we then derive two specific structures for the extraction of two different qualitative features in Section 4.3. In particular, we examine the texture attribute of "regularity" and the shape attribute of "line trend" as specific qualitative features. An integrated structure for simultaneous multiple-feature extraction is discussed in the final section of this chapter. In Chapter 5 we discuss applications of these structures. The remainder of this introduction provides additional background and development for the FES design.

In the previous two chapters we have reviewed the concepts of both associative processing and cellular logic processing. Associative processing possessed the distinct advantage of speed due to the inherent parallelism of the associative memory. Speed was the primary advantage presented for cellular logic processing as well. This was due to the spatial parallelism of its structure. The primary cost for this speed, in both cases, was reflected in the hardware required for implementation. In the 25 years since the conception of both of these ideas, however, advances in VLSI design have dramatically decreased hardware costs. This trend is expected to continue, and as such, it is anticipated that larger associative memories and sizeable cellular logic structures will be economically feasible in the foreseeable future.

Cellular logic was also presented as the "natural" structure for computer vision in Chapter 3. By augmenting this structure with associative memory it would seem that a more efficient system for computer vision would be possible. Several existing associative array processors were reported in Section 2.3. The major hypothesis of this thesis is that the style of architecture resulting from the "marriage" of the concepts of associative and cellular logic processing is the ideal style of architecture for computer vision systems.

A comparison of the major implementations (processors and computers) presented in the last two chapters, yields only one large-scale implementation which possesses associative memory and a cellular logic construction. This, of course, is the STARAN computer system. The STARAN, however, was not built specifically for computer vision but rather as a general purpose system to be used experimentally on a number of practical applications. Additionally, the STARAN system, which possesses 32 arrays of 256 PE's, is much too massive in size and requires an inordinant amount of control, to represent an ideal computer vision system.

It is interesting to note that Preston did not mention or reference the STARAN system in his article (Preston, 1983). Recall from Chapter 3 that his conclusion was that full-array processors may be abandoned due to border problems, the number of chip pins required, and the overhead required to process an image. As a result of Preston's analysis, we have placed special consideration in the design of our structure to avoid these problems.

In the next section we present a structure capable of performing picture processing. The output of such a processor is a quantized measure of a qualitative feature which may be used for pattern recognition or classification. The architecture is cellular logic and the structure uses associative memory. As such, the structure is small in size and scope compared to STARAN and specifically designed for computer vision applications. The problems of such structures, which were presented by Preston, are solved in the design of our structure.

Similar work in this area has been done. In (Smith, 1983) an image processor which handles 256 pixels simultaneously is described. Called SCAPE (for Single-Chip Array-Processing Element), the chip is basically a variation on the CLIP-4, with bit-serial communication between chips such that only 24 pins are required for the package. In operation, the associative memory logic selects pixels upon which data transformations may be made. Additionally, the array structure is programmable. This chip is one of the most complex yet developed for image processing tasks, containing up to 140,000 transistors in a 6 millimeter square (Smith, 1983).

In our design, the associative memory is utilized in a different manner. Rather then using the associative memory to select pixels for impending operations, we will use the memory for storing characteristics of the preprocessed image. In the process of feature extraction we are able to access, in parallel, these characteristics by content. Additionally, our design is much simpler since only feature extraction is required rather than complete image processing.

4.2 A General Structure

In this section we present the basic structure for FES. (We assume FES will fit on a chip the size of SCAPE since FES is much less complicated as far as hardware.) A block diagram of the FES chip is depicted in Figure 4.1. Easily seen from the figure are the facts that FES consists basically of a rectangular cellular logic array and associative memory. We now describe each component of FES.

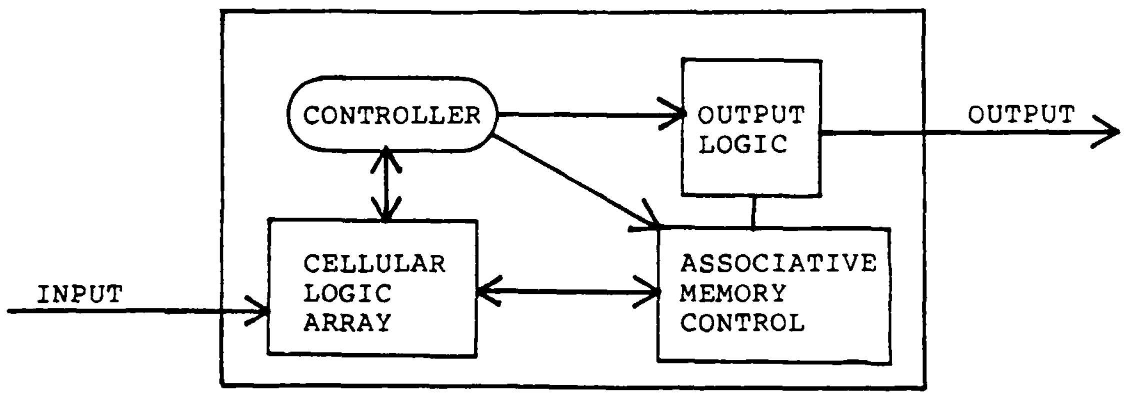


Figure 4.1 Block Diagram of the FES

The input consists of six-bit grey-level values. We have chosen this size because it is common in other implementations. As discussed earlier, in Chapter 1, we assume the sensing and digitization has been accomplished elsewhere. The input is received in a serial manner for reasons mentioned later.

The Controller is the programmable brain of the chip. It controls the timing of operations conducted in the other components. Instructions are "broadcast" to the cells of the array from the controller. These instructions will vary with the specific feature to be extracted. Keep in mind that Figure 4.1 is a model of a generalized FES. It is not adapted to extract a specific qualitative feature.

The Output Logic block will usually contain a microprocessor. (In implementation this may be the same processor as in the Controller.) Its purpose is to accumulate and output the result of the operations conducted on the chip as a feature. Like the controller, the internal structure of this element depends upon the specific feature required.

The Associative Memory Control contains the comparand, mask and indicator registers, as well as the majority of the match logic required for the chip to possess an operative associative memory. Word size and other logic considerations will vary in this block dependent upon the specific feature. The associative memory cells, themselves, are colocated with the cellular logic cells.

The Cellular Logic Array is redrawn in more detail in Figure 4.2. Here we see the rectangular architecture of FES. The array is 20 x 20 cells in size. The interior 16 x 16 cells are bordered to indicate that these cells possess an associative memory element. The pictorial input is thus 20 x 20, but only 16 x 16 cells partake in image processing by the FES. The border cells then simply represent digitally that area of the image and provide the information to neighbors within the interior as required.

We have selected this size for the FES because we feel that implementation on a single chip is possible. This is based upon the success of implementing the SCAPE (of comparable size) on a single chip, as was reported earlier.

The purpose of the border of cells may not be intuitively obvious. The 144 cells comprising the border represent considerable overhead. In this case, however, we feel the benefits incurred outweigh the additional cost in

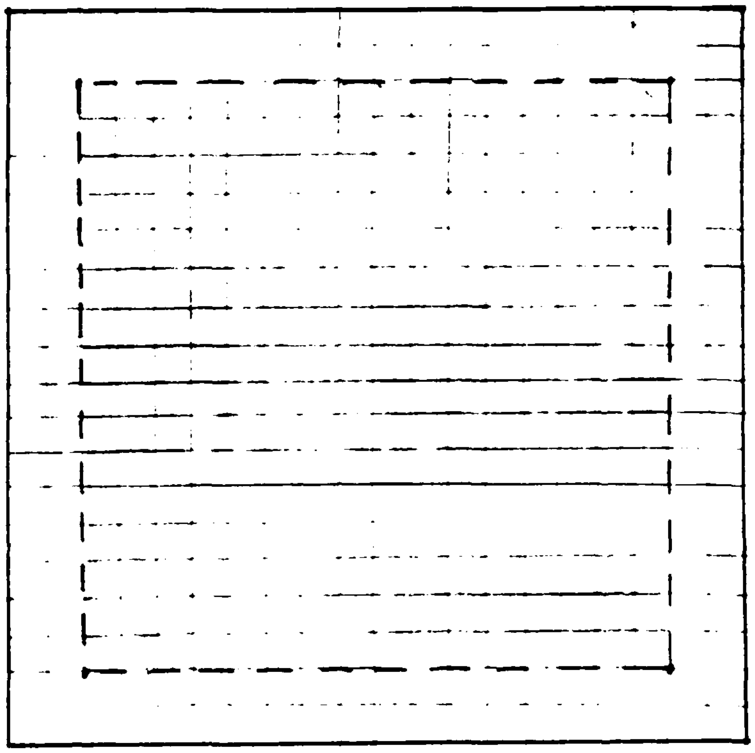


Figure 4.2   The Cellular Logic Array in FES. Interconnections are represented by proximity. Interior cells contain associative memory elements.

hardware. Operationally, the border allows for accurate data processing at the edges of the 16 x 16 array. Most computer vision preprocessing operations, such as those described in Chapter 3, require "pixelwise" computations within a neighborhood of pixels. Our structure, therefore, will allow for up to 5 x 5 sized neighborhood operations. If we want to "stack" FES's so that larger pictures may be processed, we discover that extensive interconnections are not required between chips. This overlapping essentially reduces the required pin count of the chip by at least 64. The pin count is also reduced due to the fact that input is serial. Thus, the pin count of this chip is not unmanageable. Finally, a border cell is much more simple than an interior cell. It has no processing function and contains no associative memory element. Thus, this overhead is not as significant as it appears.

A block diagram of an interior cell of the array is depicted in Figure 4.3. Valid input is latched into the input register, an internal PE register, as instructed by the Controller. The processing element is then able to perform arithmetic/logical operations on this data and input data from neighboring cells as directed by the chip Controller. The particular processed data (dependent on feature) is then stored in the associative memory element.

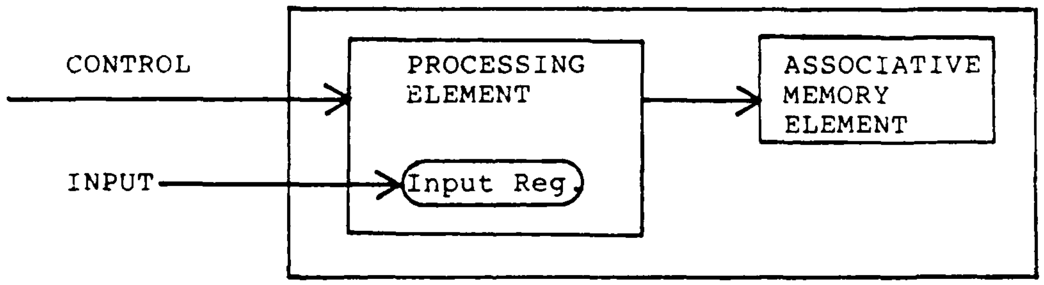


Figure 4.3 An Interior Cell in the FES Cellular Logic Array

The FES chip incurs some overhead due to the serial loading of its cells. But, the chip has a manageable pin count and when implemented together with other chips no border problems are manifested. At this point, the "general structure" may seem ambiguous or unclear. This is due to the immense flexibility incorporated into the structure. In the next section we will examine specific structures for specific feature extraction in order to illustrate the power of this structure.

4.3 Specific Structures

In this section we adapt the general structure of FES to a particular purpose. This will be accomplished in some detail for two features: the texture attribute of "regularity" and the shape attribute of "line trend". Each section ends with a summary of other features which may be extracted using similarly adapted FES's. Example applications of these specific structures will be covered in Chapter 5.

4.3.1 "Texture" Structures

Despite the tremendous amount of research that has been conducted, the field of texture analysis in computer vision is a highly controversial and ill-defined subject. According to Levine: "At present there does not exist a generally accepted model for texture" (Levine, 1985). And "the notion of texture admits to no rigid description" (Ballard and Brown, 1982). In this thesis we use the word "texture" rather loosely, sometimes in reference to a particular textural feature, and at other times to represent the sum of all texture attributes.

Much work has been done in qualitative and quantitive analysis of texture. "Texture can be qualitatively evaluated as having one or more of the properties of fineness, coarseness, smoothness, granulation, randomness, lineation, or being mottled, irregular, or hummocky." (Haralick, 1979). Levine discusses how to quantify these concepts mathematically. The analysis of texture from this viewpoint is sketched in Figure 4.4, which has been included to emphasize the complexity of this field.

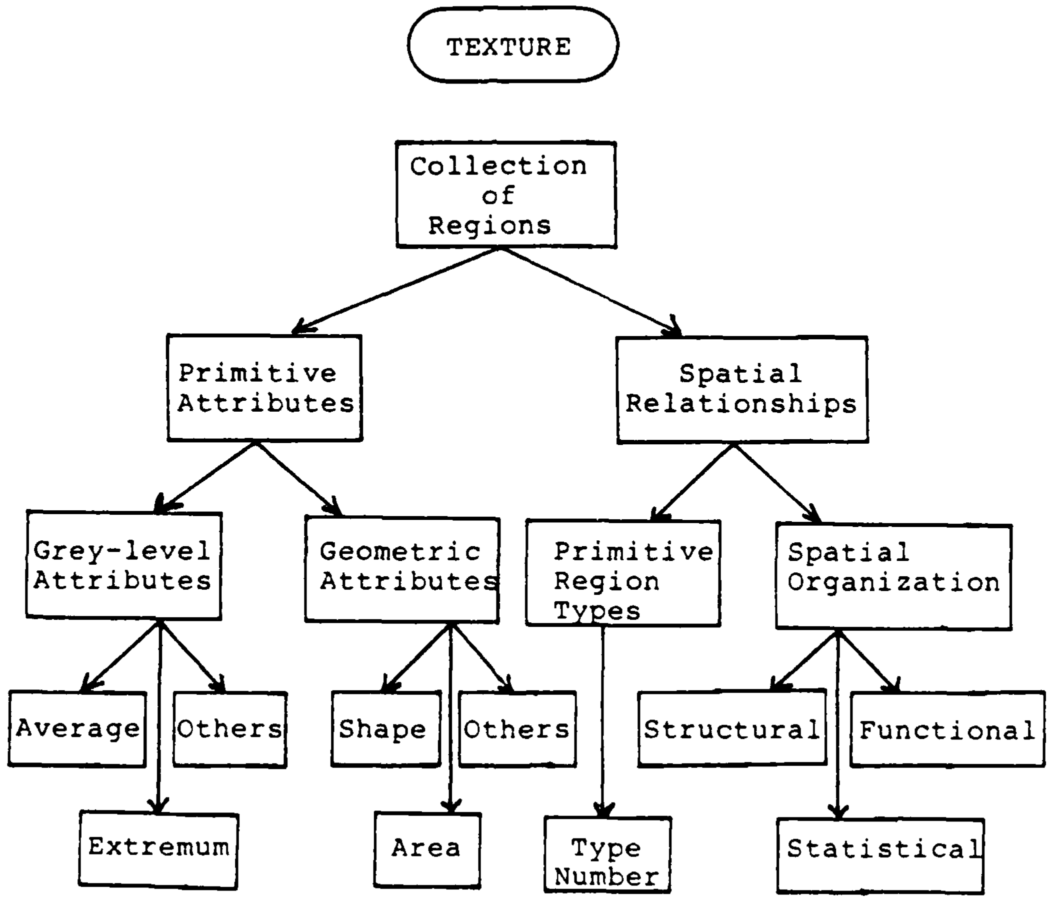


Figure 4.4 A Texture Paradigm

Levine reformulates all texture analysis into just two groups of analysis techniques: statistical and structural. We will not discuss the structural techniques here, however, a good review of most existing computational approaches to textural analysis may be found in (Haralick, 1979).

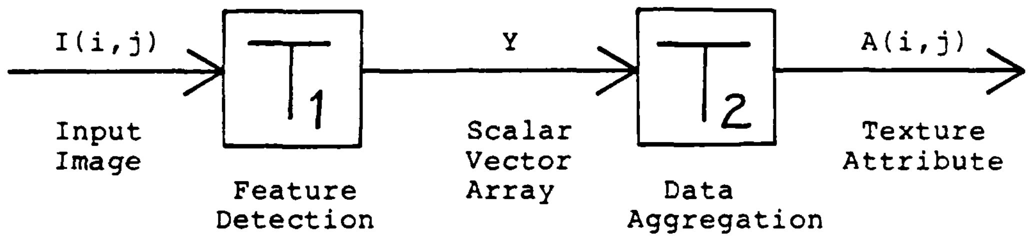


Figure 4.5 The Process of Statistical Texture Analysis

In Figure 4.5 we depict the process of statistical textural analysis by a block diagram. Depending upon the specific technique, transformation T_1 measures a statistical feature. If the output of T_1 is not scalar, the data is usually compressed via transformation T_2. In this manner a texture attribute describing the texture of the input image results.

Statistical techniques of texture analysis may be further divided into three groups (Levine, 1985):

(1) Texture as Spatial Frequency: In this technique the textural attribute of "regularity" is exploited. Popular indicators of this attribute are the autocorrelation function and the two-dimensional Fourier

transform.

(2) Second-Order Statistics: Extremely complex mathematically, these techniques yield good results for the analysis of texture on a regional level. These methods are primarily an extension of first-order statistics which we consider next.

(3) Simple Texture Methods: We will confine further discussion of texture analysis to these techniques. The idea in these techniques is to compute a histogram of the input in transformation T_1 and in transformation T_2 to compute a corresponding textural attribute. (Refer to Figure 4.5) These attributes are usually the so-called "central moments" of the histogram: mean, variance, skewness and kurtosis. The mean and variance are indicators of the regularity of the input image. Symmetry and peakedness are indicated by skewness and kurtosis respectively.

There are several popular variations in the construction of the histogram. In a first method the grey-level value of each pixel is used to make the histogram. In another method, rather than these values, the histogram is formed by some local grey-level property. For example, the sum of the grey-level differences within a neighborhood is often used. In a third method grey-level "run-lengths", where run-lengths indicate areas of similar grey-level values, are used to compute the histogram. In a final method a property called local rank correlation is computed to be used in histogramming. Fairly complicated, this method may be found in (Harwood, 1985).

Having paused for this brief review of the methods of texture analysis, we now proceed to adapt the FES for textural feature extraction. Recall from Figure 4.5 that two transformations are required for the extraction of a textural attribute (or feature). If we design a specific FES such that the cellular array and associative memory logic can accomplish the transformation T_1, and the output logic perform transformation T_2, we will have a specific FES for texture.

For an example design, we use the first variation histogramming method mentioned previously. Figure 4.6 depicts a specific FES which will accomplish the transformations required.

Once the input image has been serially entered into the array, the next step is to latch the grey-level values of each pixel into their respective associative memory cells. Assuming that the grey-levels are quantized into 64 discrete levels, the associative memory will be six bits wide.

The histogram is then computed via the associative memory logic. In this case the comparand is simply a six-bit counter and the masking register is not required. The match logic compares the value of the comparand with the values of all the associative memory cells in parallel. The 1's which result from this comparison in the indicator

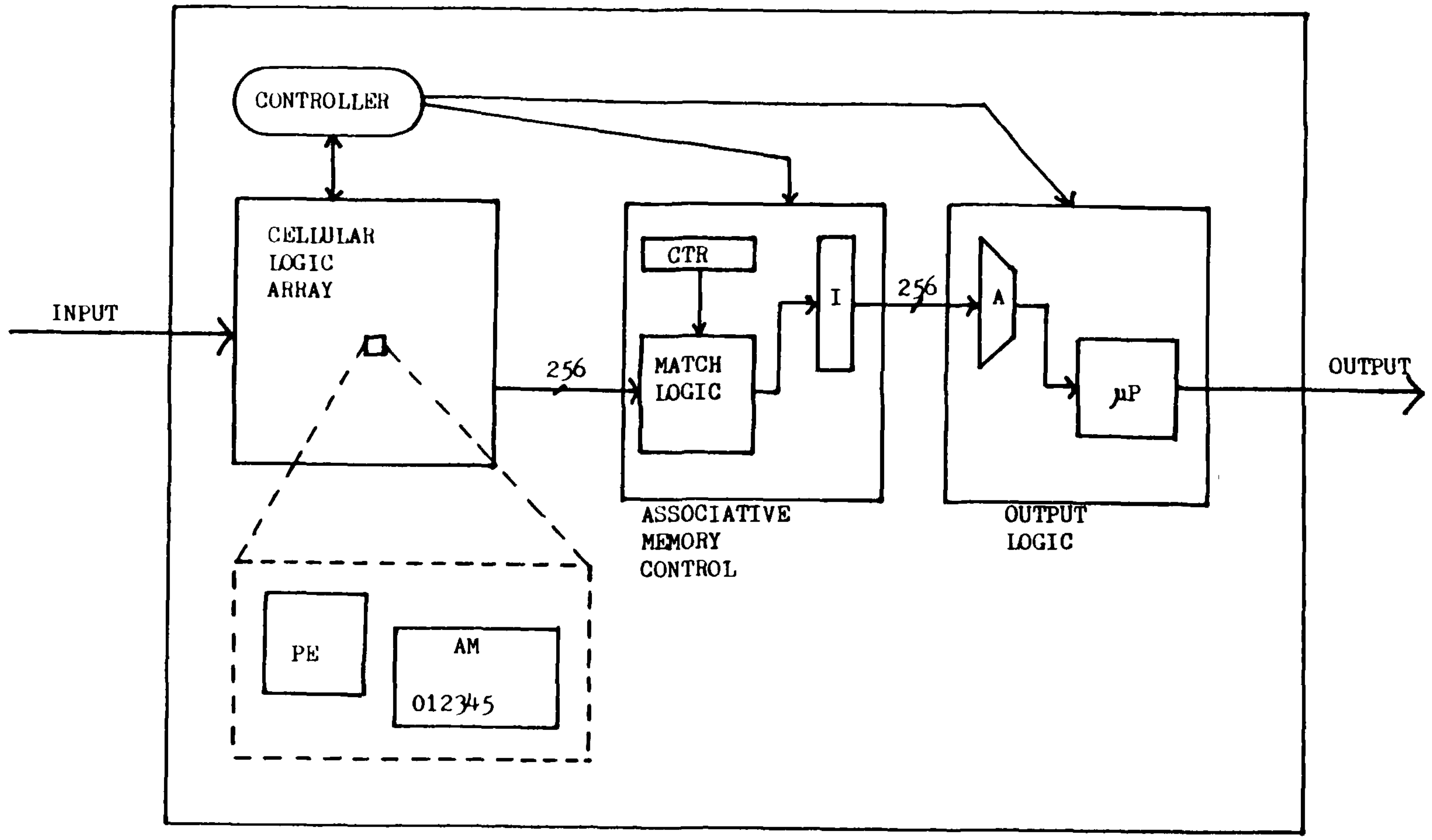


Figure 4.6 FES for Texture Features

register signify matches. Since the pixel array is 16 x 16 the indicator register must be 256 bits.

The indicator register is hardwired to the adder of the Output Logic. The adder could consist of a carry-save adder tree structure with a carry-propagate adder (Kuck, 1978). Using a 7:3 carry-save adder size this addition requires seven adder stages as shown in Figure 4.7. (Other "adder trees" are possible and, indeed, other adding methods.)

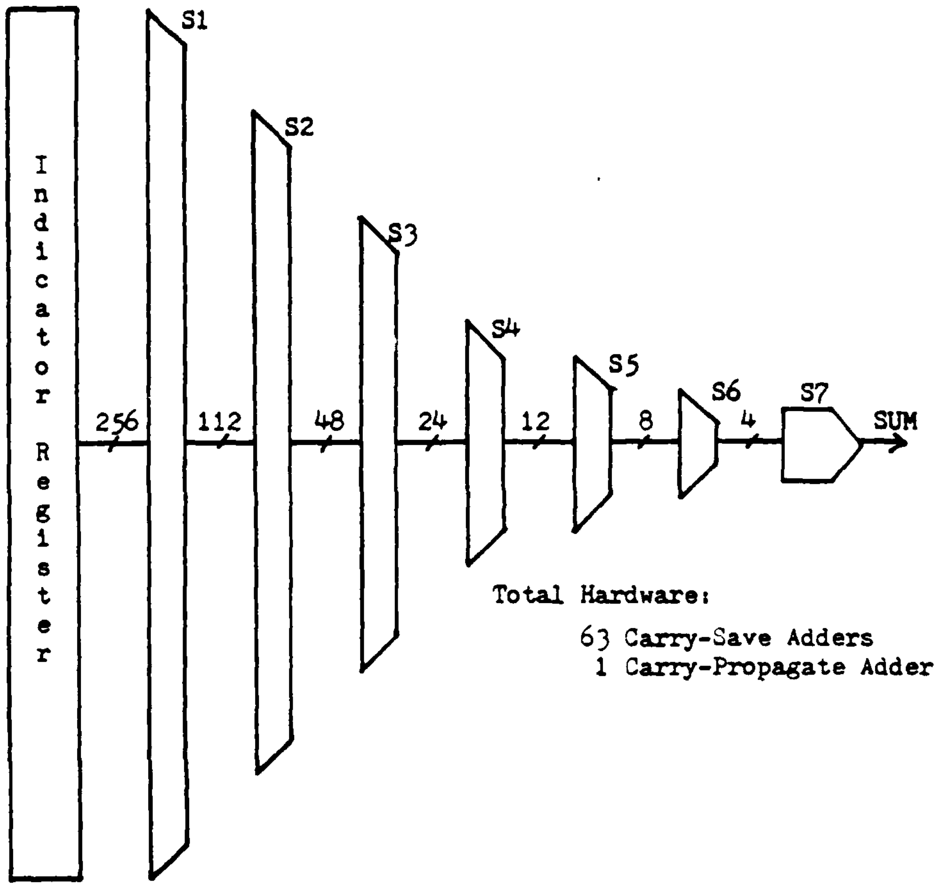


Figure 4.7 The Output Logic "Adder"

The Output Logic microprocessor uses this data to produce, when all grey-level values have been matched and summed, a texture attribute such as the mean, variance, skewness, or kurtosis. Each of these attributes can be computed by "traversing" the histogram once as discussed above. For a more complete explanation of some of the mathematics involved see Section 5.2 wherein this "texture structure" is used in an application. For general references, both (Ballard and Brown, 1982) and (Levine, 1985) cover these mathematics in detail.

The controller in Figure 4.6 is programmed to control the operations detailed above. If we assume that the associative memory search, some part of the adder stages, and steps in attribute formulation operations can be made to proceed in uniform periods of time, part of the process can be pipelined for greater speed. In summary, the entire process would follow these steps:

1. Input Digital Signals
2. Latch grey-level values into associative memory
3. Conduct associative search: counter = 0
4. Sum Indicator Register
5. Compute and sum "partial" texture attribute
6. Repeat Steps 3 - 5 for counter = 1 to 63
7. Output texture attribute

Steps 3 - 5 may be pipelined for efficient computation.

The other simple texture analysis techniques mentioned earlier could be accomplished using this same structure with slight programming modifications to the controller. Recall that instead of the grey-levels themselves these variations required pixel-processing prior to histogramming. It takes little imagination to see that additional instructions could be broadcast to the cells of the array between steps 3 and 4. These instructions would allow for the computation of a local grey-level property or rank correlation as described previously. In step 5, these values would then be latched into the associative memory element for histogramming and final attribute processing.

We have not attempted to derive an adaptation of the FES which will accomplish the computations necessary for the spatial frequency method or the second-order statistics method of statistical texture analysis techniques. This is beyond the scope of this paper. However, it seems entirely possible that each can be implemented in a structure similar to Figure 4.6. The only real significant difference in these methods from the simple texture analysis method is that of computational complexity. It should be pointed out, however, that "comparison studies have indicated that the performance of the spatial frequency approach [Fourier Transform] is poorer than that of other methods." (Levine, 1985).

4.3.2 "Shape" Structures

"Shape" analysis is as large and nebulous a subject as that of texture. Many theories of shape description and recognition exist, where each attempts to explain some specific aspect of the problem. The general approach is tc describe a given shape in terms of simple shape features. A characteristic of the method is whether or not the original shape can be reconstructed from the descriptors. Popular approaches may then be categorized as information-preserving or non-information-preserving based upon this characteristic. A further distinction in shape analysis lies in the type of method employed. For example, there exist spatial domain approaches and scalar transform techniques (Levine, 1985).

In this section we will restrict discussion of shape analysis to the identification and extraction of simple shape features. Exactly what constitutes a set of simple shape features depends on the method used. In our discussion we will simply think of a shape feature as an indicator of a shape property. Virtually any adjective used to describe a shape may be such an indicator. Some examples may be perimeter, area, horizontalness, verticalness, closure, curvature, eccentricity, compactness, concavity, etc. Nearly all of these indicators require preprocessing of the input image for detection. The most common preprocessing steps are highlighting edges, threshholding, and boundary determination.

To best illustrate how the FES may be utilized in extracting a shape feature we consider an example. Suppose the feature we desire is that of line trend, where we define line trend as the sum of the measures of horizontalness and verticalness. Specifically, we would like to characterize the input image by its line trend. A specific FES which will accomplish this operation is shown in Figure 4.8.

Once the grey-level values have been entered into the array, we perform a neighborhood operation. This operation can be thought of as a masking operation where a mask which highlights vertical or horizontal lines is applied to the entire array (Figure 4.9). Each cell of the array performs the operation such that the entire process is done in parallel. Example masking operations are shown in Section 5.2.3. Further information on masks for edge enhancement may be found in (Iliffe, 1984), (Wiejak, 1985), and (Chaudhuri, 1983).

Another neighborhood operation, that of thresholding, is now applied to the data in the array. For simple thresholding each cell compares its value with a value broadcast from the controller. If the cell's value is lower, then a 0 is stored within the cell, otherwise, a 1 is stored. We now have a "binary" image stored in the array. See (Rosenfeld and Kak, Vol. 2, 1982) for more

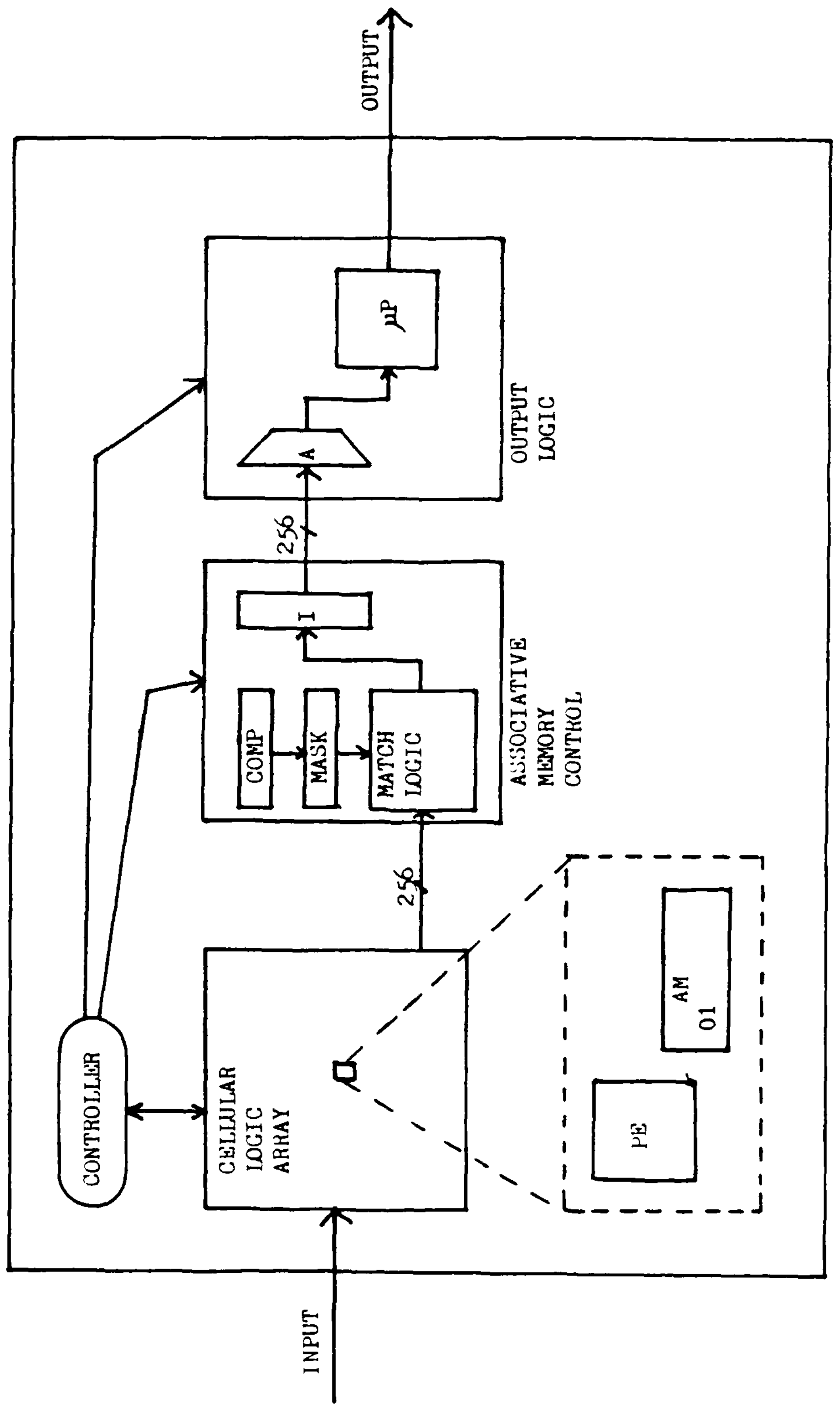


Figure 4.8 FES for Shape Feature

$$\begin{array}{ccc} X & 0 & Y \\ X & 0 & Y \\ X & 0 & Y \end{array} \qquad \begin{array}{ccc} X & X & X \\ 0 & 0 & 0 \\ Y & Y & Y \end{array}$$

(a) (b)

Figure 4.9 Masks for Highlighting Edges (a) Vertical edges. (b) Horizontal edges. Normally the variables X and Y are equal but of opposite sign. Emphasis determined by magnitudes.

information on thresholding. This image reflects either horizontal or vertical edges of the input image depending upon which mask was used.

We now compare the binary value of each pixel with its neighbors . We arbitrarily define a pixel to be in a horizontal line if it has a value of 1 and both its east and west neighbors have values of 1 as well. Likewise, a pixel is in a vertical line if it contains a 1 and both its north and south neighbors contain 1's. A 0 is stored in the associative memory otherwise.

If all we require is this line trend information we can restrict the size of the associative memory to only two-bits wide. Once the neighborhood operations have been completed for horizontal lines we store a 1 in the first bit position of the associative memory cell if its associated pixel was in a horizontal line. During this same time, we can (sequentially within each step) perform the same operations for vertical lines and store a 1 in the second bit position of the associative memory if the pixel was in a vertical line.

We can now initiate an associative memory search for horizontal or vertical aspects of the input image. The Output Logic adder sums the number of occurrences and the microprocessor computes the line trend attribute as a scalar representing horizontalness, verticalness, or any combination of the two such as their ratio. A more complete visualization of this process may be found in Section 5.3 wherein we apply this structure as an aid to image orientation. If the ratio of horizontal to vertical measures was required, feature extraction would require the following steps:

1. Input Digital Signals
2. Highlight horizontal/vertical lines via masking
3. Form Binary Images via thresholding
4. Compare pixel value with neighbors (E and W for horizontal, N and S for vertical). Store line "membership" in associative memory
5. Associative memory search for horizontal lines
6. Sum Indicator Register
7. Repeat 5 - 6 for vertical lines
8. Divide sums and output ratio

By adapting the FES similarly, other shape features can be extracted. If the desired feature were perimeter, a value of 1 could be stored in the associative memory if the pixel was a part of the border of the object in the input image. These 1's could then be added, via associative search and adder steps, to output a measure of the perimeter based upon pixel size. In a similar manner we could store a 1 in the associative memory if a pixel is in the interior or boundary of an object. Summing this characteristic would lead to a measure of the area of an object in the input image. These computational steps, simply described here, are non-trivial local operations. See (Rosenfeld and Kak, Vol 1, 1982) for more information on neighborhood operations and algorithms for contouring (boundary determination).

For other FES structures which would efficiently produce other shape features (some of which were mentioned previously) all that is required is a decomposition of the feature into a set of characteristics which can be computed by neighborhood operations.

4.4 Multi-Feature Structures

In examining the specific structures presented for texture and shape, Figure 4.6 and Figure 4.8, the reader should notice that the only differences which exist are in the width of the associative memory and in the components present in the Associative Memory Control. In operation, of course, there are many differences in the amount and type of operations performed in the cellular array. Therefore, Controller programming is feature specific. Also, note that the data stored in the associative memory differs as to what characteristic it represents. Levine considers both shape and texture attributes to be parts of a greater image processing concept of image segmentation. (Levine, 1985) As such, the fact that the resulting specific structures are similar should not be surprising.

In summary, by modifying FES, it is possible to produce a structure which is capable of extracting a simple feature. If this is not an exciting development in itself, imagine an adapted FES capable of extracting a small set of combined textural and shape type features. We can characterize the necessary changes in FES structure. This would require a larger-width associative memory, perhaps divided into fields for each feature. The Controller programming would need to be complicated and efficient. The output logic would need be more complicated so that it could either serially or in parallel (by vectors) process and output the attributes. A multi-feature extracting structure of this type would seem to possess tremendous potential in computer vision.

Chapter 5 APPLICATIONS

5.1 Introduction

From a casual survey of computer vision literature (See Bibliography) it is evident that a tremendous amount of successful research has been accomplished with regard to computer enhancement of pictorial information. The need for such enhancement stems primarily from the physical inability to receive high enough quality input via receptors. For example, the input may contain noise or may not be in focus. In Section 5.2, an aid to focusing will be developed by an application of the specific texture structure (Figure 4.6).

The second major application, presented in Section 5.3, will be an application of the line trend structure (Figure 4.8) which was also developed in Chapter 4. Imagine, for example, a remote sensing device of some type which observes a conveyor belt carrying automotive parts. Further, suppose this device is part of a system responsible for identifying and counting these parts for inventory purposes. Identification is accomplished ultimately by template matching of the input image to a library of part images stored in memory. By using an implementation of the line trend structure to automatically control rotation of the sensing device, the part image could be properly oriented such that the process of matching would be easier.

In Section 5.4 additional applications are briefly discussed for the general structure developed in this Thesis. By no means a complete listing of possible applications, this section serves to emphasize the versatility and power of an associative preprocessing structure for qualitative feature extraction.

5.2 Focusing by Texture Analysis

In photography, focusing is extremely important. This industry has placed much time, effort, and expense into designing systems which will focus automatically. As a result, there exist slide projectors which focus automatically based upon maximizing the reflected light intensity received from the projection. There are cameras and video equipment which accomplish the focus process via an acoustic system which determines object range. Other, more sophisticated methods exist or are in the research stage, but nearly all of these systems require external apparatus such as sound or light sensors. It should be possible to develop a system which can focus based only upon the pictorial input.

A new product on the market, "from the mind of

Minolta", is a line of cameras (The MAXXUM 7000 and 9000) which possess a SLR Autofocus feature capable of focusing based only on input data. The system incorporates twin separator lenses which project dual images on CCD sensors. A microcomputer compares these signals with a reference signal thus providing controls for the lens movement. When these signals are "in phase" the picture is in focus. The system is composed of optical, electrical, and mechanical components under the direction of a main CPU. The other IC components include a CCD sensor, autofocus interface IC, ROM ICs on each lens, and autofocus CPU, which total to over 150,000 transistors. The controlled four-speed micromotor allows for focusing of the lenses at a rate substantiating Minolta's claim that "only the human eye focuses faster." (Information taken from Minolta's MAXXUM 7000 and MAXXUM 9000 advertising brochures.)

Despite Minolta's success, their system seems far too complex. In this Section we will use the texture structure and first-order statistical method of analysis to construct a focusing system which can focus using only the input data.

5.2.1 Development

In order to focus based upon the input, a parameter of the input which changes with a change in focus must be developed. Consider the digitized input "scene" in Figure

5.1. Here the values represent a quantized level from 0 to 63 of the light intensity received on the sensors via an A/D conversion. The scene depicts a rather bright, small square block on a dark background.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	50	50	0	0	0	0	0	0	0
0	0	0	0	0	0	0	50	50	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.1 A 16 X 16 Pixel "Small Block"

In Figure 5.2 the same "picture", but not in focus, is displayed. Note that the values have been effectively "smoothed." Smoothing evokes an analogy to physically changing the texture of a surface. Smoothing here does make the texture of a pictorial input less rough. As such we consider a first-order statistical approach to the textural changes that occur in the pictorial information in the focusing process. If the data from these pictures are placed into histograms an effect of this smoothing is apparent, as shown in Figure 5.3.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	6	11	11	6	0	0	0	0	0	0
0	0	0	0	0	1	11	20	20	11	1	0	0	0	0	0
0	0	0	0	0	1	11	20	20	11	1	0	0	0	0	0
0	0	0	0	0	0	6	11	11	6	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.2 Unfocused 16 X 16 Pixel "Small Block"

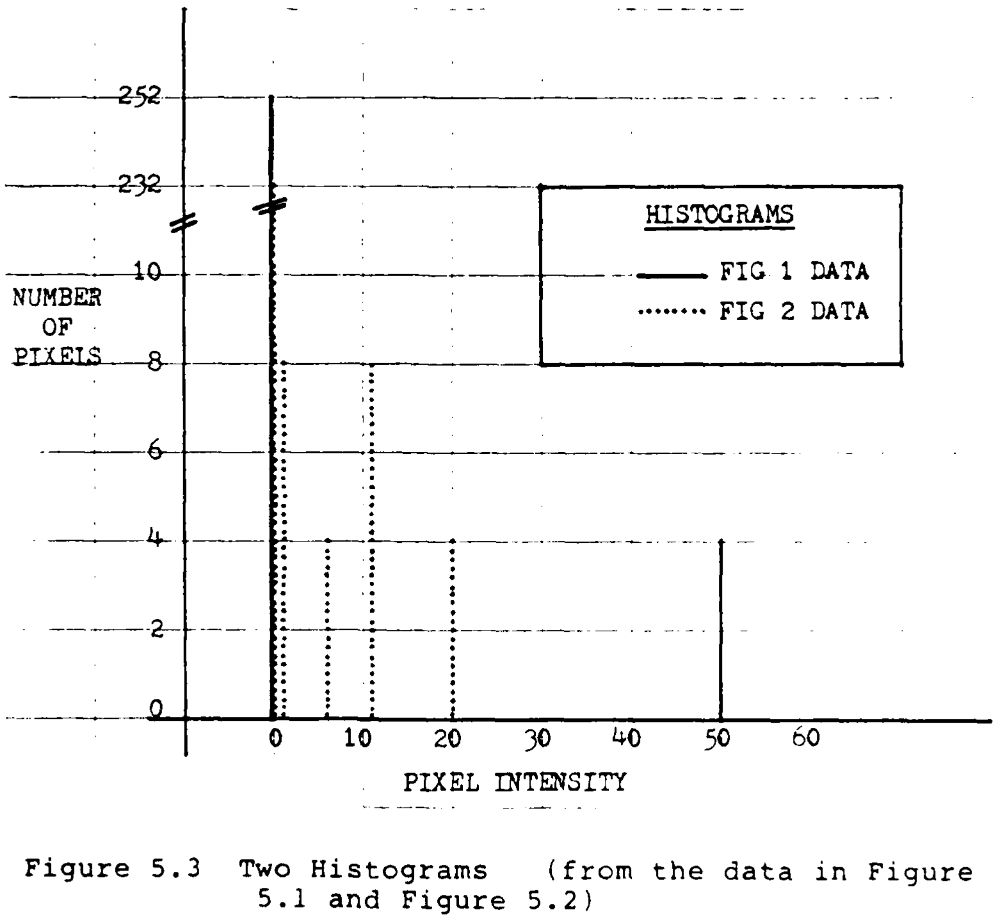


Figure 5.3 Two Histograms (from the data in Figure 5.1 and Figure 5.2)

Note that in each picture the sum of the individual intensity values is the same. In this case we have a constant "power level" of 200. Therefore, as can be easily verified, averaging the histogram would not provide the desired parameter with which focusing could be accomplished. Additionally, it is seen that the tendency is for the bright areas of the picture to darken and the dark areas to brighten as the picture is defocused. At some defocused point it seems likely that the histogram will converge with all pixels having the same value equal to the average power.

The obvious difference between the two histograms is that of distribution of intensities. As such the statistical Gaussian measure of standard deviation would seem promising. Recall that the standard deviation is the square root of the variance which was shown to be a texture analysis tool in Chapter 4. Define the mean MN and the standard deviation SD by:

$$MN = (1/n^2)\sum_{a}^{b} I*H(I)$$

$$SD = \mathrm{sqrt}[\ (1/n^2)\sum_{a}^{b} H(I)(I-MN)^2\ ]$$

where H(I) = Number of pixels with intensity I

I = Pixel intensity

a = Minimum pixel intensity = 0

b = Maximum pixel intensity = 63

n = Size of the n X n picture = 16

Computations performed on the picture data in Figure 5.1 and Figure 5.2 yield the same mean equal to 0.78 as expected. The standard deviations differed, being 24.80 for the former and 12.66 for the latter. Thus, we have a measure for the textural difference between focused and unfocused representations of the same picture. The standard deviation seems promising as the parameter by which to focus.

5.2.2 Algorithms

In this section we assume the standard deviation (SD) to be a reliable measure of focus as suggested, but not proven, in the preceeding section. As such, we develop a simple algorithm, we will call the Single-Step Algorithm (SSA), by which focusing may be accomplished using the texture structure detailed in Chapter 4. A flow-chart for this algorithm is shown in Figure 5.4.

In summary, the algorithm begins by first computing a standard deviation for the initial input. The lens is then moved in a positive direction (closer to the sensors) and the standard deviation is computed for this new input. The standard deviations are compared. If the second measurement is higher in value the direction for lens movement is correct and the algorithm directs repetition of positive lens movement and associated standard deviation computation and comparison until a value for the standard

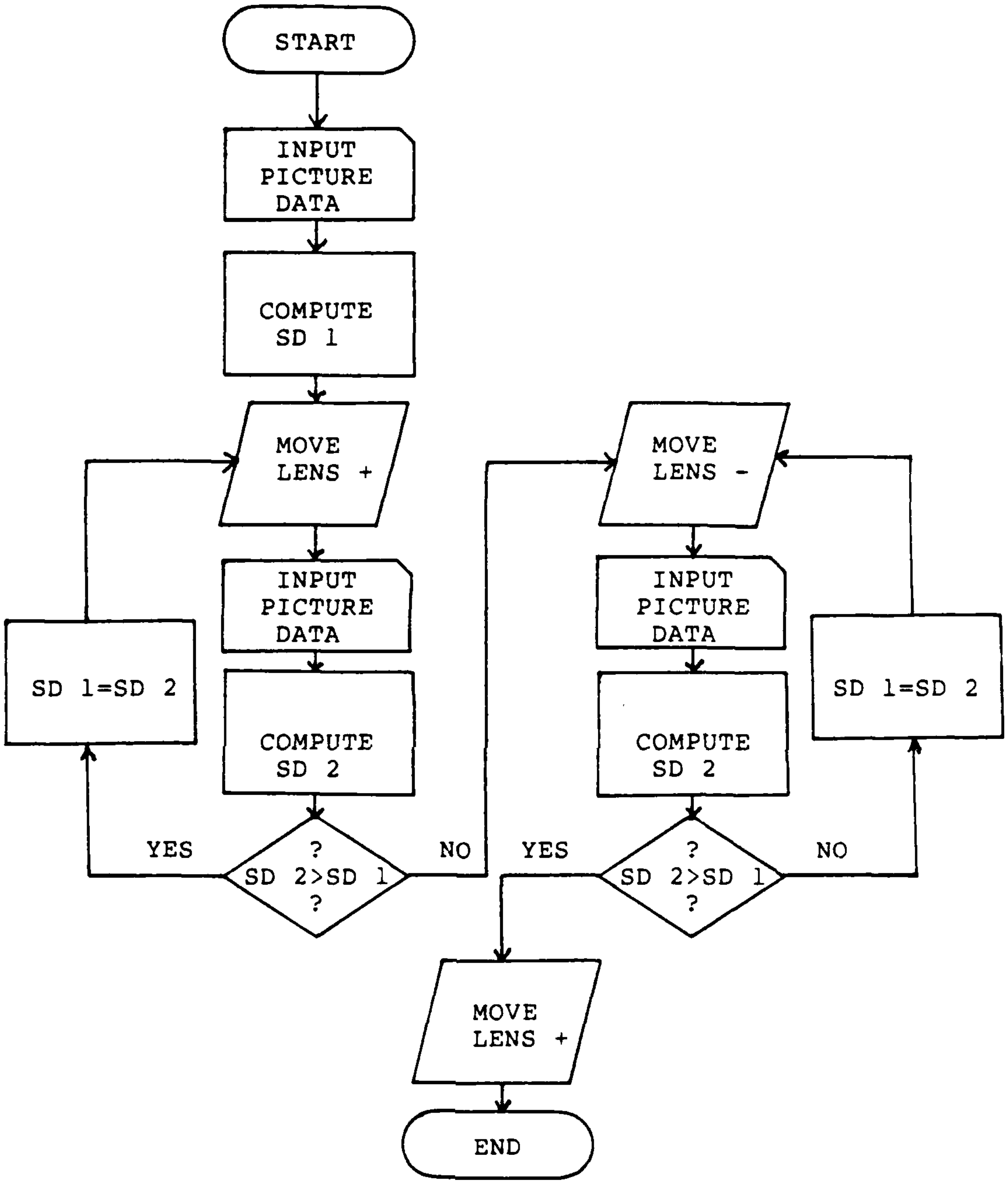


Figure 5.4 Flowchart for the Single-Step Algorithm

deviation is computed, which is less than the previous. At this point, the lens is moved once in the negative direction to insure correctness. Finally, the lens is moved once in the positive direction and at this point the picture is in focus. Conversely, had the second measurement been lower in value than the first, lens movement in the negative direction would have been indicated. The algorithm would then proceed iteratively as before until once again a smaller value was encountered for the standard deviation, at which time the lens would be moved once in the positive direction to insure focus.

Table 5.1 Simple Example Data

Lens Position (Relative to Focused Position)	Standard Deviation
-4	1
-3	2
-2	3
-1	4
0	5
1	4
2	3
3	2
4	1

For an extremely simple example, suppose that at focus a picture input results in a standard deviation of "5." When the picture is defocused by lens movement in either direction the resulting standard deviations are lower, as shown in Table 5.1. Figure 5.5 shows how focusing would occur for three different initial situations according to the Single-Step Algorithm applied to this example.

	Lens Position										
Step	-4	-3	-2	-1	0	1	2	3	4	SD 1/SD 2	Comments
1			*							3 / -	Lens +
2				*						4 / 3	Lens +
3					*					5 / 4	Lens +
4						*				4 / 5	Lens -
5					*					5 / 4	Lens -
6				*						4 / 5	Lens +
7					*						IN FOCUS

(a)

	Lens Position										
Step	-4	-3	-2	-1	0	1	2	3	4	SD 1/SD 2	Comments
1							*			3 / -	Lens +
2								*		2 / 3	Lens -
3							*			3 / 2	Lens -
4						*				4 / 3	Lens -
5					*					5 / 4	Lens -
6				*						4 / 5	Lens +
7					*						IN FOCUS

(b)

	Lens Position										
Step	-4	-3	-2	-1	0	1	2	3	4	SD 1/SD 2	Comments
1					*					5 / -	Lens +
2						*				4 / 5	Lens -
3					*					5 / 4	Lens -
4				*						4 / 5	Lens +
5					*						IN FOCUS

(c)

Figure 5.5 Focusing Example. For the simple example using the SSA when (a) lens too far from sensor, (b) lens too close to sensor, and (c) initial position at focus. Lens position relative to the focused position.

As mentioned previously, the algorithm presented is simple and correspondingly inefficient in that it focuses one lens movement at a time. This is especially apparent in the example where the algorithm directs four lens movements to focus a focused picture. The performance of the algorithm can be expressed by the number of lens movements required to focus. (This is proportional to the time required to focus.)

$M = 1 + D + 3$

where M = number of lens movements

D = number of lens movements from focus

1 movement required to determine direction

3 movements required for convergence

Since the basic operation performed by the algorithm is to find the maximum value of the standard deviation, other more efficient algorithms are possible. Some possible algorithms are:

(1) In-Echelon: Once the correct lens direction has been determined the algorithm could direct a "jump" of a specified, constant number of lens positions. Once this "coarse-tuning" is completed the Single-Step Algorithm could then "fine tune" the focusing process. Performance would be improved for pictures well out of focus.

$M = 1 + D/S + \mathrm{MOD}_S D + 3$

Where S = Jump size

(2) Binary: Once the correct lens direction is established the algorithm could direct a jump half-way to the other end of the lens movement capability. The correct

direction from this point would then be determined. Based upon this determination, the next jump would either move the lens half-way back to the initial position or move the lens another half-distance towards the lens stop position. The process of (1) determine direction and (2) jump half-way would continue until convergence at the point of focus. Obviously, this method requires memory of past lens movements. An example of this algorithm is given in Figure 5.6. Here the range of lens positions is represented by the "dots", "X" is the position where the picture is in focus, and the numbered points indicate current lens

```
                X
. . . . . . . . . . . . . . . . . . . . . . . . . .
    1 2                                           *
```

(a) Direction determined to be to the right. Jump is 11 positions to right. (Half the distance to lens stop)

```
                X
. . . . . . . . . . . . . . . . . . . . . . . . . .
      *                     1 2                   *
```

(b) Direction now left. Jump left 5 positions.

```
                X
. . . . . . . . . . . . . . . . . . . . . . . . . .
      *           1 2       *
```

(c) Direction still left. Jump left 3 positions.

```
                X
. . . . . . . . . . . . . . . . . . . . . . . . . .
      *     1 2   *
```

(d) Direction right. Jump 1 position to right. Picture is now in focus.

Figure 5.6 Operation of the Binary Algorithm

position. The positions denoted with a "*" must be maintained in memory. The number of lens movements would depend upon the current lens position, the direction, and the focusing range. Performance would then be on the order of:

$$[\ 1 + \mathrm{LOG}_2(P_{max} - P_{current})\ ]$$

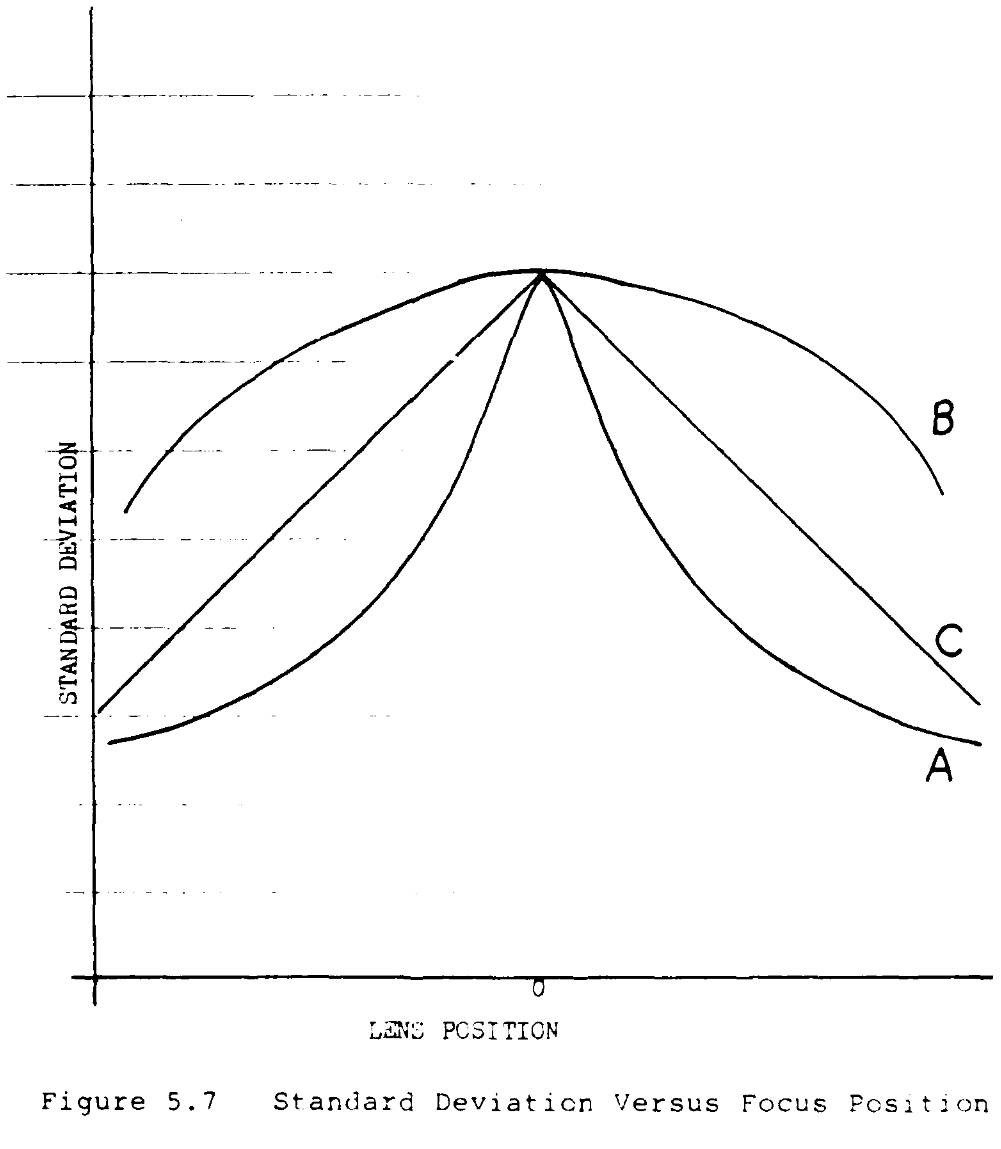


Figure 5.7 Standard Deviation Versus Focus Position

(3) Slope: Once the correct lens direction is established the algorithm would direct a variable length jump inversely proportional to the magnitude of the difference between successive standard deviations. Of course, this would require that the standard deviation versus input curve be monotone on each side of the peak. Performance would naturally depend upon this curvature. For example, three such possible curves are shown in Figure 5.7. This method would work well for curve A since the slope increases as the point of focus is approached. The smaller the slope computed the larger would be the jump directed by the algorithm. For curve B the same, but opposite, procedure would be used since the slope decreases to the point of focus. If the curve resembled curve C, where the slope does not vary, this method would not be significantly better than the Single-Step Algorithm.

Obviously, these algorithms are much more complex logically when compared to the Single-Step Algorithm. As such, implementations would require more controls and require memory to keep track of where the lens had been. This means more hardware and greater cost. On the other hand, the Single-Step Algorithm requires no memory of lens movement, requires simple controls, requires minimal hardware, and always converges to focus (providing the curve resembles those of Figure 5.7).

5.2.3 Modeling Input

At this point a structure (Chapter 4) and an algorithm (Section 5.2.2) have been devised for focusing pictures in our application. The next step is verification, but due to the limited resources available to this researcher, input will be modeled. In (Levine, 1985) a low-pass operator for the spatial frequency domain is used to blur or defocus pictures. For different degrees of blurring, templates of successively larger size, but equally weighted, were used. This basic idea is used to develop a method which will produce our required input. Input, in this case, refers to a series of digital pictures representing focused and successively blurred images of the same scene. We desire this input to approximate the real images that would occur using an imaging system and, as such, we begin with a basic image model.

Point projection is the fundamental model traditionally used for the imaging process conducted by our eye, camera, or other imaging devices. These devices act like a pin hole camera in that the image results from projecting scene points onto an image plane. (Ballard and Brown, 1982.) An schematic representation is shown in Figure 5.8. In part (a) of this figure is a diagram depicting the point to be in focus. This occurs since the image plane is at a distance from the lens which equals the focal length f of the lens. In parts (b) and (c) of the

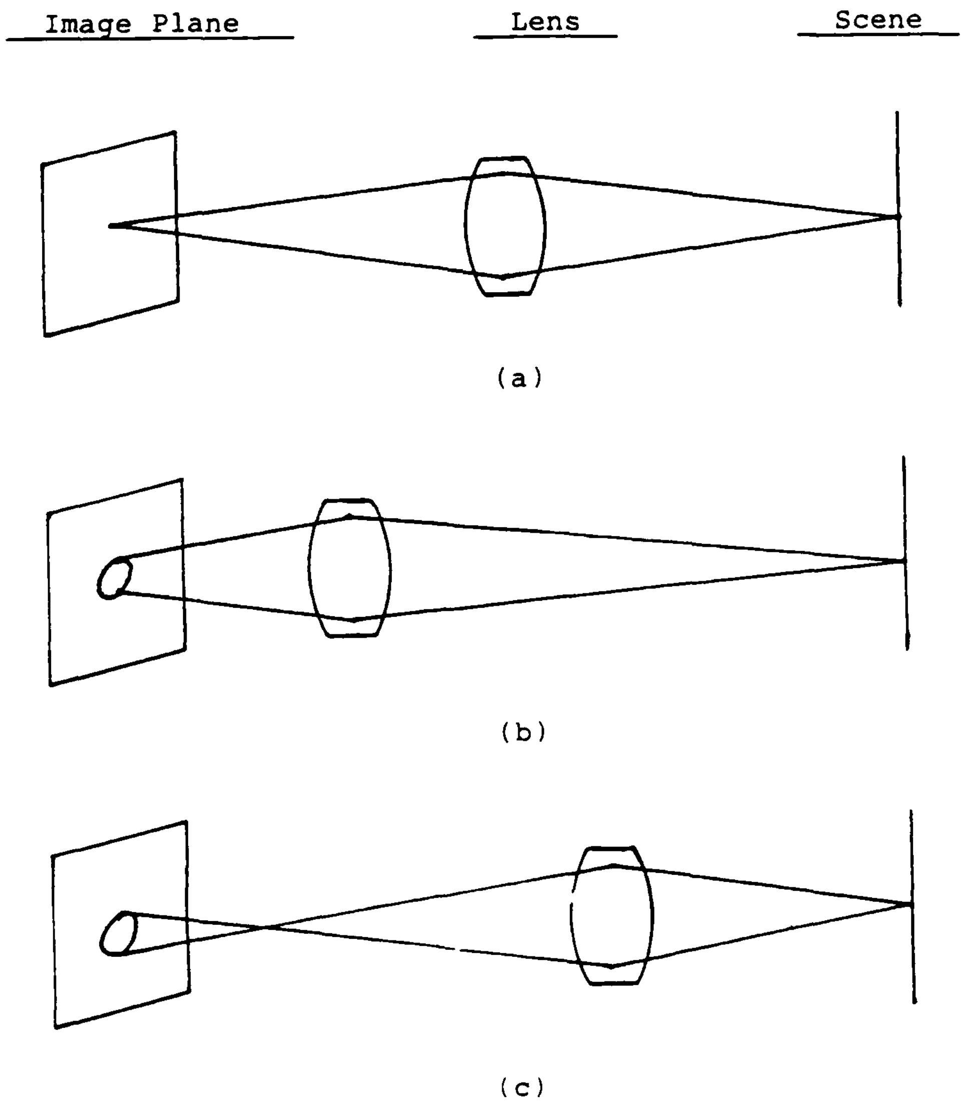


Figure 5.8 Point Projection Model. In (a) the point is in focus, (b) out of focus positive direction, and (c) out of focus negative direction.

same figure the point is shown to be equally out of focus. In the former the distance between image plane and lens is less than f (termed positive) and in the latter this distance is greater than f (termed negative).

Several assumptions are necessary for the model. We will assume that fringing effects due to different light wavelengths are negligible. This will allow us to use an image which represents blurring in the positive direction for an image in the negative direction where the absolute difference between the lens position and focal length is the same. The light from the scene point is assumed to impact the image plane in a circular, uniform pattern when defocused. Other photometric concepts such as variations in sensor sensitivity to light wavelength, scene geometry, reflectance, illumination, etc. will be assumed to be negligible as well. The sensors are assumed to be linear with respect to light energy.

We now expand the point into an "area" or "patch," and retain the assumed behavior. Additionally, we will ignore the fact that the image is "reversed" when out of focus in the negative direction as this does not affect standard deviation calculations. An "area" consists of a neighborhood of points. By sampling, the scene can be divided into an array of M x N areas. An array of photosensitive devices is placed in the image plane such that each device measures the total "brightness" or "power" of a single area of the scene. The power incident on the sensor is converted to an electric signal. The analog signal is then converted to digital and quantized into a set of grey level values. The result of this sampling and quantizing is a digital picture. In our modeling the scene is sampled such that a 16 x 16 "pixel" picture is produced where each "pixel" value may range from 0 to 63 in grey level values. Resolution can be expressed by the product of the number of grey levels and the number of pixels in a picture. (Rosenfeld and Kak, 1982)

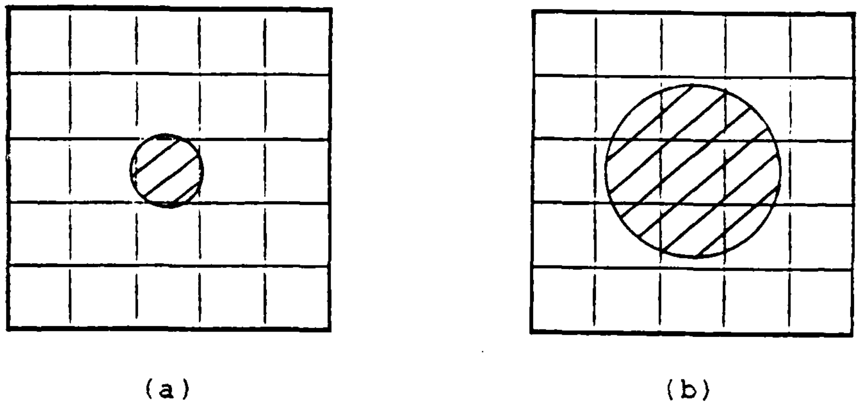


Figure 5.9 The Image Plane. Here (a) represents incident light when the scene is in focus and (b) when the scene is out of focus.

Figure 5.9(a) represents a portion of the image plane. Each square is a sensor in the sensor array. The shaded area is that area on which light strikes from one scene patch when the picture is in focus. If the lens is moved in either direction, the scene is defocused, and the shaded area becomes larger as shown in 5.9(b). The total power of both shaded areas are equal. Therefore, the center pixel in (b) has a grey level value less than the center pixel in (a), having "given" some percentage of its power to neighboring pixels.

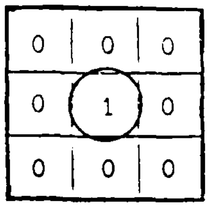
Imagine the entire sensor array as above. When defocused, each pixel's grey level value would be comprised of a percentage of the value it would have at focus summed with amounts received from its neighbors. This idea is used directly to form templates which will be used as spatial frequency operators. However, these "masks" differ from the equal valued templates used in (Levine, 1985) in that they are weighted in a manner such that defocused images computed using convolution approximates real input.

The mathematical process used is discrete 2-D convolution. (Rosenfeld and Kak, 1982; Ballard and Brown, 1982) Given two functions f and g, where f represents the focused picture and g the template or mask, convolution can be expressed as:

$$C(i,j) = \sum_{p=-m}^{+m} \sum_{q=-n}^{+n} g(i-p,j-q)*f(p,q)$$

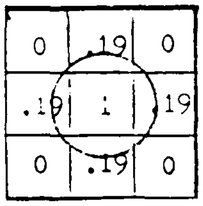
Intuitively, g is "rubbed over" f and the value of the convolution at any displacement is the sum of the product of the relatively displaced functions f and g.

Twelve different masks were created to produce a wide range of defocusing as shown in Figure 5.10. Part (a) shows the focused situation. We ignore the shape difference between the square pixel and the circle which represents the incident light. Their areas are the same. The masks represented for "fuzz factor 0" is then a 1 x 1 template with value 1. Convolution of this mask with the focused picture yields the same picture.



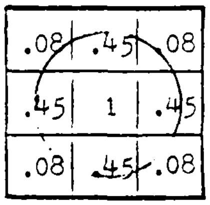
0	0	0
0	1	0
0	0	0

(a) Fuzz Factor 0



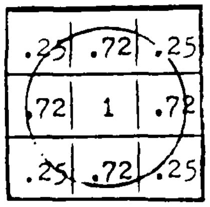
0	.19	0
.19	1	.19
0	.19	0

(b) Fuzz Factor 1 or -1



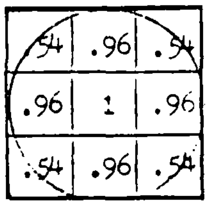
.08	.45	.08
.45	1	.45
.08	.45	.08

(c) Fuzz Factor 2 or -2



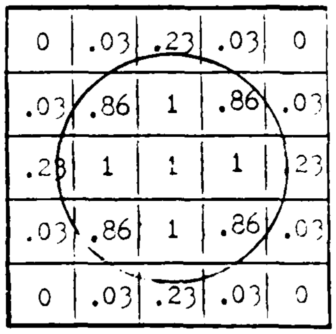
.25	.72	.25
.72	1	.72
.25	.72	.25

(d) Fuzz Factor 3 or -3



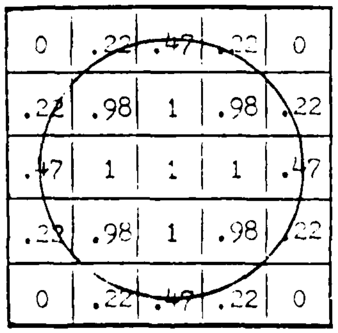
.54	.96	.54
.96	1	.96
.54	.96	.54

(e) Fuzz Factor 4 or -4



0	.03	.23	.03	0
.03	.86	1	.86	.03
.23	1	1	1	.23
.03	.86	1	.86	.03
0	.03	.23	.03	0

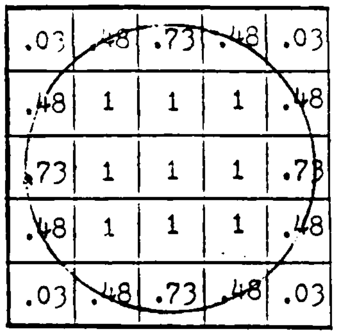
(f) Fuzz Factor 5 or -5



0	.22	.47	.22	0
.22	.98	1	.98	.22
.47	1	1	1	.47
.22	.98	1	.98	.22
0	.22	.47	.22	0

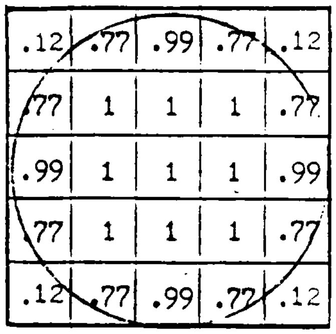
(g) Fuzz Factor 6 or -6

Figure 5.10 Blurring Templates
(Continued on the next page.)



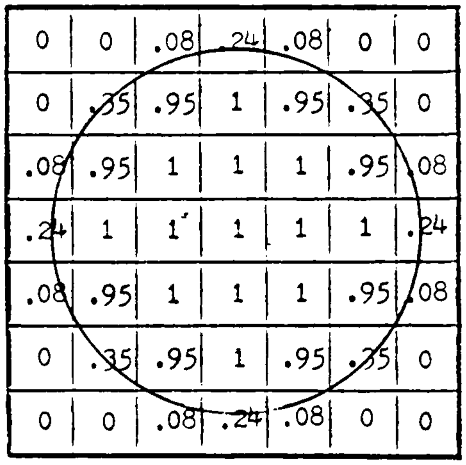
.03	.48	.73	.48	.03
.48	1	1	1	.48
.73	1	1	1	.73
.48	1	1	1	.48
.03	.48	.73	.48	.03

(h) Fuzz Factor 7 or -7



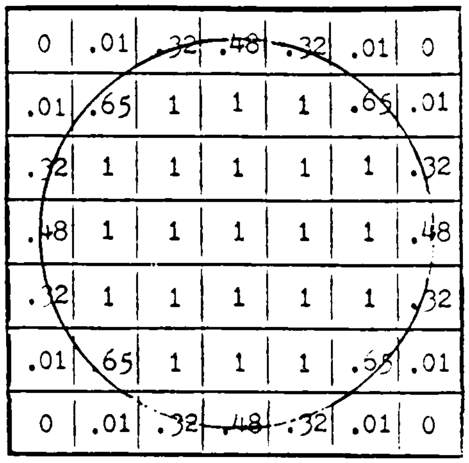
.12	.77	.99	.77	.12
.77	1	1	1	.77
.99	1	1	1	.99
.77	1	1	1	.77
.12	.77	.99	.77	.12

(i) Fuzz Factor 8 or -8



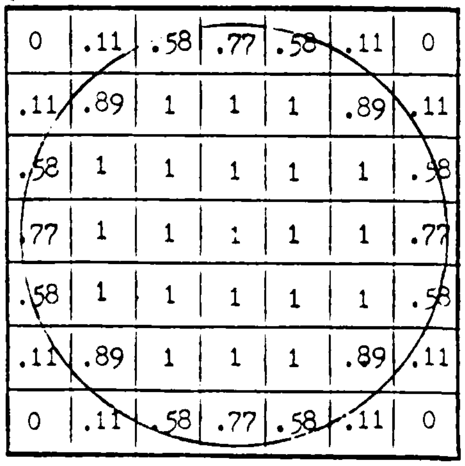
0	0	.08	.24	.08	0	0
0	.35	.95	1	.95	.35	0
.08	.95	1	1	1	.95	.08
.24	1	1	1	1	1	.24
.08	.95	1	1	1	.95	.08
0	.35	.95	1	.95	.35	0
0	0	.08	.24	.08	0	0

(j) Fuzz Factor 9 or -9



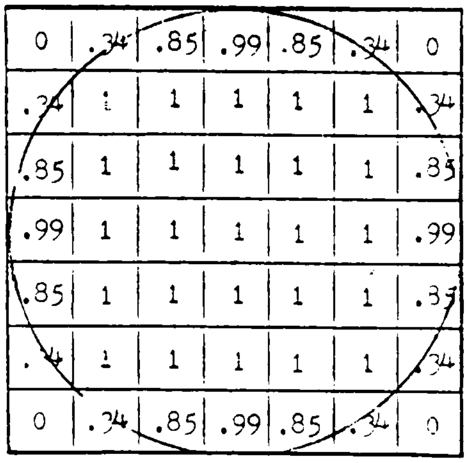
0	.01	.32	.48	.32	.01	0
.01	.65	1	1	1	.65	.01
.32	1	1	1	1	1	.32
.48	1	1	1	1	1	.48
.32	1	1	1	1	1	.32
.01	.65	1	1	1	.65	.01
0	.01	.32	.48	.32	.01	0

(k) Fuzz Factor 10 or -10



0	.11	.58	.77	.58	.11	0
.11	.89	1	1	1	.89	.11
.58	1	1	1	1	1	.58
.77	1	1	1	1	1	.77
.58	1	1	1	1	1	.58
.11	.89	1	1	1	.89	.11
0	.11	.58	.77	.58	.11	0

(l) Fuzz Factor 11 or -11



0	.34	.85	.99	.85	.34	0
.34	1	1	1	1	1	.34
.85	1	1	1	1	1	.85
.99	1	1	1	1	1	.99
.85	1	1	1	1	1	.85
.34	1	1	1	1	1	.34
0	.34	.85	.99	.85	.34	0

(m) Fuzz Factor 12 or -12

Figure 5.10 Blurring Templates (continued)

In part (b) we have expanded the circle (increased the radius by .25 where each square represents an area of 1.) This represents a lens movement in eiher the postive or negative direction away from focus. The resulting mask for fuzz factor 1 or -1 is then 3 x 3 in size with the values shown. Note that unit squares partially covered by the circle have a value proportional to the area covered and are computed to the nearest hundredth. The same process is completed for masks 2 - 12, as shown in parts (c) - (m), to produce masks with fuzz factors 2,-2,...,12,-12. When these masks are convolved with the focused picture different pictures representing greater and greater degrees of blurring result. (Normalization is accomplished after convolution by dividing each C(i,j) value by the area of the circle. This normalization insures that the total power of the resulting pictures remains constant.)

Modeled input can now be produced for analysis using the following procedure:

(1) Select a scene and digitize into a focused picture.

(2) Convolve the focused picture with Mask 1.

(3) Normalize.

(4) Repeat steps (2) and (3) using Masks 2 - 12.

A PASCAL program which implements this procedure (and performs other tasks) appears in Appendix A1.

5.2.4 Experimental Results

Eight focused pictures were selected as examples to test our focusing algorithm (Section 5.2.2). Effort was expended to make the pictures possess different features with respect to size, power, and regularity. These examples are listed in Table 5.2.

Table 5.2 Example Inputs

Example	Title	Dominant Features
1	Small Block A	Low Power, Small Size
2	Small Block B	High Power, Small Size
3	Large Block	Large Size, High Density
4	Large Square	Large Size, Low Density
5	Checkered	Diagonally Periodic
6	Striped	Vertically Periodic
7	Random	Randomness
8	Mixed	All the above

By running the PASCAL program of Appendix A1 on an APPLE II Plus computer, these example pictures were blurred following the procedure presented in the previous section. The eight sets of resulting data appear in Appendix A1. The program computes, as well, the histogram and associated standard deviation for each image. This data is also contained in Appendix A1 following each blur set.

The final results of the experiment can best be seen graphically. In Figure 5.11 the standard deviations of each histogram set versus the fuzz factor are graphed.

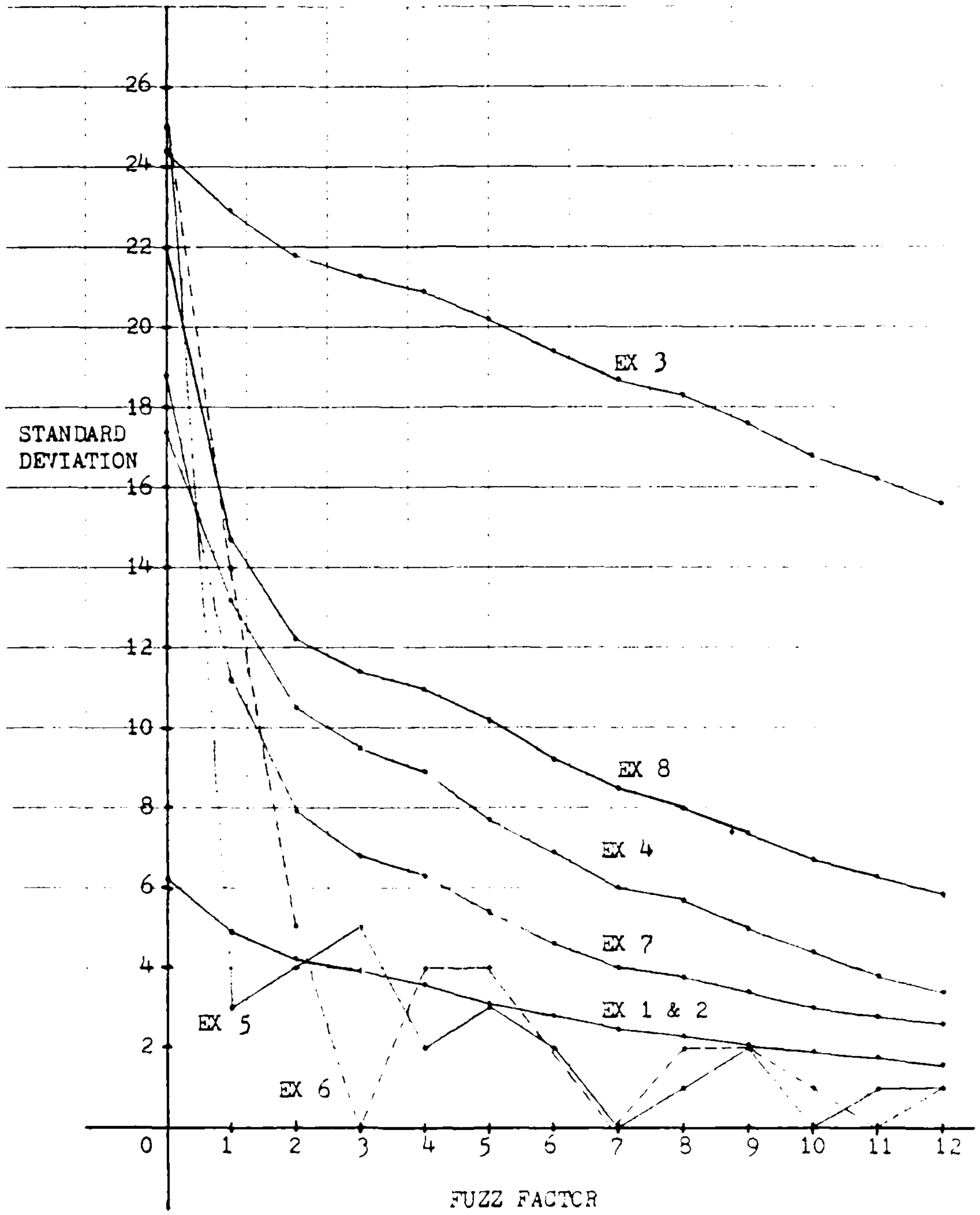


Figure 5.11 Graphical Results. Standard deviation versus fuzz factor display for the data derived from the example inputs.

(Note that in Appendix A2 only the positive direction blurring is reflected in images, histograms, and standard deviations. Likewise only this data is graphed. But, recall that the negative direction data is a mirror reflection of the positive direction data displayed. Therefore, when looking at Figure 5.9 realize that the graphs are "folded over" and that the center of the graph is the vertical axis.)

5.2.5 Analysis and Conclusions

Casual examination of the graphs in Figure 5.11 shows that each of the inputs yield a graph which peaks at fuzz factor 0. Elsewhere the standard deviation decreases in value as the blurring is increased, with the exception of the periodic inputs. The obvious conclusion is that the Single-Step Algorithm implemented on the texture structure will accomplish successful focusing for the aperiodic inputs. Results show that size, power, and density input features do not affect this success. Each curve can be approximated by a curve of the form: y=a/x, where a is a constant. The "bumpiness" of the curve is assumed to be caused by round-off errors.

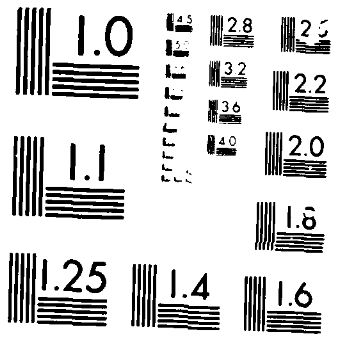
The results from the random input are better than expected. "Noise" in pictures is generally represented by random input. We conclude that "noisy" pictures can be focused successfully by our application. (Although data is not provided in this report, this has been shown experimentally. When a random picture is added to an object picture, via matrix addition, the resulting standard deviation graph for this "picture sum" possesses the same well behaved characteristics for focusing.)

Examination of the graphs produced from the periodic inputs reveal that our simple Single-Step Algorithm will fail unless we begin focusing very near the actual focus. Why the data does not follow the expected results is due to the convolution process performed. Recall that convolution is simply multiplication in the frequency domain of the Fourier Transform. If a picture is periodic the corresponding transform will contain impulses. These components in the transform are responsible for the oscillating appearance of the periodic input results. This effect can be alleviated by surrounding the input picture with a border of zeros. (Levine, 1985) (When, experimentally this was done to our periodic inputs the results were, in fact, similar to the results of the other nonperiodic inputs.) This is not a good solution, however, as a large "border of zeros" is not physically practical.

It should be noted that in a real imaging system obtaining a perfectly periodic input is extremely difficult. Even if the scene were perfectly periodic the edge effects caused by the lens would create some distortion in the picture. Considering this, all we need at most is a slight modification to our algorithm for

AD-A169 147 SPECIAL ASSOCIATIVE PREPROCESSING STRUCTURES FOR QUALITATIVE FEATURE EXTRACTION(U) ARMY MILITARY PERSONNEL CENTER ALEXANDRIA VA M C BIBBY 11 JUN 86 2/2

UNCLASSIFIED F/G 9/2 NL



Mi Ro [illegible]

successful focusing to occur. Note that the periodic input graphs in Figure 5.9 have, at focus, a peak value much greater than at any other point. The other peaks are relatively small in comparison. (In a real system distortion would make these peaks even smaller!) Therefore, we could modify our algorithm such that convergence does not occur unless the peak is greater, by a specified amount, than the surrounding points.

5.3 Orientation by Line Trend

In Section 5.1 we introduced an industrial situation where proper image orientation could aid in the process of template matching. There are many more applications of this process than in the monitoring of a conveyor belt of parts. For example, microscopic cells and cellular structures in microbiology have been identified using template matching procedures (Sternberg, 1983). Template matching has been successfully used to differentiate and identify alphanumeric characters (Hall, 1979).

"The simplest approach to scene matching is called template matching" (Hall, 1979). Basically, template matching may be described as calculating the measure of similarity of an image and a template. This is done by shifting the template across the image and at each position determining the "correlation". The result is a "correlation array" which may then be analyzed for identification or classification (Ballard and Brown, 1982). Besides the two references already mentioned in this paragraph, more information on template matching may be found in (Levine, 1985). This reference has a particularly good explanation of the mathematics involved in template matching.

It should be obvious that if the template and the image are not aligned, match results will be, at best, misleading. Thus, a good deal of additional processing is required on the image or the template in aligning the two pictures for valid matching results. In order to make this process easier, we will apply the shape structure which was developed in Chapter 4.

5.3.1 Development

The basic idea for this application is that if we can orient the image with regard to some parameter, then the number of templates required to be stored in memory for matching may be minimized and the amount of image processing required for alignment will be greatly reduced.

For example, suppose the binary representation of an image (16x16) contains a rectangular object. In memory we have stored the same object as a template. These pictures are shown in Figure 5.12. From the figure it is easy to see that good match results will not be possible because

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

Binary Image

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

Template

Figure 5.12   A Template Matching Example

the pictures are not aligned. We could expand our memory and store another template of the object at this orientation, or we could rotate, via processing, the image to get better results. Now suppose the rectangle in the image was oriented such that there were no horizontal or vertical edges (at an angle). Obviously, we do not want to store many templates of the same object. Likewise, excessive image processing for transforming the image through 360 degrees of rotation is undesirable. A system that would orient input images, then, would be a tremendous aid in template matching.

Based upon this example, it would seem that good parameters for orientation of the input image would be "verticalness" and "horizontalness". In the example above, if the input image was maximized with regard to either horizontalness or verticalness only one template would need be stored and no additional processing would be required on the image for alignment.

We note that the example object, a rectangle, is symmetric in shape. Other objects may not have this property. Therefore, when the object is oriented with regard to horizontalness, there will be two possibilities of the resultant image where one is an upside-down version of the other. The same comment applies to orientati n by verticalness, where two mirror images are possible. This is not considered to be too great a problem. Rotation of the input image by 90 or 180 degrees is a very simple procedure.

In our application we will use the shape structure of Figure 4.8 to extract a feature, which is a measure of the sum of verticalness and horizontalness. (Hereafter referred to as the HV Sum, this sum will be maximum for a line which is either vertical or horizontal and less at other angles.) The methodology for this extraction was described in Section 4.3.2. This measure will be maximized by rotating the sensor/lens portion of the vision system with respect to the scene. Template matching is, thus, made easier as only one template is required per object in memory and, at most, only a couple of simple image transformations (90 or 180 degree rotations) would be required for angular alignment.

5.3.2 Algorithm

Based upon the previous discussions we will assume that the HV Sum is a reliable measure of angular alignment. As such, we develop an algorithm to accomplish this objective. A flow chart for this algorithm is shown in Figure 5.13.

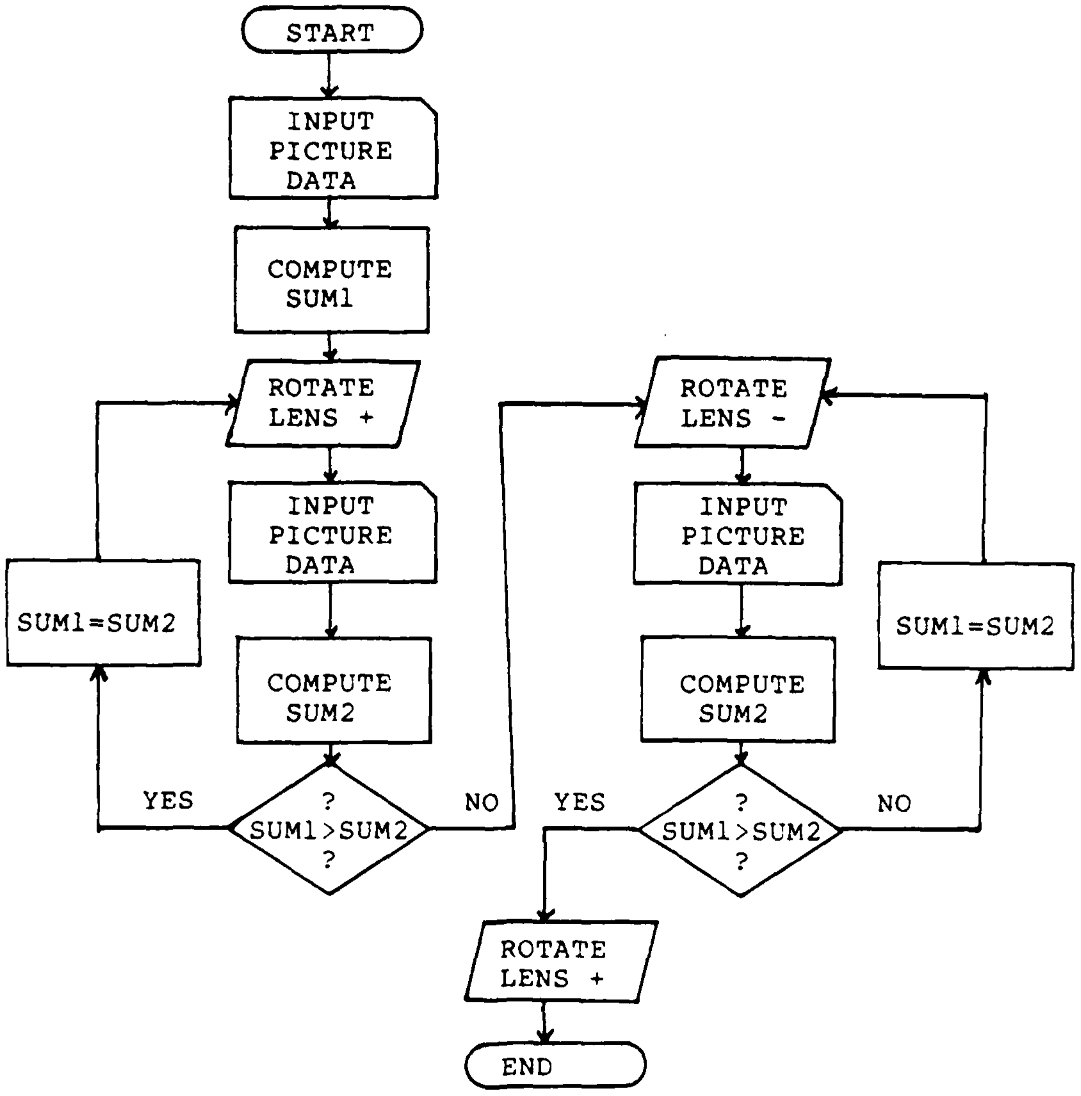


Figure 5.13 Flowchart of Orientation Algorithm

Note that the only differences between this algorithm and the Single-step Algorithm (Figure 5.4) are in the type lens movement and feature computations.

Using this algorithm to manage the activities of the line trend structure will orient scenes composed mostly of parallel rectangular areas to one of four possible positions: right-side-up, upside-down, sideways, and inverted sideways. A worst case for algorithm performance would occur for objects initially oriented 45 degrees from the desired position. As was the case for the focusing algorithm we expect that more efficient algorithms are possible, although these will not be discussed here. The algorithm presented would be a part of a higher level algorithm for template matching.

5.3.3 An Example

In this section we use a simple, hand-produced example to illustrate the activities of the line trend structure that are involved in calculating the HV Sum. (Review methodology presented in Section 4.3.2.) The object we will use is a simple image of a "house" as shown in Figure 5.14 (a).

In Figure 5.14 (b) and (c), we show the results of (after thresholding) applying horizontal and vertical masks to the input picture. Next we apply the operation which determines line "membership" in either a vertical or a

$$\begin{matrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&7&7&7&7&7&7&7&7&0&0&0\\
0&0&8&9&9&9&9&9&9&8&7&6&0&0\\
0&5&7&7&7&7&7&7&8&5&5&5&5&0\\
0&5&5&5&5&5&5&5&4&3&3&3&3&0\\
0&5&5&5&5&5&1&5&4&3&3&3&3&0\\
0&5&5&5&5&5&1&5&4&3&3&3&3&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{matrix}$$

(a) A "house" image.

$$\begin{matrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&1&1&1&1&0&0&0\\
0&0&1&0&0&0&0&0&0&1&0&1&0&0\\
0&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&1&1&1&1&1&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{matrix}$$

(b) Image after horizontal masking and thresholding.

$$\begin{matrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&0&0&1&0&1&0&0\\
0&1&0&0&0&0&0&0&1&0&0&0&1&0\\
0&1&0&0&0&0&0&0&1&0&0&0&1&0\\
0&1&0&0&0&0&1&0&1&0&0&0&1&0\\
0&1&0&0&0&0&1&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{matrix}$$

(c) Image after vertical masking and thresholding.

$$\begin{matrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&1&1&1&1&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&1&1&1&1&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{matrix}$$

(d) Matrix of values which are stored in the first bit position of the associative memory.

$$\begin{matrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&1&0&0&0&1&0\\
0&1&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{matrix}$$

(e) Matrix of values which are stored in the second bit position of the associative memory.

Figure 5.14 Example Data Transformations for HV Sum Calculations

horizontal line. Recall from an earlier discussion that to be considered part of a line the individual pixel must be within a line of at least three pixels. This data, respectively, is then stored in the associative memory, as shown in matrix format in Figure 5.14 (d) and (e). The first matrix represents the first-bit values and the other matrix represents the second-bit values of the two-bit associative memory used in the line trend structure (Figure 4.8).

To complete the computation, the Output Logic would then sum the values in each bit position of the associative memory. Finally, these sums would be combined to form the HV Sum. In this example the HV Sum is equal to 32.

We have purposely selected a house image which was already oriented for simplicity. It is obvious that if the house was at an angle the HV Sum computed above would be smaller. It should be equally obvious that the house could be oriented, depending on its original representation, to an upside-down or either of two separate sideways representations. All four representations would possess the same HV Sum as defined previously. If we graph this sum versus rotational position, a graph such as shown in Figure 5.15 is to be expected. The peaks of the graph represent rotations of the object where the HV Sum is a relative maximum. (The curve has been drawn to highlight these peaks and may not necessarily have this smooth shape.)

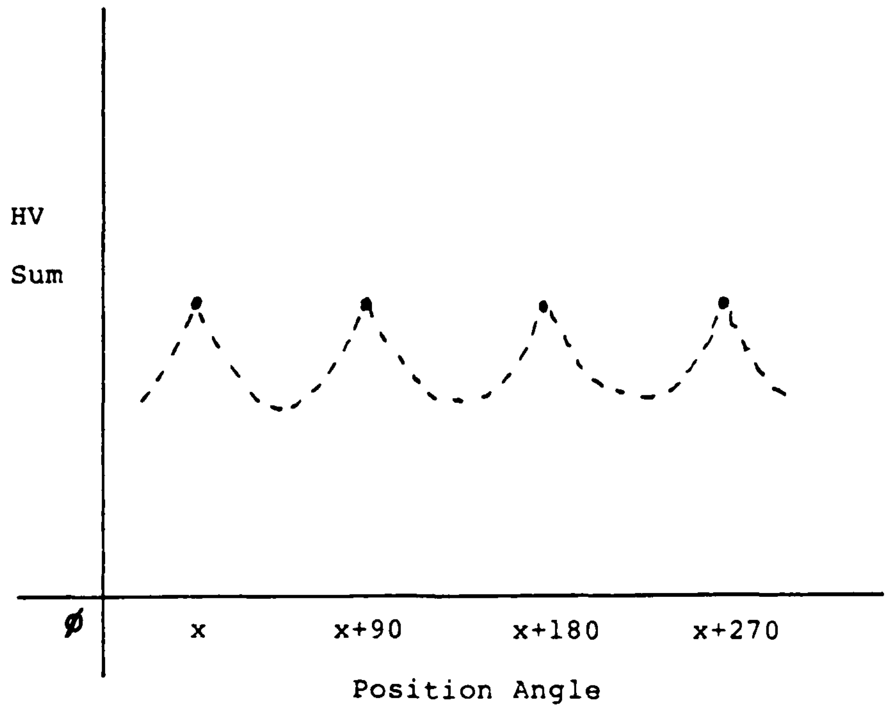


Figure 5.15 HV Sum versus Position Angle
Orientation occurs at the "peaks" and x is dependent upon initial object position.

5.3.4 Conclusion

Based upon the single crude example of the previous section, we conclude that an application of the line trend structure to orient the sensor/lens of a vision system seems viable. One very simple experiment is not sufficient to prove successful results in all cases, and the discussion of "future work" in Section 6.4 suggests how to develop more evidence.

5.4 Other Applications

In this section we consider, in rapidly decreasing detail, four possible applications for the FES. These applications involve more than the determination of a single feature. Therefore, an integrated or multi-feature FES would be required. The power and versatility of this structure is such that a comprehensive list of applications is not possible.

A first possible application continues in the theme of automatic sensor control for vision systems. In this application the desire is to utilize the FES such that a feature, which can be used to control the "zoom" of a vision system, is extracted. This could be used as an aid to template matching by assisting in the size alignment of image and template. The FES should then be able to determine (1) the size of the object in the input image, (2) recognize it as being too small or too large, (3) direct lens movement for magnification, and (4) refocus.

In step (1) the image would need to be edge highlighted and thresholded. The next operation would be an area determination. Step (2) would involve the comparison of this area to a programmed, desired area based upon system resolution to determine zoom direction. Step (3) would be accomplished by the algorithm controlling the FES and Step (4) could be programmed to occur after every zoom lens movement. All of these steps have been discussed previously in the text of this thesis, therefore, this application seems reasonable.

In a second example, suppose we have aerial images as input. What is desired is an estimate by vegetation type of crop yields. (Or maybe we would like an estimate of the area of vegetation destroyed by pests, drought or fire.)

We assume that these estimates may be made by the analysis of a couple of texture features, a color feature and a shape feature. The algorithm for our second application would then be very complicated. By extracting the average, variance and color features, it would seem that quite a number of vegetation types could be distinguished. This information could be stored in an associative memory comprised of three fields. Area (the shape feature) computations could be made for each vegetation type. Based on these areas an estimate of crop yield could then be determined.

In a third situation inspired by a realistic need at the researcher's university, suppose we want to identify the composition of a bushel of seeds by seed type. (Or maybe determine the percentage of white to red blood cells?) An expert in seeds (or blood cells) would need to define differentiating characteristics. If these characteristics can be expressed as extractable features, there is nothing to preclude the successful application of a FES for preprocessing in this situation.

Finally, as a fourth application, consider a remote vision system (encircling Mars; in a radioactive environment; at the bottom of an ocean). The system is required to have the capability of focusing, rotating and zooming with regard to observed objects under preprogrammed control. Continuous transmission of this system reports observations to its owners. Granted, the programming for such a system would be a difficult task, but is it not intuitively obvious that a valid FES application exists here?

Chapter 6 SUMMARY

6.1 General

The general conclusion of this thesis is that the marriage of the principles of associative memory and cellular logic arrays provides for the design of preprocessing structures that are ideally suited for computer vision applications. Furthermore, such a structure seems to be capable of operation in a real-time environment.

In this chapter we will review the concepts and conclusions presented in our investigation of design concepts for special associative preprocessing structures for qualitative feature extraction. In Section 6.2 we analyze the success of the FES with regard to the goals and objectives established in the Introduction. The next section relates an estimation of the performance of the FES. In the last section we itemize further research tasks which need be accomplished to fully develop the FES as a realizable structure. The remainder of this section is devoted to a brief review, by chapter, of the pertinent points presented in this thesis.

In Chapter 1, we introduced the subject of computer vision as the analysis of pictorial information. This activity was divided into two distinct parts: "low-level" analysis or preprocessing and "high-level" analysis or interpretation. Next, we set the stage for the design of a structure which would perform "low-level" analysis by defining goals and objectives.

In the next two chapters we reviewed the literature regarding associative memory and cellular logic arrays. The use of associative memory was shown to have inherent parallelism in processing data. Cellular logic arrays were presented as a "natural" structure for computer vision systems. Current implementations of both associative processors and cellular array computers were reported. Applications of these systems to computer vision were highlighted.

In Chapter 4, we "married" the concepts presented in the previous chapters by designing an associative preprocessing structure for qualitative feature extraction. Termed a FES, for Feature Extracting Structure, the design was first presented in a general configuration. We then designed two specific FES's for the extraction of specific features.

FES applications was the topic for discussion in Chapter 5. Here we applied the specific FES's as an aid to focusing or rotation of images in a vision system. Based upon low-level simulation, we concluded that these applications were viable. Other applications were discussed as practical uses for the FES.

6.2 Analysis of Goals and Objectives

In this section we analyze the FES with respect to the goals and objectives which were established in the Introduction, Chapter 1. The three categories listed were simplicity, parallelism and flexibility.

We consider the architecture of the FES to be simple in the respect that it contains only a few components. The largest and most complicated component is the associative cellular array, which achieves a measure of simplicity through regularity of construction. The array is composed of a rectangular pattern of cells, where each cell is composed of relatively simple logic.

One of the problems discussed in Chapter 2 was the overhead required to read or write to the associative memory. In our design, by colocating the associative memory elements with the cells of the array, we have simplified this overhead. Data is written by the single operation of latching the input data, in parallel, into the associative memory. In operation, in the applications covered, there is no need to read this stored data. Feature extraction is accomplished by comparative searching operations which highlight contextual qualities of the data.

The FES structure particularly appears simple with respect to the computational work it accomplishes. In the example of Section 5.2, the input to the FES is a 20 x 20 array of digital information, or 2,400 bits of information (20 x 20 x 6 bits grey code). Output for the focusing example consisted of a standard deviation measure which at its greatest value would be numerically less than 50, or 6 bits. This is a remarkable decrease in data bandwidth.

The structure uses n x n parallel operations whenever possible. Sequential operations required in applications may be pipelined as discussed in Chapter 4. Neighborhood operations are conducted in parallel within the cellular logic array. Additionally, the parallel search capability of the associative memory is utilized in each application discussed. We will defer discussion of the speed thereby gained due to this highly parallel structure until the next section.

The flexibility of the FES structure was established in Chapters 4 and 5. This structure has potential for many and varied applications in computer vision. The structure is capable of performing a variety of neighborhood operations. As such, it is capable of performing standard preprocessing such as noise removal and enhancement. These operations, however, are invisible at the output as the FES is designed for feature extraction and the output is a coded measure of the feature. Another consideration which was expressed in the Introduction under flexibility was expandability. This will be discussed in the next section.

6.3 FES Performance

In this section the intent is to estimate several performance factors for the FES. The factors selected for analysis were cost, size and speed.

The cost of a FES implementation will probably be very expensive. As a relatively "new" architecture, the development of detailed gate-level designs, chip layout and templates for production will be expensive with regard to both money and time. However, this is to be expected in implementing any new product for special application. In Chapter 3 it was reported that a rectangular array of cellular logic was very expensive. In Chapter 2 it was reported that associative memory was about one and a half times as expensive as comparable sized RAM. Therefore, even if we ignore development costs we cannot avoid the fact that an FES implementation will be very expensive using current technology. Recall, however, a recurring theme in this thesis is that these costs are declining.

In our FES design we selected a 22 x 22 array size primarily because we felt such a design could be implemented on a single chip. This was based upon the successful chip implementation SCAPE, which is comparable in size to the FES. Recall that only the center 16 x 16 cells process picture information. In order to expand a system to "small scale" resolution such as 64 x 64, 16 FES chips would be required. It is not unreasonable to assume that 16 FES chips could be placed on a small printed circuit board. To approximate the resolution of a television screen (512 x 512), however, would require 64 such circuit boards. Undoubtedly, "large scale" resolution, such as satellite images of 1000 x 1000, are beyond the capability of the FES unless "windowing" or other techniques are employed. However, consider another trend of technology: the amount of logic possible per chip is increasing.

Now let us consider the speed of operation obtainable by the FES. Specifically, we will try to estimate the time it would take for "focusing" as described in the example application of Section 5.2. If we examine this particular FES structure (Figure 4.6), we can realistically assume that the most time consuming operation will occur in the "adder" stages of the Output Logic. Let us assume a conservative figure of 50 nanoseconds for this operation. Using the steps listed in Section 4.3.1 as a guideline, we can estimate the time required to analyze the pictorial information. Assuming the pipelining discussed, a rough calculation yields: 64 (grey levels) + 2 (latching and output steps) + 9 (pipeline overhead) + 20 (serial input estimate) which equals 95 periods of time. Multiplying by 50 nanoseconds yields a rough figure of 5 microseconds to complete an analysis. Even if this calculation is off by an order of magnitude, the analysis is completed in much less time than required to move the lens of the visual system. Therefore, we conclude that the FES is capable of real-time picture preprocessing using today's technology.

6.4 Further Research Tasks

To eventually develop an integrated chip for low-level picture processing, we need to consider a number of topics, some of which have been investigated for years at various research institutions. The lists presented here are by no means exhaustive. However, accomplishment of these tasks should aid in the realization of the desired integrated system.

We have divided this section into three lists of further research tasks based upon their level of "future work" activity. The first level addresses specific, "next-step" tasks which would further develop the concepts presented in this thesis. These tasks could be pursued as MS or PhD projects. The second level addresses the same topics, except that, these tasks require greater resources then are normally available to the graduate student. The third level is concerned with global research directions in computer vision. We realize that some tasks may overlap these levels. However, to avoid redundancy we have listed different tasks in each level.

First Level Tasks

* Develop simple yet realistic models for input pictures.

* Develop efficient algorithms and programs which simulate neighborhood operations on sequential computers.

* Investigate additional applications for the FES.

* Using the modeling and simulation developed above, develop verification procedures for FES applications/operations.

* Develop gate-level designs for FES components.

* Study Controller microprogramming, control and timing for FES operations.

Second Level Tasks

* Develop multiple-feature extracting FES's to include designs, algorithms and applications.

* Develop FES Chip floorplans for implementation. Conduct high-level simulations to verify designs.

* Implement in hardware and further study cellular logic arrays possessing associative memory.

* Actually construct and test FES components.

* Design and conduct experiments to measure FES performance in real-time.

Third Level Tasks

* Develop image description languages and image manipulation languages.

* Develop efficient algorithms for pattern recognition and classification using these languages.

* Develop standard "sets" of features which when extracted during preprocessing provide sufficient input for the algorithms developed above.

* Research more efficient methods for performing neighborhood operations, histogramming and the addition of large numbers of 1-bit values.

BIBLIOGRAPHY

Aggarwal, J.K., Richard O. Duda, and Azriel Rosenfeld, eds., Computer Methods in Image Analysis , IEEE Press, New York, 1977.

Aleksander, I. and M.J. Dobree Wilson, "Adaptive Windows for Image Processing", IEE Proceedings-Part E Computers and Digital Techniques , 132:233-45, Sept 1985.

Ballard, Dana H. and Christopher M. Brown, Computer Vision Prentice-Hall, Englewood Cliffs, 1982.

Chaudhuri, B.B., "Efficient Algorithm for Image Enhancement", IEE Proceedings-Part E Computers and Digital Techniques , 130:95-7, May 1983.

Chaudhuri, B.B., "Applications of Quadtree, Octree, and Binary Tree Decomposition Techniques to Shape Analysis and Pattern Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence , 7:652-61, Nov 1985.

Duff, M.J.B., ed., Computing Structures for Image Processing , Academic Press, New York, 1983.

Foster, Caxton C., Content Addressable Parallel Processors , Van Nostrand Reinhold Co., New York, 1976.

Fu, K.S. and Azriel Rosenfeld, "Pattern Recognition and Computer Vision", Computer , 17:274-83, Oct 10, 1984.

Gargantini, I. and H.H. Atkinson, "Linear Quadtrees: A Blocking Technique for Contour Filling", Pattern Recognition , 17:285-93, 1984.

Hall, Ernest L., Computer Image Processing and Recognition , Academic Press, New York, 1979.

Haralick, R.M., "Statistical and Structural Approaches to Texture", Proceedings of the IEEE , 67:786-804, May 1979.

Harwood, David, M. Subbarao, and L.S. Davis, "Texture Classification by Local Rank Correlation", Computer Vision, Graphics, and Image Processing , 32:404-11, Dec 1985.

Hinden, Harvey J., "Algorithms Still Key to Computer Vision Systems", Computer Design , 24:69-74, May 1985.

Hwang, Kai and King-sun Fu, "Integrated Computer Architectures for Image Processing and Database Management", Computer , 16:51-60, Jan 10, 1983.

Hwang, Kai and Faye A. Briggs, Computer Architecture and Parallel Processing , McGraw-Hill Book Co., New York, 1984.

Iliffe, J.K., Advanced Computer Design , Prentice-Hall Int., London, 1982.

Jaggernauth, Jeff, Alexander C.P. Loui, and Anastasios N. Venetsanopoulos, "Real-Time Image Processing by Distributed Arithmetic Implementation of Two-Dimensional Digital Filters", IEEE Transactions on Acoustics, Speech, and Signal Processing , 33:1546-55, Dec 1985.

Kidode, Masatsugu, "Image Processing Machines in Japan", Computer , 16:68-80, Jan 10, 1983.

Kuck, David J., The Structure of Computers and Computations , vol 1, John Wiley and Sons, New York, 1978.

Levine, Martin D., Vision in Man and Machine , McGraw-Hill Book Co., New York, 1985.

McIlroy, C.D., R. Linggard, and W. Monteith, "Hardware for Real-time Image Processing", IEE Proceedings-Part E Computers and Digital Techniques , 131:223-9, Nov 1984.

Merelli, D., P. Mussio, and M. Padula, "An Approach to the Definition, Description, and Extraction of Structures in Binary Digital Images", Computer Vision, Graphics, and Image Processing , 31:19-49, July 1985.

Onoe, Morio, Kendall Preston, and Azriel Rosenfeld, eds., Real-Time/Parallel Computing: Image Analysis , Plenum Press, New York, 1978.

Potter, J. L., "Image Processing on the Massively Parallel Processor", Computer , 16:62-70, Jan 10, 1983.

Preston, Kendall Jr., "Cellular Logic Computers for Pattern Recognition", Computer , 16:36-47, Jan 10, 1983.

Rosenfeld, Azriel and Avinash C. Kak, Digital Picture Processing , Vols 1 and 2, Academic Press, New York, 1982.

Rosenfeld, Azriel, "Parallel Image Processing Using Cellular Arrays", Computer , 16:14-20, Jan 10, 1983.

Smith, K., "Image Processor Handles 256 Pixels Simultaneously", Electronics , 56:81-2, Aug 11, 1983.

Sternberg, Stanley R., "Biomedical Image Processing", Computer , 16:22-34, Jan 10, 1983.

Stoffel, James C., Graphical and Binary Image Processing and Applications , Artech House, Dedham, MA, 1982.

Temma, Tsutomu, et al., "Data Flow Processor Chip for Image Processing", IEEE Transactions on Electron Devices , 32:1784-91, Sept 1985.

Thakkar, Shreekant S., "A High Performance Memory Management Scheme", Computer , 19:8-22, May 1986.

Wiejak, J.S., H. Buxton, and B.F. Buxton, "Convolution with Separable Masks for Early Image Processing", Computer Vision, Graphics, and Image Processing , 32:279-90, Dec 1985.

Yau S.S. and H.S. Fung, "Associative Processor Architecture: A Survey", ACM Computing Surveys , 9:3-28, Mar 1977.

APPENDIX

APPENDIX

The experimental data for the focus application (Section 5.2.2) is contained in this appendix. A PASCAL program which reads an input picture, defocuses this input, computes histograms, and calculates standard deviation is displayed first. This is followed, in succession, by the data generated from this program for the eight example pictures listed in Table 5.2. The data shown consists of each focused picture followed by a series of twelve defocused images of the same picture. The histograms and standard deviation results immediately follow each set of pictures.

```
PROGRAM DEFOCUS (PICTIN,OUTPUT,INPUT);

(*******************************************************************)
(* THIS PROGRAM READS AND DEFOCUSES AN INPUT PICTURE. HISTOGRAMS AND *)
(*   ASSOCIATED STANDARD DEVIATIONS ARE COMPUTED AND PRINTED.        *)
(*          MICHAEL BIBBY    MAY 1986                                *)
(*******************************************************************)


CONST NUM=22; NUMB=16;

TYPE MATX=ARRAY[1..NUM,1..NUM] OF INTEGER;
     MATY=ARRAY[1..NUMB,1..NUMB] OF INTEGER;
     HISTO=ARRAY[0..63,0..12] OF INTEGER;

VAR XMAT:MATX;
    YMAT:MATY;
    HIST:HISTO;
    FUZFAC:INTEGER;
    DONE,INIT:BOOLEAN;
    DATAOUT,DATAIN,PICTIN:TEXT;
    AREA,AA,BB,CC,DD,EE,FF,GG,HH:REAL;


PROCEDURE HGRAM(YMAT:MATY;VAR FUZFAC:INTEGER; VAR HIST:HISTO; VAR INIT:
                BOOLEAN;VAR DONE:BOOLEAN);

(* THIS SUBROUTINE COMPUTES THE HISTOGRAM FOR AN IMAGE *)


VAR ROW,COL,DIFF:INTEGER;

BEGIN
   IF(NOT DONE)
      THEN BEGIN
           IF(NOT INIT)
              THEN BEGIN
                   FOR ROW:=0 TO 63 DO
                      FOR COL:=0 TO 12 DO
                         HIST[ROW,COL]:=0;
                   INIT:=TRUE;
                   END;
         FOR ROW:=1 TO 16 DO
            FOR COL:=1 TO 16 DO
               HIST[YMAT[ROW,COL],FUZFAC]:=HIST[YMAT[ROW,COL],FUZFAC]+1;
         END;
END;


PROCEDURE PRHIST(HIST:HISTO);

(* THIS SUBROUTINE PRINTS THE HISTOGRAMS *)


VAR PAP,START,ROW,COL:INTEGER;

BEGIN
```

```
   WRITELN('PROCEED?');
   READLN(START);
   WRITELN(DATAOUT);
   WRITELN(DATAOUT);
   WRITELN(DATAOUT);
   WRITELN(DATAOUT);
   WRITELN(DATAOUT);
   WRITELN(DATAOUT,'                    INTENSITY HISTOGRAMS FOR EXAMPLE');
   WRITE(DATAOUT,'          ------------------');
   WRITELN(DATAOUT,'---------------------------------------');
   WRITELN(DATAOUT);
   WRITELN(DATAOUT,'                                    FUZZ FACTOR');
   WRITE(DATAOUT,'            I  /');
   FOR ROW:=0 TO 12 DO
      WRITE(DATAOUT,ROW:3,' ');
   WRITELN(DATAOUT);
   WRITE(DATAOUT,'          ------------------------------------');
   WRITELN(DATAOUT,'-------------------');
   FOR ROW:=0 TO 63
      DO BEGIN
         IF ROW=45
            THEN BEGIN
                 WRITELN('INSERT PAPER');
                 READLN(PAP);
                 WRITELN(DATAOUT);
                 WRITELN(DATAOUT);
                 WRITELN(DATAOUT);
                 WRITELN(DATAOUT);
                 WRITELN(DATAOUT);
                 WRITE(DATAOUT,'                    ');
                 WRITELN(DATAOUT,'INTENSITY HISTOGRAMS (CONTINUED)');
                 WRITE(DATAOUT,'             -------------------------');
                 WRITELN(DATAOUT,'-------------------------------');
                 WRITELN(DATAOUT);
                 END;
         WRITE(DATAOUT,'            ',ROW:3,' /');
         FOR COL:=0 TO 12 DO
            WRITE(DATAOUT,HIST[ROW,COL]:3,' ');
         WRITELN(DATAOUT);
         END;
END;

PROCEDURE PRFOC (HIST:HISTO);

(* THIS SUBROUTINE CALCULATES AND PRINTS HISTOGRAM STATISTICS *)


TYPE FOCUS=ARRAY[0..12] OF REAL;

VAR MN,STD:FOCUS;
    ROW,COL:INTEGER;
    SQROOT,PSUM,SUM:REAL;

BEGIN
   FOR COL:=0 TO 12
      DO BEGIN
```

```
         SUM:=0;
         FOR ROW:=0 TO 63
            DO BEGIN
               PSUM:=ROW*HIST[ROW,COL];
               SUM:=SUM+PSUM;
               END;
         MN[COL]:=SUM/256;
         SUM:=0;
         FOR ROW:=0 TO 63
            DO BEGIN
               PSUM:=SQR(ROW-MN[COL])*HIST[ROW,COL];
               SUM:=SUM+PSUM;
               END;
         WRITELN(SUM:12:3);
         READLN(SQROOT);
         STD[COL]:=0.0625*SQROOT;
         END;
   WRITELN(DATAOUT);
   WRITELN(DATAOUT);
   WRITELN(DATAOUT,'                    HISTOGRAM ANALYSIS');
   WRITE(DATAOUT,'          -------------------');
   WRITELN(DATAOUT,'--------------------------------------');
   WRITELN(DATAOUT);
   WRITE(DATAOUT,'          ');
   WRITELN(DATAOUT,'     FUZZ FACTOR     MEAN     STANDARD DEVIATION');
   WRITE(DATAOUT,'          ----');
   WRITELN(DATAOUT,'-----------------------------------------------------');
   FOR COL:=0 TO 12
      DO WRITELN(DATAOUT,'                      ',COL:2,'      ',MN[COL]:7:2,
           '         ',STD[COL]:7:2);
END;


PROCEDURE LOADMAT (VAR XMAT:MATX; VAR DATAIN,DATAOUT:TEXT; VAR YMAT:MATY;
                   VAR FUZFAC:INTEGER; VAR HIST:HISTO;VAR INIT,DONE:BOOLEAN);

(* THIS SUBROUTINE READS THE INPUT PICTURE *)


VAR COL, ROW:INTEGER;

BEGIN
   FOR ROW:=1 TO NUM
      DO BEGIN
            FOR COL:=1 TO NUM
               DO BEGIN
                     READ(DATAIN,XMAT[ROW,COL]);
                     IF((ROW>3) AND (ROW<20) AND (COL>3) AND (COL<20))
                        THEN BEGIN
                                YMAT[ROW-3,COL-3]:=XMAT[ROW,COL];
                             END;
                     END;
            READLN(DATAIN);
            END;
   FUZFAC:=0;
```

```
  HGRAM(YMAT,FUZFAC,HIST,INIT,DONE);
END;

PROCEDURE CALCA(XMAT:MATX;ROW,COL:INTEGER;AA,BB,CC,DD,EE:REAL;
                VAR S1,S2,S3,S4,S5:REAL);

(* THIS SUBROUTINE PERFORMS BLURRING COMPUTATIONS *)


BEGIN
                S1:=AA*(XMAT[ROW,COL+1]+XMAT[ROW,COL-1]+XMAT[ROW-1,COL]+
                    XMAT[ROW+1,COL]);
                S2:=BB*(XMAT[ROW-1,COL-1]+XMAT[ROW-1,COL+1]+
                    XMAT[ROW+1,COL-1]+XMAT[ROW+1,COL+1]);
                S3:=CC*(XMAT[ROW,COL+2]+XMAT[ROW,COL-2]+XMAT[ROW-2,COL]+
                    XMAT[ROW+2,COL]);
                S4:=DD*(XMAT[ROW-2,COL-1]+XMAT[ROW-2,COL+1]+
                 XMAT[ROW-1,COL-2]+XMAT[ROW-1,COL+2]+XMAT[ROW+1,COL-2]+
                 XMAT[ROW+1,COL+2]+XMAT[ROW+2,COL-1]+XMAT[ROW+2,COL+1]);
                S5:=EE*(XMAT[ROW-2,COL-2]+XMAT[ROW-2,COL+2]+
                    XMAT[ROW+2,COL-2]+XMAT[ROW+2,COL+2]);
END;

PROCEDURE CALCB(XMAT:MATX;ROW,COL:INTEGER;FF,GG,HH:REAL;
                VAR S6,S7,S8:REAL);

(* THIS SUBROUTINE PERFORMS BLURRING COMPUTATIONS. *)


BEGIN
                S6:=FF*(XMAT[ROW,COL+3]+XMAT[ROW,COL-3]+XMAT[ROW-3,COL]+
                    XMAT[ROW+3,COL]);
                S7:=GG*(XMAT[ROW-3,COL-1]+XMAT[ROW-3,COL+1]+
                 XMAT[ROW-1,COL-3]+XMAT[ROW-1,COL+3]+XMAT[ROW+1,COL-3]+
                 XMAT[ROW+1,COL+3]+XMAT[ROW+3,COL-1]+XMAT[ROW+3,COL+1]);
                S8:=HH*(XMAT[ROW-3,COL-2]+XMAT[ROW-3,COL+2]+
                 XMAT[ROW-2,COL-3]+XMAT[ROW-2,COL+3]+XMAT[ROW+2,COL-3]+
                 XMAT[ROW+2,COL+3]+XMAT[ROW+3,COL-2]+XMAT[ROW+3,COL+2]);
END;


PROCEDURE CALC(XMAT:MATX;VAR YMAT:MATY;VAR DATAOUT:TEXT;AREA,AA,BB,CC,
     DD,EE,FF,GG,HH:REAL;DONE:BOOLEAN);

(* THIS SUBROUTINE PERFORMS BLURRING COMPUTATIONS. *)


VAR ROW,COL:INTEGER;
    S1,S2,S3,S4,S5,S6,S7,S8:REAL;

BEGIN
  IF(NOT DONE)
    THEN BEGIN
    FOR ROW:=4 TO 19
```

```
      DO BEGIN
         FOR COL:=4 TO 19
            DO BEGIN
                  CALCA(XMAT,ROW,COL,AA,BB,CC,DD,EE,S1,S2,S3,S4,S5);
                  CALCB(XMAT,ROW,COL,FF,GG,HH,S6,S7,S8);
               YMAT[ROW-3,COL-3]:=ROUND((XMAT[ROW,COL]+S1+S2+S3+S4+S5+
                   S6+S7+S8)/AREA);
               END;
         END;
     END;
END;



PROCEDURE FUZZ(XMAT:MATX;VAR YMAT:MATY;VAR DATAOUT:TEXT;VAR DONE:BOOLEAN;
          VAR AREA,AA,BB,CC,DD,EE,FF,GG,HH:REAL; VAR FUZFAC:INTEGER);

(* THIS SUBROUTINE INITIALIZES VARIABLES FOR DEFOCUSING COMPUTATIONS. *)


BEGIN
   WRITELN(FUZFAC:2);
   IF((FUZFAC<1) OR (FUZFAC>12))
      THEN BEGIN
              DONE:=TRUE
           END
      ELSE BEGIN
              BB:=0;CC:=0;DD:=0;EE:=0;FF:=0;GG:=0;HH:=0;
              CASE FUZFAC OF
                 1: BEGIN
                       AREA:=1.78;AA:=0.195;
                    END;
                 2: BEGIN
                       AREA:=3.14;AA:=0.455;BB:=0.08;
                    END;
                 3: BEGIN
                       AREA:=4.92;AA:=0.73;BB:=0.25;
                    END;
                 4: BEGIN
                       AREA:=7.08;AA:=0.97;BB:=0.55;
                    END;
                 5: BEGIN
                       AREA:=9.6;AA:=1;BB:=0.86;CC:=0.23;DD:=0.03;
                    END;
                 6: BEGIN
                       AREA:=12.56;AA:=1;BB:=0.98;CC:=0.47;DD:=0.22;
                    END;
                 7: BEGIN
                       AREA:=15.88;AA:=1;BB:=1;CC:=0.73;DD:=0.48;
                       EE:=0.03;
                    END;
                 8: BEGIN
                       AREA:=19.6;AA:=1;BB:=1;CC:=0.99;DD:=0.77;
                       EE:=0.12;
                    END;
                 9: BEGIN
```

```
                       AREA:=23.68;AA:=1;BB:=1;CC:=1;DD:=0.96;
                       EE:=0.35;FF:=0.24;GG:=0.08;
                     END;
                 10: BEGIN
                        AREA:=28.24;AA:=1;BB:=1;CC:=1;DD:=1;
                        EE:=0.65;FF:=0.48;GG:=0.32;HH:=0.02;
                     END;
                 11: BEGIN
                        AREA:=33.16;AA:=1;BB:=1;CC:=1;DD:=1;
                        EE:=0.89;FF:=0.77;GG:=0.58;HH:=0.11;
                     END;
                 12: BEGIN
                        AREA:=38.48;AA:=1;BB:=1;CC:=1;DD:=1;
                        EE:=1;FF:=0.99;GG:=0.85;HH:=0.34;
                     END;
                 END;
              END;
END;

(*********************************************************************)
(*********************************************************************)
(***************      MAIN PROGRAM      ******************************)
(*********************************************************************)
(*********************************************************************)


BEGIN
   RESET(DATAIN,'PICTIN.TEXT');
   REWRITE(DATAOUT,'PRINTER:');
   DONE:=FALSE;
   INIT:=FALSE;
   LOADMAT(XMAT,DATAIN,DATAOUT,YMAT,FUZFAC,HIST,INIT,DONE);
   WHILE NOT DONE
      DO BEGIN
         FOR FUZFAC:=1 TO 13
          DO BEGIN
             FUZZ(XMAT,YMAT,DATAOUT,DONE,AREA,AA,BB,CC,DD,EE,FF,GG,HH,FUZFAC);
             CALC(XMAT,YMAT,DATAOUT,AREA,AA,BB,CC,DD,EE,FF,GG,HH,DONE);
             HGRAM(YMAT,FUZFAC,HIST,INIT,DONE);
             END;
          END;
   PRHIST(HIST);
   PRFOC(HIST);
   CLOSE(DATAOUT,CRUNCH);
END.
```

EXAMPLE 1 "SMALL BLOCK A"

```
                    THE FOCUSED PICTURE
----------------------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0 50 50  0  0  0  0  0  0  0
  0  0  0  0  0  0  0 50 50  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

EXAMPLE 1   BLURRED IMAGES OF "SMALL BLOCK A"

```
                 FUZZ FACTOR = 1
----------------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  5  5  0  0  0  0  0  0  0
 0  0  0  0  0  0  5 39 39  5  0  0  0  0  0  0
 0  0  0  0  0  0  5 39 39  5  0  0  0  0  0  0
 0  0  0  0  0  0  0  5  5  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                 FUZZ FACTOR = 3
----------------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  3 10 10  3  0  0  0  0  0  0
 0  0  0  0  0  0 10 28 28 10  0  0  0  0  0  0
 0  0  0  0  0  0 10 28 28 10  0  0  0  0  0  0
 0  0  0  0  0  0  3 10 10  3  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                 FUZZ FACTOR = 2
----------------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  1  9  9  1  0  0  0  0  0  0
 0  0  0  0  0  0  9 32 32  9  0  0  0  0  0  0
 0  0  0  0  0  0  9 32 32  9  0  0  0  0  0  0
 0  0  0  0  0  0  1  9  9  1  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                 FUZZ FACTOR = 4
----------------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  4 11 11  4  0  0  0  0  0  0
 0  0  0  0  0  0 11 25 25 11  0  0  0  0  0  0
 0  0  0  0  0  0 11 25 25 11  0  0  0  0  0  0
 0  0  0  0  0  0  4 11 11  4  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

EXAMPLE 1   BLURRED IMAGES OF "SMALL BLOCK A"   (continued)

```
                FUZZ FACTOR = 5
-----------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0
  0  0  0  0  0  0  5 11 11  5  0  0  0  0  0  0
  0  0  0  0  0  1 11 20 20 11  1  0  0  0  0  0
  0  0  0  0  0  1 11 20 20 11  1  0  0  0  0  0
  0  0  0  0  0  0  5 11 11  5  0  0  0  0  0  0
  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 7
-----------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  2  4  4  2  0  0  0  0  0  0
  0  0  0  0  0  2  6 10 10  6  2  0  0  0  0  0
  0  0  0  0  0  4 10 13 13 10  4  0  0  0  0  0
  0  0  0  0  0  4 10 13 13 10  4  0  0  0  0  0
  0  0  0  0  0  2  6 10 10  6  2  0  0  0  0  0
  0  0  0  0  0  0  2  4  4  2  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 6
-----------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  1  3  3  1  0  0  0  0  0  0
  0  0  0  0  0  1  6 11 11  6  1  0  0  0  0  0
  0  0  0  0  0  3 11 16 16 11  3  0  0  0  0  0
  0  0  0  0  0  3 11 16 16 11  3  0  0  0  0  0
  0  0  0  0  0  1  6 11 11  6  1  0  0  0  0  0
  0  0  0  0  0  0  1  3  3  1  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 8
-----------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  2  4  4  2  0  0  0  0  0  0
  0  0  0  0  0  2  7 10 10  7  2  0  0  0  0  0
  0  0  0  0  0  4 10 10 10 10  4  0  0  0  0  0
  0  0  0  0  0  4 10 10 10 10  4  0  0  0  0  0
  0  0  0  0  0  2  7 10 10  7  2  0  0  0  0  0
  0  0  0  0  0  0  2  4  4  2  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

EXAMPLE 1   BLURRED IMAGES OF "SMALL BLOCK A"   (continued)

```
              FUZZ FACTOR = 9                                     FUZZ FACTOR = 11
-------------------------------------------------       -------------------------------------------------

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

              FUZZ FACTOR = 10                                    FUZZ FACTOR = 12
-------------------------------------------------       -------------------------------------------------

0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0          0  0  0  0  0  0  2  2  2  2  0  0  0  0  0  0
0  0  0  0  0  1  4  5  5  4  1  0  0  0  0  0          0  0  0  0  0  2  4  5  5  4  2  0  0  0  0  0
0  0  0  0  1  4  6  7  7  6  4  1  0  0  0  0          0  0  0  0  2  4  5  5  5  5  4  2  0  0  0  0
0  0  0  0  1  5  7  7  7  7  5  1  0  0  0  0          0  0  0  0  2  5  5  5  5  5  5  2  0  0  0  0
0  0  0  0  1  5  7  7  7  7  5  1  0  0  0  0          0  0  0  0  2  5  5  5  5  5  5  2  0  0  0  0
0  0  0  0  1  4  6  7  7  6  4  1  0  0  0  0          0  0  0  0  2  4  5  5  5  5  4  2  0  0  0  0
0  0  0  0  0  1  4  5  5  4  1  0  0  0  0  0          0  0  0  0  0  2  4  5  5  4  2  0  0  0  0  0
0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0          0  0  0  0  0  0  2  2  2  2  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

INTENSITY HISTOGRAMS FOR EXAMPLE 1

		FUZZ FACTOR												
I	/	0	1	2	3	4	5	6	7	8	9	10	11	12
0	/	252	244	240	240	240	232	224	224	224	212	204	204	204
1	/	0	0	4	0	0	8	8	0	0	12	20	8	0
2	/	0	0	0	0	0	0	0	8	8	0	0	12	20
3	/	0	0	0	4	0	0	8	0	0	8	0	0	0
4	/	0	0	0	0	4	0	0	8	8	0	8	8	8
5	/	0	8	0	0	0	4	0	0	0	8	8	8	24
6	/	0	0	0	0	0	0	4	4	0	0	4	16	0
7	/	0	0	0	0	0	0	0	0	4	4	12	0	0
8	/	0	0	0	0	0	0	0	0	0	12	0	0	0
9	/	0	0	8	0	0	0	0	0	0	0	0	0	0
10	/	0	0	0	8	0	0	0	8	12	0	0	0	0
11	/	0	0	0	0	8	8	8	0	0	0	0	0	0
12	/	0	0	0	0	0	0	0	0	0	0	0	0	0
13	/	0	0	0	0	0	0	0	4	0	0	0	0	0
14	/	0	0	0	0	0	0	0	0	0	0	0	0	0
15	/	0	0	0	0	0	0	0	0	0	0	0	0	0
16	/	0	0	0	0	0	0	4	0	0	0	0	0	0
17	/	0	0	0	0	0	0	0	0	0	0	0	0	0
18	/	0	0	0	0	0	0	0	0	0	0	0	0	0
19	/	0	0	0	0	0	0	0	0	0	0	0	0	0
20	/	0	0	0	0	0	4	0	0	0	0	0	0	0
21	/	0	0	0	0	0	0	0	0	0	0	0	0	0
22	/	0	0	0	0	0	0	0	0	0	0	0	0	0
23	/	0	0	0	0	0	0	0	0	0	0	0	0	0
24	/	0	0	0	0	0	0	0	0	0	0	0	0	0
25	/	0	0	0	0	4	0	0	0	0	0	0	0	0
26	/	0	0	0	0	0	0	0	0	0	0	0	0	0
27	/	0	0	0	0	0	0	0	0	0	0	0	0	0
28	/	0	0	0	4	0	0	0	0	0	0	0	0	0
29	/	0	0	0	0	0	0	0	0	0	0	0	0	0
30	/	0	0	0	0	0	0	0	0	0	0	0	0	0
31	/	0	0	0	0	0	0	0	0	0	0	0	0	0
32	/	0	0	4	0	0	0	0	0	0	0	0	0	0
33	/	0	0	0	0	0	0	0	0	0	0	0	0	0
34	/	0	0	0	0	0	0	0	0	0	0	0	0	0
35	/	0	0	0	0	0	0	0	0	0	0	0	0	0
36	/	0	0	0	0	0	0	0	0	0	0	0	0	0
37	/	0	0	0	0	0	0	0	0	0	0	0	0	0
38	/	0	0	0	0	0	0	0	0	0	0	0	0	0
39	/	0	4	0	0	0	0	0	0	0	0	0	0	0
40	/	0	0	0	0	0	0	0	0	0	0	0	0	0
41	/	0	0	0	0	0	0	0	0	0	0	0	0	0
42	/	0	0	0	0	0	0	0	0	0	0	0	0	0
43	/	0	0	0	0	0	0	0	0	0	0	0	0	0
44	/	0	0	0	0	0	0	0	0	0	0	0	0	0

INTENSITY HISTOGRAMS (CONTINUED)

45 /	0	0	0	0	0	0	0	0	0	0	0	0	0
46 /	0	0	0	0	0	0	0	0	0	0	0	0	0
47 /	0	0	0	0	0	0	0	0	0	0	0	0	0
48 /	0	0	0	0	0	0	0	0	0	0	0	0	0
49 /	0	0	0	0	0	0	0	0	0	0	0	0	0
50 /	4	0	0	0	0	0	0	0	0	0	0	0	0
51 /	0	0	0	0	0	0	0	0	0	0	0	0	0
52 /	0	0	0	0	0	0	0	0	0	0	0	0	0
53 /	0	0	0	0	0	0	0	0	0	0	0	0	0
54 /	0	0	0	0	0	0	0	0	0	0	0	0	0
55 /	0	0	0	0	0	0	0	0	0	0	0	0	0
56 /	0	0	0	0	0	0	0	0	0	0	0	0	0
57 /	0	0	0	0	0	0	0	0	0	0	0	0	0
58 /	0	0	0	0	0	0	0	0	0	0	0	0	0
59 /	0	0	0	0	0	0	0	0	0	0	0	0	0
60 /	0	0	0	0	0	0	0	0	0	0	0	0	0
61 /	0	0	0	0	0	0	0	0	0	0	0	0	0
62 /	0	0	0	0	0	0	0	0	0	0	0	0	0
63 /	0	0	0	0	0	0	0	0	0	0	0	0	0

HISTOGRAM ANALYSIS

FUZZ FACTOR	MEAN	STANDARD DEVIATION
0	0.78	6.20
1	0.77	4.90
2	0.80	4.23
3	0.80	3.86
4	0.80	3.63
5	0.77	3.14
6	0.81	2.83
7	0.80	2.51
8	0.77	2.34
9	0.78	2.07
10	0.78	1.90
11	0.78	1.77
12	0.75	1.61

EXAMPLE 2 "SMALL BLOCK B"

```
                    THE FOCUSED PICTURE
-----------------------------------------------------------

 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50  0  0 50 50 50 50 50 50 50
 50 50 50 50 50 50 50  0  0 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

EXAMPLE 2   BLURRED IMAGES OF "SMALL BLOCK B"

```
                FUZZ FACTOR = 1                                        FUZZ FACTOR = 3
-----------------------------------------------        -----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 45 45 50 50 50 50 50 50 50        50 50 50 50 50 50 47 40 40 47 50 50 50 50 50 50
50 50 50 50 50 50 45 11 11 45 50 50 50 50 50 50        50 50 50 50 50 50 40 22 22 40 50 50 50 50 50 50
50 50 50 50 50 50 45 11 11 45 50 50 50 50 50 50        50 50 50 50 50 50 40 22 22 40 50 50 50 50 50 50
50 50 50 50 50 50 50 45 45 50 50 50 50 50 50 50        50 50 50 50 50 50 47 40 40 47 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50

                FUZZ FACTOR = 2                                        FUZZ FACTOR = 4
-----------------------------------------------        -----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 49 41 41 49 50 50 50 50 50 50        50 50 50 50 50 50 46 39 39 46 50 50 50 50 50 50
50 50 50 50 50 50 41 18 18 41 50 50 50 50 50 50        50 50 50 50 50 50 39 25 25 39 50 50 50 50 50 50
50 50 50 50 50 50 41 18 18 41 50 50 50 50 50 50        50 50 50 50 50 50 39 25 25 39 50 50 50 50 50 50
50 50 50 50 50 50 49 41 41 49 50 50 50 50 50 50        50 50 50 50 50 50 46 39 39 46 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50        50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

EXAMPLE 2   BLURRED IMAGES OF "SMALL BLOCK B"   (continued)

```
                  FUZZ FACTOR = 5
-----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 49 49 50 50 50 50 50 50 50
50 50 50 50 50 50 45 39 39 45 50 50 50 50 50 50
50 50 50 50 50 49 39 30 30 39 49 50 50 50 50 50
50 50 50 50 50 49 39 30 30 39 49 50 50 50 50 50
50 50 50 50 50 50 45 39 39 45 50 50 50 50 50 50
50 50 50 50 50 50 50 49 49 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

```
                  FUZZ FACTOR = 6
-----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 49 47 47 49 50 50 50 50 50 50
50 50 50 50 50 49 44 39 39 44 49 50 50 50 50 50
50 50 50 50 50 47 39 34 34 39 47 50 50 50 50 50
50 50 50 50 50 47 39 34 34 39 47 50 50 50 50 50
50 50 50 50 50 49 44 39 39 44 49 50 50 50 50 50
50 50 50 50 50 50 49 47 47 49 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

```
                  FUZZ FACTOR = 7
-----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 48 46 46 48 50 50 50 50 50 50
50 50 50 50 50 48 44 40 40 44 48 50 50 50 50 50
50 50 50 50 50 46 40 37 37 40 46 50 50 50 50 50
50 50 50 50 50 46 40 37 37 40 46 50 50 50 50 50
50 50 50 50 50 48 44 40 40 44 48 50 50 50 50 50
50 50 50 50 50 50 48 46 46 48 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

```
                  FUZZ FACTOR = 8
-----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 48 46 46 48 50 50 50 50 50 50
50 50 50 50 50 48 43 40 40 43 48 50 50 50 50 50
50 50 50 50 50 46 40 40 40 40 46 50 50 50 50 50
50 50 50 50 50 46 40 40 40 40 46 50 50 50 50 50
50 50 50 50 50 48 43 40 40 43 48 50 50 50 50 50
50 50 50 50 50 50 48 46 46 48 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

EXAMPLE 2   BLURRED IMAGES OF "SMALL BLOCK B"   (continued)

```
                 FUZZ FACTOR = 9
-----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 49 49 50 50 50 50 50 50 50
50 50 50 50 50 49 47 45 45 47 49 50 50 50 50 50
50 50 50 50 50 47 43 42 42 43 47 50 50 50 50 50
50 50 50 50 49 45 42 42 42 42 45 49 50 50 50 50
50 50 50 50 49 45 42 42 42 42 45 49 50 50 50 50
50 50 50 50 50 47 43 42 42 43 47 50 50 50 50 50
50 50 50 50 50 49 47 45 45 47 49 50 50 50 50 50
50 50 50 50 50 50 50 49 49 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

```
                 FUZZ FACTOR = 10
-----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 49 49 49 49 50 50 50 50 50 50
50 50 50 50 50 49 46 45 45 46 49 50 50 50 50 50
50 50 50 50 49 46 44 43 43 44 46 49 50 50 50 50
50 50 50 50 49 45 43 43 43 43 45 49 50 50 50 50
50 50 50 50 49 45 43 43 43 43 45 49 50 50 50 50
50 50 50 50 49 46 44 43 43 44 46 49 50 50 50 50
50 50 50 50 50 49 46 45 45 46 49 50 50 50 50 50
50 50 50 50 50 50 49 49 49 49 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

```
                 FUZZ FACTOR = 11
-----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 49 48 48 49 50 50 50 50 50 50
50 50 50 50 50 48 46 45 45 46 48 50 50 50 50 50
50 50 50 50 49 46 44 44 44 44 46 49 50 50 50 50
50 50 50 50 48 45 44 44 44 44 45 48 50 50 50 50
50 50 50 50 48 45 44 44 44 44 45 48 50 50 50 50
50 5C 50 50 49 46 44 44 44 44 46 49 50 50 50 50
50 50 50 50 50 48 46 45 45 46 48 50 50 50 50 50
50 50 50 50 50 50 49 48 48 49 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

```
                 FUZZ FACTOR = 12
-----------------------------------------------

50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 48 48 48 48 50 50 50 50 50 50
50 50 50 50 50 48 46 45 45 46 48 50 50 50 50 50
50 50 50 50 48 46 45 45 45 45 46 48 50 50 50 50
50 50 50 50 48 45 45 45 45 45 45 48 50 50 50 50
50 50 50 50 48 45 45 45 45 45 45 48 50 50 50 50
50 50 50 50 48 46 45 45 45 45 46 48 50 50 50 50
50 50 50 50 50 48 46 45 45 46 48 50 50 50 50 50
50 50 50 50 50 50 48 48 48 48 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
```

INTENSITY HISTOGRAMS FOR EXAMPLE 2

	FUZZ FACTOR												
I /	0	1	2	3	4	5	6	7	8	9	10	11	12
0 /	4	0	0	0	0	0	0	0	0	0	0	0	0
1 /	0	0	0	0	0	0	0	0	0	0	0	0	0
2 /	0	0	0	0	0	0	0	0	0	0	0	0	0
3 /	0	0	0	0	0	0	0	0	0	0	0	0	0
4 /	0	0	0	0	0	0	0	0	0	0	0	0	0
5 /	0	0	0	0	0	0	0	0	0	0	0	0	0
6 /	0	0	0	0	0	0	0	0	0	0	0	0	0
7 /	0	0	0	0	0	0	0	0	0	0	0	0	0
8 /	0	0	0	0	0	0	0	0	0	0	0	0	0
9 /	0	0	0	0	0	0	0	0	0	0	0	0	0
10 /	0	0	0	0	0	0	0	0	0	0	0	0	0
11 /	0	4	0	0	0	0	0	0	0	0	0	0	0
12 /	0	0	0	0	0	0	0	0	0	0	0	0	0
13 /	0	0	0	0	0	0	0	0	0	0	0	0	0
14 /	0	0	0	0	0	0	0	0	0	0	0	0	0
15 /	0	0	0	0	0	0	0	0	0	0	0	0	0
16 /	0	0	0	0	0	0	0	0	0	0	0	0	0
17 /	0	0	0	0	0	0	0	0	0	0	0	0	0
18 /	0	0	4	0	0	0	0	0	0	0	0	0	0
19 /	0	0	0	0	0	0	0	0	0	0	0	0	0
20 /	0	0	0	0	0	0	0	0	0	0	0	0	0
21 /	0	0	0	0	0	0	0	0	0	0	0	0	0
22 /	0	0	0	4	0	0	0	0	0	0	0	0	0
23 /	0	0	0	0	0	0	0	0	0	0	0	0	0
24 /	0	0	0	0	0	0	0	0	0	0	0	0	0
25 /	0	0	0	0	4	0	0	0	0	0	0	0	0
26 /	0	0	0	0	0	0	0	0	0	0	0	0	0
27 /	0	0	0	0	0	0	0	0	0	0	0	0	0
28 /	0	0	0	0	0	0	0	0	0	0	0	0	0
29 /	0	0	0	0	0	0	0	0	0	0	0	0	0
30 /	0	0	0	0	0	4	0	0	0	0	0	0	0
31 /	0	0	0	0	0	0	0	0	0	0	0	0	0
32 /	0	0	0	0	0	0	0	0	0	0	0	0	0
33 /	0	0	0	0	0	0	0	0	0	0	0	0	0
34 /	0	0	0	0	0	0	4	0	0	0	0	0	0
35 /	0	0	0	0	0	0	0	0	0	0	0	0	0
36 /	0	0	0	0	0	0	0	0	0	0	0	0	0
37 /	0	0	0	0	0	0	0	4	0	0	0	0	0
38 /	0	0	0	0	0	0	0	0	0	0	0	0	0
39 /	0	0	0	0	8	8	8	0	0	0	0	0	0
40 /	0	0	0	8	0	0	0	8	12	0	0	0	0
41 /	0	0	8	0	0	0	0	0	0	0	0	0	0
42 /	0	0	0	0	0	0	0	0	0	12	0	0	0
43 /	0	0	0	0	0	0	0	0	4	4	12	0	0
44 /	0	0	0	0	0	0	4	4	0	0	4	16	0

```
          INTENSITY HISTOGRAMS (CONTINUED)
------------------------------------------------------------

 45 /  0    8    0    0    0    4    0    0    0    8    8    8   24
 46 /  0    0    0    0    4    0    0    8    8    0    8    8    8
 47 /  0    0    0    4    0    0    8    0    0    8    0    0    0
 48 /  0    0    0    0    0    0    0    8    8    0    0   12   20
 49 /  0    0    4    0    0    8    8    0    0   12   20    8    0
 50 /252  244  240  240  240  232  224  224  224  212  204  204  204
 51 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 52 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 53 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 54 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 55 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 56 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 57 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 58 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 59 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 60 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 61 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 62 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 63 /  0    0    0    0    0    0    0    0    0    0    0    0    0


          HISTOGRAM ANALYSIS
------------------------------------------------------------

      FUZZ FACTOR      MEAN      STANDARD DEVIATION
------------------------------------------------------------
           0          49.22            6.20
           1          49.23            4.90
           2          49.20            4.23
           3          49.20            3.86
           4          49.20            3.63
           5          49.23            3.14
           6          49.19            2.83
           7          49.20            2.51
           8          49.23            2.34
           9          49.22            2.07
          10          49.22            1.90
          11          49.22            1.77
          12          49.25            1.61
```

EXAMPLE 3 "LARGE BLOCK"

```
                    THE FOCUSED PICTURE
 ---------------------------------------------------------

  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

EXAMPLE 3   BLURRED IMAGES OF "LARGE BLOCK"

```
                FUZZ FACTOR = 1
------------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  5  5  5  5  5  5  5  5  5  5  0  0  0
 0  0  5 39 45 45 45 45 45 45 45 45 39  5  0  0
 0  0  5 45 50 50 50 50 50 50 50 50 45  5  0  0
 0  0  5 45 50 50 50 50 50 50 50 50 45  5  0  0
 0  0  5 45 50 50 50 50 50 50 50 50 45  5  0  0
 0  0  5 45 50 50 50 50 50 50 50 50 45  5  0  0
 0  0  5 45 50 50 50 50 50 50 50 50 45  5  0  0
 0  0  5 45 50 50 50 50 50 50 50 50 45  5  0  0
 0  0  5 45 50 50 50 50 50 50 50 50 45  5  0  0
 0  0  5 45 50 50 50 50 50 50 50 50 45  5  0  0
 0  0  5 39 45 45 45 45 45 45 45 45 39  5  0  0
 0  0  0  5  5  5  5  5  5  5  5  5  5  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 3
------------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  3 10 13 13 13 13 13 13 13 13 10  3  0  0
 0  0 10 28 38 38 38 38 38 38 38 38 28 10  0  0
 0  0 13 38 50 50 50 50 50 50 50 50 38 13  0  0
 0  0 13 38 50 50 50 50 50 50 50 50 38 13  0  0
 0  0 13 38 50 50 50 50 50 50 50 50 38 13  0  0
 0  0 13 38 50 50 50 50 50 50 50 50 38 13  0  0
 0  0 13 38 50 50 50 50 50 50 50 50 38 13  0  0
 0  0 13 38 50 50 50 50 50 50 50 50 38 13  0  0
 0  0 13 38 50 50 50 50 50 50 50 50 38 13  0  0
 0  0 13 38 50 50 50 50 50 50 50 50 38 13  0  0
 0  0 10 28 38 38 38 38 38 38 38 38 28 10  0  0
 0  0  3 10 13 13 13 13 13 13 13 13 10  3  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 2
------------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  1  9 10 10 10 10 10 10 10 10  9  1  0  0
 0  0  9 32 40 40 40 40 40 40 40 40 32  9  0  0
 0  0 10 40 50 50 50 50 50 50 50 50 40 10  0  0
 0  0 10 40 50 50 50 50 50 50 50 50 40 10  0  0
 0  0 10 40 50 50 50 50 50 50 50 50 40 10  0  0
 0  0 10 40 50 50 50 50 50 50 50 50 40 10  0  0
 0  0 10 40 50 50 50 50 50 50 50 50 40 10  0  0
 0  0 10 40 50 50 50 50 50 50 50 50 40 10  0  0
 0  0 10 40 50 50 50 50 50 50 50 50 40 10  0  0
 0  0 10 40 50 50 50 50 50 50 50 50 40 10  0  0
 0  0  9 32 40 40 40 40 40 40 40 40 32  9  0  0
 0  0  1  9 10 10 10 10 10 10 10 10  9  1  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 4
------------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  4 11 15 15 15 15 15 15 15 15 11  4  0  0
 0  0 11 25 35 35 35 35 35 35 35 35 25 11  0  0
 0  0 15 35 50 50 50 50 50 50 50 50 35 15  0  0
 0  0 15 35 50 50 50 50 50 50 50 50 35 15  0  0
 0  0 15 35 50 50 50 50 50 50 50 50 35 15  0  0
 0  0 15 35 50 50 50 50 50 50 50 50 35 15  0  0
 0  0 15 35 50 50 50 50 50 50 50 50 35 15  0  0
 0  0 15 35 50 50 50 50 50 50 50 50 35 15  0  0
 0  0 15 35 50 50 50 50 50 50 50 50 35 15  0  0
 0  0 15 35 50 50 50 50 50 50 50 50 35 15  0  0
 0  0 11 25 35 35 35 35 35 35 35 35 25 11  0  0
 0  0  4 11 15 15 15 15 15 15 15 15 11  4  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

EXAMPLE 3   BLURRED IMAGES OF "LARGE BLOCK"   (continued)

```
                 FUZZ FACTOR = 5
------------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  1  2  2  2  2  2  2  2  2  1  0  0  0
  0  0  5 11 16 16 16 16 16 16 16 16 11  5  0  0
  0  1 11 23 33 34 34 34 34 34 34 33 23 11  1  0
  0  2 16 33 47 48 48 48 48 48 48 47 33 16  2  0
  0  2 16 34 48 50 50 50 50 50 50 48 34 16  2  0
  0  2 16 34 48 50 50 50 50 50 50 48 34 16  2  0
  0  2 16 34 48 50 50 50 50 50 50 48 34 16  2  0
  0  2 16 34 48 50 50 50 50 50 50 48 34 16  2  0
  0  2 16 34 48 50 50 50 50 50 50 48 34 16  2  0
  0  2 16 34 48 50 50 50 50 50 50 48 34 16  2  0
  0  2 16 33 47 48 48 48 48 48 48 47 33 16  2  0
  0  1 11 23 33 34 34 34 34 34 34 33 23 11  1  0
  0  0  5 11 16 16 16 16 16 16 16 16 11  5  0  0
  0  0  0  1  2  2  2  2  2  2  2  2  1  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                 FUZZ FACTOR = 7
------------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  2  4  5  6  6  6  6  6  6  5  4  2  0  0
  0  2  6 12 16 18 18 18 18 18 18 16 12  6  2  0
  0  4 12 20 28 32 32 32 32 32 32 28 20 12  4  0
  0  5 16 28 39 44 44 44 44 44 44 39 28 16  5  0
  0  6 18 32 44 50 50 50 50 50 50 44 32 18  6  0
  0  6 18 32 44 50 50 50 50 50 50 44 32 18  6  0
  0  6 18 32 44 50 50 50 50 50 50 44 32 18  6  0
  0  6 18 32 44 50 50 50 50 50 50 44 32 18  6  0
  0  6 18 32 44 50 50 50 50 50 50 44 32 18  6  0
  0  6 18 32 44 50 50 50 50 50 50 44 32 18  6  0
  0  5 16 28 39 44 44 44 44 44 44 39 28 16  5  0
  0  4 12 20 28 32 32 32 32 32 32 28 20 12  4  0
  0  2  6 12 16 18 18 18 18 18 18 16 12  6  2  0
  0  0  2  4  5  6  6  6  6  6  6  5  4  2  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                 FUZZ FACTOR = 6
------------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  1  3  4  4  4  4  4  4  4  4  3  1  0  0
  0  1  6 12 16 17 17 17 17 17 17 16 12  6  1  0
  0  3 12 21 30 33 33 33 33 33 33 30 21 12  3  0
  0  4 16 30 43 46 46 46 46 46 46 43 30 16  4  0
  0  4 17 33 46 50 50 50 50 50 50 46 33 17  4  0
  0  4 17 33 46 50 50 50 50 50 50 46 33 17  4  0
  0  4 17 33 46 50 50 50 50 50 50 46 33 17  4  0
  0  4 17 33 46 50 50 50 50 50 50 46 33 17  4  0
  0  4 17 33 46 50 50 50 50 50 50 46 33 17  4  0
  0  4 17 33 46 50 50 50 50 50 50 46 33 17  4  0
  0  4 16 30 43 46 46 46 46 46 46 43 30 16  4  0
  0  3 12 21 30 33 33 33 33 33 33 30 21 12  3  0
  0  1  6 12 16 17 17 17 17 17 17 16 12  6  1  0
  0  0  1  3  4  4  4  4  4  4  4  4  3  1  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                 FUZZ FACTOR = 8
------------------------------------------------

  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  2  5  7  7  7  7  7  7  7  7  5  2  0  0
  0  2  7 12 16 19 19 19 19 19 19 16 12  7  2  0
  0  5 12 19 27 31 31 31 31 31 31 27 19 12  5  0
  0  7 16 27 36 43 43 43 43 43 43 36 27 16  7  0
  0  7 19 31 43 50 50 50 50 50 50 43 31 19  7  0
  0  7 19 31 43 50 50 50 50 50 50 43 31 19  7  0
  0  7 19 31 43 50 50 50 50 50 50 43 31 19  7  0
  0  7 19 31 43 50 50 50 50 50 50 43 31 19  7  0
  0  7 19 31 43 50 50 50 50 50 50 43 31 19  7  0
  0  7 19 31 43 50 50 50 50 50 50 43 31 19  7  0
  0  7 16 27 36 43 43 43 43 43 43 36 27 16  7  0
  0  5 12 19 27 31 31 31 31 31 31 27 19 12  5  0
  0  2  7 12 16 19 19 19 19 19 19 16 12  7  2  0
  0  0  2  5  7  7  7  7  7  7  7  7  5  2  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

EXAMPLE 3   BLURRED IMAGES OF "LARGE BLOCK"   (continued)

```
                FUZZ FACTOR = 9
-----------------------------------------------

 0  0  0  1  1  1  1  1  1  1  1  1  1  0  0  0
 0  1  3  6  8  8  8  8  8  8  8  8  6  3  1  0
 0  3  7 12 16 19 19 19 19 19 19 16 12  7  3  0
 1  6 12 19 25 30 31 31 31 31 30 25 19 12  6  1
 1  8 16 25 34 41 42 42 42 42 41 34 25 16  8  1
 1  8 19 30 41 48 49 49 49 49 48 41 30 19  8  1
 1  8 19 31 42 49 50 50 50 50 49 42 31 19  8  1
 1  8 19 31 42 49 50 50 50 50 49 42 31 19  8  1
 1  8 19 31 42 49 50 50 50 50 49 42 31 19  8  1
 1  8 19 31 42 49 50 50 50 50 49 42 31 19  8  1
 1  8 19 30 41 48 49 49 49 49 48 41 30 19  8  1
 1  8 16 25 34 41 42 42 42 42 41 34 25 16  8  1
 1  6 12 19 25 30 31 31 31 31 30 25 19 12  6  1
 0  3  7 12 16 19 19 19 19 19 19 16 12  7  3  0
 0  1  3  6  8  8  8  8  8  8  8  8  6  3  1  0
 0  0  0  1  1  1  1  1  1  1  1  1  1  0  0  0
```

```
                FUZZ FACTOR = 11
-----------------------------------------------

 0  0  1  2  3  3  3  3  3  3  3  3  2  1  0  0
 0  2  4  7  9 11 11 11 11 11 11  9  7  4  2  0
 1  4  8 12 16 19 20 20 20 20 19 16 12  8  4  1
 2  7 12 18 23 28 30 30 30 30 28 23 18 12  7  2
 3  9 16 23 30 36 39 39 39 39 36 30 23 16  9  3
 3 11 19 28 36 44 47 47 47 47 44 36 28 19 11  3
 3 11 20 30 39 47 50 50 50 50 47 39 30 20 11  3
 3 11 20 30 39 47 50 50 50 50 47 39 30 20 11  3
 3 11 20 30 39 47 50 50 50 50 47 39 30 20 11  3
 3 11 20 30 39 47 50 50 50 50 47 39 30 20 11  3
 3 11 19 28 36 44 47 47 47 47 44 36 28 19 11  3
 3  9 16 23 30 36 39 39 39 39 36 30 23 16  9  3
 2  7 12 18 23 28 30 30 30 30 28 23 18 12  7  2
 1  4  8 12 16 19 20 20 20 20 19 16 12  8  4  1
 0  2  4  7  9 11 11 11 11 11 11  9  7  4  2  0
 0  0  1  2  3  3  3  3  3  3  3  3  2  1  0  0
```

```
                FUZZ FACTOR = 10
-----------------------------------------------

 0  0  1  1  2  2  2  2  2  2  2  2  1  1  0  0
 0  1  4  6  9 10 10 10 10 10 10  9  6  4  1  0
 1  4  8 12 16 19 20 20 20 20 19 16 12  8  4  1
 1  6 12 18 24 29 30 30 30 30 29 24 18 12  6  1
 2  9 16 24 32 38 40 40 40 40 38 32 24 16  9  2
 2 10 19 29 38 46 48 48 48 48 46 38 29 19 10  2
 2 10 20 30 40 48 50 50 50 50 48 40 30 20 10  2
 2 10 20 30 40 48 50 50 50 50 48 40 30 20 10  2
 2 10 20 30 40 48 50 50 50 50 48 40 30 20 10  2
 2 10 20 30 40 48 50 50 50 50 48 40 30 20 10  2
 2 10 19 29 38 46 48 48 48 48 46 38 29 19 10  2
 2  9 16 24 32 38 40 40 40 40 38 32 24 16  9  2
 1  6 12 18 24 29 30 30 30 30 29 24 18 12  6  1
 1  4  8 12 16 19 20 20 20 20 19 16 12  8  4  1
 0  1  4  6  9 10 10 10 10 10 10  9  6  4  1  0
 0  0  1  1  2  2  2  2  2  2  2  2  1  1  0  0
```

```
                FUZZ FACTOR = 12
-----------------------------------------------

 0  0  2  3  4  4  4  4  4  4  4  4  3  2  0  0
 0  2  5  7 10 11 12 12 12 12 11 10  7  5  2  0
 2  5  8 12 16 19 20 20 20 20 19 16 12  8  5  2
 3  7 12 17 22 27 30 30 30 30 27 22 17 12  7  3
 4 10 16 22 29 34 38 38 38 38 34 29 22 16 10  4
 4 11 19 27 34 41 46 46 46 46 41 34 27 19 11  4
 4 12 20 30 38 46 50 50 50 50 46 38 30 20 12  4
 4 12 20 30 38 46 50 50 50 50 46 38 30 20 12  4
 4 12 20 30 38 46 50 50 50 50 46 38 30 20 12  4
 4 12 20 30 38 46 50 50 50 50 46 38 30 20 12  4
 4 11 19 27 34 41 46 46 46 46 41 34 27 19 11  4
 4 10 16 22 29 34 38 38 38 38 34 29 22 16 10  4
 3  7 12 17 22 27 30 30 30 30 27 22 17 12  7  3
 2  5  8 12 16 19 20 20 20 20 19 16 12  8  5  2
 0  2  5  7 10 11 12 12 12 12 11 10  7  5  2  0
 0  0  2  3  4  4  4  4  4  4  4  4  3  2  0  0
```

INTENSITY HISTOGRAMS FOR EXAMPLE 3

		FUZZ FACTOR												
I	/	0	1	2	3	4	5	6	7	8	9	10	11	12
0	/	156	116	112	112	112	72	64	64	64	20	12	12	12
1	/	0	0	4	0	0	8	8	0	0	44	20	8	0
2	/	0	0	0	0	0	32	0	8	8	0	32	12	12
3	/	0	0	0	4	0	0	8	0	0	8	0	32	8
4	/	0	0	0	0	4	0	32	8	0	0	8	8	32
5	/	0	40	0	0	0	4	0	8	8	0	0	0	8
6	/	0	0	0	0	0	0	4	28	0	8	8	0	0
7	/	0	0	0	0	0	0	0	0	36	4	0	8	8
8	/	0	0	0	0	0	0	0	0	0	32	4	4	4
9	/	0	0	8	0	0	0	0	0	0	0	8	8	0
10	/	0	0	32	8	0	0	0	0	0	0	24	0	8
11	/	0	0	0	0	8	8	0	0	0	0	0	24	8
12	/	0	0	0	0	0	0	8	8	8	8	8	8	24
13	/	0	0	0	32	0	0	0	0	0	0	0	0	0
14	/	0	0	0	0	0	0	0	0	0	0	0	0	0
15	/	0	0	0	0	32	0	0	0	0	0	0	0	0
16	/	0	0	0	0	0	32	8	8	8	8	8	8	8
17	/	0	0	0	0	0	0	24	0	0	0	0	0	4
18	/	0	0	0	0	0	0	0	24	0	0	4	4	0
19	/	0	0	0	0	0	0	0	0	28	28	8	8	8
20	/	0	0	0	0	0	0	0	4	0	0	16	16	16
21	/	0	0	0	0	0	0	4	0	0	0	0	0	0
22	/	0	0	0	0	0	0	0	0	0	0	0	0	8
23	/	0	0	0	0	0	4	0	0	0	0	0	8	0
24	/	0	0	0	0	0	0	0	0	0	0	8	0	0
25	/	0	0	0	0	4	0	0	0	0	8	0	0	0
26	/	0	0	0	0	0	0	0	0	0	0	0	0	0
27	/	0	0	0	0	0	0	0	0	8	0	0	0	8
28	/	0	0	0	4	0	0	0	8	0	0	0	8	0
29	/	0	0	0	0	0	0	0	0	0	0	8	0	4
30	/	0	0	0	0	0	0	8	0	0	8	16	20	16
31	/	0	0	0	0	0	0	0	0	24	16	0	0	0
32	/	0	0	4	0	0	0	0	24	0	0	4	0	0
33	/	0	0	0	0	0	8	24	0	0	0	0	0	0
34	/	0	0	0	0	0	24	0	0	0	4	0	0	8
35	/	0	0	0	0	32	0	0	0	0	0	0	0	0
36	/	0	0	0	0	0	0	0	0	4	0	0	8	0
37	/	0	0	0	0	0	0	0	0	0	0	0	0	0
38	/	0	0	0	32	0	0	0	0	0	0	8	0	16
39	/	0	4	0	0	0	0	0	4	0	0	0	16	0
40	/	0	0	32	0	0	0	0	0	0	0	16	0	0
41	/	0	0	0	0	0	0	0	0	0	8	0	0	4
42	/	0	0	0	0	0	0	0	0	0	16	0	0	0
43	/	0	0	0	0	0	0	4	0	24	0	0	0	0
44	/	0	0	0	0	0	0	0	24	0	0	0	4	0

INTENSITY HISTOGRAMS (CONTINUED)

45 /	0	32	0	0	0	0	0	0	0	0	0	0	0
46 /	0	0	0	0	0	0	24	0	0	0	4	0	16
47 /	0	0	0	0	0	4	0	0	0	0	0	16	0
48 /	0	0	0	0	0	24	0	0	0	4	16	0	0
49 /	0	0	0	0	0	0	0	0	0	16	0	0	0
50 /	100	64	64	64	64	36	36	36	36	16	16	16	16
51 /	0	0	0	0	0	0	0	0	0	0	0	0	0
52 /	0	0	0	0	0	0	0	0	0	0	0	0	0
53 /	0	0	0	0	0	0	0	0	0	0	0	0	0
54 /	0	0	0	0	0	0	0	0	0	0	0	0	0
55 /	0	0	0	0	0	0	0	0	0	0	0	0	0
56 /	0	0	0	0	0	0	0	0	0	0	0	0	0
57 /	0	0	0	0	0	0	0	0	0	0	0	0	0
58 /	0	0	0	0	0	0	0	0	0	0	0	0	0
59 /	0	0	0	0	0	0	0	0	0	0	0	0	0
60 /	0	0	0	0	0	0	0	0	0	0	0	0	0
61 /	0	0	0	0	0	0	0	0	0	0	0	0	0
62 /	0	0	0	0	0	0	0	0	0	0	0	0	0
63 /	0	0	0	0	0	0	0	0	0	0	0	0	0

HISTOGRAM ANALYSIS

FUZZ FACTOR	MEAN	STANDARD DEVIATION
0	19.53	24.39
1	19.52	22.91
2	19.55	21.77
3	19.67	21.33
4	19.55	20.93
5	19.55	20.18
6	19.56	19.38
7	19.52	18.73
8	19.53	18.34
9	19.55	17.64
10	19.55	16.79
11	19.53	16.19
12	19.52	15.57

EXAMPLE 4 "LARGE SQUARE"

```
                    THE FOCUSED PICTURE
 ------------------------------------------------------------

   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
   0   0   0  50   0   0   0   0   0   0   0   0  50   0   0   0
   0   0   0  50   0   0   0   0   0   0   0   0  50   0   0   0
   0   0   0  50   0   0   0   0   0   0   0   0  50   0   0   0
   0   0   0  50   0   0   0   0   0   0   0   0  50   0   0   0
   0   0   0  50   0   0   0   0   0   0   0   0  50   0   0   0
   0   0   0  50   0   0   0   0   0   0   0   0  50   0   0   0
   0   0   0  50   0   0   0   0   0   0   0   0  50   0   0   0
   0   0   0  50   0   0   0   0   0   0   0   0  50   0   0   0
   0   0   0  50  50  50  50  50  50  50  50  50  50   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

EXAMPLE 4   BLURRED IMAGES OF "LARGE SQUARE"

```
                  FUZZ FACTOR = 1
-----------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  5  5  5  5  5  5  5  5  5  5  0  0  0
 0  0  5 39 39 39 39 39 39 39 39 39 39  5  0  0
 0  0  5 39 11  5  5  5  5  5  5 11 39  5  0  0
 0  0  5 39  5  0  0  0  0  0  0  5 39  5  0  0
 0  0  5 39  5  0  0  0  0  0  0  5 39  5  0  0
 0  0  5 39  5  0  0  0  0  0  0  5 39  5  0  0
 0  0  5 39  5  0  0  0  0  0  0  5 39  5  0  0
 0  0  5 39  5  0  0  0  0  0  0  5 39  5  0  0
 0  0  5 39  5  0  0  0  0  0  0  5 39  5  0  0
 0  0  5 39 11  5  5  5  5  5  5 11 39  5  0  0
 0  0  5 39 39 39 39 39 39 39 39 39 39  5  0  0
 0  0  0  5  5  5  5  5  5  5  5  5  5  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                  FUZZ FACTOR = 3
-----------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  3 10 13 13 13 13 13 13 13 13 10  3  0  0
 0  0 10 25 28 25 25 25 25 25 25 28 25 10  0  0
 0  0 13 28 22 13 13 13 13 13 13 22 28 13  0  0
 0  0 13 25 13  0  0  0  0  0  0 13 25 13  0  0
 0  0 13 25 13  0  0  0  0  0  0 13 25 13  0  0
 0  0 13 25 13  0  0  0  0  0  0 13 25 13  0  0
 0  0 13 25 13  0  0  0  0  0  0 13 25 13  0  0
 0  0 13 25 13  0  0  0  0  0  0 13 25 13  0  0
 0  0 13 25 13  0  0  0  0  0  0 13 25 13  0  0
 0  0 13 28 22 13 13 13 13 13 13 22 28 13  0  0
 0  0 10 25 28 25 25 25 25 25 25 28 25 10  0  0
 0  0  3 10 13 13 13 13 13 13 13 13 10  3  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                  FUZZ FACTOR = 2
-----------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  1  9 10 10 10 10 10 10 10 10  9  1  0  0
 0  0  9 30 32 30 30 30 30 30 30 32 30  9  0  0
 0  0 10 32 18 10 10 10 10 10 10 18 32 10  0  0
 0  0 10 30 10  0  0  0  0  0  0 10 30 10  0  0
 0  0 10 30 10  0  0  0  0  0  0 10 30 10  0  0
 0  0 10 30 10  0  0  0  0  0  0 10 30 10  0  0
 0  0 10 30 10  0  0  0  0  0  0 10 30 10  0  0
 0  0 10 30 10  0  0  0  0  0  0 10 30 10  0  0
 0  0 10 30 10  0  0  0  0  0  0 10 30 10  0  0
 0  0 10 32 18 10 10 10 10 10 10 18 32 10  0  0
 0  0  9 30 32 30 30 30 30 30 30 32 30  9  0  0
 0  0  1  9 10 10 10 10 10 10 10 10  9  1  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                  FUZZ FACTOR = 4
-----------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  4 11 15 15 15 15 15 15 15 15 11  4  0  0
 0  0 11 21 25 21 21 21 21 21 21 25 21 11  0  0
 0  0 15 25 25 15 15 15 15 15 15 25 25 15  0  0
 0  0 15 21 15  0  0  0  0  0  0 15 21 15  0  0
 0  0 15 21 15  0  0  0  0  0  0 15 21 15  0  0
 0  0 15 21 15  0  0  0  0  0  0 15 21 15  0  0
 0  0 15 21 15  0  0  0  0  0  0 15 21 15  0  0
 0  0 15 21 15  0  0  0  0  0  0 15 21 15  0  0
 0  0 15 21 15  0  0  0  0  0  0 15 21 15  0  0
 0  0 15 25 25 15 15 15 15 15 15 25 25 15  0  0
 0  0 11 21 25 21 21 21 21 21 21 25 21 11  0  0
 0  0  4 11 15 15 15 15 15 15 15 15 11  4  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

EXAMPLE 4   BLURRED IMAGES OF "LARGE SQUARE"   (continued)

```
                FUZZ FACTOR = 5
-----------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  1  2  2  2  2  2  2  2  2  1  0  0  0
 0  0  5 11 14 14 14 14 14 14 14 14 11  5  0  0
 0  1 11 18 21 18 18 18 18 18 18 21 18 11  1  0
 0  2 14 21 24 16 14 14 14 14 16 24 21 14  2  0
 0  2 14 18 16  3  2  2  2  2  3 16 18 14  2  0
 0  2 14 18 14  2  0  0  0  0  2 14 18 14  2  0
 0  2 14 18 14  2  0  0  0  0  2 14 18 14  2  0
 0  2 14 18 14  2  0  0  0  0  2 14 18 14  2  0
 0  2 14 18 14  2  0  0  0  0  2 14 18 14  2  0
 0  2 14 18 16  3  2  2  2  2  3 16 18 14  2  0
 0  2 14 21 24 16 14 14 14 14 16 24 21 14  2  0
 0  1 11 18 21 18 18 18 18 18 18 21 18 11  1  0
 0  0  5 11 14 14 14 14 14 14 14 14 11  5  0  0
 0  0  0  1  2  2  2  2  2  2  2  2  1  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 7
-----------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  2  4  5  6  6  6  6  6  6  5  4  2  0. 0
 0  2  6 10 12 13 12 12 12 12 13 12 10  6  2  0
 0  4 10 14 16 16 14 14 14 14 16 16 14 10  4  0
 0  5 12 16 19 16 12 12 12 12 16 19 16 12  5  0
 0  6 13 16 16 11  6  6  6  6 11 16 16 13  6  0
 0  6 12 14 12  6  0  0  0  0  6 12 14 12  6  0
 0  6 12 14 12  6  0  0  0  0  6 12 14 12  6  0
 0  6 12 14 12  6  0  0  0  0  6 12 14 12  6  0
 0  6 12 14 12  6  0  0  0  0  6 12 14 12  6  0
 0  6 13 16 16 11  6  6  6  6 11 16 16 13  6  0
 0  5 12 16 19 16 12 12 12 12 16 19 16 12  5  0
 0  4 10 14 16 16 14 14 14 14 16 16 14 10  4  0
 0  2  6 10 12 13 12 12 12 12 13 12 10  6  2  0
 0  0  2  4  5  6  6  6  6  6  6  5  4  2  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 6
-----------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  1  3  4  4  4  4  4  4  4  4  3  1  0  0
 0  1  6 11 14 14 14 14 14 14 14 14 11  6  1  0
 0  3 11 16 19 17 16 16 16 16 17 19 16 11  3  0
 0  4 14 19 21 16 14 14 14 14 16 21 19 14  4  0
 0  4 14 17 16  7  4  4  4  4  7 16 17 14  4  0
 0  4 14 16 14  4  0  0  0  0  4 14 16 14  4  0
 0  4 14 16 14  4  0  0  0  0  4 14 16 14  4  0
 0  4 14 16 14  4  0  0  0  0  4 14 16 14  4  0
 0  4 14 16 14  4  0  0  0  0  4 14 16 14  4  0
 0  4 14 17 16  7  4  4  4  4  7 16 17 14  4  0
 0  4 14 19 21 16 14 14 14 14 16 21 19 14  4  0
 0  3 11 16 19 17 16 16 16 16 17 19 16 11  3  0
 0  1  6 11 14 14 14 14 14 14 14 14 11  6  1  0
 0  0  1  3  4  4  4  4  4  4  4  4  3  1  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
                FUZZ FACTOR = 8
-----------------------------------------------

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  2  5  7  7  7  7  7  7  7  7  5  2  0  0
 0  2  6 10 12 12 12 12 12 12 12 12 10  6  2  0
 0  5 10 13 15 15 13 13 13 13 15 15 13 10  5  0
 0  7 12 15 17 16 12 12 12 12 16 17 15 12  7  0
 0  7 12 15 16 14  7  7  7  7 14 16 15 12  7  0
 0  7 12 13 12  7  0  0  0  0  7 12 13 12  7  0
 0  7 12 13 12  7  0  0  0  0  7 12 13 12  7  0
 0  7 12 13 12  7  0  0  0  0  7 12 13 12  7  0
 0  7 12 13 12  7  0  0  0  0  7 12 13 12  7  0
 0  7 12 15 16 14  7  7  7  7 14 16 15 12  7  0
 0  7 12 15 17 16 12 12 12 12 16 17 15 12  7  0
 0  5 10 13 15 15 13 13 13 13 15 15 13 10  5  0
 0  2  6 10 12 12 12 12 12 12 12 12 10  6  2  0
 0  0  2  5  7  7  7  7  7  7  7  7  5  2  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

EXAMPLE 4   BLURRED IMAGES OF "LARGE SQUARE"   (continued)

```
                FUZZ FACTOR = 9
------------------------------------------------

  0  0  0  1  1  1  1  1  1  1  1  1  1  0  0  0
  0  1  3  5  7  8  8  8  8  8  8  7  5  3  1  0
  0  3  7  9 11 11 11 11 11 11 11 11  9  7  3  0
  1  5  9 12 13 14 12 12 12 12 14 13 12  9  5  1
  1  7 11 13 15 15 11 11 11 11 15 15 13 11  7  1
  1  8 11 14 15 15  8  8  8  8 15 15 14 11  8  1
  1  8 11 12 11  8  2  1  1  2  8 11 12 11  8  1
  1  8 11 12 11  8  1  0  0  1  8 11 12 11  8  1
  1  8 11 12 11  8  1  0  0  1  8 11 12 11  8  1
  1  8 11 12 11  8  2  1  1  2  8 11 12 11  8  1
  1  8 11 14 15 15  8  8  8  8 15 15 14 11  8  1
  1  7 11 13 15 15 11 11 11 11 15 15 13 11  7  1
  1  5  9 12 13 14 12 12 12 12 14 13 12  9  5  1
  0  3  7  9 11 11 11 11 11 11 11 11  9  7  3  0
  0  1  3  5  7  8  8  8  8  8  8  7  5  3  1  0
  0  0  0  1  1  1  1  1  1  1  1  1  1  0  0  0
```

```
                FUZZ FACTOR = 11
------------------------------------------------

  0  0  1  2  3  3  3  3  3  3  3  3  2  1  0  0
  0  2  4  6  7  8  8  8  8  8  8  7  6  4  2  0
  1  4  6  8  9 10  9  9  9  9 10  9  8  6  4  1
  2  6  8 10 11 12 11 10 10 11 12 11 10  8  6  2
  3  7  9 11 12 13 11  9  9 11 13 12 11  9  7  3
  3  8 10 12 13 13 11  8  8 11 13 13 12 10  8  3
  3  8  9 11 11 11  6  3  3  6 11 11 11  9  8  3
  3  8  9 10  9  8  3  0  0  3  8  9 10  9  8  3
  3  8  9 10  9  8  3  0  0  3  8  9 10  9  8  3
  3  8  9 11 11 11  6  3  3  6 11 11 11  9  8  3
  3  8 10 12 13 13 11  8  8 11 13 13 12 10  8  3
  3  7  9 11 12 13 11  9  9 11 13 12 11  9  7  3
  2  6  8 10 11 12 11 10 10 11 12 11 10  8  6  2
  1  4  6  8  9 10  9  9  9  9 10  9  8  6  4  1
  0  2  4  6  7  8  8  8  8  8  8  7  6  4  2  0
  0  0  1  2  3  3  3  3  3  3  3  3  2  1  0  0
```

```
                FUZZ FACTOR = 10
------------------------------------------------

  0  0  1  1  2  2  2  2  2  2  2  2  1  1  0  0
  0  1  4  6  7  8  8  8  8  8  8  7  6  4  1  0
  1  4  6  8 10 11 10 10 10 10 11 10  8  6  4  1
  1  6  8 11 12 13 11 11 11 11 13 12 11  8  6  1
  2  7 10 12 14 14 11 10 10 11 14 14 12 10  7  2
  2  8 11 13 14 14 10  8  8 10 14 14 13 11  8  2
  2  8 10 11 11 10  4  2  2  4 10 11 11 10  8  2
  2  8 10 11 10  8  2  0  0  2  8 10 11 10  8  2
  2  8 10 11 10  8  2  0  0  2  8 10 11 10  8  2
  2  8 10 11 11 10  4  2  2  4 10 11 11 10  8  2
  2  8 11 13 14 14 10  8  8 10 14 14 13 11  8  2
  2  7 10 12 14 14 11 10 10 11 14 14 12 10  7  2
  1  6  8 11 12 13 11 11 11 11 13 12 11  8  6  1
  1  4  6  8 10 11 10 10 10 10 11 10  8  6  4  1
  0  1  4  6  7  8  8  8  8  8  8  7  6  4  1  0
  0  0  1  1  2  2  2  2  2  2  2  2  1  1  0  0
```

```
                FUZZ FACTOR = 12
------------------------------------------------

  0  0  2  3  4  4  4  4  4  4  4  4  3  2  0  0
  0  2  4  6  7  7  7  7  7  7  7  7  6  4  2  0
  2  4  6  8  9  9  9  9  9  9  9  9  8  6  4  2
  3  6  8  9 10 11 11  9  9 11 11 10  9  8  6  3
  4  7  9 10 11 12 12  9  9 12 12 11 10  9  7  4
  4  7  9 11 12 13 11  7  7 11 13 12 11  9  7  4
  4  7  9 11 12 11  9  4  4  9 11 12 11  9  7  4
  4  7  9  9  9  7  4  0  0  4  7  9  9  9  7  4
  4  7  9  9  9  7  4  0  0  4  7  9  9  9  7  4
  4  7  9 11 12 11  9  4  4  9 11 12 11  9  7  4
  4  7  9 11 12 13 11  7  7 11 13 12 11  9  7  4
  4  7  9 10 11 12 12  9  9 12 12 11 10  9  7  4
  3  6  8  9 10 11 11  9  9 11 11 10  9  8  6  3
  2  4  6  8  9  9  9  9  9  9  9  9  8  6  4  2
  0  2  4  6  7  7  7  7  7  7  7  7  6  4  2  0
  0  0  2  3  4  4  4  4  4  4  4  4  3  2  0  0
```

INTENSITY HISTOGRAMS FOR EXAMPLE 4

	FUZZ FACTOR												
I /	0	1	2	3	4	5	6	7	8	9	10	11	12
0 /	220	152	148	148	148	88	80	80	80	24	16	16	16
1 /	0	0	4	0	0	8	8	0	0	52	20	8	0
2 /	0	0	0	0	0	48	0	8	8	4	40	12	12
3 /	0	0	0	4	0	4	8	0	0	8	0	40	8
4 /	0	0	0	0	4	0	48	8	0	0	12	8	48
5 /	0	64	0	0	0	4	0	8	8	8	0	0	0
6 /	0	0	0	0	0	0	4	44	4	0	12	16	12
7 /	0	0	0	0	0	0	4	0	48	12	8	8	40
8 /	0	0	0	0	0	0	0	0	0	40	40	40	8
9 /	0	0	8	0	0	0	0	0	0	8	0	32	56
10 /	0	0	56	8	0	0	0	8	8	0	40	20	8
11 /	0	4	0	0	8	8	8	4	0	48	36	32	28
12 /	0	0	0	0	0	0	0	40	48	20	8	12	16
13 /	0	0	0	56	0	0	0	8	20	8	8	12	4
14 /	0	0	0	0	0	48	48	20	4	8	16	0	0
15 /	0	0	0	0	56	0	0	0	16	16	0	0	0
16 /	0	0	0	0	0	8	28	24	8	0	0	0	0
17 /	0	0	0	0	0	0	8	0	4	0	0	0	0
18 /	0	0	4	0	0	28	0	0	0	0	0	0	0
19 /	0	0	0	0	0	0	8	4	0	0	0	0	0
20 /	0	0	0	0	0	0	0	0	0	0	0	0	0
21 /	0	0	0	0	28	8	4	0	0	0	0	0	0
22 /	0	0	0	4	0	0	0	0	0	0	0	0	0
23 /	0	0	0	0	0	0	0	0	0	0	0	0	0
24 /	0	0	0	0	0	4	0	0	0	0	0	0	0
25 /	0	0	0	28	12	0	0	0	0	0	0	0	0
26 /	0	0	0	0	0	0	0	0	0	0	0	0	0
27 /	0	0	0	0	0	0	0	0	0	0	0	0	0
28 /	0	0	0	8	0	0	0	0	0	0	0	0	0
29 /	0	0	0	0	0	0	0	0	0	0	0	0	0
30 /	0	0	28	0	0	0	0	0	0	0	0	0	0
31 /	0	0	0	0	0	0	0	0	0	0	0	0	0
32 /	0	0	8	0	0	0	0	0	0	0	0	0	0
33 /	0	0	0	0	0	0	0	0	0	0	0	0	0
34 /	0	0	0	0	0	0	0	0	0	0	0	0	0
35 /	0	0	0	0	0	0	0	0	0	0	0	0	0
36 /	0	0	0	0	0	0	0	0	0	0	0	0	0
37 /	0	0	0	0	0	0	0	0	0	0	0	0	0
38 /	0	0	0	0	0	0	0	0	0	0	0	0	0
39 /	0	36	0	0	0	0	0	0	0	0	0	0	0
40 /	0	0	0	0	0	0	0	0	0	0	0	0	0
41 /	0	0	0	0	0	0	0	0	0	0	0	0	0
42 /	0	0	0	0	0	0	0	0	0	0	0	0	0
43 /	0	0	0	0	0	0	0	0	0	0	0	0	0
44 /	0	0	0	0	0	0	0	0	0	0	0	0	0

```
          INTENSITY HISTOGRAMS (CONTINUED)
--------------------------------------------------------------

45 /  0    0    0    0    0    0    0    0    0    0    0    0    0
46 /  0    0    0    0    0    0    0    0    0    0    0    0    0
47 /  0    0    0    0    0    0    0    0    0    0    0    0    0
48 /  0    0    0    0    0    0    0    0    0    0    0    0    0
49 /  0    0    0    0    0    0    0    0    0    0    0    0    0
50 / 36    0    0    0    0    0    0    0    0    0    0    0    0
51 /  0    0    0    0    0    0    0    0    0    0    0    0    0
52 /  0    0    0    0    0    0    0    0    0    0    0    0    0
53 /  0    0    0    0    0    0    0    0    0    0    0    0    0
54 /  0    0    0    0    0    0    0    0    0    0    0    0    0
55 /  0    0    0    0    0    0    0    0    0    0    0    0    0
56 /  0    0    0    0    0    0    0    0    0    0    0    0    0
57 /  0    0    0    0    0    0    0    0    0    0    0    0    0
58 /  0    0    0    0    0    0    0    0    0    0    0    0    0
59 /  0    0    0    0    0    0    0    0    0    0    0    0    0
60 /  0    0    0    0    0    0    0    0    0    0    0    0    0
61 /  0    0    0    0    0    0    0    0    0    0    0    0    0
62 /  0    0    0    0    0    0    0    0    0    0    0    0    0
63 /  0    0    0    0    0    0    0    0    0    0    0    0    0
```

```
          HISTOGRAM ANALYSIS
--------------------------------------------------------------

     FUZZ FACTOR      MEAN     STANDARD DEVIATION
--------------------------------------------------------------
           0          7.03           17.38
           1          6.91           13.20
           2          7.05           10.50
           3          7.16            9.46
           4          7.16            8.92
           5          7.00            7.69
           6          7.25            6.91
           7          7.03            5.99
           8          7.13            5.73
           9          7.13            4.98
          10          7.09            4.36
          11          7.02            3.80
          12          7.00            3.40
```

EXAMPLE 5   "CHECKERED"

```
                    THE FOCUSED PICTURE
 ------------------------------------------------------

  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0
   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50
  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0
   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50
  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0
   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50
  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0
   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50
  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0
   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50
  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0
   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50
  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0
   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50
  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0
   0  50   0  50   0  50   0  50   0  50   0  50   0  50   0  50
```

EXAMPLE 5   BLURRED IMAGES OF "CHECKERED"

```
               FUZZ FACTOR = 1                                      FUZZ FACTOR = 3
-----------------------------------------------      -----------------------------------------------

28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20
28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20
28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20
28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20
28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20
28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20
28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20
28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      30 20 30 20 30 20 30 20 30 20 30 20 30 20 30 20

               FUZZ FACTOR = 2                                      FUZZ FACTOR = 4
-----------------------------------------------      -----------------------------------------------

21 29 21 29 21 29 21 29 21 29 21 29 21 29 21 29      23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21      27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
21 29 21 29 21 29 21 29 21 29 21 29 21 29 21 29      23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21      27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
21 29 21 29 21 29 21 29 21 29 21 29 21 29 21 29      23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21      27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
21 29 21 29 21 29 21 29 21 29 21 29 21 29 21 29      23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21      27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
21 29 21 29 21 29 21 29 21 29 21 29 21 29 21 29      23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21      27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
21 29 21 29 21 29 21 29 21 29 21 29 21 29 21 29      23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21      27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
21 29 21 29 21 29 21 29 21 29 21 29 21 29 21 29      23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21      27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
21 29 21 29 21 29 21 29 21 29 21 29 21 29 21 29      23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21      27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
```

EXAMPLE 5   BLURRED IMAGES OF "CHECKERED"   (continued)

```
                  FUZZ FACTOR = 5                                    FUZZ FACTOR = 7
 -----------------------------------------------      -----------------------------------------------

 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
 22 28 22 28 22 28 22 28 22 28 22 28 22 28 22 28      25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25

                  FUZZ FACTOR = 6                                    FUZZ FACTOR = 8
 -----------------------------------------------      -----------------------------------------------

 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23      24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27      26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23      24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27      26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23      24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27      26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23      24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27      26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23      24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27      26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23      24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27      26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23      24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27      26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23      24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27      26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
```

EXAMPLE 5   BLURRED IMAGES OF "CHECKERED"   (continued)

```
              FUZZ FACTOR = 9
------------------------------------------------

23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23
```

```
              FUZZ FACTOR = 10
------------------------------------------------

25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
```

```
              FUZZ FACTOR = 11
------------------------------------------------

26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
```

```
              FUZZ FACTOR = 12
------------------------------------------------

26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26
```

INTENSITY HISTOGRAMS FOR EXAMPLE 5

		FUZZ FACTOR												
I	/	0	1	2	3	4	5	6	7	8	9	10	11	12
0	/	128	0	0	0	0	0	0	0	0	0	0	0	0
1	/	0	0	0	0	0	0	0	0	0	0	0	0	0
2	/	0	0	0	0	0	0	0	0	0	0	0	0	0
3	/	0	0	0	0	0	0	0	0	0	0	0	0	0
4	/	0	0	0	0	0	0	0	0	0	0	0	0	0
5	/	0	0	0	0	0	0	0	0	0	0	0	0	0
6	/	0	0	0	0	0	0	0	0	0	0	0	0	0
7	/	0	0	0	0	0	0	0	0	0	0	0	0	0
8	/	0	0	0	0	0	0	0	0	0	0	0	0	0
9	/	0	0	0	0	0	0	0	0	0	0	0	0	0
10	/	0	0	0	0	0	0	0	0	0	0	0	0	0
11	/	0	0	0	0	0	0	0	0	0	0	0	0	0
12	/	0	0	0	0	0	0	0	0	0	0	0	0	0
13	/	0	0	0	0	0	0	0	0	0	0	0	0	0
14	/	0	0	0	0	0	0	0	0	0	0	0	0	0
15	/	0	0	0	0	0	0	0	0	0	0	0	0	0
16	/	0	0	0	0	0	0	0	0	0	0	0	0	0
17	/	0	0	0	0	0	0	0	0	0	0	0	0	0
18	/	0	0	0	0	0	0	0	0	0	0	0	0	0
19	/	0	0	0	0	0	0	0	0	0	0	0	0	0
20	/	0	0	0	128	0	0	0	0	0	0	0	0	0
21	/	0	0	128	0	0	0	0	0	0	0	0	0	0
22	/	0	128	0	0	0	128	0	0	0	0	0	0	0
23	/	0	0	0	0	128	0	128	0	0	128	0	0	0
24	/	0	0	0	0	0	0	0	0	128	0	0	128	128
25	/	0	0	0	0	0	0	0	256	0	0	256	0	0
26	/	0	0	0	0	0	0	0	0	128	0	0	128	128
27	/	0	0	0	0	128	0	128	0	0	128	0	0	0
28	/	0	128	0	0	0	128	0	0	0	0	0	0	0
29	/	0	0	128	0	0	0	0	0	0	0	0	0	0
30	/	0	0	0	128	0	0	0	0	0	0	0	0	0
31	/	0	0	0	0	0	0	0	0	0	0	0	0	0
32	/	0	0	0	0	0	0	0	0	0	0	0	0	0
33	/	0	0	0	0	0	0	0	0	0	0	0	0	0
34	/	0	0	0	0	0	0	0	0	0	0	0	0	0
35	/	0	0	0	0	0	0	0	0	0	0	0	0	0
36	/	0	0	0	0	0	0	0	0	0	0	0	0	0
37	/	0	0	0	0	0	0	0	0	0	0	0	0	0
38	/	0	0	0	0	0	0	0	0	0	0	0	0	0
39	/	0	0	0	0	0	0	0	0	0	0	0	0	0
40	/	0	0	0	0	0	0	0	0	0	0	0	0	0
41	/	0	0	0	0	0	0	0	0	0	0	0	0	0
42	/	0	0	0	0	0	0	0	0	0	0	0	0	0
43	/	0	0	0	0	0	0	0	0	0	0	0	0	0
44	/	0	0	0	0	0	0	0	0	0	0	0	0	0

```
        INTENSITY HISTOGRAMS (CONTINUED)
----------------------------------------------------------

 45 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 46 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 47 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 48 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 49 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 50 /128    0    0    0    0    0    0    0    0    0    0    0    0
 51 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 52 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 53 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 54 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 55 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 56 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 57 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 58 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 59 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 60 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 61 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 62 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 63 /  0    0    0    0    0    0    0    0    0    0    0    0    0

        HISTOGRAM ANALYSIS
----------------------------------------------------------

     FUZZ FACTOR      MEAN      STANDARD DEVIATION
----------------------------------------------------------
             0       25.00             25.00
             1       25.00              3.00
             2       25.00              4.00
             3       25.00              5.00
             4       25.00              2.00
             5       25.00              3.00
             6       25.00              2.00
             7       25.00              0.00
             8       25.00              1.00
             9       25.00              2.00
            10       25.00              0.00
            11       25.00              1.00
            12       25.00              1.00
```

EXAMPLE 6 "STRIPED"

```
                     THE FOCUSED PICTURE
--------------------------------------------------------------

  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
  0 50  0 50  0 50  0 50  0 50  0 50  0 50  0 50
```

EXAMPLE 6   BLURRED IMAGES OF "STRIPED"

```
                FUZZ FACTOR = 1
-----------------------------------------------

11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
11 39 11 39 11 39 11 39 11 39 11 39 11 39 11 39
```

```
                FUZZ FACTOR = 3
-----------------------------------------------

25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
```

```
                FUZZ FACTOR = 2
-----------------------------------------------

20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
20 30 20 30 20 30 20 30 20 30 20 30 20 30 20 30
```

```
                FUZZ FACTOR = 4
-----------------------------------------------

29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21
```

EXAMPLE 6   BLURRED IMAGES OF "STRIPED"   (continued)

```
                FUZZ FACTOR = 5                                        FUZZ FACTOR = 7
-----------------------------------------------     -----------------------------------------------

29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
29 21 29 21 29 21 29 21 29 21 29 21 29 21 29 21     25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25

                FUZZ FACTOR = 6                                        FUZZ FACTOR = 8
-----------------------------------------------     -----------------------------------------------

27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
27 23 27 23 27 23 27 23 27 23 27 23 27 23 27 23     23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27
```

EXAMPLE 6   BLURRED IMAGES OF "STRIPED"   (continued)

```
              FUZZ FACTOR = 9                                   FUZZ FACTOR = 11
------------------------------------------------  ------------------------------------------------

23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
23 27 23 27 23 27 23 27 23 27 23 27 23 27 23 27   25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25

              FUZZ FACTOR = 10                                  FUZZ FACTOR = 12
------------------------------------------------  ------------------------------------------------

24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
24 26 24 26 24 26 24 26 24 26 24 26 24 26 24 26   26 24 26 24 26 24 26 24 26 24 26 24 26 24 26 24
```

INTENSITY HISTOGRAMS FOR EXAMPLE 6

	FUZZ FACTOR												
I /	0	1	2	3	4	5	6	7	8	9	10	11	12
0 /	128	0	0	0	0	0	0	0	0	0	0	0	0
1 /	0	0	0	0	0	0	0	0	0	0	0	0	0
2 /	0	0	0	0	0	0	0	0	0	0	0	0	0
3 /	0	0	0	0	0	0	0	0	0	0	0	0	0
4 /	0	0	0	0	0	0	0	0	0	0	0	0	0
5 /	0	0	0	0	0	0	0	0	0	0	0	0	0
6 /	0	0	0	0	0	0	0	0	0	0	0	0	0
7 /	0	0	0	0	0	0	0	0	0	0	0	0	0
8 /	0	0	0	0	0	0	0	0	0	0	0	0	0
9 /	0	0	0	0	0	0	0	0	0	0	0	0	0
10 /	0	0	0	0	0	0	0	0	0	0	0	0	0
11 /	0	128	0	0	0	0	0	0	0	0	0	0	0
12 /	0	0	0	0	0	0	0	0	0	0	0	0	0
13 /	0	0	0	0	0	0	0	0	0	0	0	0	0
14 /	0	0	0	0	0	0	0	0	0	0	0	0	0
15 /	0	0	0	0	0	0	0	0	0	0	0	0	0
16 /	0	0	0	0	0	0	0	0	0	0	0	0	0
17 /	0	0	0	0	0	0	0	0	0	0	0	0	0
18 /	0	0	0	0	0	0	0	0	0	0	0	0	0
19 /	0	0	0	0	0	0	0	0	0	0	0	0	0
20 /	0	0	128	0	0	0	0	0	0	0	0	0	0
21 /	0	0	0	0	128	128	0	0	0	0	0	0	0
22 /	0	0	0	0	0	0	0	0	0	0	0	0	0
23 /	0	0	0	0	0	0	128	0	128	128	0	0	0
24 /	0	0	0	0	0	0	0	0	0	0	128	0	128
25 /	0	0	0	256	0	0	0	256	0	0	0	256	0
26 /	0	0	0	0	0	0	0	0	0	0	128	0	128
27 /	0	0	0	0	0	0	128	0	128	128	0	0	0
28 /	0	0	0	0	0	0	0	0	0	0	0	0	0
29 /	0	0	0	0	128	128	0	0	0	0	0	0	0
30 /	0	0	128	0	0	0	0	0	0	0	0	0	0
31 /	0	0	0	0	0	0	0	0	0	0	0	0	0
32 /	0	0	0	0	0	0	0	0	0	0	0	0	0
33 /	0	0	0	0	0	0	0	0	0	0	0	0	0
34 /	0	0	0	0	0	0	0	0	0	0	0	0	0
35 /	0	0	0	0	0	0	0	0	0	0	0	0	0
36 /	0	0	0	0	0	0	0	0	0	0	0	0	0
37 /	0	0	0	0	0	0	0	0	0	0	0	0	0
38 /	0	0	0	0	0	0	0	0	0	0	0	0	0
39 /	0	128	0	0	0	0	0	0	0	0	0	0	0
40 /	0	0	0	0	0	0	0	0	0	0	0	0	0
41 /	0	0	0	0	0	0	0	0	0	0	0	0	0
42 /	0	0	0	0	0	0	0	0	0	0	0	0	0
43 /	0	0	0	0	0	0	0	0	0	0	0	0	0
44 /	0	0	0	0	0	0	0	0	0	0	0	0	0

```
          INTENSITY HISTOGRAMS (CONTINUED)
----------------------------------------------------------

 45 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 46 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 47 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 48 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 49 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 50 /128    0    0    0    0    0    0    0    0    0    0    0    0
 51 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 52 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 53 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 54 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 55 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 56 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 57 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 58 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 59 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 60 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 61 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 62 /  0    0    0    0    0    0    0    0    0    0    0    0    0
 63 /  0    0    0    0    0    0    0    0    0    0    0    0    0

          HISTOGRAM ANALYSIS
----------------------------------------------------------

       FUZZ FACTOR      MEAN     STANDARD DEVIATION
----------------------------------------------------------
             0          25.00          25.00
             1          25.00          14.00
             2          25.00           5.00
             3          25.00           0.00
             4          25.00           4.00
             5          25.00           4.00
             6          25.00           2.00
             7          25.00           0.00
             8          25.00           2.00
             9          25.00           2.00
            10          25.00           1.00
            11          25.00           0.00
            12          25.00           1.00
```

EXAMPLE 7   "RANDOM"

```
                  THE FOCUSED PICTURE
-------------------------------------------------

 26 13 10 59 49  6  5 49 19 14  0 45 22 12 20 47
 48 37 11  7 35  6 19 39 34 43 37 34  1 28 51  8
 37 38 58 25 13 17 58 57 14 63 46 59 40 39  0 12
 40 19 46 36 63 35  0 22 17 43 57 11 36 43 12 13
 47 14 33  1 60 59 50  9 10 47 54  6 14 63  8 54
 24 42 25 17 62 56 61 23  2 31 19 52 55  7 15 17
  2 17  8 21  5 36 39 28 58 27 37  3  4 51 13  4
 59  0 33 57 42 44 37 55 35 21  1 49 10 44 12 24
 48 18 45 10 35  4 40 22 32 61 26 42 51  4 12  4
  0 38 50 63 42 20 11 33 46 28 54 27 46 29 20 62
 51 12 30 48 20 13 39 29 34  7 63 62 13 62  0 45
 28 57  9  4 21 24 12  1 16 51 60 13  9 50 27 40
 15 23  0 58 16 49 52  5 18 26 30 54 53 31  3 23
 32  5 28 19 45 38 32 40 45 58 53 44 41 56 39 45
 51 41 29 25 33 58  4 19 37  7 20 60 10 25 42 41
 59  9  1  3 55 33 25 10 51 35 62 31 32 23 48 22
```

EXAMPLE 7   BLURRED IMAGES OF "RANDOM"

```
                 FUZZ FACTOR = 1
------------------------------------------------

29 21 17 43 39 16 11 39 26 21 15 35 25 16 27 40
40 33 18 18 28 12 23 39 32 40 34 35 14 27 35 17
38 38 46 27 23 22 43 47 27 51 50 47 37 34 12 16
38 26 42 35 51 35 18 21 19 44 49 23 32 41 14 22
38 23 28 17 54 55 42 17 14 42 44 18 25 43 20 40
28 32 25 21 50 55 53 24 14 28 30 38 39 24 13 22
19 15 15 21 20 36 40 35 43 31 26 17 15 36 16 11
42 14 31 44 39 38 40 44 38 25 15 34 22 33 17 19
41 24 37 28 30 17 31 30 36 46 32 40 40 17 11 17
16 30 47 52 39 19 21 30 40 34 46 38 39 31 23 45
36 26 30 40 25 19 29 28 30 23 55 48 27 45 17 37
33 40 15 17 19 24 19  7 20 41 51 28 19 42 25 36
20 21 13 37 28 42 40 15 20 32 38 46 45 35 15 26
28 16 22 28 37 42 33 34 42 47 46 48 41 46 38 42
47 33 27 23 39 44 17 21 34 20 31 45 23 28 40 40
52 20 11 16 39 36 26 21 39 34 47 41 31 31 39 29
```

```
                 FUZZ FACTOR = 3
------------------------------------------------

33 27 23 29 30 21 20 31 35 29 29 29 26 21 29 34
34 30 26 27 24 19 26 36 33 35 35 33 26 25 24 26
37 37 35 30 28 26 31 35 36 42 48 38 34 29 22 21
35 33 35 36 39 37 32 23 26 42 43 34 32 34 20 28
32 31 25 32 47 51 36 21 19 35 37 29 32 29 27 30
31 25 23 25 39 52 44 27 24 28 34 28 29 31 18 25
31 18 22 25 34 39 41 39 32 30 23 26 26 25 19 18
32 24 26 32 33 33 39 38 39 30 27 24 29 24 19 17
33 30 34 40 31 28 27 35 38 35 35 37 33 25 16 25
27 28 41 42 33 21 25 30 34 38 41 44 36 29 25 29
27 33 32 33 28 22 22 26 27 36 46 38 35 31 32 30
33 27 22 24 22 25 25 17 22 34 43 38 32 33 27 29
24 21 22 23 33 35 30 23 25 37 43 40 39 36 28 30
26 24 21 31 35 40 33 29 35 37 41 46 41 37 36 38
40 28 22 24 39 36 27 25 33 34 40 37 33 33 39 39
46 32 22 26 29 35 28 28 28 32 37 44 35 36 35 35
```

```
                 FUZZ FACTOR = 2
------------------------------------------------

32 25 21 34 33 20 17 34 32 26 25 31 26 19 29 36
36 31 23 24 25 16 25 38 32 37 34 34 22 26 27 23
38 38 38 28 27 25 35 39 33 45 49 41 35 31 19 19
36 31 38 35 44 36 28 22 23 44 45 30 31 37 17 26
34 28 26 26 50 53 38 20 17 37 39 25 31 34 25 33
30 27 24 24 43 53 47 26 21 28 33 31 32 30 16 24
27 17 19 23 29 37 40 38 35 31 23 24 22 28 18 16
35 21 28 36 36 34 40 40 39 29 23 27 27 27 19 17
36 28 34 36 30 25 27 33 37 38 34 38 35 23 14 23
24 28 44 45 35 20 24 30 36 37 42 42 36 30 24 35
29 32 31 35 27 21 24 27 28 32 50 41 33 36 27 32
34 31 20 23 20 25 23 13 22 36 46 35 27 36 26 32
23 21 19 27 32 37 33 20 23 35 42 42 41 36 23 29
26 22 20 31 35 41 33 30 38 40 43 47 41 40 37 40
43 30 24 23 39 38 24 23 33 29 37 39 30 31 39 39
48 27 18 23 32 36 27 26 32 33 40 44 33 35 36 33
```

```
                 FUZZ FACTOR = 4
------------------------------------------------

35 28 25 26 28 21 23 30 37 31 32 29 27 23 29 33
32 30 27 29 24 21 28 35 35 32 36 33 29 24 23 27
36 37 32 31 28 27 29 32 37 40 46 36 33 28 23 22
34 35 33 36 37 38 34 25 28 41 43 35 33 31 24 29
32 32 25 35 45 50 35 22 21 33 36 32 33 28 27 29
31 24 21 26 37 50 42 29 26 30 33 27 28 29 22 26
32 20 23 27 36 41 42 38 31 28 24 26 29 24 20 19
31 25 25 30 31 32 37 38 38 32 29 24 29 23 19 17
31 32 34 42 33 30 28 35 38 34 35 36 33 26 20 26
28 30 38 40 31 23 25 31 33 39 41 44 36 28 26 26
27 33 33 32 28 23 20 25 27 39 43 38 36 29 35 29
32 25 25 24 25 26 25 20 22 33 42 40 35 30 29 26
25 21 23 22 33 33 29 24 27 39 44 39 39 36 32 32
26 25 23 30 36 39 33 28 32 34 40 43 41 35 35 37
37 28 20 25 37 35 28 27 33 38 41 37 35 34 38 39
45 35 25 27 29 34 29 29 26 31 35 43 37 37 35 37
```

EXAMPLE 7   BLURRED IMAGES OF "RANDOM"   (continued)

```
                 FUZZ FACTOR = 5
-----------------------------------------------

36 30 27 25 25 23 27 30 35 33 33 29 26 26 27 32
32 30 29 29 24 24 27 33 36 32 37 32 32 24 23 29
35 35 30 32 29 28 28 30 36 40 42 36 31 28 24 26
33 36 31 35 35 38 34 27 30 38 41 37 34 28 27 28
32 31 28 36 42 46 36 25 25 31 35 34 32 29 26 28
31 24 22 28 37 46 39 32 27 31 31 27 27 28 25 25
31 24 24 28 37 41 41 36 31 28 27 27 30 23 20 22
31 27 26 29 30 33 35 37 36 33 30 26 29 23 19 20
29 31 34 40 34 30 30 34 37 33 35 34 31 27 23 25
30 32 34 37 29 26 25 31 32 38 41 41 36 29 27 25
28 31 33 31 28 24 22 24 28 40 40 39 36 29 34 28
29 25 27 25 27 26 25 24 23 33 40 40 38 29 30 25
27 23 23 24 31 32 27 25 29 37 43 40 37 35 34 32
27 25 24 27 36 36 33 28 30 34 40 41 40 34 35 37
35 28 21 28 33 35 31 29 31 39 40 39 37 36 36 39
42 35 28 28 30 32 31 30 27 30 35 40 39 36 36 38
```

```
                 FUZZ FACTOR = 7
-----------------------------------------------

36 33 31 26 23 27 32 32 33 34 32 29 28 28 26 30
34 31 31 28 27 27 27 29 35 37 35 33 31 27 25 31
34 32 30 32 32 31 28 29 34 36 36 35 31 27 27 30
34 34 32 33 35 36 34 31 33 34 37 38 32 27 29 28
31 30 31 34 36 38 36 31 31 31 33 35 31 28 27 27
29 26 26 33 38 39 36 33 31 29 29 29 28 26 27 26
30 29 25 30 35 39 38 35 31 30 30 28 28 24 22 25
29 30 29 30 33 34 34 35 33 33 31 30 28 24 22 24
29 30 33 33 33 31 32 34 34 35 35 33 30 28 26 24
32 32 32 32 30 29 27 29 33 36 38 36 33 30 28 26
29 29 31 30 28 26 25 24 30 36 37 39 36 31 28 27
26 26 28 29 29 27 25 27 30 35 38 40 38 32 31 28
28 26 25 27 29 29 27 28 30 34 39 41 37 34 35 33
31 26 24 26 32 33 30 28 29 36 39 38 36 35 35 36
33 29 27 29 30 32 34 31 30 35 38 40 41 38 35 39
37 34 30 29 31 30 31 30 31 33 35 37 39 37 37 40
```

```
                 FUZZ FACTOR = 6
-----------------------------------------------

37 32 29 25 24 25 30 31 34 34 33 29 27 27 26 31
33 31 30 29 26 26 27 31 35 34 36 32 32 26 24 30
34 33 30 32 31 30 28 29 35 38 39 35 31 27 26 28
34 35 31 34 35 37 34 29 32 36 39 38 33 27 29 28
32 30 30 36 39 41 36 28 28 31 34 35 31 28 26 28
30 25 24 31 37 42 37 33 29 30 30 28 28 26 26 26
31 27 25 29 36 40 40 35 31 29 29 27 29 23 21 23
30 29 27 29 32 34 34 36 34 33 30 29 28 23 20 22
29 31 33 36 33 30 31 34 36 34 35 33 31 27 25 24
31 32 33 34 30 28 26 30 33 37 40 39 34 29 27 25
29 30 32 30 28 25 23 24 29 38 38 39 36 30 31 27
27 25 28 27 28 26 25 26 27 34 39 40 38 31 31 27
28 25 24 25 30 31 27 27 30 35 41 40 37 34 35 33
29 26 24 26 35 34 31 28 29 35 40 39 38 35 35 37
34 28 24 29 31 33 33 30 30 37 39 39 39 37 35 39
40 35 29 28 30 31 31 30 29 31 35 38 39 36 37 39
```

```
                 FUZZ FACTOR = 8
-----------------------------------------------

36 34 32 27 23 29 33 33 32 34 32 29 28 28 26 29
34 32 32 28 27 28 27 29 35 38 35 34 31 28 26 31
33 31 31 31 32 31 29 29 33 35 34 35 31 26 29 31
34 33 32 32 35 35 33 32 33 33 35 39 32 27 30 28
31 29 32 33 35 35 36 34 33 31 33 35 31 27 28 27
29 27 28 34 38 37 36 34 32 29 29 30 28 26 27 27
29 30 26 30 35 39 37 34 31 31 30 29 27 24 23 25
29 31 30 30 34 35 34 34 33 33 32 32 28 24 23 25
30 30 32 31 32 31 32 34 34 35 35 32 29 28 27 24
32 33 32 30 31 29 28 28 33 36 37 35 33 30 28 26
29 29 30 29 28 27 25 25 31 35 37 39 36 31 27 26
25 27 28 30 29 27 25 28 32 35 37 41 37 34 32 29
28 27 26 28 29 29 27 29 30 33 38 40 37 33 35 33
31 26 24 26 30 31 29 29 30 37 39 37 35 36 35 36
32 29 30 30 30 31 35 32 29 34 38 40 42 38 35 38
35 34 30 29 31 30 31 31 32 34 36 36 38 37 38 40
```

EXAMPLE 7   BLURRED IMAGES OF "RANDOM"   (continued)

```
                 FUZZ FACTOR = 9
-------------------------------------------------

36 34 32 28 25 29 32 33 33 34 32 29 28 27 27 29
35 32 31 28 28 29 29 29 33 37 34 34 30 29 28 30
32 31 31 31 32 32 30 30 33 34 33 34 31 27 29 32
34 32 32 32 34 34 33 33 34 32 34 37 31 28 29 29
31 30 32 33 34 34 35 35 34 31 32 34 30 27 28 27
29 29 30 34 37 35 36 34 33 29 29 31 28 25 27 27
29 30 28 31 35 37 35 34 32 31 31 30 26 25 25 26
29 31 30 31 34 35 35 33 32 32 33 32 27 25 24 25
30 31 31 29 32 32 32 33 33 36 34 32 30 28 27 25
32 32 31 29 31 30 29 28 33 35 36 34 32 30 28 26
30 29 29 29 28 28 26 27 32 34 37 38 35 32 27 26
26 27 29 29 29 28 26 28 33 35 37 40 37 34 32 30
29 27 27 28 29 28 28 29 31 34 37 39 37 34 34 33
31 27 26 27 28 30 30 30 30 35 38 38 36 36 35 36
32 30 30 29 30 31 33 32 31 33 37 39 40 38 36 38
34 33 31 30 30 30 32 31 32 34 35 35 38 38 38 40
```

```
                 FUZZ FACTOR = 11
-------------------------------------------------

33 34 31 29 29 29 29 31 33 34 32 30 28 27 29 29
35 34 31 30 30 31 31 31 32 34 33 32 30 29 31 29
33 31 31 30 32 32 32 31 32 33 33 33 31 29 29 31
32 30 32 32 32 32 32 35 33 32 32 32 31 29 28 29
32 31 31 33 33 32 34 37 35 32 32 31 30 28 28 28
31 31 33 33 33 34 35 35 33 32 31 31 29 26 26 27
30 30 33 33 35 34 33 33 33 32 32 31 27 26 27 27
30 30 31 32 33 35 34 32 32 32 34 31 28 27 27 26
31 31 29 29 31 33 33 31 32 34 34 32 30 28 27 26
30 30 29 28 30 31 30 30 32 34 34 33 32 30 28 26
30 29 28 29 30 29 28 31 33 33 36 36 35 33 29 27
29 29 29 29 29 28 28 30 33 34 36 37 36 35 31 31
29 28 28 29 28 27 28 30 32 35 36 37 37 36 33 33
30 30 29 29 26 27 30 31 32 33 36 38 38 36 36 35
31 32 30 29 29 30 30 31 33 33 36 38 37 38 38 37
31 31 32 31 30 31 32 32 32 34 34 35 36 37 37 38
```

```
                 FUZZ FACTOR = 10
-------------------------------------------------

34 34 31 28 27 29 30 32 33 34 32 30 28 27 28 29
35 33 31 29 29 30 30 30 33 35 34 32 30 29 29 30
32 31 31 30 32 32 31 30 32 33 33 34 31 28 29 32
33 30 32 32 33 33 33 34 34 32 33 34 31 29 28 29
32 31 32 33 34 33 34 36 35 32 32 33 30 27 28 27
30 30 32 34 35 34 35 34 33 31 30 31 29 26 26 27
29 30 31 32 35 35 34 33 33 32 31 30 26 25 26 26
29 30 31 32 34 35 34 33 32 32 33 31 28 26 26 26
30 31 30 29 31 33 33 32 32 35 34 32 30 28 27 25
31 31 30 29 30 30 30 29 32 34 34 34 32 30 28 26
30 29 29 29 29 28 27 29 32 33 37 37 35 32 28 26
27 28 29 29 29 28 27 29 33 35 37 38 37 35 31 31
29 28 28 29 29 27 28 29 31 35 36 38 37 35 34 33
31 28 28 28 27 28 30 30 31 34 37 38 37 36 35 35
31 31 30 29 30 31 31 32 32 33 36 38 39 38 38 37
32 32 32 30 30 31 32 31 32 34 35 35 37 38 37 39
```

```
                 FUZZ FACTOR = 12
-------------------------------------------------

33 33 31 29 30 29 29 30 33 34 32 30 28 27 29 29
35 34 31 30 31 31 32 32 32 32 33 31 30 29 31 30
33 32 31 31 32 32 32 32 32 32 33 33 31 30 30 31
31 30 31 32 32 31 32 35 33 32 32 30 30 30 28 30
32 31 32 32 33 32 33 37 35 32 31 30 30 29 27 28
31 32 34 33 32 33 34 35 34 32 32 31 29 27 26 26
30 30 34 34 34 33 33 33 33 32 32 31 27 27 28 27
30 30 31 32 33 35 34 32 32 33 33 31 28 27 28 26
31 31 29 29 30 33 32 31 32 33 33 32 30 28 27 26
30 30 28 28 29 31 30 31 31 33 33 33 32 30 28 26
29 29 28 29 30 30 30 32 33 33 36 35 35 33 30 28
30 29 29 29 28 28 29 30 33 34 36 36 36 35 31 31
29 29 29 29 28 27 29 30 33 35 36 36 36 36 33 33
30 31 30 29 26 27 30 31 33 32 35 38 38 37 36 34
31 32 31 29 28 29 29 31 33 33 35 37 36 37 39 36
30 31 32 31 30 32 31 32 32 33 34 35 35 37 37 37
```

INTENSITY HISTOGRAMS FOR EXAMPLE 7

							FUZZ FACTOR							
I	/	0	1	2	3	4	5	6	7	8	9	10	11	12
0	/	7	0	0	0	0	0	0	0	0	0	0	0	0
1	/	5	0	0	0	0	0	0	0	0	0	0	0	0
2	/	2	0	0	0	0	0	0	0	0	0	0	0	0
3	/	3	0	0	0	0	0	0	0	0	0	0	0	0
4	/	7	0	0	0	0	0	0	0	0	0	0	0	0
5	/	4	0	0	0	0	0	0	0	0	0	0	0	0
6	/	3	0	0	0	0	0	0	0	0	0	0	0	0
7	/	4	1	0	0	0	0	0	0	0	0	0	0	0
8	/	3	0	0	0	0	0	0	0	0	0	0	0	0
9	/	4	0	0	0	0	0	0	0	0	0	0	0	0
10	/	6	0	0	0	0	0	0	0	0	0	0	0	0
11	/	3	4	0	0	0	0	0	0	0	0	0	0	0
12	/	7	2	0	0	0	0	0	0	0	0	0	0	0
13	/	7	2	1	0	0	0	0	0	0	0	0	0	0
14	/	4	5	1	0	0	0	0	0	0	0	0	0	0
15	/	2	8	0	0	0	0	0	0	0	0	0	0	0
16	/	2	7	3	1	0	0	0	0	0	0	0	0	0
17	/	5	12	5	2	1	0	0	0	0	0	0	0	0
18	/	2	4	2	3	0	0	0	0	0	0	0	0	0
19	/	6	8	6	4	2	1	0	0	0	0	0	0	0
20	/	5	7	7	2	6	2	1	0	0	0	0	0	0
21	/	3	9	5	7	5	1	1	0	0	0	0	0	0
22	/	4	5	5	10	5	3	1	2	0	0	0	0	0
23	/	4	8	16	6	10	8	4	1	3	0	0	0	0
24	/	3	4	11	10	9	12	8	6	4	1	0	0	0
25	/	5	6	9	15	15	15	11	6	6	7	2	0	0
26	/	3	6	10	13	13	8	14	14	10	8	10	7	6
27	/	3	6	15	12	12	21	17	18	17	16	11	11	10
28	/	5	11	9	12	16	20	18	19	19	20	20	17	16
29	/	3	3	6	15	19	16	23	24	29	29	29	31	27
30	/	2	7	9	11	8	17	24	25	21	26	32	29	37
31	/	3	7	12	9	11	19	25	26	25	24	29	38	36
32	/	4	5	11	13	17	14	9	16	25	32	35	38	42
33	/	5	5	12	18	18	11	14	21	19	19	24	32	36
34	/	3	7	10	11	7	12	19	17	19	28	23	16	12
35	/	5	7	13	19	18	14	17	17	23	15	18	12	13
36	/	3	6	16	10	11	17	10	14	9	8	4	11	11
37	/	6	5	9	10	13	11	10	8	10	10	10	8	7
38	/	3	10	11	7	9	5	7	10	8	8	7	6	2
39	/	5	11	7	10	7	6	13	7	4	2	2	0	1
40	/	5	14	7	5	4	11	7	3	3	3	0	0	0
41	/	3	5	5	5	5	6	2	2	1	0	0	0	0
42	/	5	9	4	3	4	3	1	0	1	0	0	0	0
43	/	3	4	3	3	4	1	0	0	0	0	0	0	0
44	/	3	5	4	3	2	0	0	0	0	0	0	0	0

INTENSITY HISTOGRAMS (CONTINUED)

45 /	6	4	3	0	2	0	0	0	0	0	0	0	0
46 /	4	6	1	3	1	2	0	0	0	0	0	0	0
47 /	3	6	2	1	0	0	0	0	0	0	0	0	0
48 /	4	2	1	1	0	0	0	0	0	0	0	0	0
49 /	4	1	1	0	0	0	0	0	0	0	0	0	0
50 /	3	2	2	0	2	0	0	0	0	0	0	0	0
51 /	7	3	0	1	0	0	0	0	0	0	0	0	0
52 /	2	2	0	1	0	0	0	0	0	0	0	0	0
53 /	2	1	2	0	0	0	0	0	0	0	0	0	0
54 /	4	1	0	0	0	0	0	0	0	0	0	0	0
55 /	3	3	0	0	0	0	0	0	0	0	0	0	0
56 /	2	0	0	0	0	0	0	0	0	0	0	0	0
57 /	4	0	0	0	0	0	0	0	0	0	0	0	0
58 /	6	0	0	0	0	0	0	0	0	0	0	0	0
59 /	5	0	0	0	0	0	0	0	0	0	0	0	0
60 /	3	0	0	0	0	0	0	0	0	0	0	0	0
61 /	2	0	0	0	0	0	0	0	0	0	0	0	0
62 /	5	0	0	0	0	0	0	0	0	0	0	0	0
63 /	5	0	0	0	0	0	0	0	0	0	0	0	0

HISTOGRAM ANALYSIS

FUZZ FACTOR	MEAN	STANDARD DEVIATION
0	30.73	18.77
1	30.88	11.22
2	30.99	7.92
3	31.04	6.82
4	31.17	6.28
5	31.21	5.39
6	31.29	4.60
7	31.36	4.04
8	31.43	3.81
9	31.45	3.39
10	31.44	3.02
11	31.48	2.77
12	31.44	2.61

EXAMPLE 8   "MIXED"

```
                   THE FOCUSED PICTURE
-------------------------------------------------------------

  26  13  10  59  49   6   5  49  19  14  25  25  25  50  50  50
  48  37  11   7  35   6  19  39  34  25  25  25  25  25  50  50
  37  38  58  25  13  17  58  57  25  25  25  25  25  25  25  50
  40  19  46  36  63  35   0  25  25  25  25  25  25  25  25  25
  47  14  33   1  60  59  25  25  25  25  25  25  25  25  25  25
   0  50   0  50   0  50   0   0   0   0   0  10  30  30  30  30
  50   0  50   0  50   0   0   0   0   0   0  10  30  30  30  30
   0  50   0  50   0  50  10  10  10  10  10  10  10  10  10  10
  50   0  50   0  50   0   0   0   0   0   0  10  40  40  40  40
   0  50   0  50   0  50   0   0   0   0   0  10  40  40  40  40
  50   0  50   0  50   0  63  63  63  63   0  50   0  50   0  50
   0  50   0  50   0  50  63  63  63  63   0  50   0  50   0  50
  50   0  50   0  50   0  63  63  63  63   0  50   0  50   0  50
   5   5   5   5   5   5  63  63  63  63   0  50   0  50   0  50
  40  40   5   5   5   5   0   0   0   0   0  50   0  50   0  50
  40  40   5   5   5   5   5   5   5   5   0  50   0  50   0  50
```

EXAMPLE 8   BLURRED IMAGES OF "MIXED"

```
                FUZZ FACTOR = 1
-----------------------------------------------

29 21 17 43 39 16 11 39 26 22 26 25 30 45 50 50
40 33 18 18 28 12 23 39 31 25 25 25 25 30 45 50
38 38 46 27 23 22 43 48 29 25 25 25 25 25 30 45
38 26 42 35 51 35 16 26 25 25 25 25 25 25 25 30
35 24 25 20 47 52 23 22 22 22 22 23 26 26 26 26
22 30 20 28 23 35  8  3  3  3  4 13 27 29 29 29
28 22 28 22 28 16  1  1  1  1  2 11 26 28 28 28
22 28 22 28 22 29 12  8  8  8  8 10 15 15 15 15
28 22 28 22 28 16  1  1  1  1  2 12 33 37 37 37
22 28 22 28 22 28 12  7  7  7  1 17 32 41 36 41
28 22 28 22 28 23 49 56 56 49 12 35 15 38 15 38
22 28 22 28 22 35 62 63 63 56 12 39 11 39 11 39
29 17 29 17 29 18 56 63 63 56 12 39 11 39 11 39
14  8 10  4 10 11 50 56 56 49 12 39 11 39 11 39
36 32  9  5  5  4  8  7  7  7  5 39 11 39 11 39
40 36  9  5  5  5  4  4  4  4  6 39 11 39 11 39
```

```
                FUZZ FACTOR = 2
-----------------------------------------------

32 25 21 34 33 20 17 34 31 27 27 26 33 42 49 50
36 31 23 24 25 16 25 38 30 25 25 25 26 33 42 49
38 38 38 28 27 25 35 41 31 25 25 25 25 26 33 42
36 31 38 35 44 35 25 27 26 25 25 25 25 25 26 33
31 28 23 29 41 45 23 19 20 20 20 22 25 26 26 27
30 23 27 21 32 28 13  5  5  5  7 15 26 29 29 29
21 29 21 29 21 22  4  2  2  2  4 12 23 26 26 26
29 21 29 21 29 20 12  6  6  6  7 11 19 20 20 20
21 29 21 29 21 22  4  2  2  2  4 13 29 34 34 34
29 21 29 21 29 20 18 12 12 11  5 19 30 39 35 39
21 29 21 29 21 32 42 51 51 40 20 26 24 31 25 31
29 21 29 21 29 31 58 63 63 51 22 30 20 30 20 30
19 23 19 23 19 28 52 63 63 51 22 30 20 30 20 30
18 13 12  7 11 15 41 51 51 40 21 30 20 30 20 30
33 27 11  5  5  6 13 13 13 12 12 30 20 30 20 30
40 33 12  5  5  5  4  4  4  3 11 30 20 30 20 30
```

```
                FUZZ FACTOR = 3
-----------------------------------------------

33 27 23 29 30 21 20 31 34 29 28 27 34 41 49 50
34 30 26 27 24 19 26 37 30 24 24 25 26 34 41 49
37 37 35 30 28 26 31 37 32 25 25 25 25 26 34 41
35 33 35 36 39 36 29 28 27 25 25 25 25 25 26 34
30 29 24 32 39 40 23 17 19 19 19 22 25 26 26 28
31 21 28 20 34 26 15  6  6  6  8 16 25 29 29 29
20 30 20 30 20 23  7  3  3  3  4 12 21 25 25 25
30 20 30 20 30 17 11  5  5  5  6 12 20 23 23 23
20 30 20 30 20 23  7  3  3  3  4 14 27 33 33 33
30 20 30 20 30 18 20 16 16 13  8 19 29 37 35 37
20 30 20 30 20 35 40 47 47 35 23 21 28 28 30 28
30 20 30 20 30 31 55 63 63 47 28 25 25 25 25 25
16 24 16 24 16 32 50 63 63 47 28 25 25 25 25 25
20 16 13  9 11 18 36 47 47 35 25 25 25 25 25 25
31 24 12  5  5  7 15 17 17 14 16 25 25 25 25 25
40 31 14  5  5  5  4  4  4  3 13 25 25 25 25 25
```

```
                FUZZ FACTOR = 4
-----------------------------------------------

35 28 25 26 28 21 23 30 35 31 29 28 34 41 48 50
32 30 27 29 24 21 28 36 31 24 24 25 27 34 41 48
36 37 32 31 28 27 29 34 32 26 25 25 25 27 34 41
34 35 33 36 37 36 32 29 27 25 25 25 25 25 27 34
30 29 25 33 39 37 24 16 18 18 18 21 25 26 26 28
30 23 26 22 33 26 17  7  7  7  9 17 24 29 29 29
23 27 23 27 23 21 10  3  3  3  5 12 20 24 24 24
27 23 27 23 27 16 10  4  4  4  6 12 21 25 25 25
23 27 23 27 23 21 10  3  3  3  5 14 25 31 31 31
27 23 27 23 27 20 20 18 18 14 11 18 30 35 36 35
23 27 23 27 23 34 39 45 45 31 25 18 31 26 33 26
27 23 27 23 27 33 51 63 63 45 33 21 29 21 29 21
16 22 16 22 16 33 49 63 63 45 33 21 29 21 29 21
21 19 13 11 10 20 32 45 45 31 28 21 29 21 29 21
30 22 13  5  5  9 16 20 20 15 20 21 29 21 29 21
40 30 15  5  5  5  4  4  4  2 16 21 29 21 29 21
```

EXAMPLE 8   BLURRED IMAGES OF "MIXED"   (continued)

```
                  FUZZ FACTOR = 5
-------------------------------------------------

 36 30 27 25 25 23 27 30 34 31 29 29 34 41 46 50
 32 30 29 29 24 24 27 33 31 26 25 25 29 34 41 46
 35 35 30 32 29 28 28 32 31 26 25 25 25 29 34 41
 33 36 31 36 34 35 32 29 27 24 24 25 25 25 29 34
 31 27 29 32 38 31 25 17 17 17 18 21 25 27 27 30
 26 27 23 28 30 27 16 10  8  9 11 17 23 27 28 28
 28 22 28 22 27 19 13  4  4  4  6 13 20 24 24 24
 22 28 22 28 22 19  8  5  4  4  6 13 21 26 26 26
 28 22 28 22 27 17 14  5  5  5  6 15 23 30 30 31
 22 28 22 28 22 24 20 21 20 15 14 18 29 32 36 33
 28 22 28 22 28 30 39 41 41 30 25 20 30 27 32 27
 22 27 22 27 24 37 47 60 59 43 34 21 30 21 30 21
 20 19 19 18 20 32 47 59 59 43 34 21 29 21 29 21
 20 21 13 14 11 22 30 41 41 30 29 21 29 21 29 21
 29 21 14  6  6 10 17 22 22 17 21 20 29 21 29 21
 38 28 16  6  5  5  6  5  5  5 16 20 29 21 29 21
```

```
                  FUZZ FACTOR = 7
-------------------------------------------------

 36 33 31 26 23 27 32 32 32 30 27 29 34 40 45 48
 34 31 31 28 27 27 28 28 32 30 27 27 30 35 40 45
 34 32 30 32 31 30 28 28 29 27 25 25 27 30 35 40
 33 33 31 32 32 31 31 27 25 24 23 24 25 27 31 36
 29 30 31 33 31 28 25 20 16 16 18 21 24 26 28 32
 26 27 26 30 28 25 18 13 10 11 14 17 21 24 26 28
 26 25 25 25 25 20 14  8  5  6  9 14 20 24 26 26
 25 25 25 25 22 18 11  5  3  3  7 14 21 26 28 28
 25 25 25 25 23 18 15 12  9  8 11 16 23 27 30 29
 25 25 25 25 24 25 25 24 22 18 18 20 25 29 32 31
 25 25 25 25 28 30 34 38 35 30 25 24 26 30 30 31
 22 23 22 23 26 35 43 52 49 41 32 26 25 27 26 27
 22 20 19 17 23 31 42 51 49 41 33 26 25 25 25 25
 24 20 17 13 15 22 30 36 36 31 27 24 25 25 25 25
 27 22 15 10  8 13 18 23 22 21 21 22 25 25 25 25
 32 25 16  9  5  6  9 10 10 12 16 20 25 25 25 25
```

```
                  FUZZ FACTOR = 6
-------------------------------------------------

 37 32 29 25 24 25 30 31 33 31 28 29 34 40 45 49
 33 31 30 29 26 26 28 30 32 28 26 26 30 35 40 45
 34 33 30 32 31 29 28 30 30 27 25 25 26 30 35 40
 33 35 31 34 32 33 32 28 26 24 23 24 25 26 30 35
 30 29 31 32 34 29 25 19 17 17 18 21 24 26 28 31
 26 27 24 29 29 26 17 11  9 10 13 17 22 26 27 28
 28 23 27 23 27 19 14  6  5  5  8 14 20 24 25 25
 23 27 23 27 22 19  9  5  3  3  6 13 21 26 27 27
 27 23 27 23 25 18 15  9  7  7  9 16 23 28 30 30
 23 27 23 27 22 25 23 23 21 17 16 19 27 30 34 32
 27 23 27 23 29 30 37 40 38 30 25 22 28 29 31 29
 22 26 22 26 24 36 45 56 54 42 33 23 28 24 28 24
 22 19 20 17 22 31 44 55 54 42 34 24 27 23 27 23
 22 21 15 14 13 22 30 39 38 31 28 23 27 23 27 23
 28 21 15  8  7 12 17 23 22 19 21 20 27 23 27 23
 35 26 16  8  5  6  7  8  8  9 16 20 27 23 27 23
```

```
                  FUZZ FACTOR = 8
-------------------------------------------------

 36 34 32 27 23 29 33 33 31 30 27 30 34 39 44 48
 34 32 32 28 27 28 28 27 31 31 27 28 31 35 40 44
 33 31 31 31 32 30 29 27 28 27 25 25 27 31 35 40
 33 32 32 30 31 30 30 26 24 23 22 23 25 28 32 36
 29 31 31 33 29 27 25 21 16 16 18 21 24 26 29 33
 27 27 27 30 28 24 19 14 11 12 14 18 21 24 25 27
 25 26 24 26 24 21 14 10  6  7 10 15 20 24 26 26
 26 24 26 24 22 17 12  5  3  3  8 14 21 26 29 29
 24 26 24 26 22 19 16 14 11 10 12 16 22 26 30 29
 26 24 26 24 25 25 26 25 22 20 19 20 23 28 30 30
 24 26 24 26 27 30 33 37 33 30 25 25 25 31 29 32
 23 22 23 22 27 34 42 48 46 40 31 28 23 29 25 29
 22 21 19 18 23 31 40 48 46 41 32 28 23 27 23 27
 25 20 18 13 17 23 30 35 34 32 27 26 23 27 23 27
 26 22 16 12 10 14 18 23 22 23 21 23 23 27 23 27
 30 24 16 10  5  7 10 12 11 15 16 21 23 27 23 27
```

EXAMPLE 8 BLURRED IMAGES OF "MIXED" (continued)

```
                 FUZZ FACTOR = 9
------------------------------------------------

36 34 32 28 25 29 32 32 31 30 28 30 34 39 44 47
35 32 31 28 28 29 29 28 30 31 28 28 31 36 39 44
32 31 31 31 31 31 29 27 27 27 25 25 28 31 36 40
33 31 32 30 31 28 28 26 23 22 22 23 25 29 32 36
29 31 31 32 28 26 24 22 17 17 18 21 23 26 30 33
28 26 29 29 28 23 20 15 12 12 15 18 21 23 25 28
24 27 24 27 23 21 15 11  7  8 11 16 20 24 26 27
27 23 27 23 23 17 13  7  4  5  9 15 21 26 29 29
23 27 23 26 21 21 17 16 12 11 14 17 22 26 29 29
27 23 27 23 26 25 27 26 23 21 20 21 22 27 28 30
23 26 23 27 26 31 32 36 32 29 26 25 24 30 29 31
24 21 23 21 28 33 40 44 42 38 31 28 24 29 26 30
22 22 19 19 23 31 37 43 42 39 32 29 24 27 23 27
25 20 19 14 18 23 30 33 33 32 27 26 23 27 23 27
26 22 16 13 11 15 19 23 23 24 21 23 23 27 23 27
29 23 17 11  7  8 11 13 13 17 16 21 22 27 23 27
```

```
                 FUZZ FACTOR = 10
------------------------------------------------

34 34 31 28 27 29 30 31 31 30 29 30 33 38 43 46
35 33 31 29 29 30 30 29 29 30 28 29 32 36 39 43
32 31 31 30 30 30 29 26 27 27 25 26 29 32 36 40
31 30 31 31 29 28 27 25 23 22 22 23 25 29 33 37
31 30 31 30 27 26 24 21 18 17 18 20 23 26 30 33
28 28 29 29 26 23 20 17 13 13 15 18 21 23 26 29
26 26 26 27 24 20 16 11  8  9 12 16 20 24 26 28
25 25 25 24 22 18 14 10  7  7 11 15 21 25 28 29
25 25 25 24 22 20 19 17 14 13 15 18 21 26 28 29
25 25 25 25 25 25 26 26 23 22 21 21 22 26 28 30
24 24 24 25 27 30 33 33 32 29 26 25 25 28 29 30
23 22 22 22 27 32 37 39 38 35 31 28 25 28 27 29
23 21 20 20 24 30 35 38 38 36 31 29 25 27 25 27
24 21 18 17 19 23 28 32 32 31 28 26 24 26 24 26
26 21 18 13 13 16 20 22 23 24 22 23 23 26 24 26
28 23 17 12  8  9 12 14 15 17 18 20 22 26 24 26
```

```
                 FUZZ FACTOR = 11
------------------------------------------------

33 34 31 29 29 29 29 30 31 30 30 30 33 37 42 46
35 34 31 30 30 31 31 30 28 29 29 29 32 36 39 42
32 31 31 30 30 30 29 26 26 27 26 26 29 33 36 40
31 30 31 31 29 28 25 25 22 21 22 23 26 30 34 37
31 30 31 28 27 25 24 21 19 18 19 20 22 26 30 33
28 29 30 29 25 23 20 18 14 13 15 18 21 23 27 29
27 25 27 26 24 20 17 12  9 10 13 16 20 24 27 29
25 26 24 25 21 19 15 13  9  9 12 16 20 24 28 29
26 24 26 23 23 21 20 17 15 14 16 18 21 25 27 29
24 26 24 26 25 26 26 26 23 22 21 22 23 25 27 29
25 23 25 24 27 29 33 32 31 29 27 25 25 26 29 29
22 23 21 23 27 31 34 35 35 33 31 28 27 27 28 28
24 21 20 20 25 29 34 34 35 34 31 28 26 26 26 26
24 22 18 18 19 24 28 30 31 31 28 26 25 25 25 25
26 21 18 14 15 16 20 22 24 24 23 23 23 25 25 25
27 23 17 13 10 10 12 14 17 18 19 20 22 25 25 25
```

```
                 FUZZ FACTOR = 12
------------------------------------------------

33 33 31 29 30 29 29 29 30 30 31 31 33 37 41 45
35 34 31 30 30 31 31 30 28 28 29 29 32 36 39 42
32 32 31 29 29 29 28 27 25 26 26 27 30 33 36 40
30 30 31 30 28 27 24 24 22 21 22 23 26 30 34 37
31 29 31 28 27 25 23 22 20 18 19 20 22 26 29 33
28 29 30 28 24 23 20 18 15 14 16 18 21 24 27 30
27 26 28 26 24 20 17 13 10 10 13 17 20 24 27 29
25 26 24 24 21 20 17 15 11 11 14 17 20 24 27 29
26 24 26 23 23 21 21 18 16 16 17 19 21 25 27 29
24 26 24 26 25 26 26 25 24 22 22 22 23 25 27 28
24 23 24 24 27 29 32 31 31 29 28 25 25 25 28 28
23 23 21 24 26 30 32 31 32 31 31 28 28 26 30 27
24 22 20 21 25 28 32 31 32 32 31 28 27 25 27 25
24 22 18 19 20 24 27 29 31 30 29 26 25 24 26 24
26 22 18 16 16 17 21 21 24 24 24 23 24 24 26 24
27 22 18 14 11 11 13 15 18 18 20 20 23 24 26 24
```

INTENSITY HISTOGRAMS FOR EXAMPLE 8

I /	FUZZ FACTOR 0	1	2	3	4	5	6	7	8	9	10	11	12
0 /	67	0	0	0	0	0	0	0	0	0	0	0	0
1 /	1	9	0	0	0	0	0	0	0	0	0	0	0
2 /	0	2	6	0	1	0	0	0	0	0	0	0	0
3 /	0	3	1	7	6	0	2	2	2	0	0	0	0
4 /	0	7	7	5	6	5	0	0	0	1	0	0	0
5 /	19	6	9	8	7	9	4	3	2	1	0	0	0
6 /	2	1	4	4	1	7	3	2	1	0	0	0	0
7 /	1	6	3	3	3	0	4	1	2	3	2	0	0
8 /	0	7	0	2	0	2	5	3	1	2	2	0	0
9 /	0	2	0	1	2	1	5	4	0	1	2	3	0
10 /	15	3	0	0	4	2	1	4	6	0	1	3	2
11 /	1	13	5	2	2	2	1	3	3	6	2	0	4
12 /	0	8	8	3	2	0	1	2	5	3	3	3	0
13 /	2	1	6	3	2	4	3	3	1	4	5	4	3
14 /	2	1	0	3	2	4	3	4	6	2	3	4	3
15 /	0	6	2	2	2	2	3	3	2	5	4	4	3
16 /	0	4	1	8	7	3	4	5	7	4	3	4	5
17 /	1	4	1	4	2	7	7	3	2	7	6	4	6
18 /	0	3	2	2	7	3	2	7	5	3	8	8	9
19 /	3	0	6	5	1	4	6	1	4	4	2	5	3
20 /	0	2	20	20	7	9	4	7	4	4	9	10	11
21 /	0	1	18	4	22	19	6	5	8	12	9	10	10
22 /	0	22	5	1	4	17	11	9	10	9	12	9	11
23 /	0	6	7	8	17	4	25	8	20	29	13	14	11
24 /	0	1	2	7	8	9	10	10	16	9	15	14	30
25 /	35	18	19	41	17	13	11	48	14	8	22	25	14
26 /	1	9	12	9	8	6	15	14	19	16	23	22	20
27 /	0	2	7	5	21	16	24	16	24	23	11	14	16
28 /	0	21	4	10	7	17	14	12	10	17	15	11	16
29 /	0	9	20	8	18	21	11	5	11	17	20	22	19
30 /	8	5	18	21	7	15	16	14	14	9	17	16	16
31 /	0	1	8	6	9	7	10	14	14	18	14	18	18
32 /	0	2	3	3	5	8	7	12	10	12	7	3	9
33 /	1	2	7	5	8	3	6	5	8	5	6	6	5
34 /	1	0	5	7	7	8	6	4	5	2	2	7	2
35 /	2	6	4	7	5	3	5	4	3	1	3	4	1
36 /	1	3	2	3	5	4	1	4	2	5	3	2	2
37 /	2	3	0	6	3	1	2	0	1	1	2	2	2
38 /	1	5	5	0	0	2	2	1	0	1	4	0	0
39 /	1	18	2	2	2	1	1	0	1	3	2	1	1
40 /	13	2	3	3	1	0	4	3	4	2	1	1	1
41 /	0	2	3	3	3	7	0	2	1	0	0	0	1
42 /	0	1	4	0	0	0	2	1	1	2	0	2	1
43 /	0	2	0	0	0	2	0	1	0	1	2	0	0
44 /	0	0	1	0	0	0	1	0	2	3	0	0	0

INTENSITY HISTOGRAMS (CONTINUED)

45 /	0	3	1	0	6	0	3	2	0	0	0	0	1
46 /	1	1	0	0	0	2	0	0	2	0	1	1	0
47 /	1	1	0	6	0	2	0	0	0	1	0	0	0
48 /	1	1	0	0	2	0	0	1	3	0	0	0	0
49 /	2	3	2	2	1	0	1	2	0	0	0	0	0
50 /	48	4	1	2	1	1	0	0	0	0	0	0	0
51 /	0	1	6	0	1	0	0	1	0	0	0	0	0
52 /	0	1	1	0	0	0	0	1	0	0	0	0	0
53 /	0	0	0	0	0	0	0	0	0	0	0	0	0
54 /	0	0	0	0	0	0	2	0	0	0	0	0	0
55 /	0	0	0	1	0	0	1	0	0	0	0	0	0
56 /	0	7	0	0	0	0	1	0	0	0	0	0	0
57 /	1	0	0	0	0	0	0	0	0	0	0	0	0
58 /	2	0	1	0	0	0	0	0	0	0	0	0	0
59 /	2	0	0	0	0	3	0	0	0	0	0	0	0
60 /	1	0	0	0	0	1	0	0	0	0	0	0	0
61 /	0	0	0	0	0	0	0	0	0	0	0	0	0
62 /	0	1	0	0	0	0	0	0	0	0	0	0	0
63 /	17	4	4	4	4	0	0	0	0	0	0	0	0

HISTOGRAM ANALYSIS

FUZZ FACTOR	MEAN	STANDARD DEVIATION
0	25.20	21.96
1	25.09	14.66
2	25.08	12.23
3	25.04	11.44
4	25.01	11.02
5	25.04	10.16
6	25.17	9.23
7	25.17	8.52
8	25.26	8.04
9	25.32	7.44
10	25.29	6.74
11	25.33	6.29
12	25.33	5.84

END

DTIC

7-86