ADAPTIVE LIGHTING

FOR

COMPUTER VISION

ADAPTIVE LIGHTING
FOR
COMPUTER VISION

by

MARIO PEÑA CABRERA, M. Eng. (National University of Mexico)

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Engineering

McMaster University

January 1983

MASTER OF ENGINEERING
(Electrical Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario.

TITLE: Adaptive Lighting for Computer Vision

AUTHOR: Mario Peña Cabrera, M. Eng. (National University of Mexico)

SUPERVISOR: Professor R. Kitai, D.Sc.

NUMBER OF PAGES: x , 174 .

ABSTRACT

A system capable of adjusting a computer vision system to unpredictable ambient lighting has been designed and attached to a silhouette robot vision system.

Its principle of operation is based on the generation and analysis of the distribution of light in one T.V. frame.

Designed to be used in robot vision applications, high speed processing of data is achieved in the system to generate a histogram of grey levels in one frame time.

An addressable RAM technique for this purpose is explained. The system obtains two threshold values from the histogram of grey levels and places them into a threshold logic unit. A silhouette from a grey level picture is obtained as the result of the process.

Adaptability of the system is performed by using different integration times in the read out of the visual transducer.

The implementation of the system is based on a video rate histogram generator, a sensitivity control unit, a DMA circuit, an 86/12A microcomputer and a solid state T.V. camera. A graphics printer is used to print out results and a CRT terminal to communicate with the microcomputer. The custom hardware and software implementations for the system are depicted in detail.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to his supervisor Dr. R. Kitai for the support, guidance and teaching throughout the course of this work.

Also, financial support to the National Research Council of Mexico and McMaster University are greatly appreciated.

A special thanks to David Capson for his collaboration and friendship; important factors for the realization of this work.

Finally the author would like to dedicate this work to his parents Manuel and Conchita.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER I

INTRODUCTION

Industrial processes today are becoming increasingly in need of automation in many of their operations. This requirement has led to new technologies that attempt to enable man-made machines to perceive their environments by sensory means as humans and animals do.

Machines that perceive their environments and perform required tasks have an obvious usefulness for diverse application areas such as: planetary space exploration, automated medical X-ray screening, monitoring of earth resources by remote sensors, military applications and industrial assembly and inspection. They could assist in many tasks that are routine, tedious and even dangerous for humans to perform, but are difficult or impossible to automate without some perceptual capability.

The ability to control an industrial or any other process which requires such a capability, relies heavily on the quality of the sensing devices that input information into the system. Traditional sensors such as photocells, temperature transducers, and pressure transducers have been used satisfactorily . However, there are many applications that would benefit from the use of visual sensing. In this respect, the industrial environment today offers many potential applications for image processing.

The most important applications for visual sensing in this area are:

- Inspection of items involved in manufacturing processes for quality control; the items may be raw materials, partly manufactured or completed manufactured products.

- Measurements of the critical dimensions of manufactured items to ensure that they are within required tolerances for quality control.
- Identification of items involving item type, position and orientation for flow control of manufactured products, or controlling a robotic device in automatic packaging and assembly processes.
- Visual servoing of robotic devices to allow the control of a tool along a desired path to achieve a specified task.

Visual sensing requires the use of a vision system comprised of: a basic sensing device such as a vidicon T.V. camera or an array of photodiodes in one or two dimensions (solid state T.V. camera), and a computer to process the raw data from the camera. The design of the computer program is the most critical part of the construction of a vision system in a particular application, but other factors such as lighting and the optical system are very important. There is no point in producing a highly accurate, sophisticated, intelligent computer system when there are no possible means of obtaining dynamic range out of the image transducer. In this respect, lighting is certainly the major consideration in any industrial automated system that uses visual sensing.

To ensure successful picture processing, the location of the objects to be analyzed must be suitably illuminated; any possible errors which could be caused through accidental formations of shadows, reflections or outside light must be eliminated. This means that the scene illumination must be dominant and homogeneous in order to achieve

an even distribution of light within the picture area. Various methods of illumination can be employed. Among the principal ones are: flash light, direct illumination, infrared illumination and fluorescent radiation.

The lamps and lights required for this purpose have to be selected carefully. However, there is still more that can be done to improve the quality of the image of the objects being analyzed so as to ensure more accurate and more reliable methods of recognizing and measuring objects by way of vision systems. Techniques such as back lighting, reflective lighting, and light patterns of different sorts are schemas that are being used at the present time for this purpose. Another method such as filtering or thresholding can also help to improve visual sensing performance.

Most of the vision systems being used by industry today are for inspection (e.g. Octek's Image Analysis Processor , the ORS Multi-element SCANSYSTEM , System "Q" by ORS , VS-110 Inspection System by Machine Intelligence Corporation). Applications in robotics are generally still in evaluative stages [1-1]. To integrate robots with visual capability into assembly, the problem concerning illumination in the assembly environment, which is not sufficiently stable for the current systems, must be solved.

Due to the impossibility of stabilizing the ambient light of the assembly environment in some way, the vision systems must be adaptable to unpredictable light conditions. Using such a facility, the user can include in the development of the algorithm that is suitable for his particular applications, a first stage for correction of light conditions. One of the obstacles in achieving this task is the large amount of data that must be

manipulated in a very short period of time; thus, such a facility must be implemented in hardware. In many cases the user can include an image enhancement stage to bring out the best possible constrast between the object of interest and the background, or a noise reduction stage which will improve the reliability of the image analysis process.

In some applications an image enhancement stage is all that is required, particulary when the image is to be presented only for human vision inspection, as in the case of radiographic examinations. In other applications the image from the camera is of sufficient quality and contrast for the enhancement to become unnecessary, or because the enhancement technique requires additional processing of the image, there may not be sufficient time for such enhancement to take place. However, enhancement is not a substitution for inadequate lighting conditions,and considerable effort should be made to ensure the best method of lighting the scene to be analyzed in order to present a good high contrast image.

By avoiding uneven lighting conditions on the scene, a substantial improvement in the reliability of a vision system is achieved.

The system to be described (Adaptive Lighting System/ALS), provides a silhouette robot vision machine with light ambient changes adaptability within a certain adaptive range [1-2]. This vision system essentially works with binary images (silhouettes), and its objective is to identify, locate and inspect the silhouettes of 3-dimensional workpieces which are to be handled by an industrial robot. The robot/vision interaction of this system is shown in figure 1.1. In the figure, a solid state T.V. camera is viewing a predefined working area for the robot. The camera takes pictures of the scene being analyzed, and transmits them

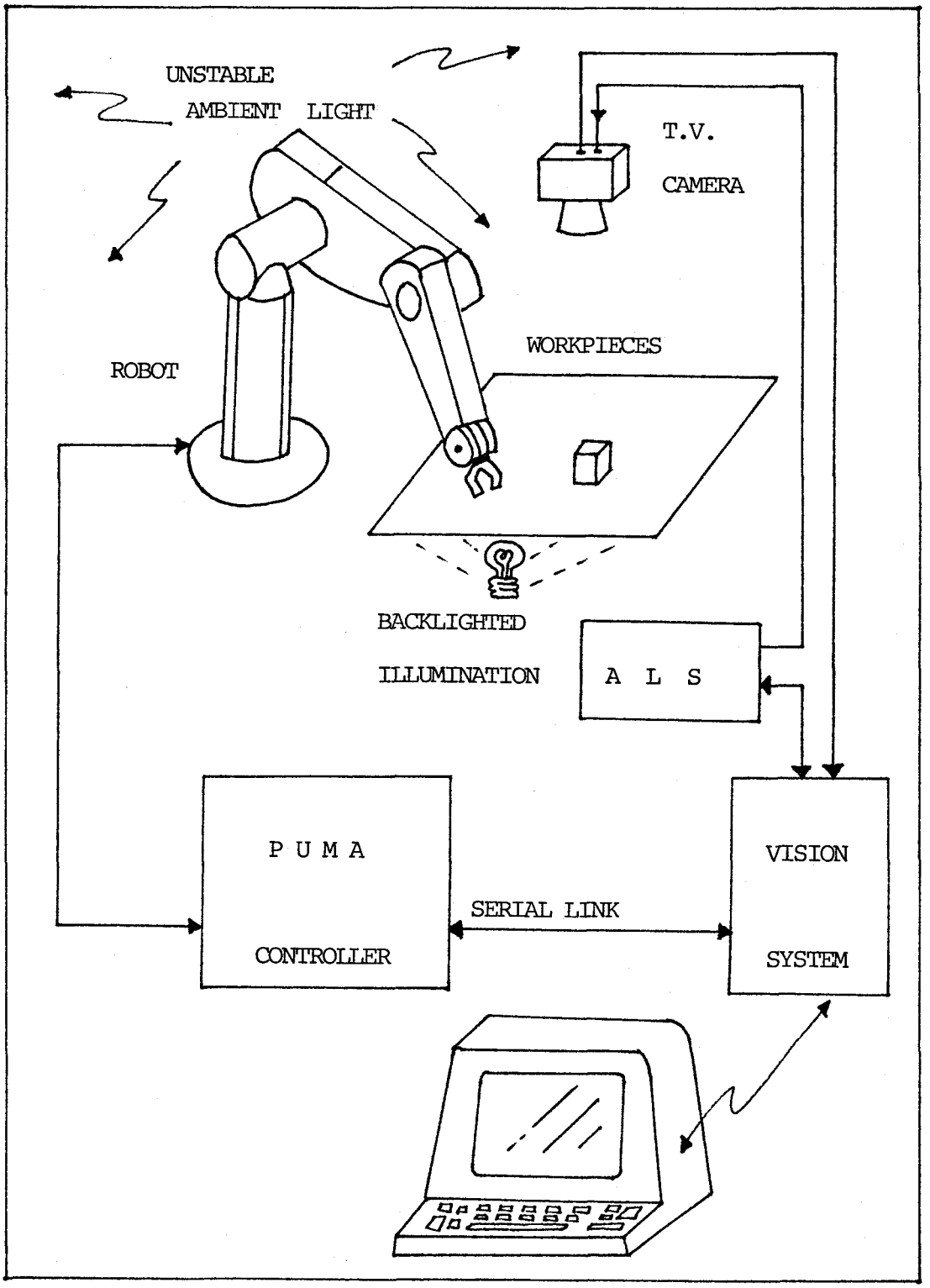


Figure 1.1 Robot/vision interaction.

to the vision system. The vision system processes the acquired image and extracts the information needed to command the robot movements for the acquisition of the workpieces via the robot controller.

The ALS analyzes the distribution of light on the scene each time a picture is taken by the camera, and sets the appropriate sensitivity of the visual transducer as well as the appropriate threshold value. To achieve this task, the ALS is incorporated in the vision system architecture to interact with other modules. In chapter III, this modular interaction is depicted in detail.

In any vision system which processes binary images, the image processing sequence may be divided into four stages: an image enhancement and correction of light conditions stage; an object-background separation stage to produce a binary image; a feature analysis stage on the binary image and a decision stage. All these subprocesses can be carried out with the machine vision involved in this report, and to provide a PUMA 600 robot with vision capability.

The object-background separation stage attempts to reduce the grey level image coming from the T.V. camera into a silhouette, where the pixels in the background are set to white and the pixels in the object are set to black. The most common method for separating the object from the background is the threshold operator. As mentioned before, the Adaptive Lighiting System described here (ALS), provides such an operator and uses the distribution of light of one frame (represented by the histogram of grey levels) for an automatic adjustment of this operator, and for the sensitivity control of the sensor transducer (integration time). The operation is essentially affected in parallel with the image acquisition,

and in the first stage of the image processing sequence of the vision system. By combining this parallel processing with serial processing to interact with other modules in the system, the silhouette is created, and a significant advance over earlier systems is obtained.

In chapter II, digital image and computer vision fundamentals are described as well as lighting and thresholding schemes being currently used in most vision systems. The visual perception process and brightness adaptability in the human eye are explained.

In chapter III, the hardware implementation and operation of the system is explained in detail. In chapter IV, the software architecture of the system is depicted as well as the algorithms used by the system.

In chapter V, measurements on the adaptive range of the system and the T.V. camera are shown. Discussion of the features and speed of the system is presented in chapter VI.

CHAPTER II

COMPUTER VISION AND DIGITAL IMAGE FUNDAMENTALS

2.1 Elements of visual perception

It is important to have a basic understanding of the visual perception process in the human eye in order to have a better understanding of the computer vision concept. Figure 2.1 shows an horizontal cross section of the human eye.

The eye is nearly spherical in form with an average diameter of aproximately 24 mm. [2-1]. It is enclosed by three membranes: the cornea and sclera (outer cover), and the choroid and the retina middle and inner membranes respectivaly. The cornea is a transparent tissue that covers the anterior surface of the eye. The sclera is continous with the cornea; it is an opaque membrane that encloses the remainder of the optical globe. The choroid lies directly below the sclera, this membrane contains a network of blood vessels which serves as the major source of nutrition to the eye. The choroid, which is heavily pigmented, helps to reduce the amount of extraneous light entering the eye and the backscatter within the optical globe. The choroid at its anterior extreme is divided into the ciliary body and the iris diaphragm. The iris diaphragm contracts or expands to control the amount of light that is permitted to enter the eye.

The central opening of the iris (the pupil) is variable in diameter from approximately 2 mm. up to 8 mm. [2-3]. The innermost membrane of the eye is the retina. When the eye is properly focused,

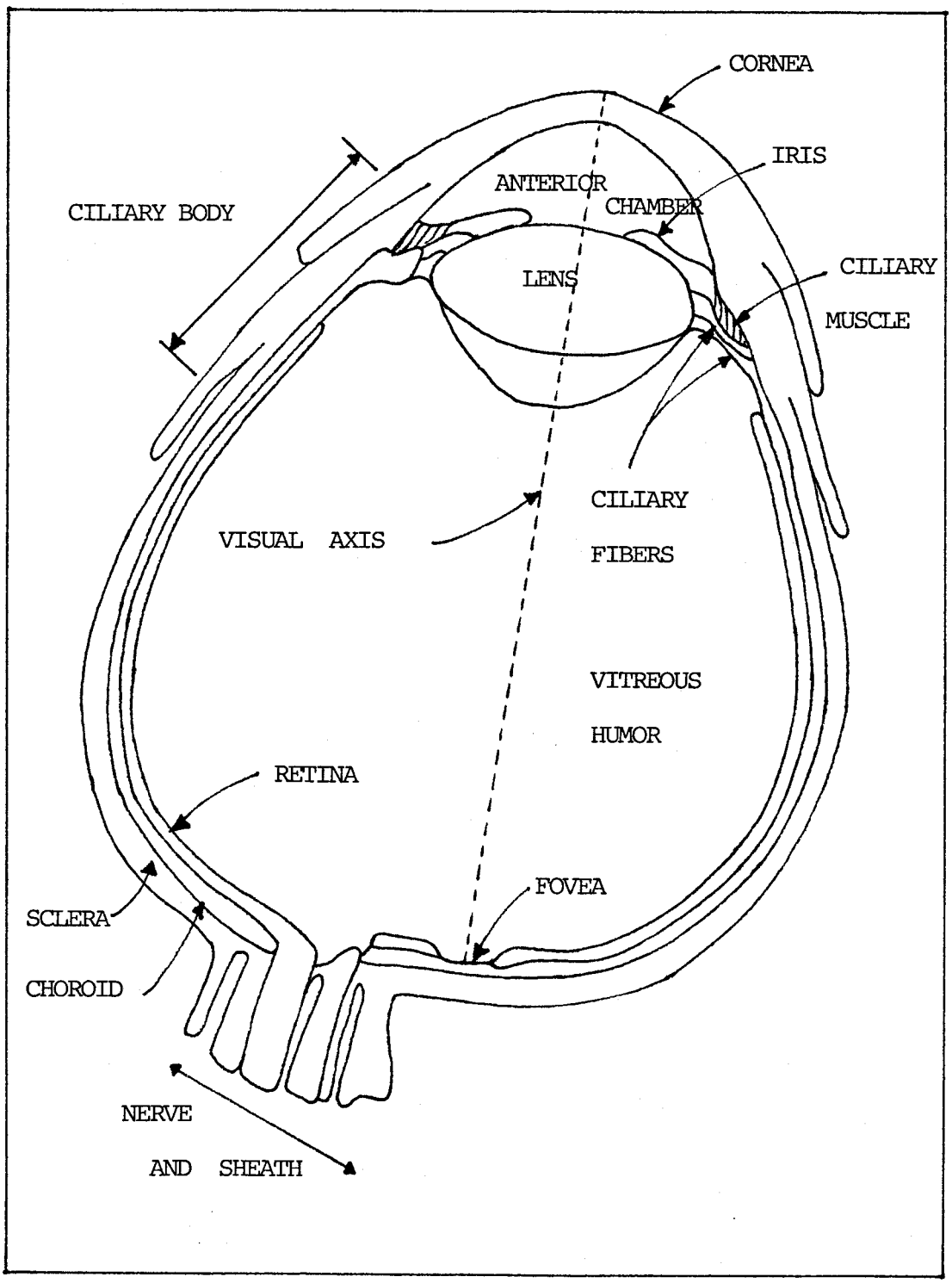


Figure 2.1 Cross section of the human eye.

reproduced from [2-2]

light from an object outside the eye is imaged on the retina. Pattern vision is afforded by the distribution of discrete light receptors over the surface of the retina. There are two classes of receptors: cones and rods . The cones in each eye number between six and seven million [2-4]. They are located in the central portion of the retina, which is called Fovea and they are highly sensitive to colour. Humans can resolve fine details with these cones because each cone is connected to its own nerve end. Muscles controlling the eye rotate the eyeball until the image of an object of interest falls on the Fovea . The number of rods is much larger ,they are in the order of 75 to 150 million [2-4] and distributed over the retinal surface. The larger area of distribution and the fact that several rods are connected to a single nerve end, reduce the amount of detail acquired for this receptor.

Rods are not involved in colour vision and are sensitive to low levels of illumination.

The lens is made up of concentric layers of fibrous cells. It measures on the average of 7-10 mm. in length, 8-10 μ width and 2-5 μ thickness [2-5], and it is suspended by fibers that attach to the ciliary body. It contains sixty to seventy percent water. It absorbs approximately eight percent of the visible light spectrum, with relatively higher absorption at shorter wavelengths. Both infrared and ultraviolet light are absorbed appreciably by proteins within the lens structure and, in excessive amounts can cause damage to the eye.

The shape of the lens is controlled by the tension in the fibers of the ciliary body. To focus on distant objects, the controlling muscles cause the lens to be relatively flattened. Similarly, these muscles allow the lens to become thicker in order to focus on objects near the eye.

The distance between the focal center of the lens and the retina ranges from 17 mm down to 14 mm. [2-6]. Once the retinal image is reflected primarily in the area of the Fovea, perception takes place by the excitation of the light receptors which transform radiant energy into electrical impulses that are ultimately decoded by the brain.

By copying the biological functions of the human eye, the concept of vision can be translated into a physical implementation developed with current known technology to reach the concept of computer vision.

2.2 Brightness adaptation

The ability of the eye to discriminate between different brightness levels is an important consideration when a variable adaptable vision system for industrial control is required, since digital images are displayed as a discrete set of brightness points.

The range of light intensity levels to which the human visual system can adapt is enormous, being in the order of 10^{10} [2-7]. There is evidence [2-8] which indicates that brightness perceived by the human eye is a logarithmic function of the light intensity incident in it (subjective brightness). This characteristic is shown in figure 2.2 , which is a plot of light intensity (mLb.) versus subjective brightness (from the scotopic threshold to the glare limit). The long solid curve represents the range of intensities to which the visual system can adapt. In photopic vision alone (vision performed by the cones), the range is about 10^{6}. The transition from scotopic to photopic vision is gradual. The eye accomplishes this variation by changes in its overall sensitivity, this phenomenon is known as brightness adaptation.

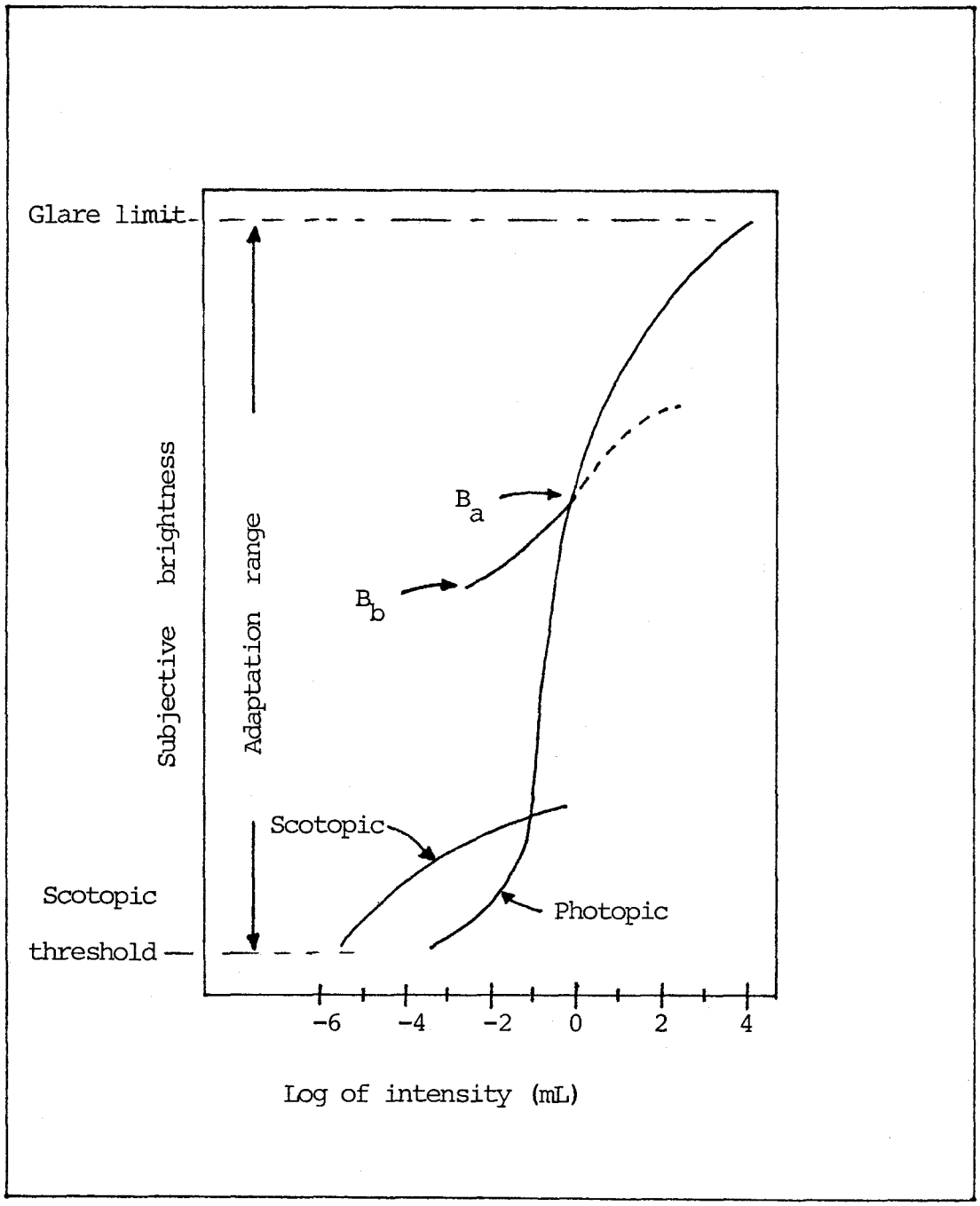


Figure 2.2 Range of subjective brightness sensations showing a particular adaptation level (reproduced from [2-9]).

The total range of intensity levels that the eye can discriminate simultaneously is small compared with the total adaptation range. The current sensitivity level of the visual system is called the brightness-adaptation level, which may correspond in the figure 2.2 to brightness Ba. The intersecting curve represents the range of subjective brightness that the eye can perceive when adapted to this level, which is restricted to a level Bb at and below which all stimuli are perceived as indistinguishable blacks. The upper portion of the curve (dashed portion) is not restricted, but if extended too far loses its meaning because higher intensities would simply raise the adaptation level to a higher value than Ba.

In a complex image, the visual system does not adapt to a single intensity level but to an average level which depends on the properties of the image, and as the eye roams about the scene, the instantaneous adaptation level changes about this average [2-10]. The result is that the eye can only detect in the neighborhood of one or two dozens intensity levels at any one point in a complex image. This does not mean however, that an image needs only be displayed in one or two dozens of intensity levels to achieve satisfactory results.

This narrow discrimination range " tracks " the adaptation level as the latter changes in order to accomodate different intensity levels following eye movements around the scene. This allows a much larger range of intensity levels discrimination so that, to obtain displays that will appear reasonably smooth to the eye and for a large class of image types, a range of over 100 intensity levels is required.

2.3 Computer vision

As mentioned before, the human eye consists of a lens with an iris which takes in the image of the object that it is required to see and by way of receptors in the retina transforms the data received into signals that send the information to the brain.

A typical computer vision system can be represented in such a manner as shown in figure 2.3. The lens and the iris are the optics and diaphragm, and the retina is a photosensor. Braccini and Gambardella [2-11] suggest the implementation of the photosensor device in the same geometrical manner as the retina (which includes the Fovea system) in order to detect three dimensional scenes and obtain a wider field of view.

The information is taken via a multiplexor system as the analogous performance of the neurons, and it is put into a computer which attempts to achieve the task of the brain. In this way, the object is imaged through optics and detected by a sensor. It is then interrogated through the image processing circuitry, to give feedback to an optical control system to view for parts, or even to make the system adaptable to unpredictable circumstances, in an attempt to make computer vision systems work like the human eye.

At each stage of a computer system, some form of decision-making is required, depending on the visual system constraints which determine the level of intelligence required. The constraints listed in figure 2.4 are the requirements for most control systems using visual sensing [2-12]. Some of the vision systems now on the market reach the sophisticated level of intelligence in some of their features. This is the case of SAM (Sensor System for Automation and Measurement) developed by the Fraunhofer Institute for Information and Data Processing which is

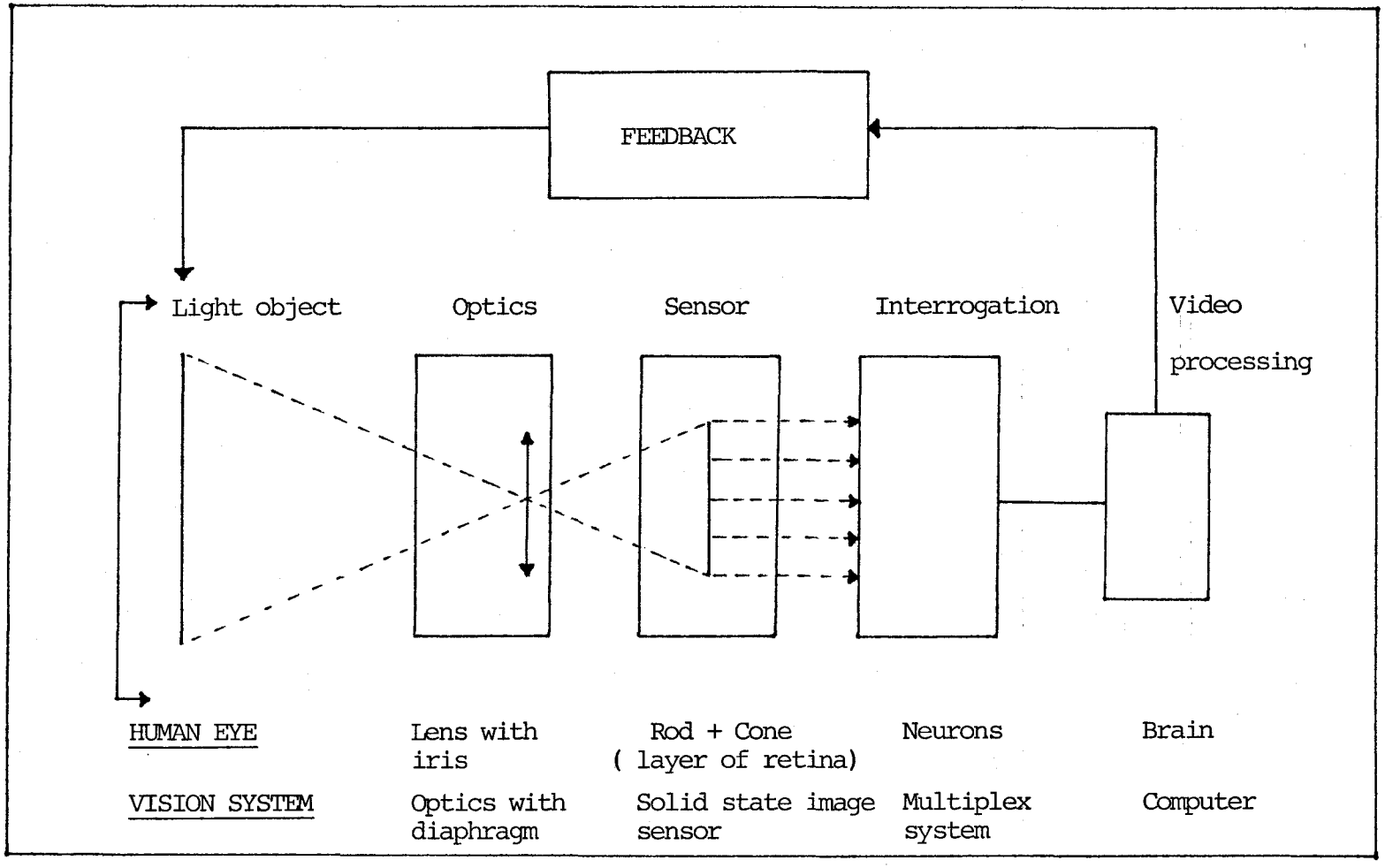


Figure 2.3 Computer vision/human eye relationship.

Figure 2.4 Vision system constraints.

INTELLIGENCE LEVEL	FUNCTIONAL REQUIREMENTS
- BASIC	ability to see
	what light wavelength?
	what intensity?
	light to electrical conversion
	ability to give out signal
- INTERMEDIARY	black/white image analysis
	timing
	spatial positioning
	level thresholding
	on/off sensing
	A to D conversion
	grey level data
- SOPHISTICATED	full video analysis
	event counting and timing
	calibration
	real time location
	control feedback
	machinery control function
	limit setting
	multi-tasking and sequencing
	computer intercommunication
	adaptability

sold commercially by Bosch in Germany [2-13].

A general computer vision process can be divided into five principal areas: sensing, segmentation, description, recognition and interpretation, which are suggested by the way computer vision systems are now generally implemented, not because human vision and reason can be so subdivided and their processes carried out independently of each other. In fact, the relationship among these functions in the human vision is not yet well understood to the point where they can be modelled analytically, so that most of the state-of-the-art computer vision systems for industrial purposes are implemented based on practical approaches.

Computer vision has became a part of the larger field of Artificial Intelligence which aims at building machines that behave intelligently, in perceptual and other domains. The capabilities of machines to see are far less than human capabilities, many fundamentals issues of processing and representation are unresolved, so that the fantasies of the science fiction writers are far away from becoming reality.

However, the current known techniques are adequate for building machines with limited, but useful abilities to see for applications to real problems.

The problems of machine vision perception are analogous to those of human visual perception that have been studied for a long time by psychologists [2-14],[2-15], but the psychological research is not advanced yet to the extent of providing many concrete models that can be programmed for a machine. The result is that most computer vision systems today accomplish very particular tasks for very particular problems, and they are far away from becoming general purpose vision systems.

2.4 Image model

The term image, refers to a two-dimensional light intensity function denoted by f(x,y), where the value of f at spatial coordinates (x,y) gives the intensity (brightness) of the image at that particular point. Light is a form of energy, therefore f(x,y) is bigger than zero and of a finite value, that is:

$$0 < f(x,y) < \infty$$

However, in practical problems, light ranges between zero and a finite value. The nature of f(x,y) may be considered as being characterized by two components: the amount of source light incident in the scene in question, and the amount of light reflected by the objects in the scene.

These components are called illumination and reflectance, and can be denoted by: i(x,y) , r(x,y) , which combined as a product form:

$$f(x,y) = i(x,y)\ r(x,y)$$

where:

$$0 < i(x,y) < \infty$$

and

$$0 < r(x,y) < 1 \ .$$

The reflectance is 0 for total absorption and 1 for total reflectance. The nature of i(x,y) is determined by the light source, and r(x,y) by the characteristics of the objects in the scene. Some typical values of these components are shown in table 2.5 [2-16].

ILLUMINATION:	
clear day	9000 foot-candles
cloudy day	1000 foot-candles
full moon night	0.01 foot-candles
commercial office room	100 foot-candles
REFLECTANCE:	
black	0.01
stainless steel	0.65
silver-plated metal	0.90
snow	0.93

Table 2.5 Typical values of illumination and reflectance.

The brightness value at coordinates (x,y) in an image is called grey-level (l) and lies in the range:

$$L_{min} \leq l \leq L_{max}$$

In theory, the requirements on L_{min} is that it must be positive and on L_{max} that it must be finite. In practice:

$$L_{min} = i_{min}\, r_{min}$$

and $$L_{max} = i_{max}\, r_{max}\ ,$$

so that, for indoors applications one may expect the values:

$L_{min} = 0.005$ and $L_{max} = 100.$

The interval $[\ L_{min}\ ,\ L_{max}\ ]$ is called the grey-scale and

is commonly in practice valued to [0,L] , where l=0 is considered black and l=L is considered white , all intermediate values are shades of grey.

An image function f(x,y) must be digitized spatially and in amplitude in order to be used in computer processing.

By uniformly sampling the image and an amplitude quantization of the grey-scale, the function f(x,y) can be represented in the form of an NxM array:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & & & \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

which is called a digital image and where each element in the array is referred to as an image element, picture element or pixel.

Each pixel takes any value G within the discretized grey-scale. In practical digital image processing, these values are integer powers of two, that is:

M and $N = 2^n$ (array dimension)

$G = 2^m$ (number of grey levels).

The number of bits required to store a digital image is then given by : $b = N \times M \times m$.

The resolution of an image is the degree of discernible detail and is directly dependent on the dimensions of the array and the number of grey-levels. It is obvious that the better resolution, the more accurate results are obtained in a particular vision application. However, the better resolution implies a greater quantity of data to be processed and consequently the more cost and complexity of a vision system. The more grey-levels and sampling-grid size are increased, the closer the digitized array will approximate the original image. Figure 2.6 shows the effect of reducing the sampling-grid size of an image, and keeping in all cases 256 grey-levels. Figure 2.7 illustrates the effects of reducing the number of grey-levels in an image and keeping in all cases sampling-grid sizes of 512x512 pixels.

2.5 Histogram of grey-levels

The grey-levels in a digital image can be histogrammed. In this case, a histogram of grey-levels represents the distribution of light intensity (brightness distribution) in that digital image, (figure 2.8 shows a typical histogram).

For a practical purpose, a histogram of grey-levels can be defined as the plot of the grey-scale versus the frequency of occurrence of each grey level in a digital image. A grey-level histogram is useful for automatic threshold adjustments when threshold selection techniques are used for image segmentation. For image enhancement techniques, where the distribution of light is a fundamental and important data when applying spatial transformations (direct manipulation of the pixels), the histogram of grey levels becomes very useful.

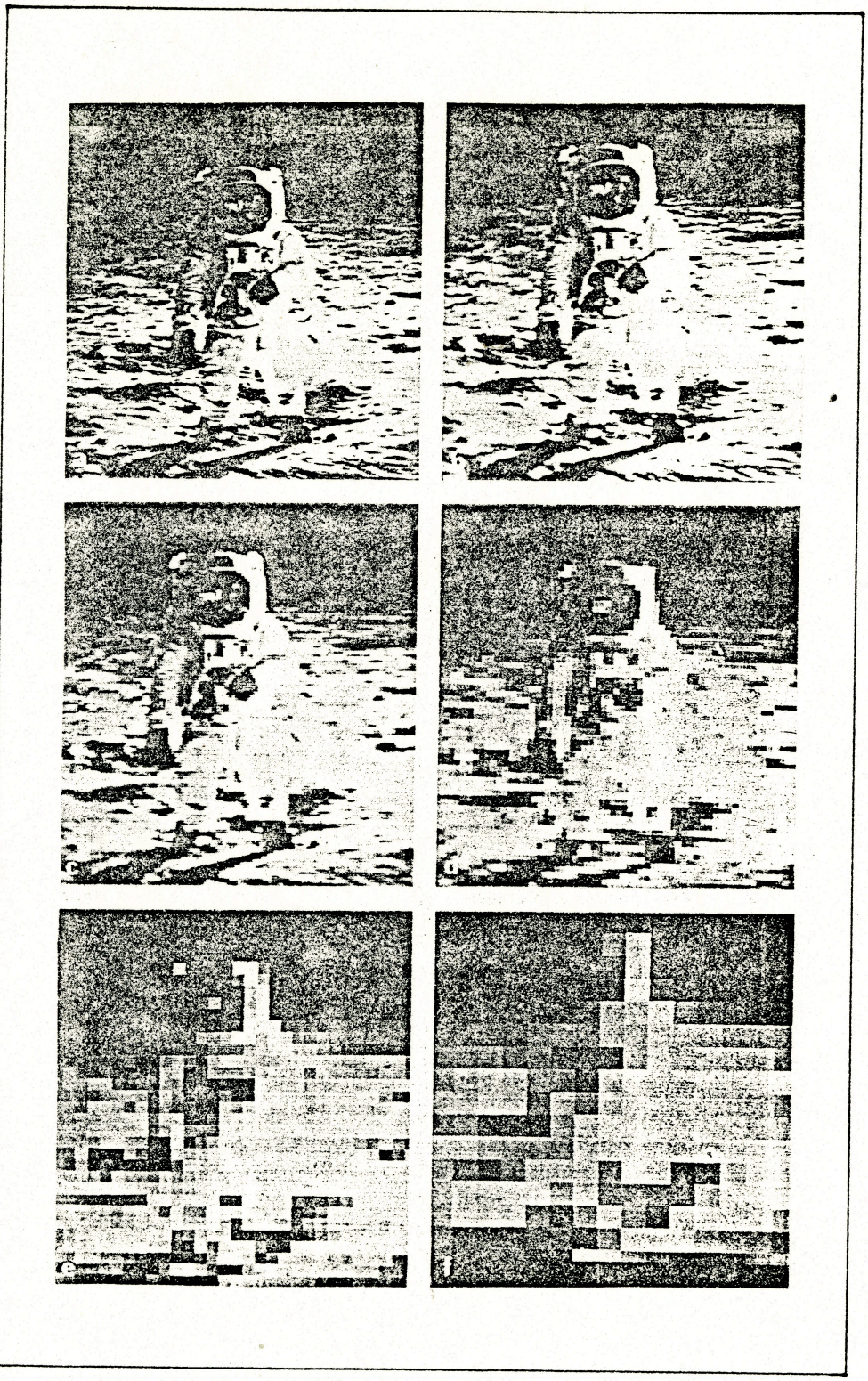


Figure 2.6 Effects of reducing sampling-grid size (reproduced from [2-17]).

Figure 2.7 Effects of reducing the number of grey-levels (reproduced from [2-17]).

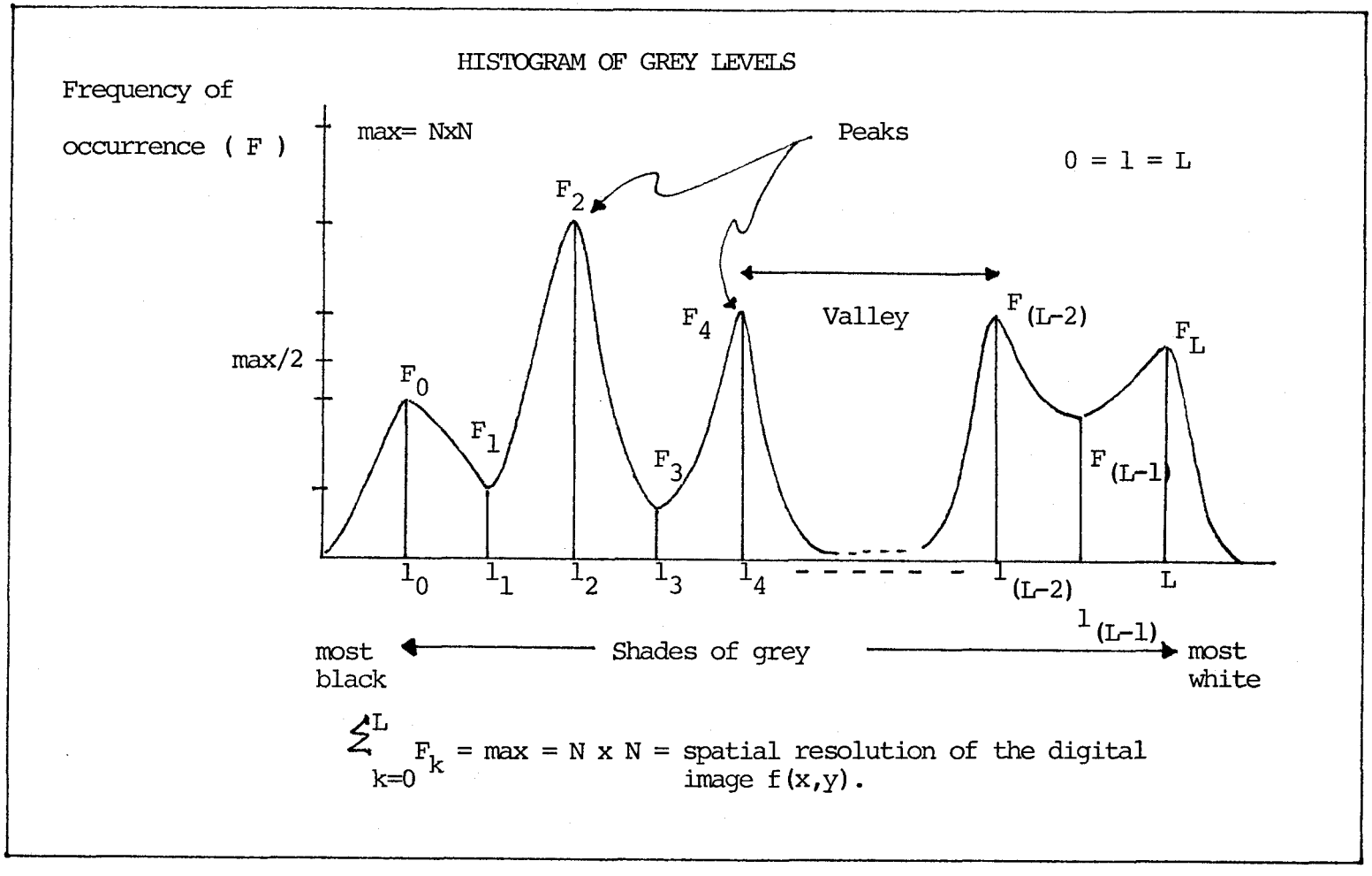


Figure 2.8 Histogram of grey-levels.

Grey-level histograms are computed generally by way of specialized hardware when used in real time applications. There are methods that improve histograms, and they can be modified for different purposes.

Histogram modification techniques for image enhancement of digital pictures is a typical example of the usefulness of such histograms. Figure 2.9 shows an original and modified image after applying a histogram equalization technique, which is based on the histogram modification of the original one.

Usually these techniques are used in a preprocessing stage of digital image processing. They increase the probability of correct pattern detection and recognition in autonomous applications such as: robotics and military systems. Woods and Gonzales [2-18] have developed a system which performs image enhancement in real-time by using histogram techniques.

In this report, such a histogram is used to accomplish lighting adaptability for the vision system.

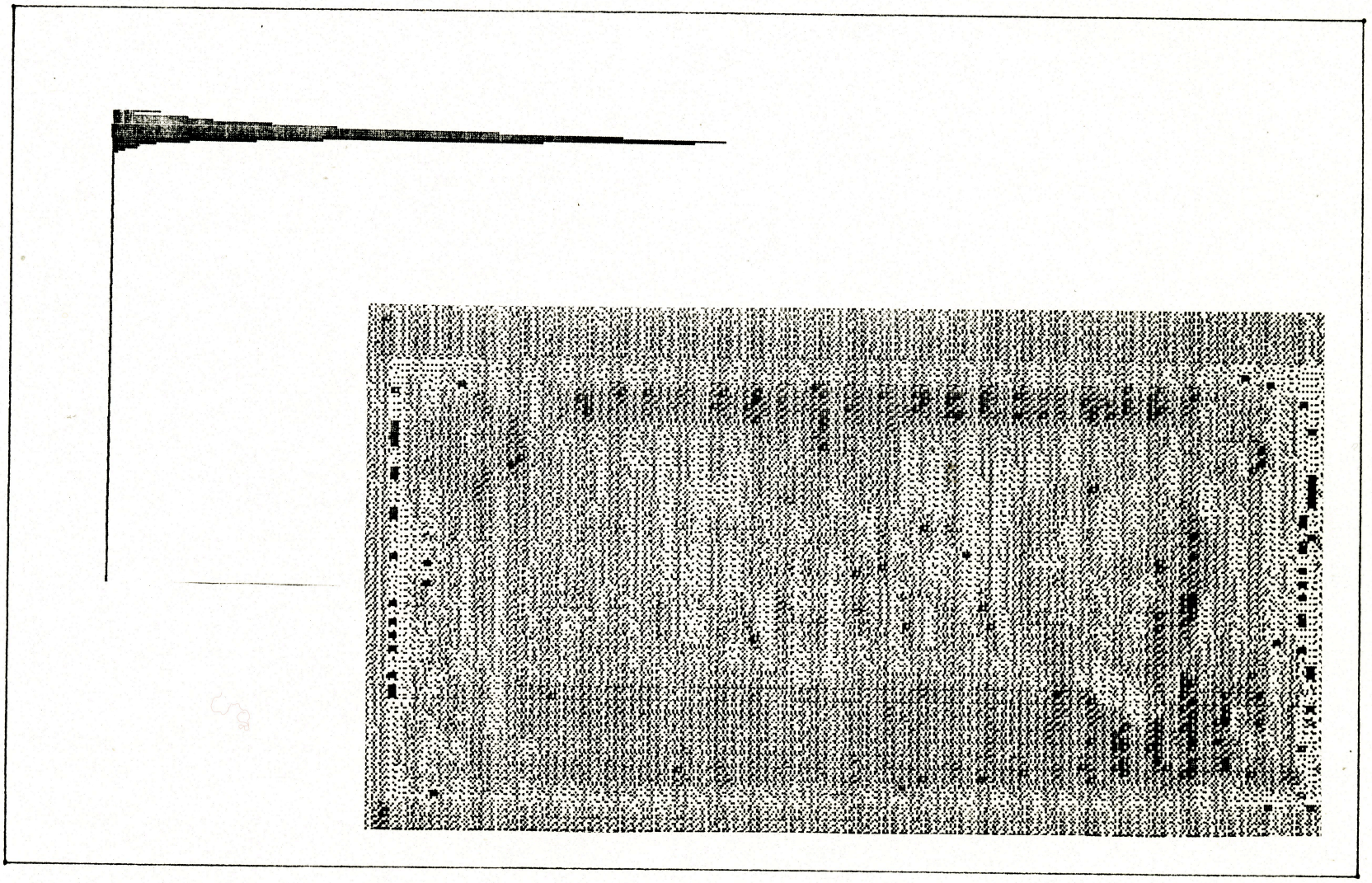


Figure 2.9 Original image and histogram.

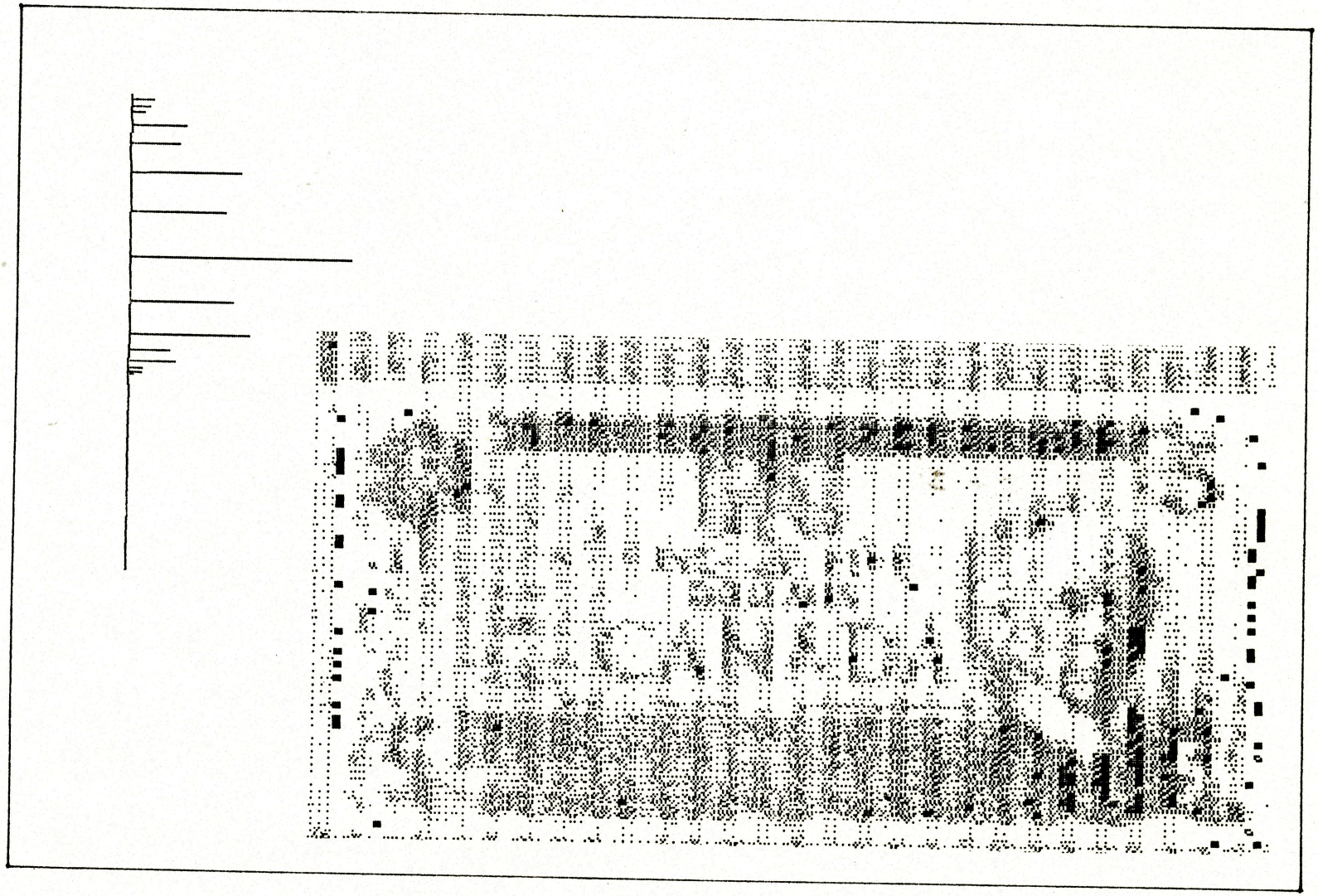


Figure 2.9 (cont.) Modified histogram and enhanced image.

2.6 Thresholding

The use of thresholding as a tool in image segmentation has been extensively studied, and is by far the most widely used approach for segmentation in industrial applications of computer vision.

The reason why thresholding techniques are used in most of the current vision systems is because they are fast and quite straightforward to implement in hardware by way of comparators.

The lighting environment is usually an unstable factor in industrial applications which results in images that are more appropriate for segmentation with a thresholding approach.

Segmentation of images using thresholding is carried out by way of a threshold operator (T_i) , which in its simplest form takes a value T , the threshold value , and sets all pixels of an image of value below T to white (or black) and all pixels of value at or above T to black (or white). This operator can not only take a value T, but a band of grey levels values $T_b = T_u - T_l$. The main problem with the threshold operator is to choose the optimal value for T or T_u and T_l that separates the object and the background into white and black respectively (threshold selection technique).

Different methods for determining the value of T can be used:

- a constant value for T ,
- a value T dependent upon the average intensity value,
- a value T calculated between two peaks of the intensity

frequency histogram ,

- a value T calculated to give a particular ratio of black to white.

Some of these methods, for example constant value and average intensity value, can be implemented with a single operator. Other methods, for example ratio value, can be done as a sequence of operators.

The ratio value may be obtained by calculating the intensity histogram, and by summing the value of successive elements of the histogram until the ratio of the sum to the total number of points minus the sum is greater than or equal to the desired ratio. The element subscript number, whose value was just added to the sum value, is the value,T, at which the image should be thresholded. These techniques are derived from methods used early in the SRI Vision Module described by Gleason and Agin [2-19]. However, it has been generalized for current vision systems to obtain the value of T from the histogram of grey levels of the image.

In a general form, thresholding can be defined by:

$$g(x,y) = k \quad \text{if} \quad T_{k-1} \leq C(x,y) \leq T_k$$

$$k = 1,2 \ldots m.$$

where:

(x,y) = pixel coordinates

$C(x,y)$ = characteristic feature function of the pixel with coordinates (x,y) (e.g. intensity of light).

$g(x,y)$ = segmented function

T_i = i_{th} threshold value

m = number of distinct labels assigned to the threshold image.

In a general form, the threshold operator T_i can be viewed as a function of the form:

$$T_i = T_i (x,y , p(x,y), C(x,y)).$$

where:

(x,y) =coordinates of the pixel

C(x,y) = the grey level of the point (x,y) in the image

p(x,y) = some local property (e.g. average intensity).

If T_i depends only on C(x,y) (the grey level of the point (x,y) in the image), it is called a global threshold; if it depends on p(x,y) (some local property) and C(x,y) it is called a local threshold; if it depends on p(x,y) , C(x,y) and the coordinates (x,y) it is called a dynamic threshold.

Global thresholding is used in applications for which the objects to be extracted exhibit high contrast from the background, as is the case of backlighted scenes. Local thresholding on the other hand, is used when the difference between objects and background is not clearly defined. The dynamic threshold is probably the most powerful way of thresholding in segmentation although the most difficult to implement.

In most practical applications that work with silhouettes, the threshold value is set empirically, but it would be very convenient to have a method for automatically setting this threshold to its optimum value.

Although different schemes for global, local and dynamic thresholding can be used, in industrial vision applications the use of global threshold values have been generalized by using global threshold selection techniques based on the analysis of the image's grey level histogram. In fact, some of the earliest techniques for automatic threshold selection were global methods based on this analysis.

A simple " p-tile " (based in percents values) method was suggested by Doyle [2-20] if the objects in an image are darker than the background, and @ccupy a fixed percentage of the picture area. This method is not applicable if the object area is unknown or varies from picture to picture. Prewitt and Mendelsohn [2-21]choose thresholds at the valleys (or antimodes) on the histogram. Their schema called the mode method, involves some smoothing of the histogram data, searching for modes and placing thresholds at the minima between them.

The Prewitt and Mendelshon method relied heavily on the structure of the grey level histogram which contains peaks and valleys corresponding to grey level subpopulations of the image. Object and background regions (represented by histogram peaks) are assumed to be of fairly constant grey level, and to differ in average grey level.

Edges (represented by valleys) are composed of intermediate grey levels and are less heavily populated than either object or background. Other work done by Wall [2-22] , has attempted to model objects and their edges by deriving the structure of the corresponding grey level histogram. Local property statistics have been used to help in the selection of global thresholds by improving the shape of the

grey-level histogram; for example, making it more bimodal and then applying the mode method described before.

Mason [2-23] has proposed a method for making histogram valleys deeper, in order to facilitate the use of the mode method. He computes a histogram in which not all points are counted equally. The lower the value of a difference operator at a particular point, the more weight is given to that point. The overall effect of the weighting process carried out by this technique makes the histogram peaks sharper and higher and the valleys deeper, so that , the bottom of the valley can more easily be selected as a threshold.

For histograms in which the valley is broad and the peaks are very unequal in size, a technique based on a digital " Laplacian " operator can be used to produce a strongly bimodal histogram. This technique was suggested by Weszka, Nagel and Rosenfeld [2-24] . The " Laplacian " operator is computed by taking absolute differences between the grey levels at each point in the image and the average of its eight neighbor grey levels, resulting in a deepened and sharpened valley bottom separating the two peaks. A complete evaluation of the different techniques is presented by Weszka and Rosenfeld in [2-25].

2.7 Lighting schemes

The output signal of a camera depends much upon the way in which the objects are presented as an image to the camera. It is very important to present these objects within the field of view with high contrast in order to obtain accurate signals of information.

Optics and illumination today already represent an area of research in industrial machine vision techniques [2-26]. The principal objective in present vision systems is to minimize the quantity of data processing complexity and time needed to extract the necessary information from the image by controlling the illumination image environment. A wide variety of techniques have evolved to reduce the computing required in vision machines, but at the present there is probably less theoretical understanding and general knowledge of specific techniques of illumination than other subjects related to machine vision.

Low image contrast, shifts in brightness, shadows, reflections , changes in the background or changes in focus can affect the performance of the pattern recognition technique being used.

An understanding of the interaction of light with the objects of interest, programmable illumination, and an adaptability to lighting for different image conditions having interaction with specific processes are needed.

Vision systems today consider the illumination a very important factor, to the point of affecting the complexity of

vision algorithms. Several basic schemes are used at the present time for illuminating parts to be inspected or manipulated. Figures 2.10,2.11,2.12 and 2.14 [2-27] show these schemes.

The diffuse lighting approach shown in figure 2.10 can be employed for objects characterized by smooth regular surfaces. The effectiveness of this technique is illustrated by Shirai and Tsuji [2-28] who used sequential diffuse lighting from several directions to obtain line drawings of polyhedral objects.

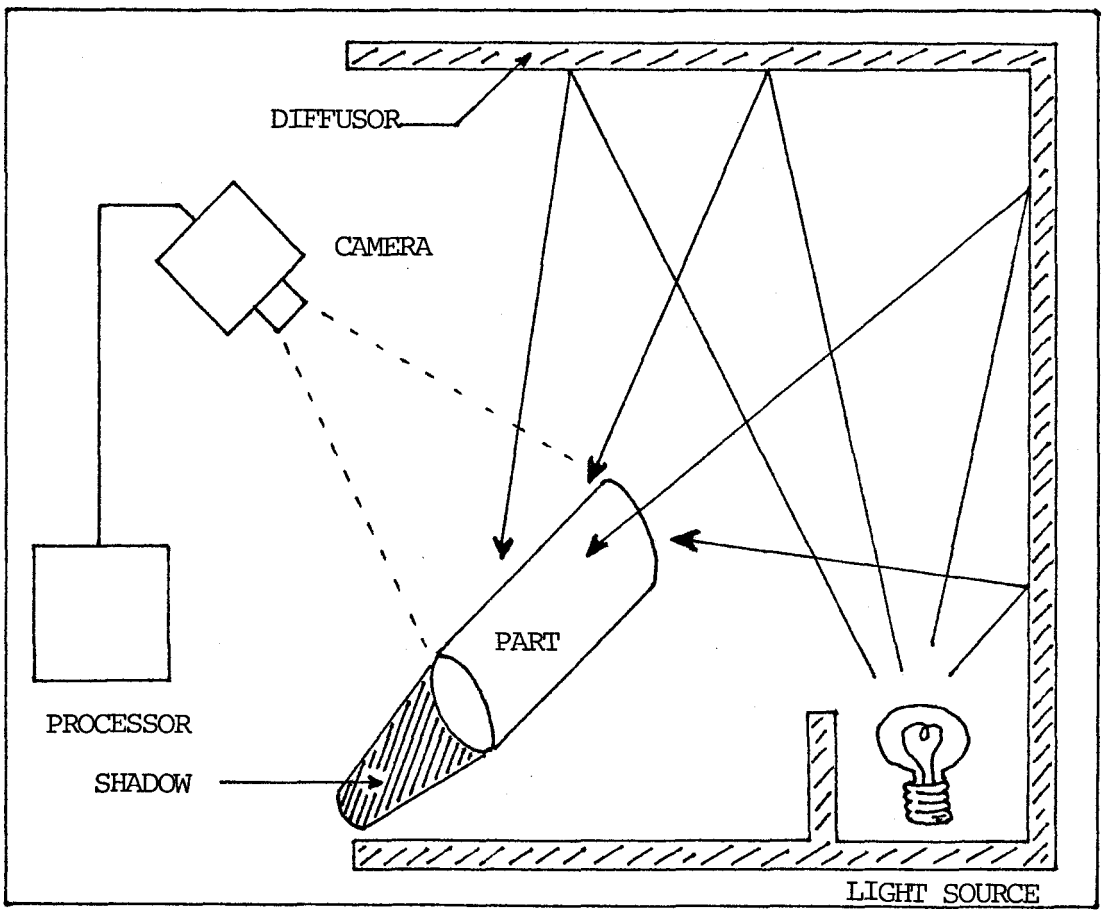


Figure 2.10 Diffuse lighting scheme (reproduced from [2-27]).

The backlighting scheme is shown in figure 2.11. This scheme is ideally suited for applications in which silhouettes of objects are sufficient for recognition and feature extraction measurements. Jarvis discusses this technique in [2-29].

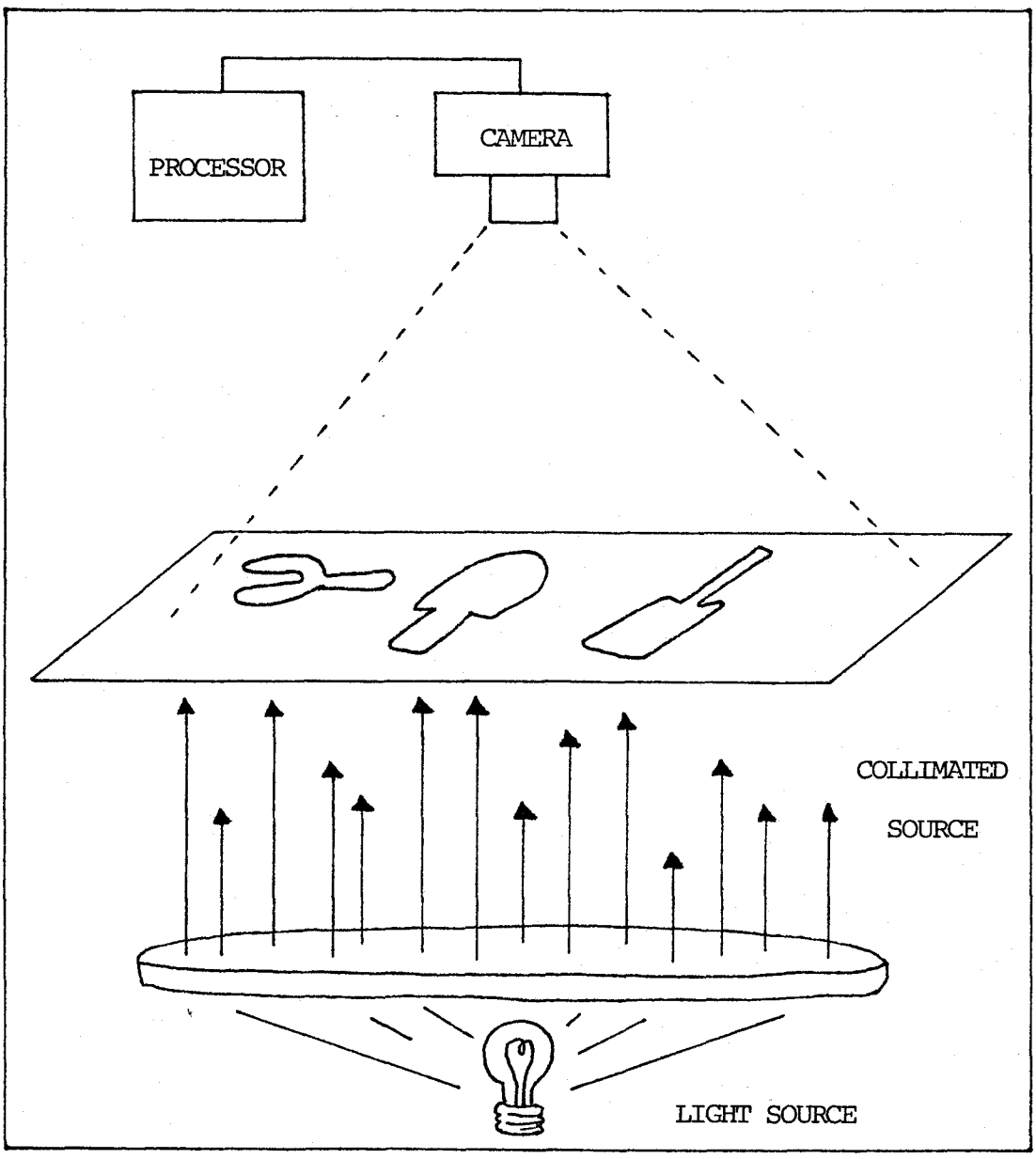


Figure 2.11 Backlighting scheme (reproduced from [2-27]).

The figure 2.12 shows a lighting scheme that uses a spatially modulated light source. Most times this technique is used for object classification, and involves projecting points, stripes and grids on an object. The curvature of the object distorts the light pattern. This distortion is detected in the digital image and is used to find the object curvature, or only the presence or absence of it.

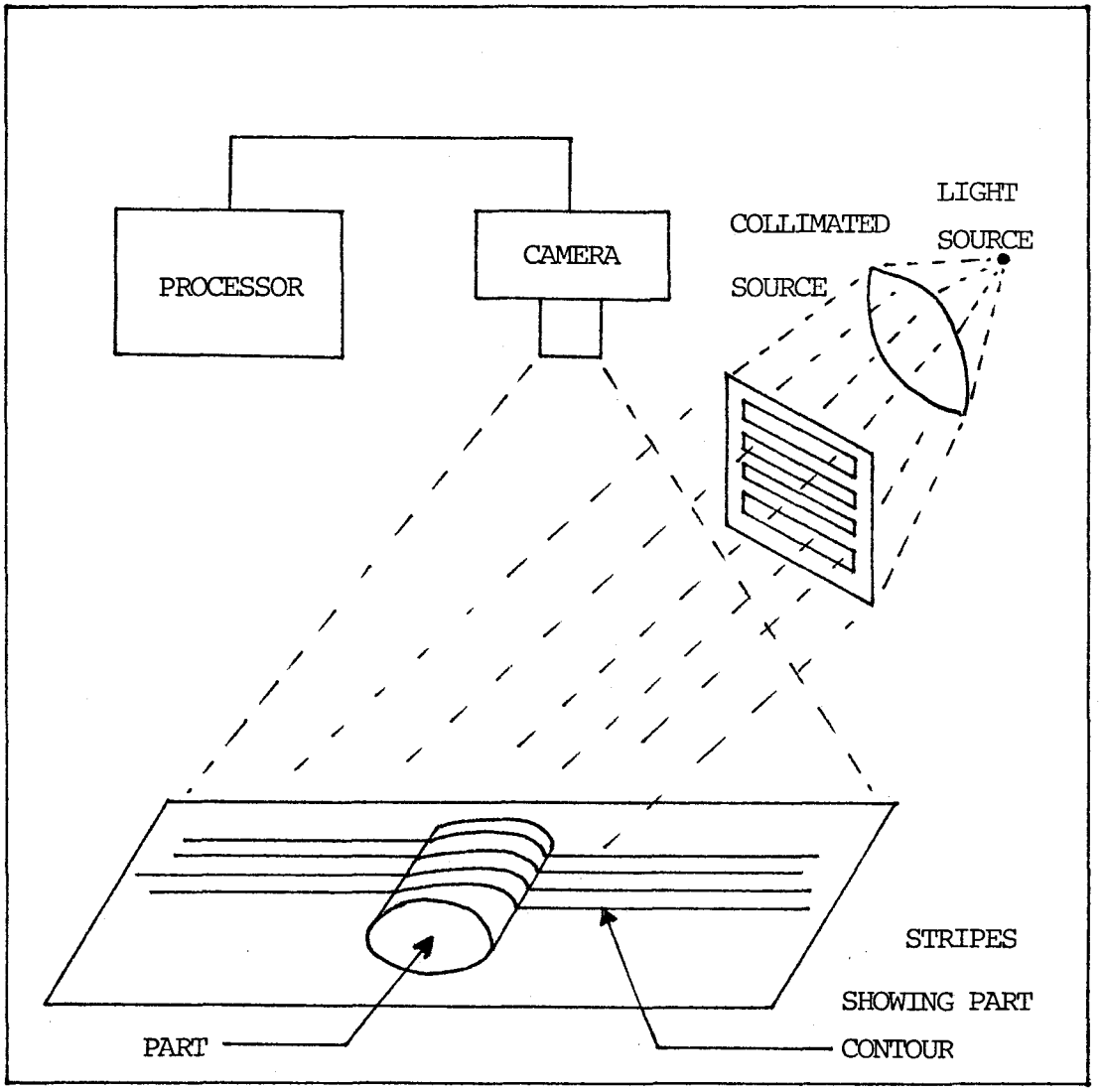


Figure 2.12 Spatially modulated scheme (reproduced from [2-27]).

The spatial modulated scheme is successfully used in the Consight-I system developed at General Motors Research Laboratory [2-30]. The lighting pattern used is a narrow intensity line projected across a moving conveyor belt surface perpendicular to the direction of motion. When objects pass below the source beam, it intercepts the light before it reaches the belt surface on the conveyor. When this is viewed from above by a linear array sensor, the line of light appears deflected and the presence of a part can be detected as shown in figure 2.13

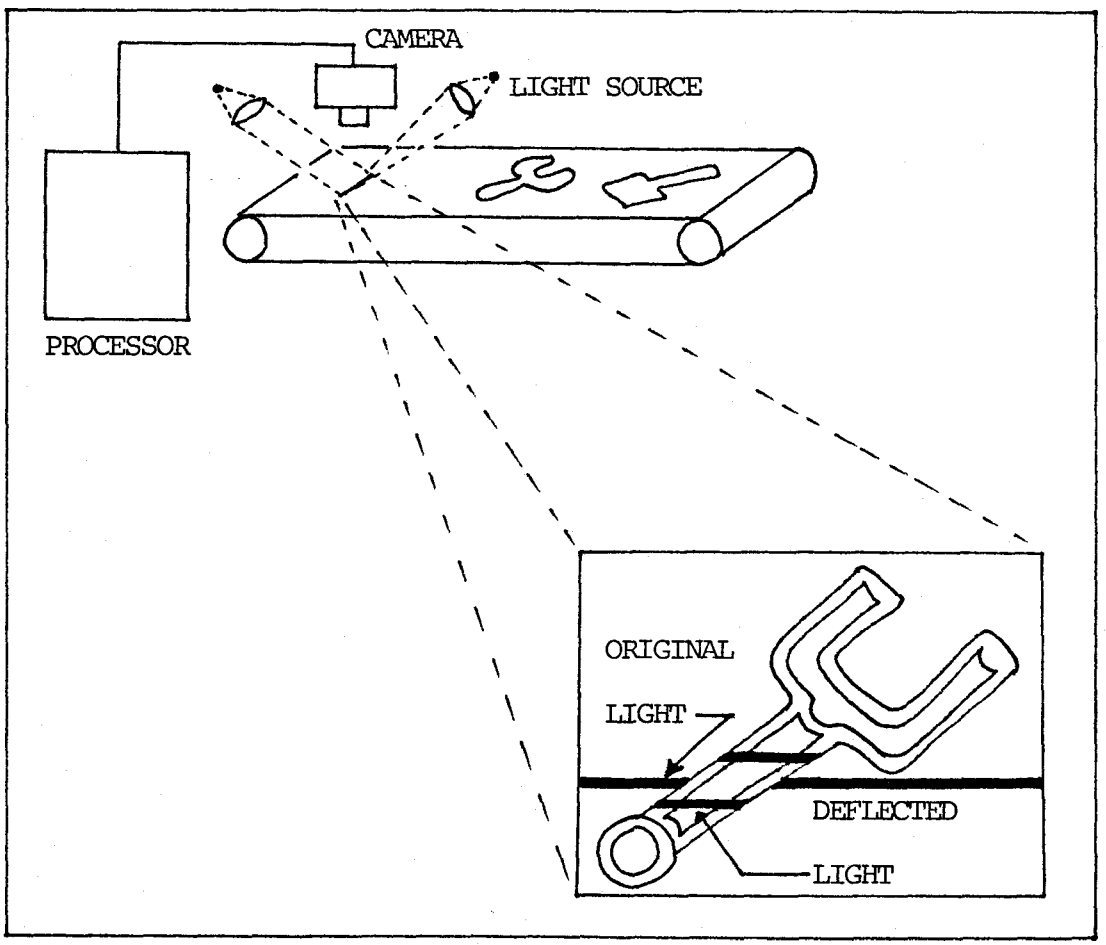


Figure 2.13 Consight-I system using the spatially modulated scheme

The directional light approach is shown in figure 2.14. This scheme is ideally suited for the inspection of rough surfaces. Imperfections on the surface can be detected by carefully choosing the projected light. When the surface shows almost no imperfections, only a little light is scattered up to the camera. When the imperfection is big enough, the scattered light is of considerable amount and the defect can be detected by the vision system.

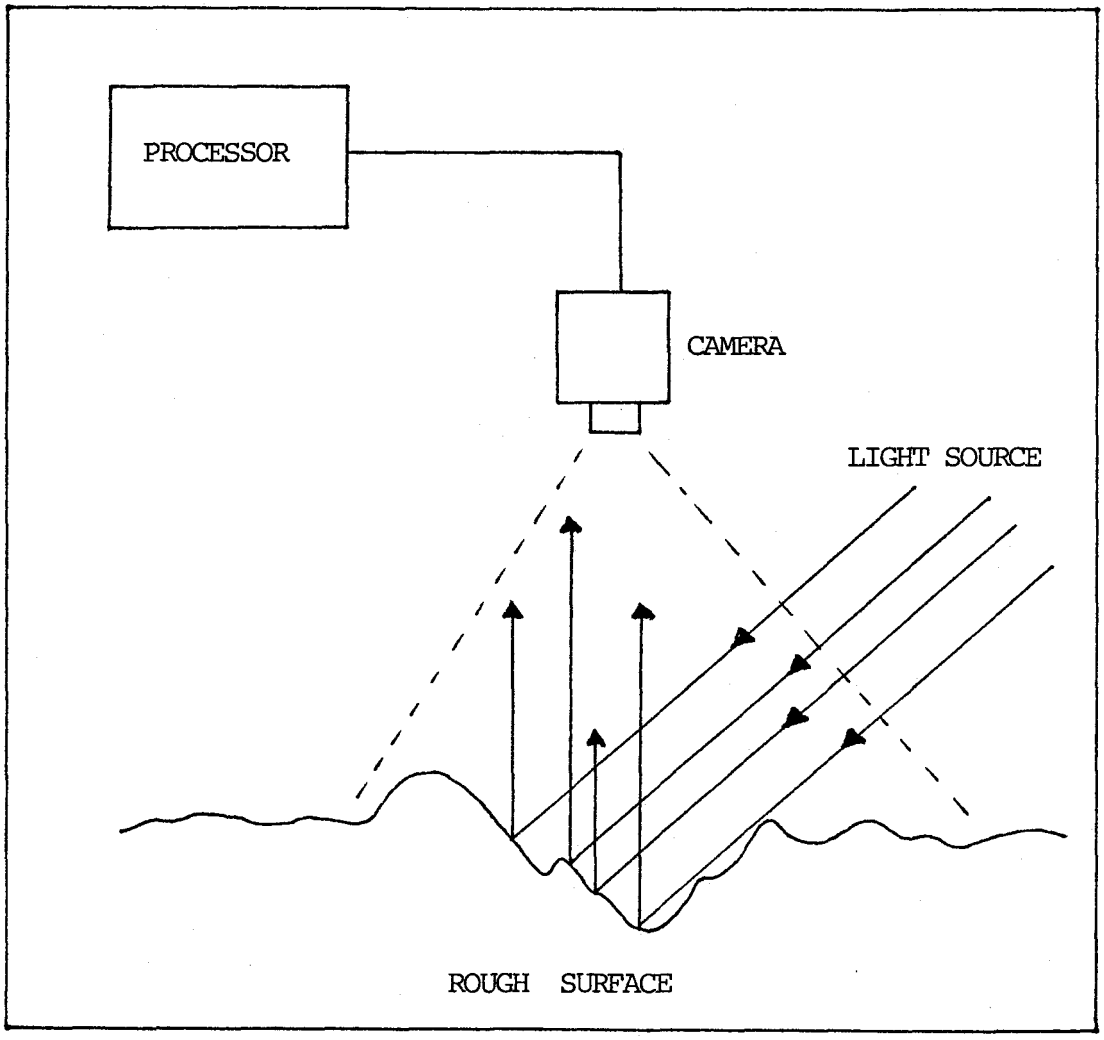


Figure 2.14 Directional light scheme (reproduced from [2-27]).

In any discussion of vision systems, it is important to point out that ordinary light sources are not always the appropriate choice; for example, many inspection applications use laser beams to improve brightness effectiveness and beam directionality. An interesting application of laser beams in robotics is the Kawasaki Vision System Model 79A [2-31] which is used for path corrections in arc welding.

A laser unit projects a light pattern onto the surface of the part to be welded. The shape of the surface distorts the light pattern, and this distortion is used by the vision system to determine the relative position of the point to be welded.

The system described in this report, does not attempt to correct the light scheme used for the application described before, but to adapt the vision system to instabilities in that light scheme and within an adaptive range.

CHAPTER III

HARDWARE IMPLEMENTATION

3.1 General architecture

The adaptive lighting system (ALS) comprises a G.E. TN2500 solid state T.V. camera, a sensitivity control unit, a video rate histogram generator (VRHG), and an 86/12A microcomputer. It uses a graphics printer to display results and a CRT terminal to communicate with the microcomputer.

The system was attached to a silhouette machine vision in order to make it adaptable to different ambient light. A dual threshold unit, a binary pixel packer unit and a status register belonging to this vision system [3-1] are used by the ALS to perform this task. In fact, the ALS can be attached to any vision system which requires lighting adaptability by making minor modifications in the hardware and software of the user vision system.

Each time that a picture is requested from the camera by the vision system, the ALS generates a histogram of grey levels of the current scene and statistical calculations are performed on the histogram to obtain an adequate threshold value for the " grey-scale to binary " image conversion. The criteria for setting the sensitivity of the visual transducer in the camera uses this information, and the sensitivity changes are achieved by adjusting the integration time in the read out of the visual transducer (CID).

The general architecture of the vision system with the ALS is shown in figure 3.1 . In the figure, the 3-dimensional blocks are the actual modules that comprises the ALS . The shaded modules interact with the ALS but are part of the user's vision system.

Video input to the system is by way of the General Electric TN2500 digital T.V. camera. As shown in figure 3.1, the internal architecture of the vision system is designed around three buses:

1.- The multibus supports the Intel iSBC 501 DMA controller (Direct Memory Access) and the iSBC 86/12A microcomputer. I/O ports in the microcomputer provide access to the serial robot interface, a status register, a threshold logic unit, and the sensitivity control unit.

2.- The video bus, which is comprised of the digitized video (8-bits), control lines, and associated timing pulses (element rate clock, end of line, end of frame, etc.) of the T.V. camera.

3.- The DMA bus, which deposits the acquired data in the RAM of the microcomputer board.

The dual threshold unit shown in figure 3.2 converts the 8-bit grey-scaled video into a single bit binary by comparing each pixel value to a pre-calculated lower (T_l) and upper (T_u) bound. These values are retained in two registers, and can be altered by the ALS every time a picture of the scene is requested. For any 8-bit pixel (p), the output of the threshold unit will be:

$$\begin{array}{l} 1 \text{ for } T_l \leq p \leq T_u \\ 0 \text{ otherwise} \end{array}$$

and a silhouette is obtained from the grey-scaled picture.

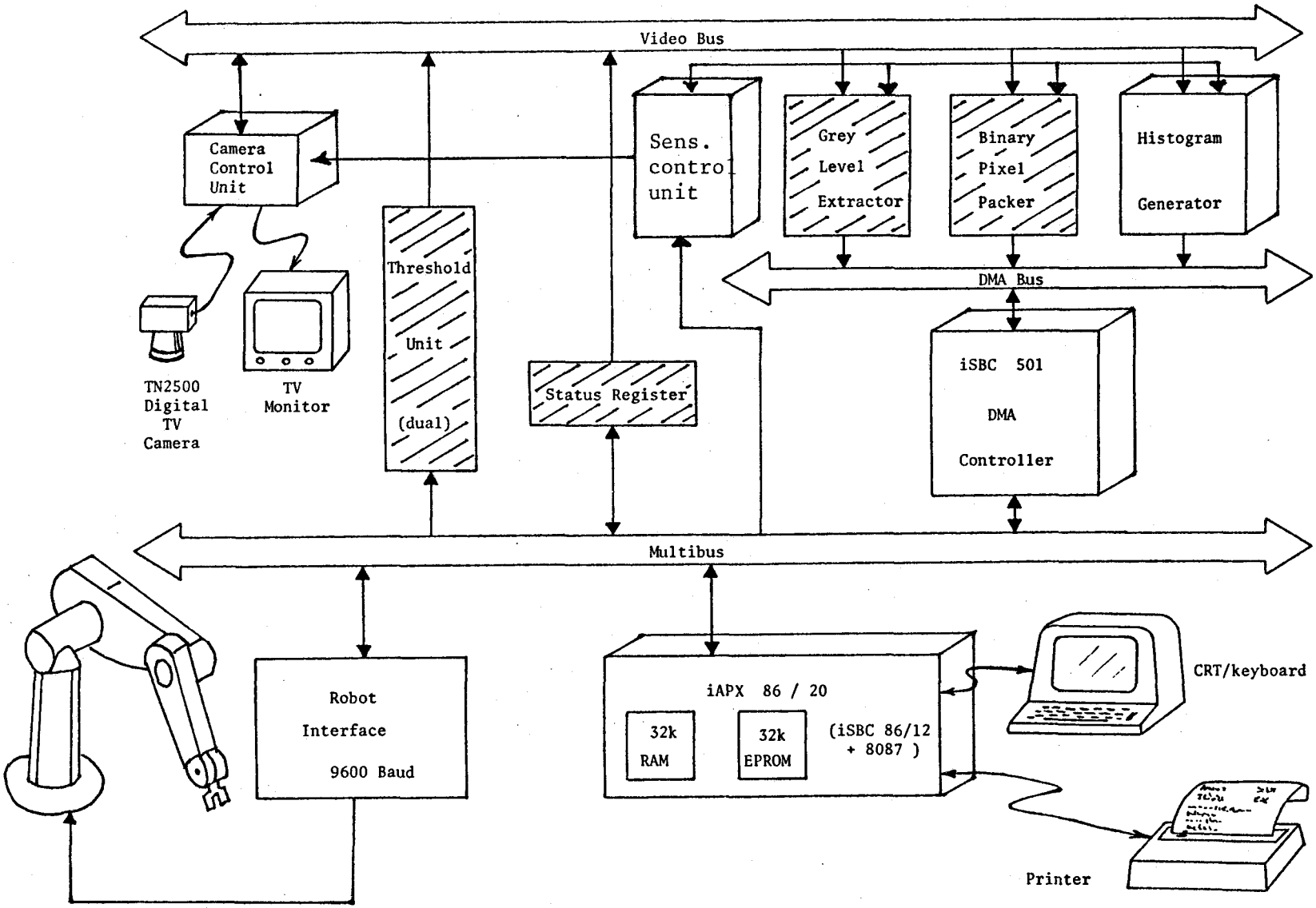


Figure 3.1 General architecture and ALS/ vision system interaction

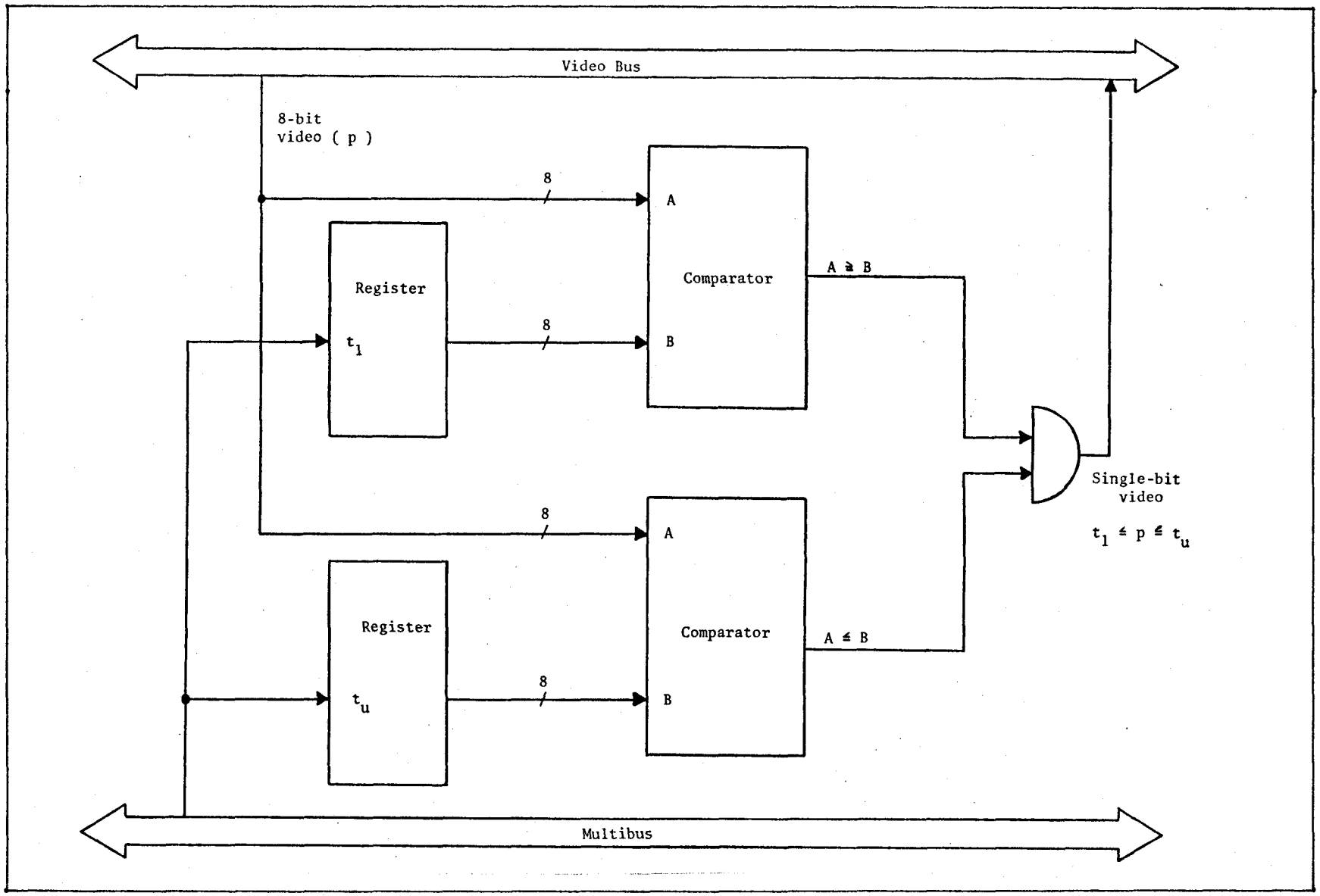


Figure 3.2 Dual threshold unit (reproduced from [3-1]).

Various types of video data can be captured from the video bus and sent to the iSBC 86/12A for processing. In each case, acquisition is accomplished in a single T.V. frame (33 ms.). For each data type, a hardware module is inserted between the video bus and the DMA bus (the ALS has its histogram generator situated in this part of the vision system). In order to choose a different data type, a status register is implemented between the video bus and the multibus. This status register, selects the background colour of the silhouettes and the control of the scan mode in the camera as well.

The binary pixel packer collects an image of 232x240 single bit pixels which are packed 8 pixels/byte and transmitted in parallel to the microcomputer RAM through the DMA bus. The implementation of the device is shown in figure 3.3. The most significant bit (MSB) of a 4-bit counter enables one of two serial in / parallel out shift registers. The first 8 binary video bits are clocked into the shift register. The MSB of the counter now toggles and this enables the second shift register to acquire the next 8 pixels. Simultaneously , a one-shot is triggered by the counter, thus issuing a transfer request to the DMA bus. This alternating process continues until a full frame has been acquired. The frame detection logic clears the counter at the beginning of each frame and line.

3.2 TN 2500 G.E. camera

As mentioned before, the video input to the system is by way of the General Electric TN 2500 solid state T.V. camera. This camera

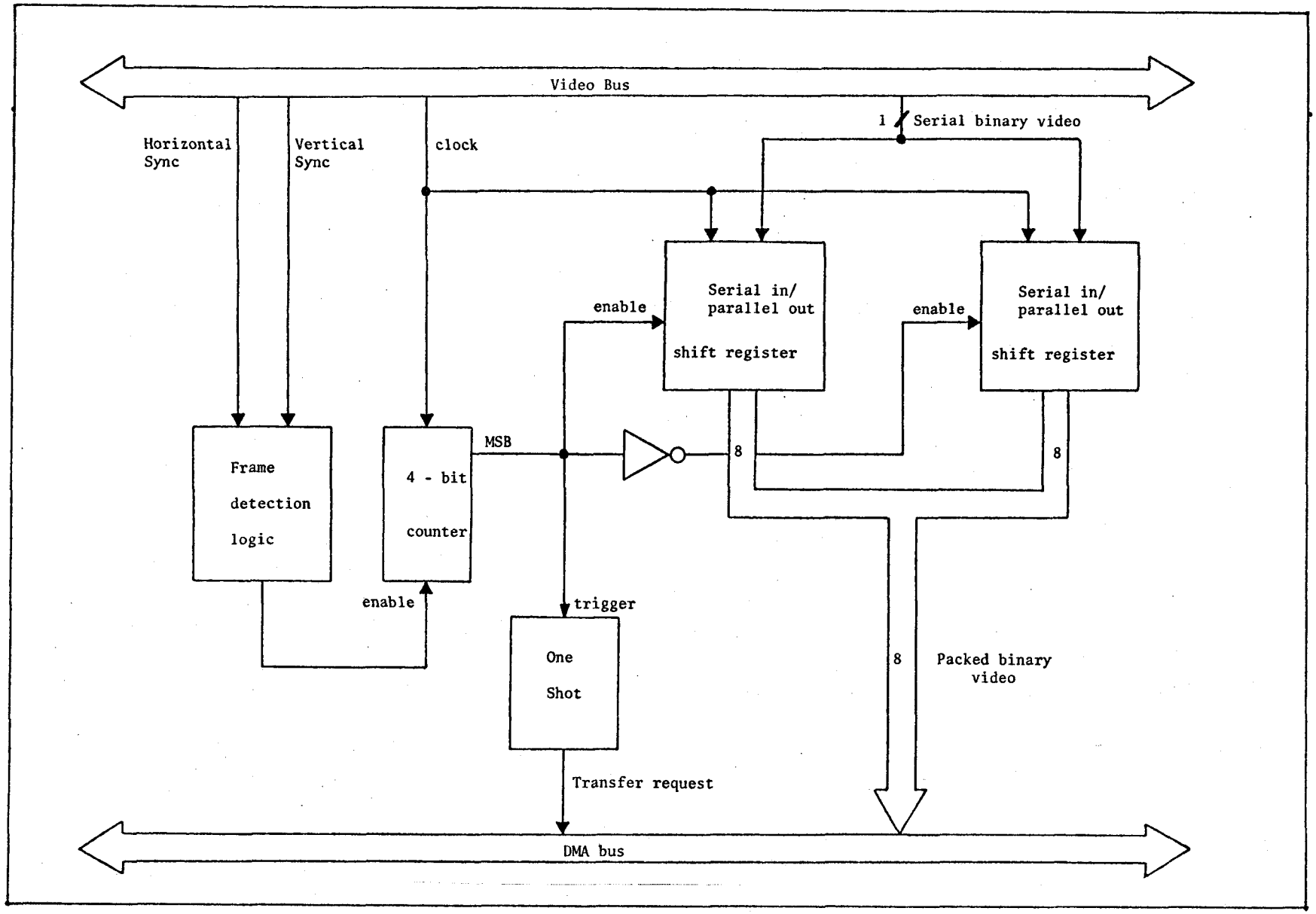


Figure 3.3 Binary Pixel Packer (reproduced from [3-1]).

contains a charge injection device (CID) imaging sensor array comprised of over 60 000 light sensing picture elements (pixels) in a 3x4 aspect ratio format. The CID is totally solid state and its format features 244 rows of 248 pixels each. The camera includes a self contained camera control unit (CCU) which generates all timing signals, and effectuates the analog to digital conversion of the video output (each grey-level coded as an 8-bit word). Video signals (e.g. the digitized video output, vertical and horizontal sync.) are extracted from the CCU and deposited on the video bus of the vision system.

3.2.1 CID sensor

Charge injection imaging makes use of a two dimensional array of coupled metal-oxide-silicon (MOS) capacitors to collect and store the photo generated charge. Coincident X-Y selection is used to inject the stored charge into the bulk silicon. The time integrals of the charging currents to the MOS capacitor plates are detected for read out.

The basic CID is an integrating photon detector [3-2], consisting of a MOS capacitor biased above threshold. The charge generated by incoming photons is collected and stored by the device; as more charge is collected, the capacitance increases. When a pulse of a lower voltage value is applied to the gate, the collected charge is injected into the substrate, where it is collected. For read out, the signal charge is detected by integrating the charging current or

measuring the capacitance change. For two dimensional operation of this device, which constitutes an image cell, two MOS capacitors at each sensing site are coupled together, so that the stored charge can be transferred from one capacitor to the other. Thus a two-axis selection method is used for scanning.

The TN 2500 camera uses a preinjection read out mode whose principal advantage is high speed performance with no column blooming due to sensor overloads [3-2]. This preinjection read out is achieved by sequential connections of the image cells to a reference voltage to produce current flows generated by the incoming photon charges stored in each cell. These current flows constitute the actual video signal.

The image cell selection is accomplished by the presence of digital pulses that normally integrate and inject the signal charge in each cell once per frame.

By altering these integration and injecting times by way of disabling and enabling the driven pulses, it is possible to change the sensitivity of the whole transducer which constitutes the inject inhibit mode operation of the camera. This mode is used by the ALS to perform its adaptive task.

3.2.2 Operating modes

The TN 2500/GE camera can be operated in three modes:

i).- normal interlaced scan mode

ii).- sequential scan mode

iii).- inject inhibit mode.

Since the system uses only the sequential and inject inhibit modes, only these are mentioned. In the sequential mode, a frame is composed of only one field; consequently the field rate equals the frame rate of 30 Hz. The ALS uses a 244 sequential format frame presentation shown in the figure 3.4 . In such a frame presentation, each CID imager line is read and displayed twice.

The inject inhibit mode is used by the ALS to adjust the integration time of the read out in the CID. These changes in the integration time permit different sensitivities in the transducer as explained in section 3.2.1 .

3.2.3 Video synchronization signals

The ALS uses the following video synchronization signals (Tri-state low power-TTL compatible) from the CCU to activate the VRHG:

i).- Digital video output (b_1 to b_8); video bits 1 through 8 comprise an eight bit binary number which represents the radiometric value of a given pixel. The digital video output data rate is 4.5045 MHz.

ii).- Synchronization clock output enable ; this signal controls the output state of the synchronization clock signals. It also is an input line to the CCU. The synchronization clock signals provide element field and video blanking information .

iii).-Element rate clock (ERC) ; a negative transition of the element rate clock (4.5045 MHz.) denotes the presentation of new digital video data.

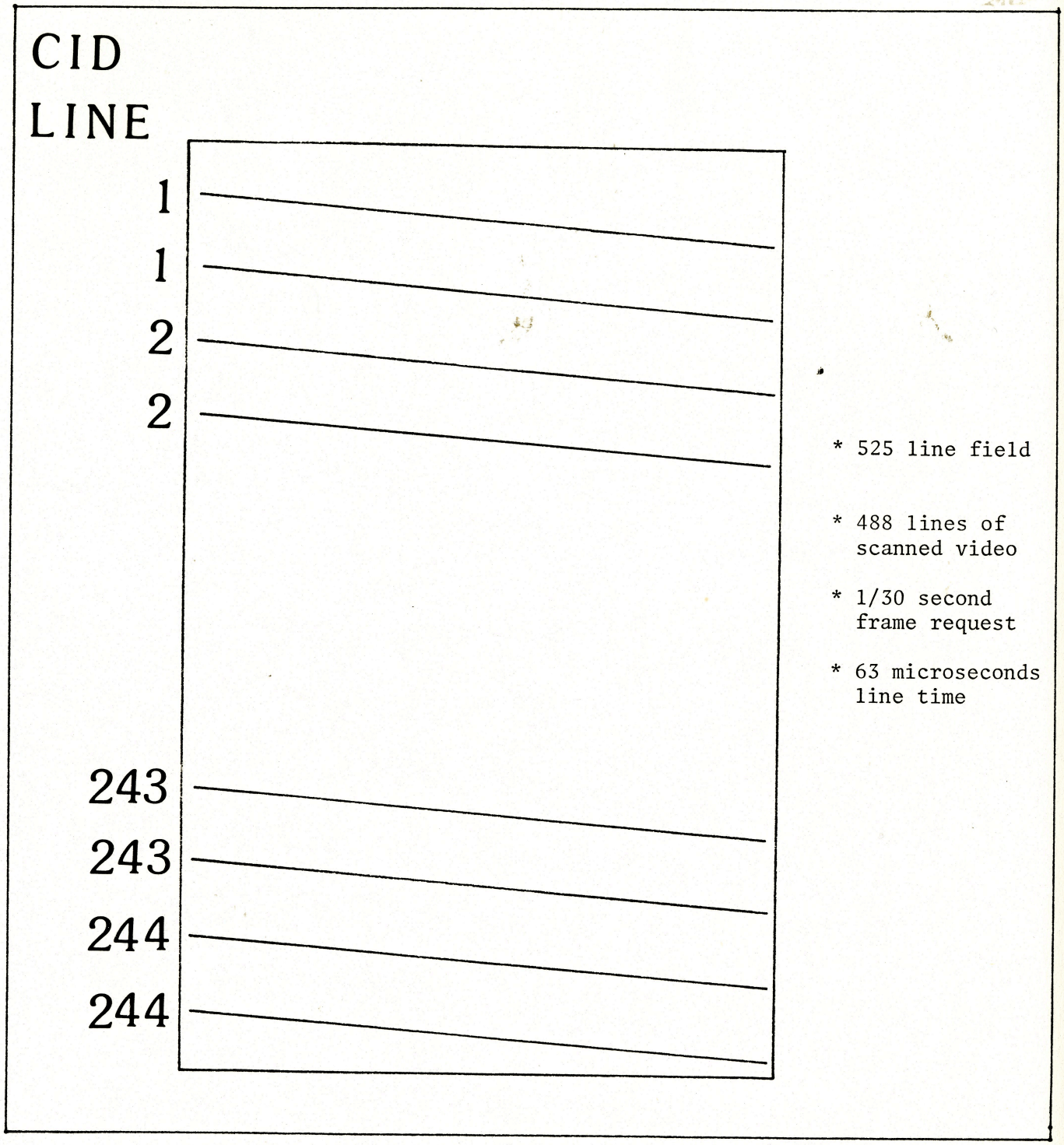


Figure 3.4 TN2500 244 line sequential mode frame presentation

iv).- Synchronized blanking ($\overline{\text{SBLNK}}$) ; a " high " TTL level of this signal coincides with the presence of digital video information. No video output is presented when this signal has a " low " TTL level.

v).- Vertical sync. (VSYNC) ; a positive transition of the vertical sync pulse denotes the start of each video frame when the sequential mode is operated in the camera.

These video synchronization signals are shown in the figure 3.5 and are connected to the video bus in the system . The timing diagram in the figure shows their interaction.

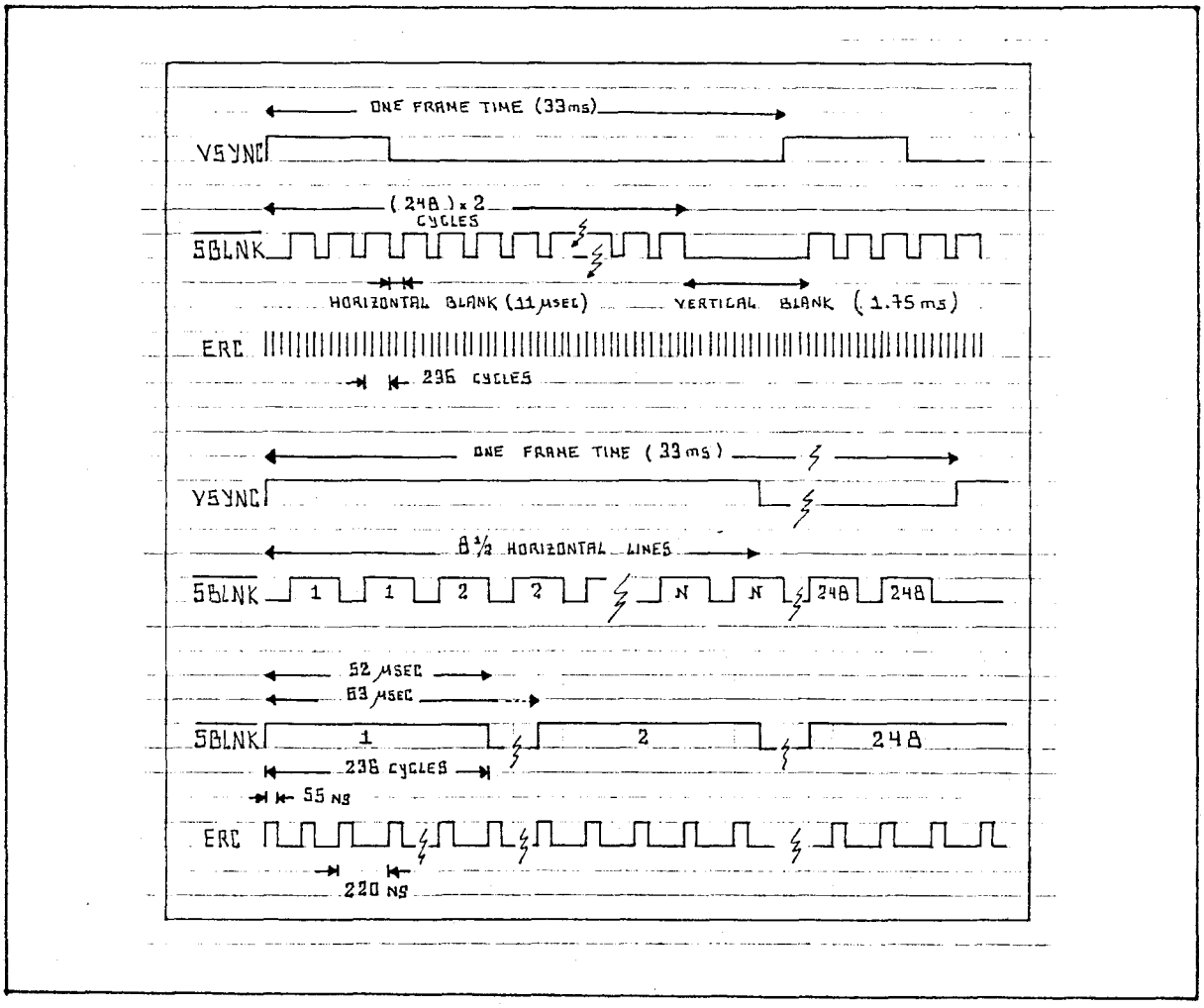


Figure 3.5 Video synchronization signals.

3.3 Video rate histogram generator (VRHG)

Section 2.4 has described the usefulness of the histogram of grey-levels for the ALS performance. The generator module of such a histogram is described below.

The VRHG has been implemented on a 6 x 12 inch Intel SBC board and with different digital logic technologies in order to optimize its functions (standard TTL, low power Schottky , HMOS-II high performance MOS technologie adn CMOS). This module generates the histogram of grey-levels of 58 528 pixels (spatial resolution used with the VRHG) in one T.V frame time (33 ms.), with dedicated hardware based on an addressable RAM technique that matches the speed of the video signals coming from the T.V. camera.

This dedicated hardware is implemented around the video and DMA buses described in section 3.1 . Its organization is shown in the figure 3.6 and consists of the following components:

i).- Histogram buffer.

ii).- Function selector.

iii).- Sequencer.

iv).- Eraser unit.

v).- Incrementer and data bus controller.

vi).- Speed converter.

vii).- Timing and control generator.

Three principal functions are accomplished in a normal operation of the VRHG in the following order and with the following commands:

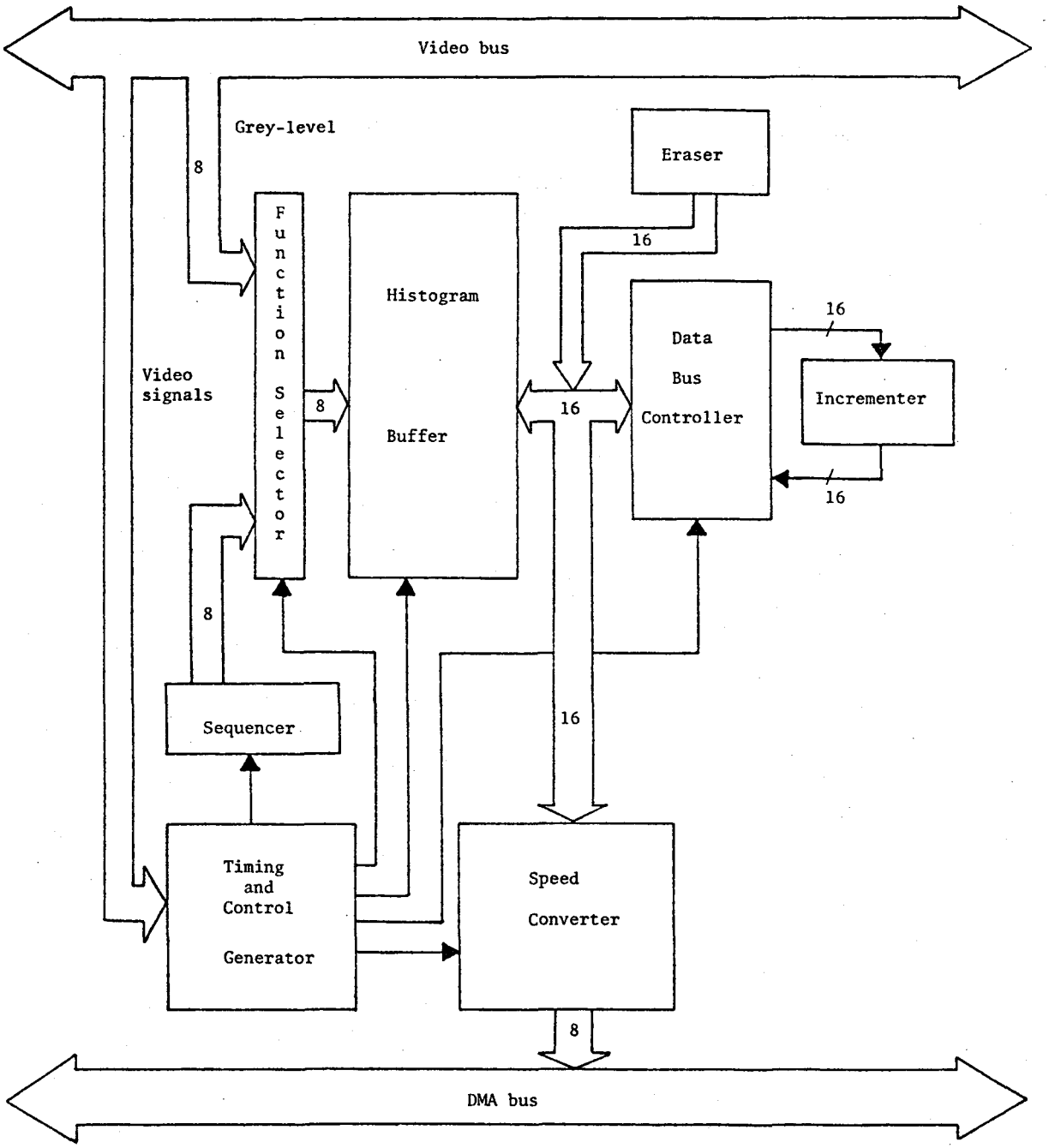


Figure 3.6 Video rate histogram generator .

CLEAR .- Erases the histogram buffer and enables the system to generate a histogram.

HISTOGRAM .- Accumulates the frequency of occurrence of each grey-level for each pixel in the image.

READ .- Dumps the acquired data.

Another two functions are available which do not have to be performed each time a histogram is to be generated:

SELECT MEMORY PAGE.- Four different histograms can be stored; this function selects the desired one.

SELECT DATA TRANSFER SPEED .- The speed of the data transfer into the 86/12A microcomputer can be selected.

The VRHG is synchronized with the solid state camera video signals, and depending on the function to be achieved, the FUNCTION SELECTOR component sets the appropriate interconnections in the circuit. Thus, a 1024 x 16 high speed RAM is erased via the ERASER and SEQUENCER components and the system is ready for data acquisition. Once a HISTOGRAM command is executed, a histogram of grey-levels is generated.

Each grey level is coded as an 8-bit word and the frequency of occurrence of each one is accumulated in one of 256/16-bit counters corresponding to the 256 different grey-levels. This function is achieved by the INCREMENTER and DATA BUS CONTROLLER components.

Once the data have been acquired, they are ready to be dumped with a READ command.

3.3.1 Histogram Buffer

The generated histogram is stored in this module. Four sets of 256/16 - bit counters are implemented with a 1024 x 16 high speed RAM for this purpose (Intel 2148-3). Three modes of operation are performed in this memory array: clear , increment and read modes .

Figure 3.7 shows the configuration of this module and its three modes of interconnection. In the clear and read modes, the address bus (lines A_0 to A_7) of the histogram buffer is connected to the output of the sequencer module. In the increment mode the address bus is connected to the grey scale lines (digitized video signals) in the video bus. With 50 ns. access time , the histogram buffer matches the high speed video signals to generate a histogram by addressing the memory array with the 8-bit grey scale coded lines. Only one mode at a time is allowed to operate. The selection of the sequence of operation modes is achieved by the function selector component and under the microcomputer control.

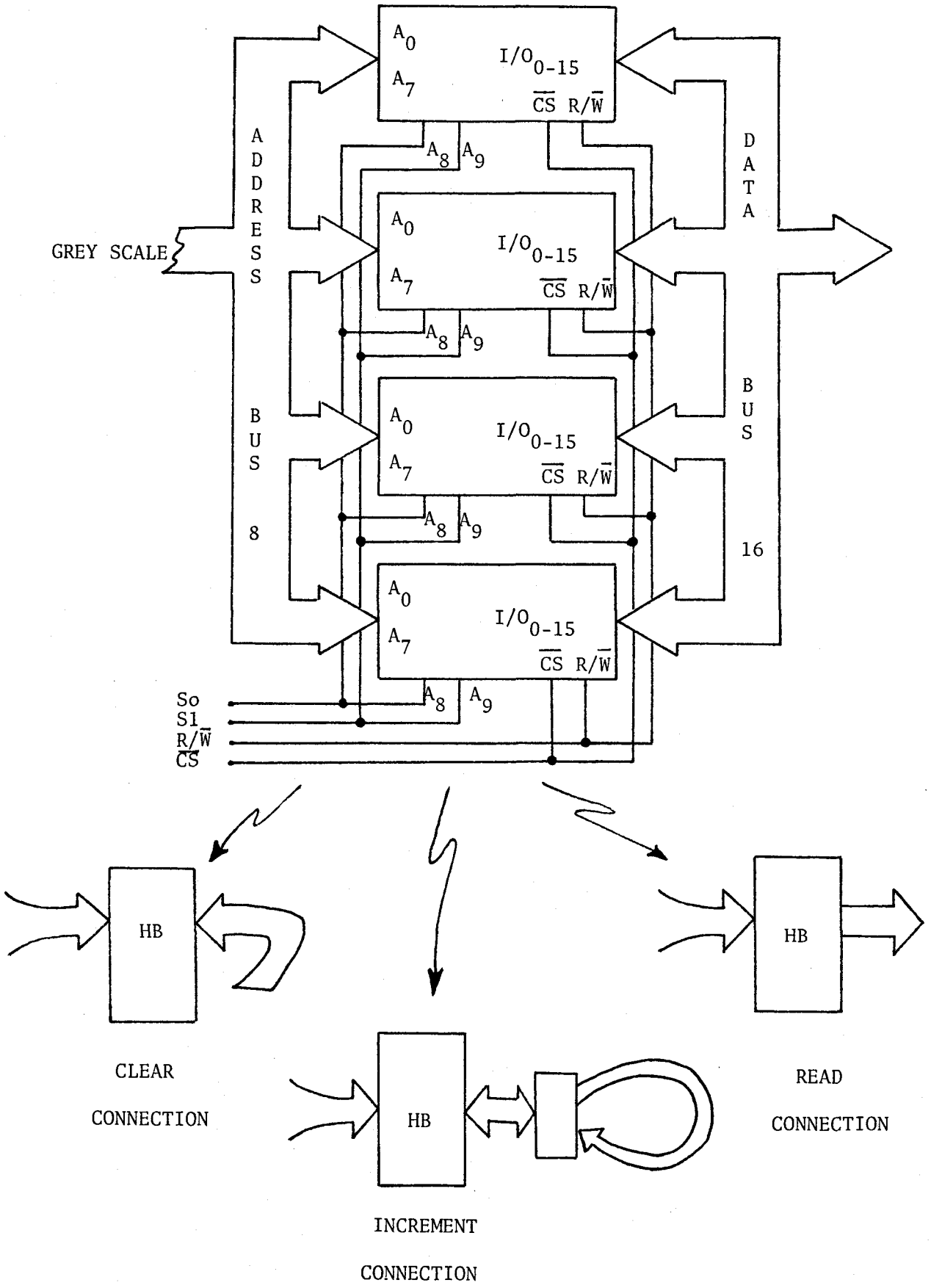
Signals R/W and CS in figure 3.7 are read/write and chip select lines supplied by the timing and control generator module.

Signals S0 ans S1 select one of the four sets of 256/16 - bit counters (histogram page), to give the histogram buffer capability of storing four histograms. Such a capability permits the ALS to remember the sensitivity step of the camera and actual distribution of light when an adaptive process is achieved.

3.3.2 Function selector

The function selector component is implemented by using

Figure 3.7 Histogram buffer component .



eight 2 to 1 multiplexers configured as shown in figure 3.8 .

The basic three functions explained before are selected here by choosing one of two different sets of eight lines:

- grey scale lines
- sequencer lines

and the appropriate read/write and chip select signals. A detailed explanation of this operation as well as the timing of the required signals for each function is shown in section 3.3.7 .

The grey scale lines are latched to compensate delays introduced by input interface circuits (schmitt trigger and drivers circuits), which isolate the CCU signals from the VRHG .

The select function signal (SF) is obtained from the sequencer module. A low level in this line connects the sequencer lines with the address bus of the histogram buffer, a high level connects the grey scale lines.

3.3.3 Sequencer

The configuration of the sequencer is shown in the figure 3.9. This module produces a sequence of 256 binary words for addressing the histogram buffer when a CLEAR or READ function is achieved. It is implemented with an 8-bit binary counter and two JK flip-flops. In the figure , when a CLEAR or READ command is executed , FF-1 and the binary counter are reset, thus the SF signal (Q of FF-1) is set to a low value to connect the eight output lines of the counter with

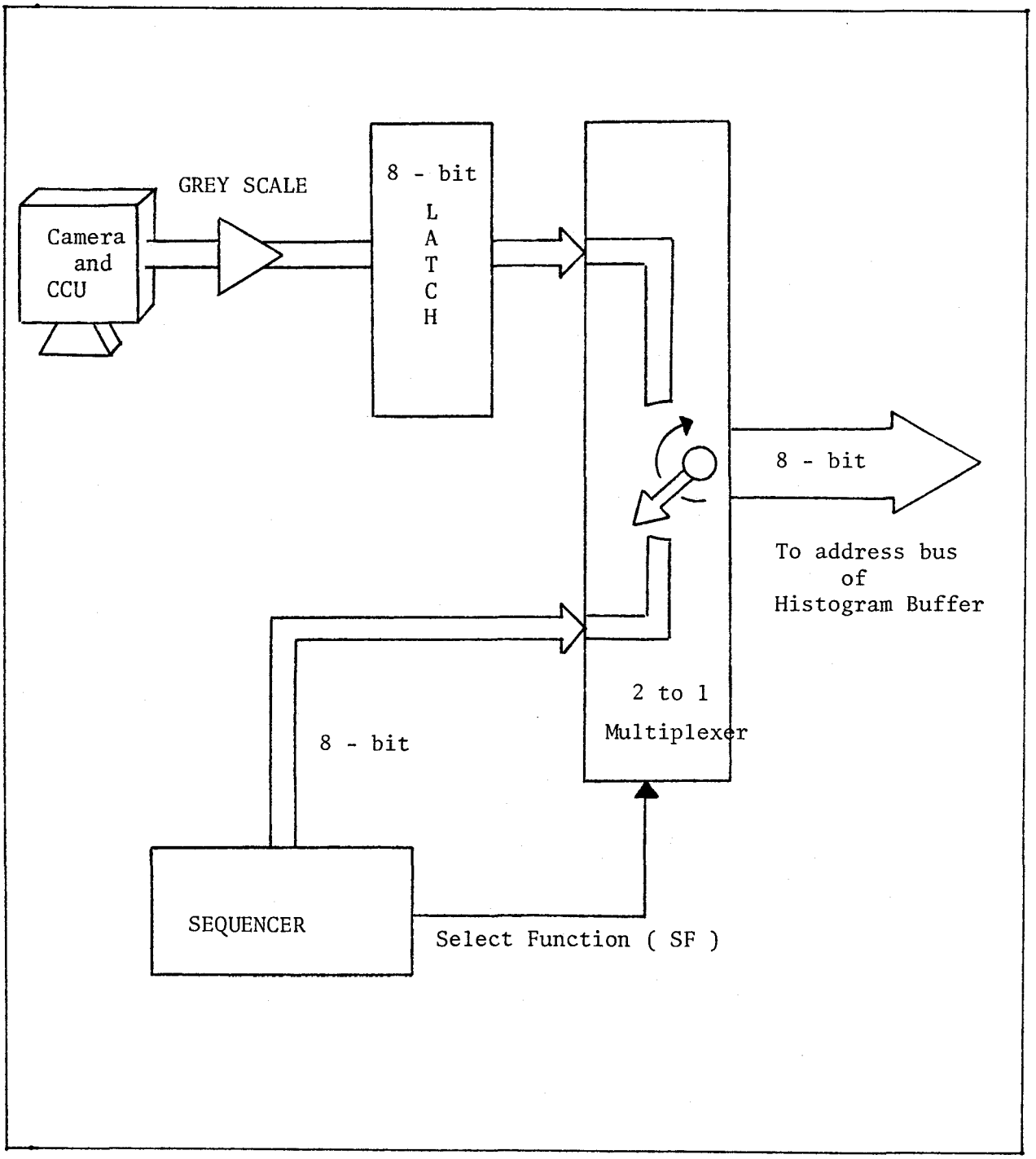


Figure 3.8 Function selector component .

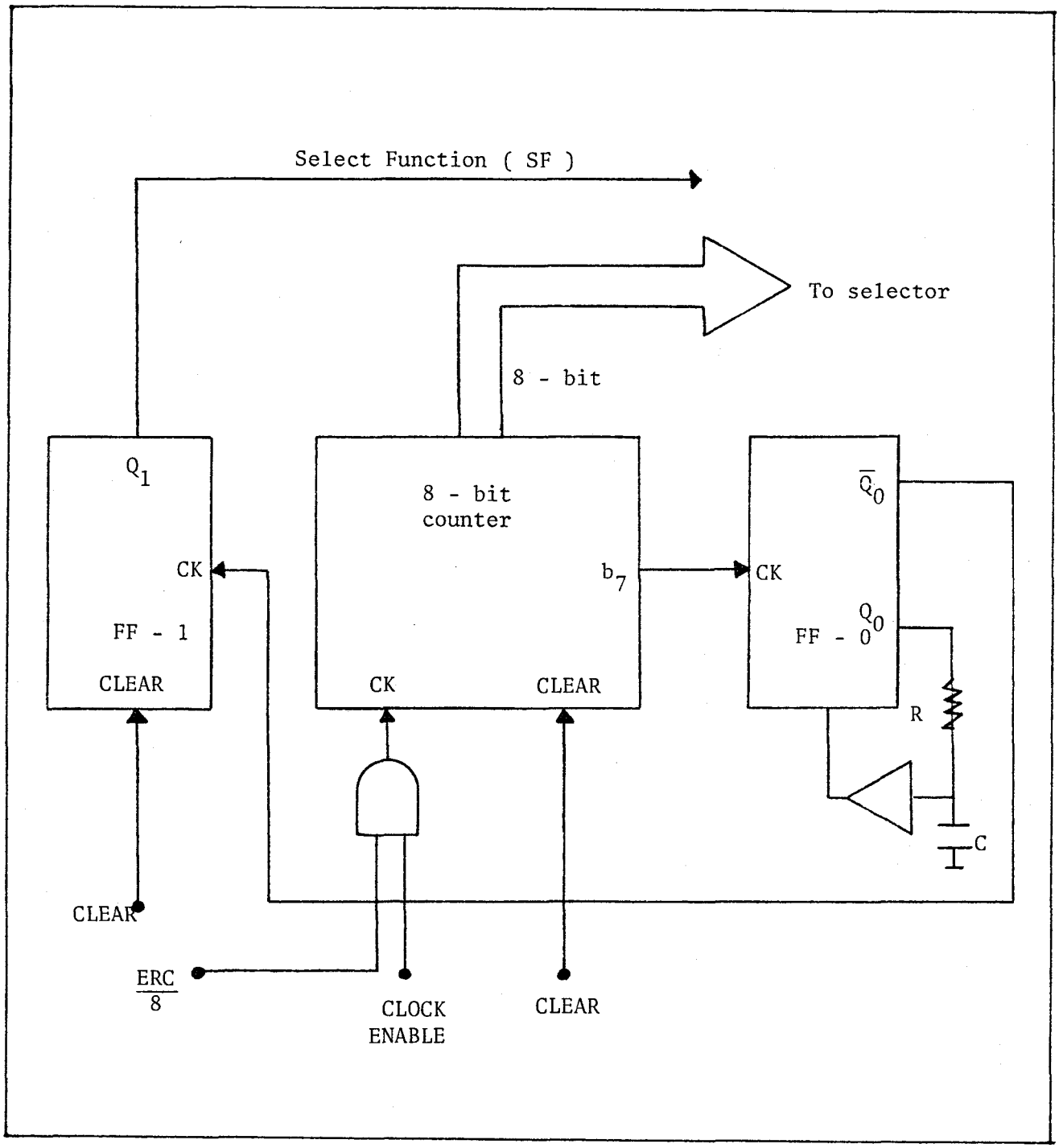


Figure 3.9 Sequencer module .

the address bus in the histogram buffer. A sequence from 00H to FFH is presented to this bus to clear or extract the 256/16-bit words of the currently selected page in the memory. When the sequence finishes, the flip-flop " 0 " in the figure (FF-0) produces a negative going pulse to clock FF-1 and set SF to a high value.

This action connects the grey scale lines to the address bus in the buffer. The process can be repeated as many times as desired and under microcomputer control, although a histogram command is expected after one sequence finishes. The " CK " and " CLOCK ENABLE " signals are supplied by the timing and control generator module.

3.3.4 Eraser unit

The function of this module is to provide low TTL values in the data bus of the histogram buffer in order to erase the currently selected page when requested.

This function is achieved by implementing an array of 16 pull down resistors and connecting it to the histogram buffer data bus. Appropriate read/write and chip select signals are given by the timing and control generator module when a CLEAR command is executed.

3.3.5 Incrementer

The implementation of this component is shown in figure 3.10, and is by way of a data bus controller, a 16-bit latch and a 16-bit adder. An increment function by one is achieved on a 16-bit length word at very high speed , and each time it is requested. The increment function and the data traffic control on the same electrical lines of the data bus is accomplished within 220 ns.

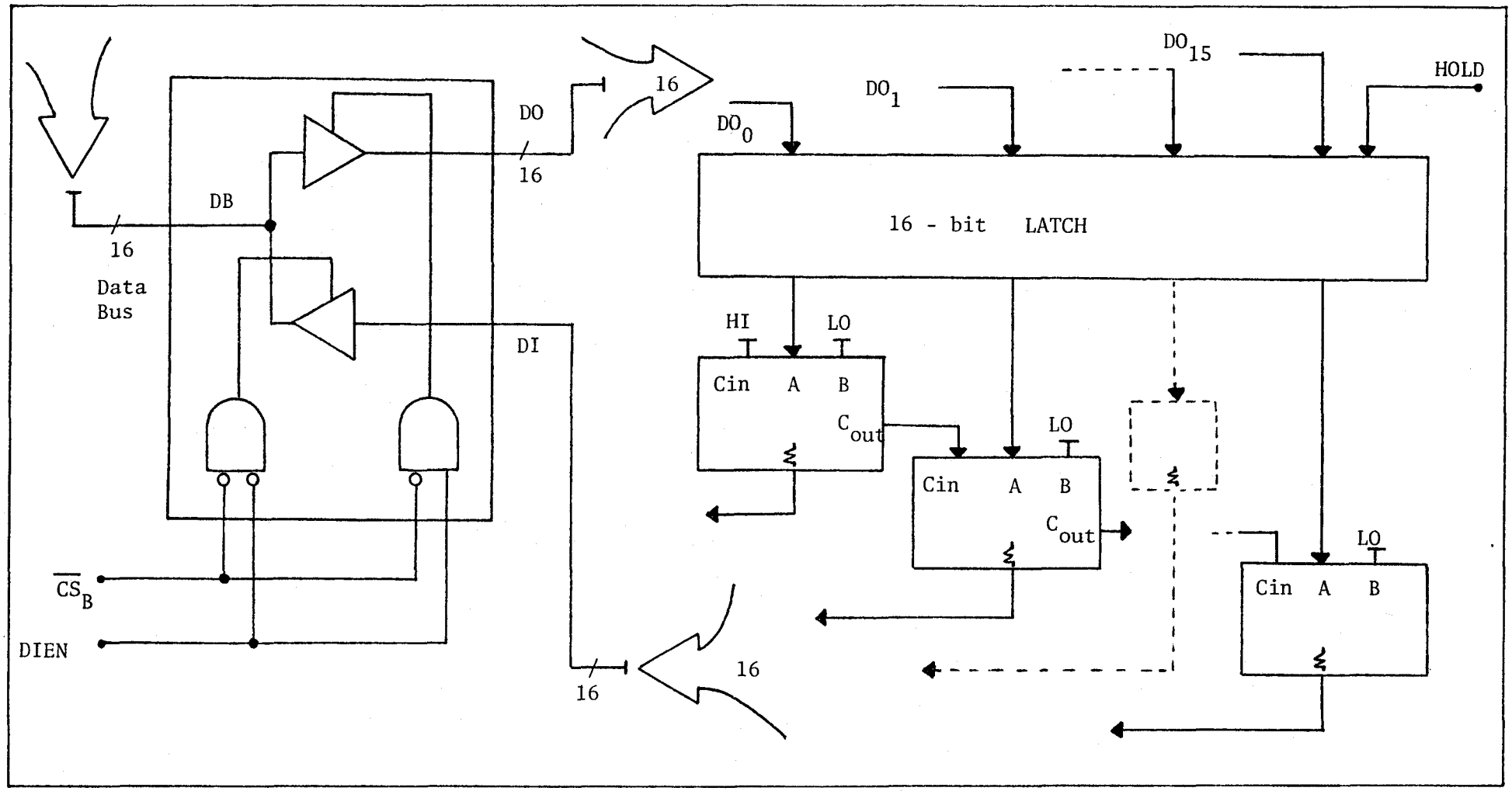


Figure 3.10 Incrementer component .

The operation is done by controlling the $\overline{CS}$, R/$\overline{W}$, $\overline{DIEN}$ and $\overline{HOLD}$ signals shown in the figure 3.10 whose timing is shown in figure 3.11 . In this timing, ERC is the element rate clock mentioned in section 3.2.3 , $\overline{CS}$ H.B. is the chip select pulse for the histogram buffer . $\overline{CS}$ D.C. is the chip select pulse for the data bus controller, R/$\overline{W}$ is the read/write pulse for the memory array, and HOLD and DIEN pulses control the actual traffic of the 16-bit word.

3.3.6 Speed converter

After a histogram has been accumulated in the histogram buffer and a READ command is exhibited, the histogram data are transferred to the microcomputer through the DMA board. Different data transfer speeds can be selected to achieve this operation. The speed converter component permits this feature as well as accomplishing a 16- to two 8-bit word conversion. The configuration of the speed converter is shown in figure 3.12, where eight 2 to 1 multiplexer realizes such a conversion, and a set of dividers for the ERC signal permits the data transfer speed selection.

A tri-state stage is suited at the output to allow the VRHG to share the DMA bus with other modules in the vision system. The transfer request signal is generated in this module to indicate to the DMA circuit the exact amount of transferred data, as explained in section 3.5 . Five different data transmission speeds can be selected by way of switches in the VRHG board:

28 microseconds/byte

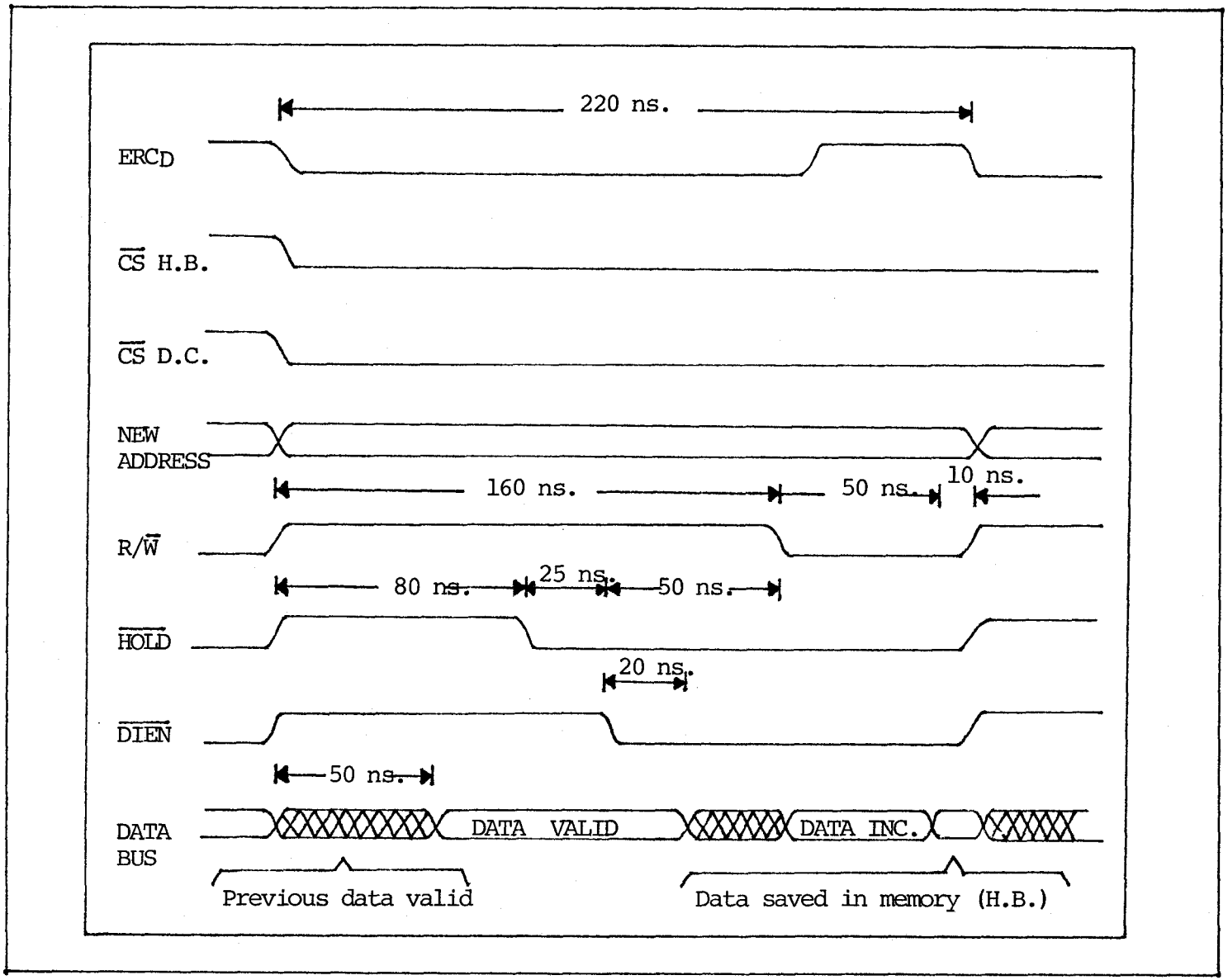


Figure 3.11 Increment function timing .

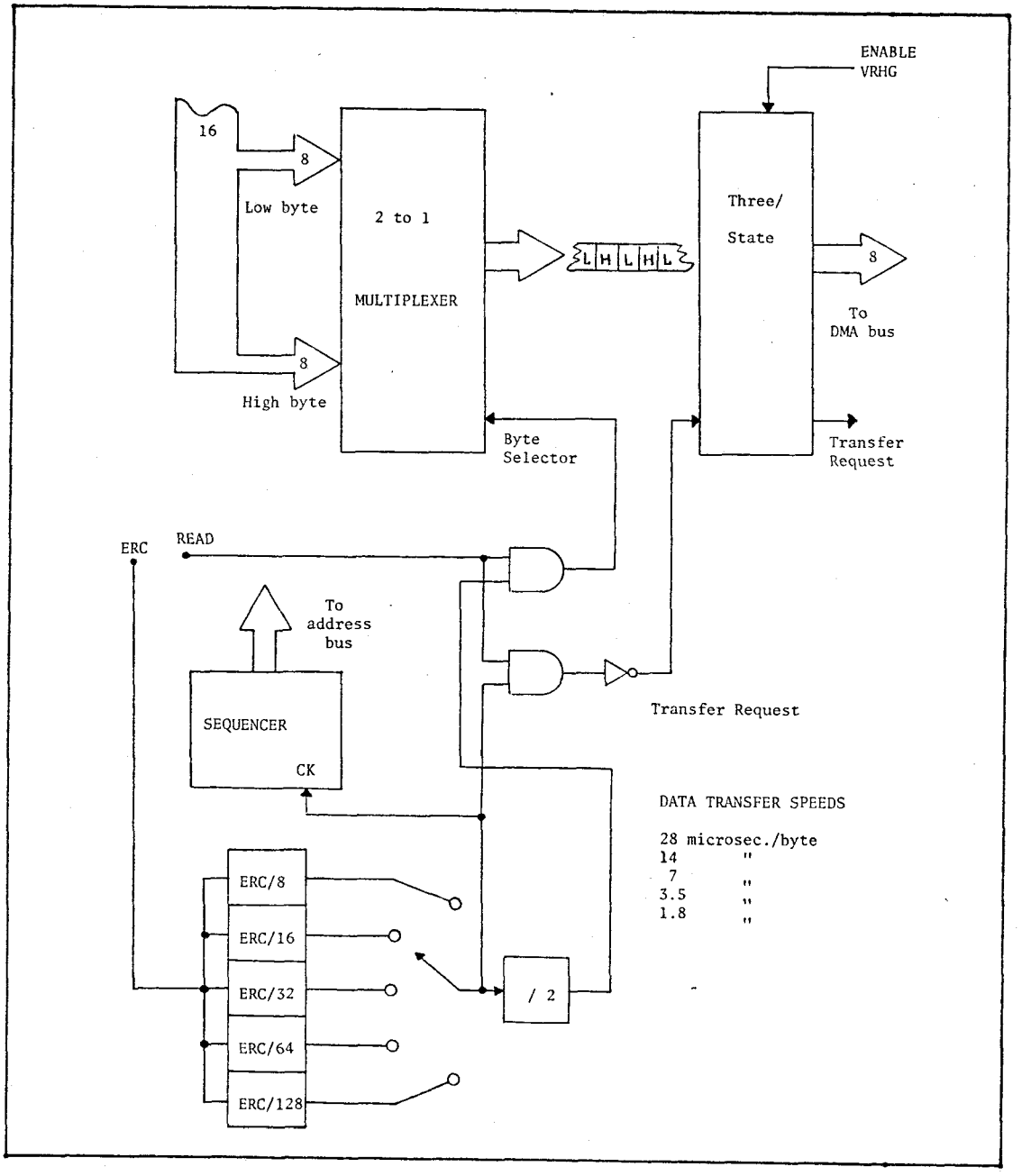


Figure 3.12 Speed converter module .

14 microseconds / byte

7 microseconds / byte

3.5 microseconds/ byte

1.8 microseconds/byte

3.3.7 Timing and control generator

This component provides the VRHG with timing and control pulses for its operation. Figure 3.13 shows the control generator interaction with other VRHG modules. This component performs programmed sequences of control pulses for each function in the VRHG. Figure 3.14 shows the timing for the sequences of pulses. Each function is activated by a negative going pulse in either the CLEAR , READ, or FRAME REQUEST lines of figure 3.13 which are exhibited under microcomputer control.

Referring to figure 3.13, the VRHG function sequences are accomplished as below:

CLEAR function .- The VRHG is initially erased by way of this function and each time a histogram is desired. A negative going pulse is presented in the CLEAR memory line to produce a negative pulse in the negative output of IC/1 ($\overline{Q}$). This pulse is applied in one input of G/2 while the other input is in a high level, because no negative going pulse has been applied in the READ line yet. Thus $\overline{Q}$ of IC/2 is in a high state. IC/1, IC/2 and IC/3 are connected as one-shot circuits for this purpose. The output of G/2 is applied to the clock input of IC/3 to produce a negative going pulse in its $\overline{Q}$ and a positive going pulse in its Q . These pulses clear the sequencer module comprised by S1, S2, S3 and G/1 , which performs as explained in section 3.3.3 .

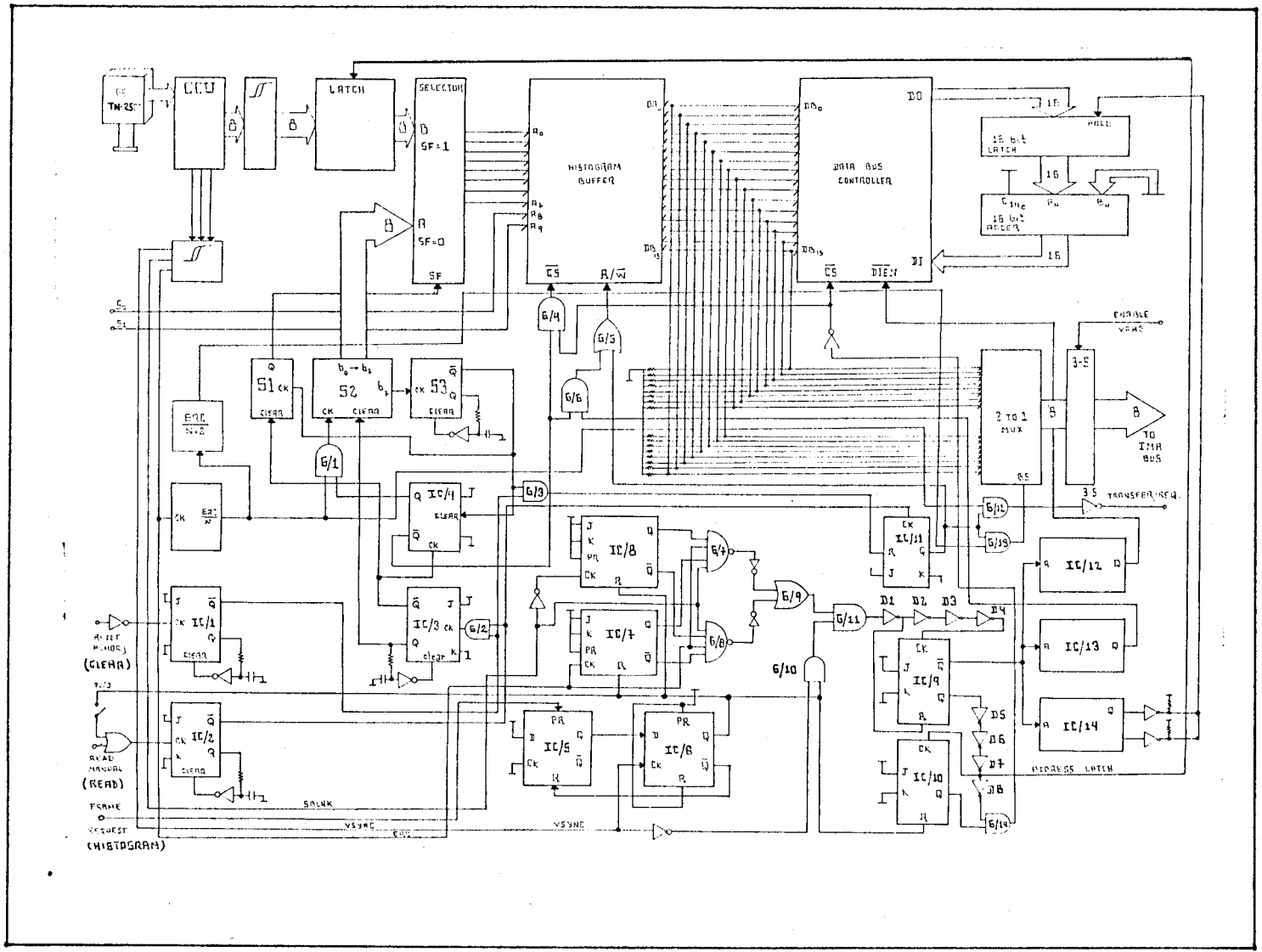


Figure 3.13 Timing and control generator/ VRHG interaction.

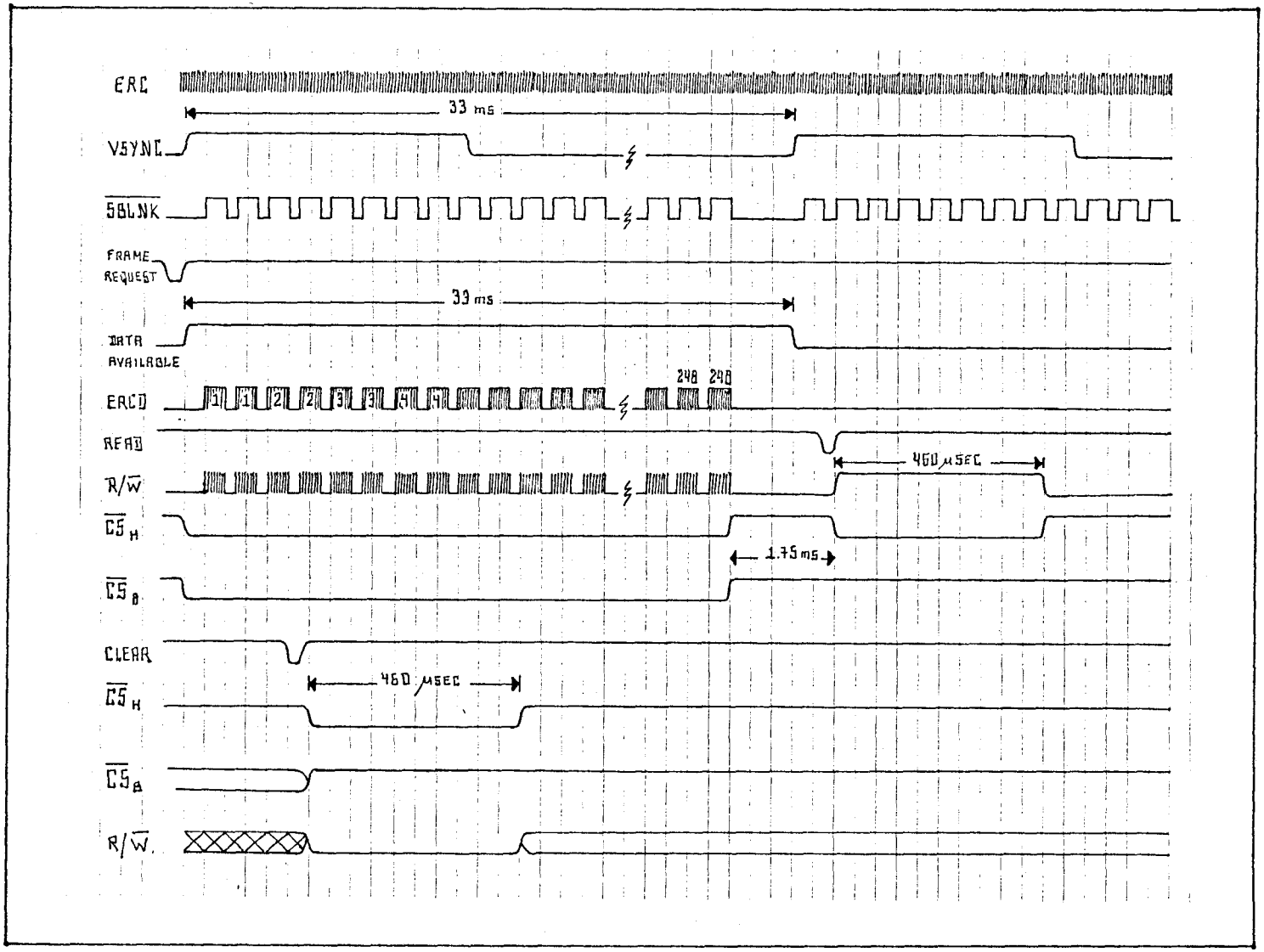


Figure 3.14 VRHG / sequence of operations .

$\overline{Q}$ of IC/3 is also applied to the clock input of IC/4 producing a high level in its Q which is presented in one input of G/1. In the other input of G/1 , clock pulses with frequency previously selected as explained in section 3.3.6, are always present. After the sequencer has been erased, a train of pulses is applied in the clock input of S2 and through G/1 for a period of time equal to 256 of these clock pulses.

After this period of time is completed, $\overline{Q}$ of S3 resets IC/4 to disable the output pulses in G/1 . At the same time that the sequencer is activated, the histogram buffer is enabled to write data by way of G/4, G/5 and G/6 and all the lines of the data bus are pulled down by the eraser unit. During this operation, all other connections to the data bus are in tri-state.

HISTOGRAM function .- A negative going pulse in the frame request line activates this function. An erased histogram buffer, and grey-scale-to-address bus lines connection are assumed before this operation is done.

When this function is activated, Q of IC/6 goes high for 33 ms. (window frame) and is synchronized with the VSYNC. This pulse activates the logic implemented with IC/7,IC/8, G/7, G/8 and G/9 to produce the master increment clock pulse (ERCD) exactly synchronized with VSYNC by way of G/10 and G/11 . This master increment clock pulse is generated as shown in figure 3.15 and produces the timing for the increment function shown in figure 3.11 .

This timing is generated with IC/9 , IC/10, IC/12 , IC/13, IC/14, G/14 and delay stages D1 through D8 of figure 3.13 . The increment function timing is repeated as many times as the master increment clock

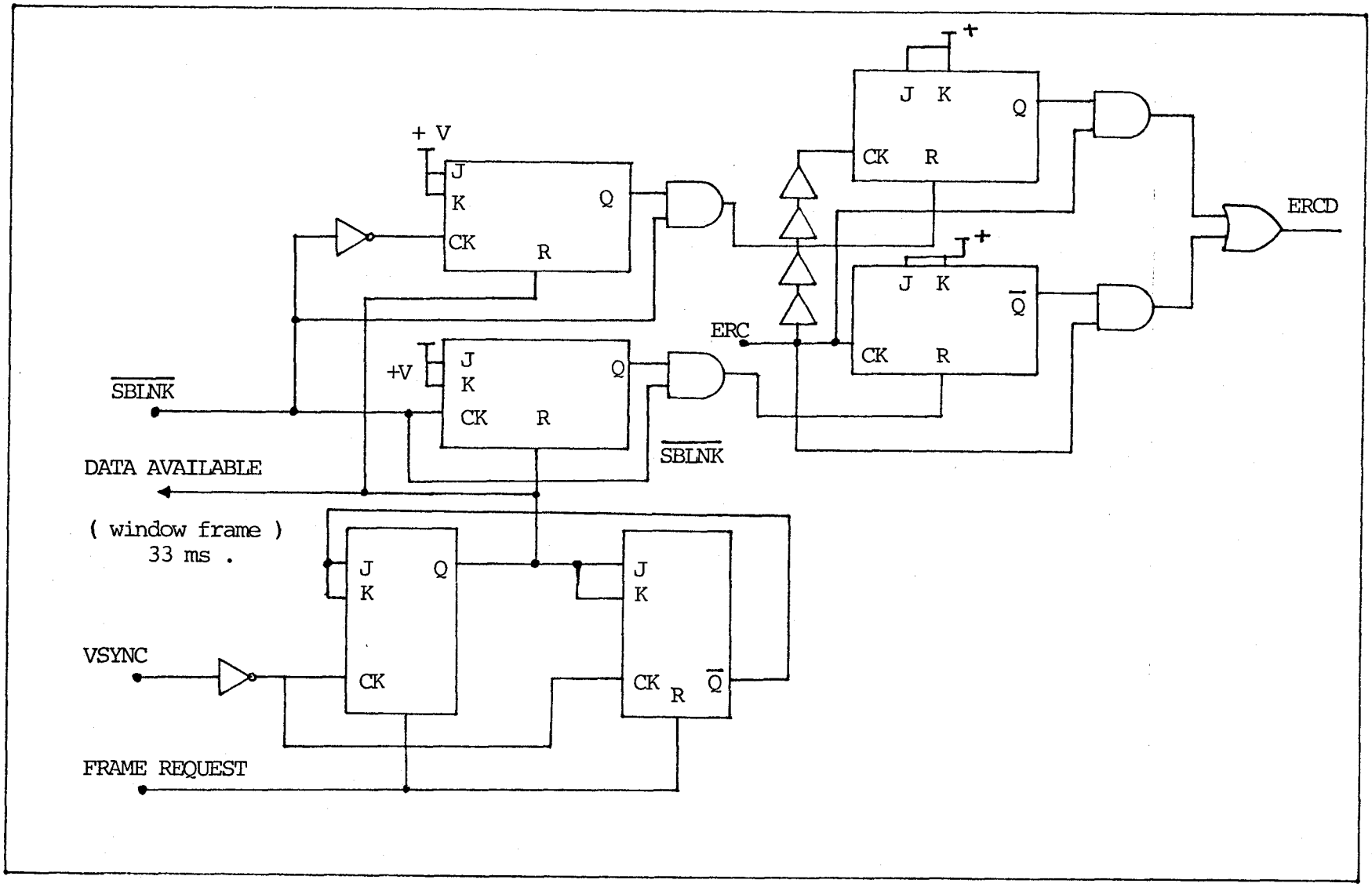


Figure 3.15 Master increment clock pulse generation .

is present, and happens to be the exact number of pixels for one picture in the 244 sequential mode frame presentation as explained in section 3.2.2 and shown in figure 3.4 .

READ function .- To perform a read function, the same sequence of events as in a clear function occurs in the sequencer component, but data is read instead of written on memory. The operation is initialized with a negative going pulse generated in the read line of figure 3.13 . A negative going pulse is produced at $\overline{Q}$ of IC/2, and applied to G/2 and IC/11 to activate the sequencer and enable the histogram buffer. IC/11 activates the speed converter, which generates the transfer request for the DMA operation and accomplishes the data transfer. An enable VRHG line is accessed by the microcomputer to select the VRHG for DMA operation within the vision system. Figure 3.16 shows the I/O lines for the VRHG.

3.4 iSBC 86/12A microcomputer

The host processor for the vision system is an Intel SBC 86/12A single board computer. It includes the 16-bit CPU, 32K RAM, 8K EPROM, an RS232 serial communications interface, 3 programmable I/O ports and a programmable interrupt controller and timer. Expansion modules have been added to constitute a total of 64K RAM and 32K EPROM.

The 8087 numerical processor has been attached to the microcomputer for floating point arithmetic operation. The iSBC 86/12A enhanced with these modules becomes the Intel iAPX 86/20 microcomputer.

The ALS uses this computer as its host processor. The memory locations for the different data types acquired (histogram of grey-

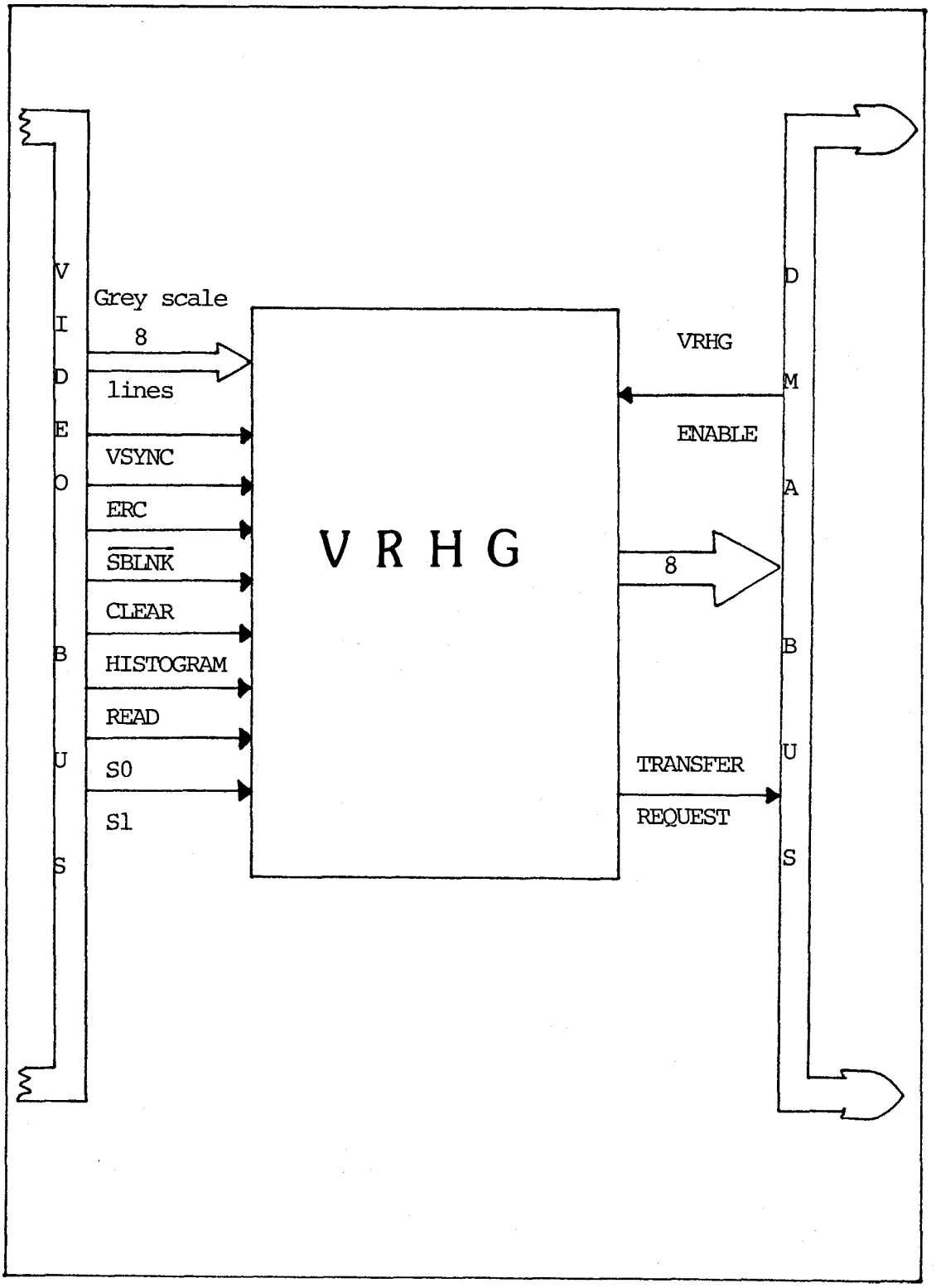


Figure 3.16 I/O lines of the VRHG .

levels, binary image) are distributed as shown in figure 3.17 .

3.5 DMA controller

Direct memory access (DMA) of 8-(or 16-) bit data between a peripheral device and the system memory is provided by the Intel SBC 501 DMA controller board in the vision system. Use was made of this controller by the three data acquisiton modules as shown in figure 3.1 . Up to 2^{16} bytes can be transferred independently of the CPU once it initiates the operation.

The iSBC 501 includes 5 I/O ports through which the CPU can communicate with external devices. The CPU initiates the transfer by programming a series of registers on the DMA board. A base address (chosen as F0H) is selected for the DMA board by way of selectable switches, and output to these registers is relative to this base.

The number of bytes to be transferred is first fed into the 16-bit length register (Base+C and Base+D). The 8-bit control register is set as shown:

bit 0 ----- write from device to memory

bit 1 ----- 8-bit transfer

bit 2 ----- DMA busy

bit 3 ----- inhibit transfer

bit 4 ----- enable interrupts

bit 5 ----- override CPU access of multibus

bit 6 ----- not used

bit 7 ----- not used

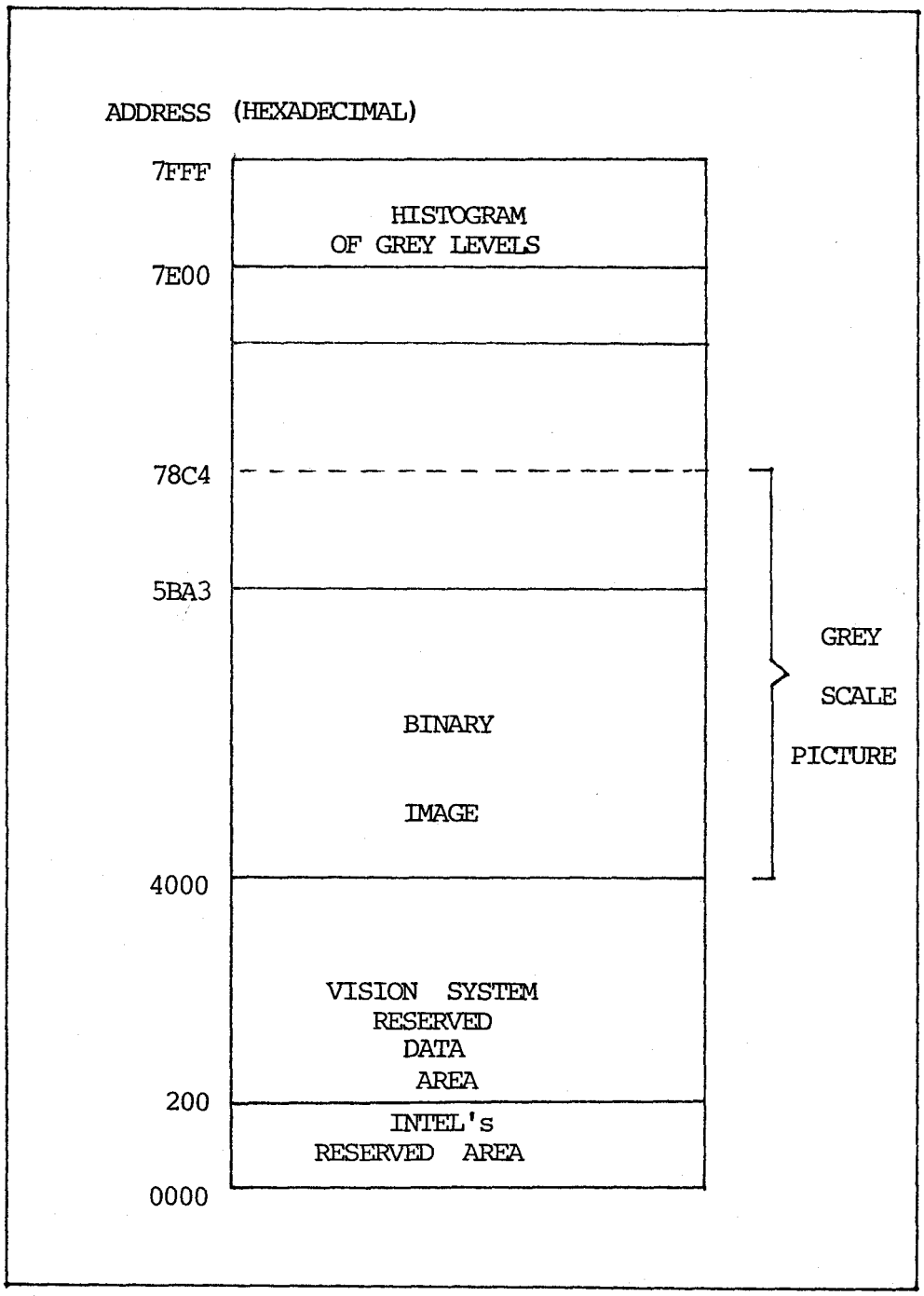


Figure 3.17 On-board RAM memory map .

The address register (Base+E and Base+F) is set to 0DE00H and through the appropriate jumpers and switches, this address is mapped into the on-board RAM beginning at location 7E00H as shown in the memory map of figure 3.17. Detailed software to program the DMA board is shown in the acquisition routines module in chapter IV.

3.6 Sensitivity control unit

The sensitivity control unit permits different sensitivities in the visual transducer by way of altering the integration time in the read out of the CID as explained in section 3.2.1 . From empirical experiments, four steps of sensitivity were chosen as follow:

STEP NUMBER	INTEGRATION TIME
1 (normal operation)	one frame
2	two frames
3	four frames
4	eight frames

The CID sensor is exposed to the light source for 1, 2, 4 or 8 frame times depending of the light conditions. The less the scene is illuminated, the more time the CID sensor needs to acquire the same image as if a well illuminated scene is acquired in normal operation (one frame time of light exposure).

As mentioned in section 3.2.1 , the alteration of the exposure time is achieved by disabling the scanning pulses for the sensor by the desired time in each case, and then enabling them again. It is necessary to capture the image contained in the first frame after the exposure

time alteration, otherwise the transduced signal will be only that of one frame time exposure.

This operation is accomplished by generating different VSYNC signals for different steps of sensitivity as shown in the timing of figure 3.18 . The implementation of this sensitivity control unit is shown is figure 3.19 and is built on the same board of the VRHG.

A digital to analog conversion of the generated histogram is also achieved with a 12-bit D/A converter circuit built in the same board in order to provide the system with real time visual feedback of the ALS operation.

Print-out of the histogram and statistical calculations are obtained by way of a graphics printer as shown in chapter V. Measurements on the system and adaptive range are shown in the same chapter.

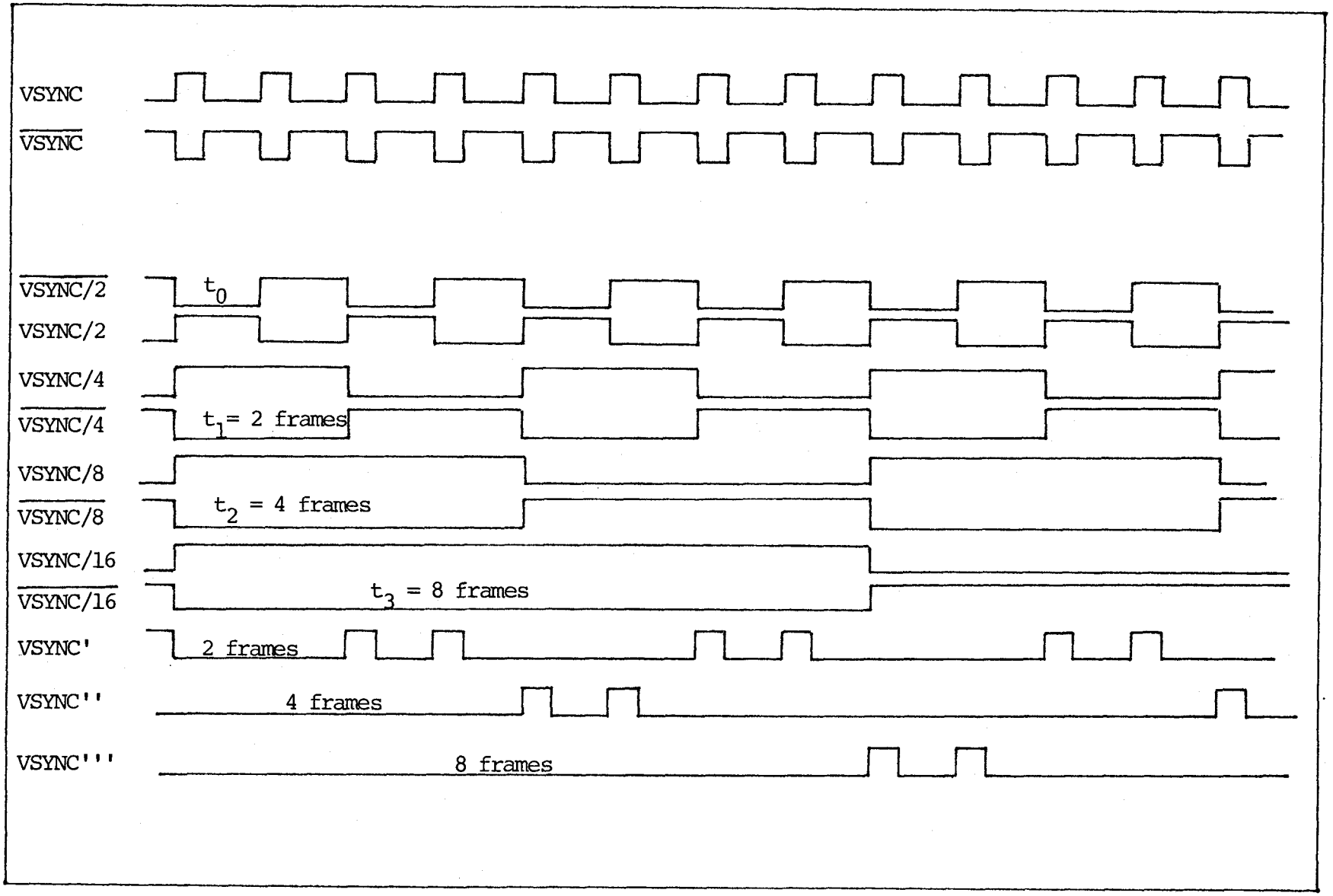


Figure 3.18 Steps sensitivity timing in the SCU.

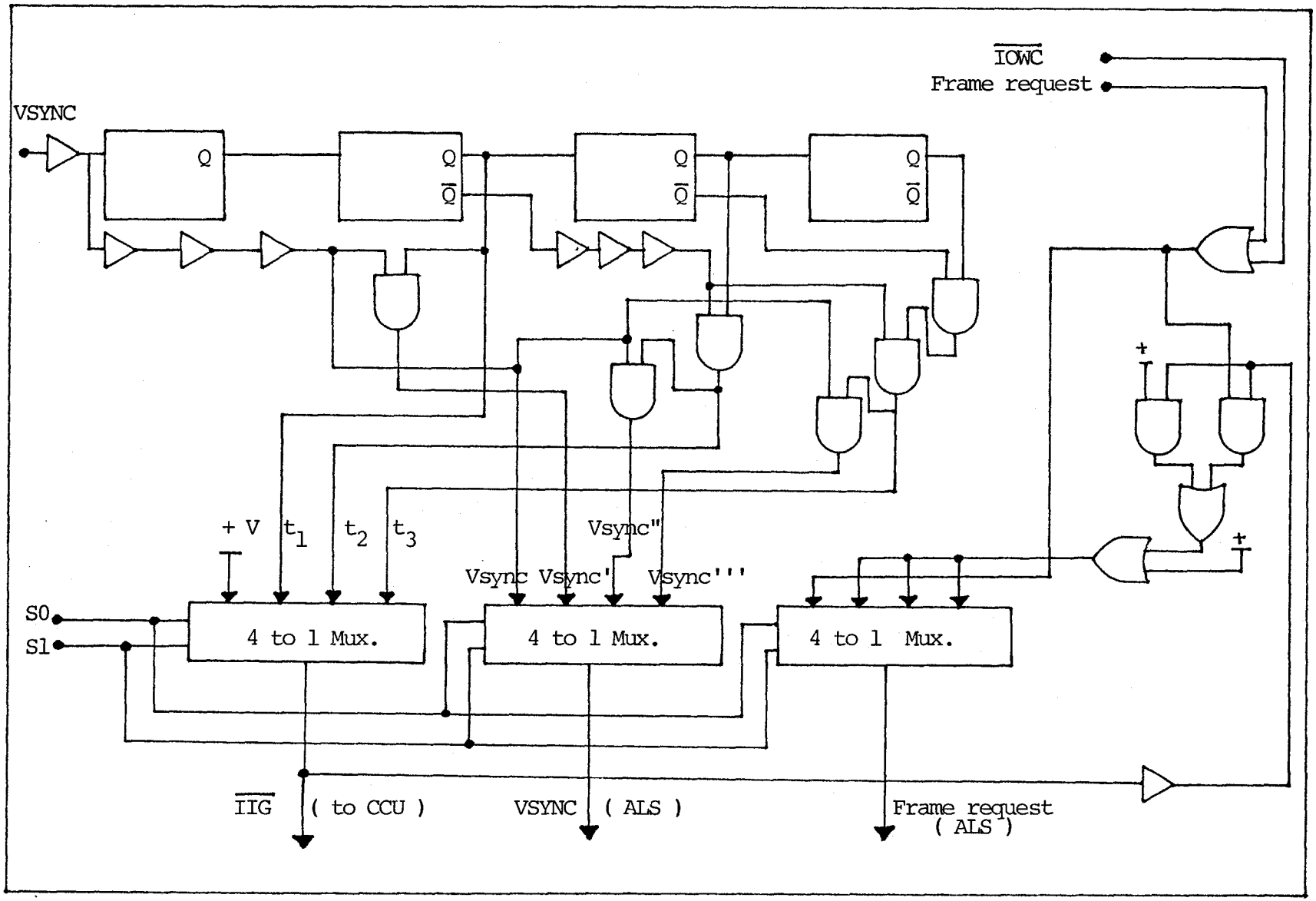


Figure 3.19 Sensitivity control unit (SCU)

CHAPTER IV

SYSTEM SOFTWARE

Programming for the ALS was written in Intel's PL/M-86 language [4-1] , and developed by using Intel's MCS-86 software development utilities [4-2] and a MDS-230 microprocessor development system.

4.1 Software structure

The general structure of the ALS software is shown in figure 4.1 . A top-down design was incorporated. The editing , compiled and list files are stored on floppy disks and the hexadecimal 8086 code resides on Intel 2732A EPROM's. The vision system software [3-1] was modified in some routines to be used exclusively by the ALS. Other routines are used either by the ALS or the vision system.

For a better understanding of the software design, a sequence of the overall process achieved by the ALS is explained in the next section and with reference to figure 3.1 .

4.2 ALS adaptive process algorithm

Each sequence starts when the user (vision system or the robot) requests an image from the scene. First the VRHG is prepared for data acquisition and the sensitivity control unit set to the sensitivity

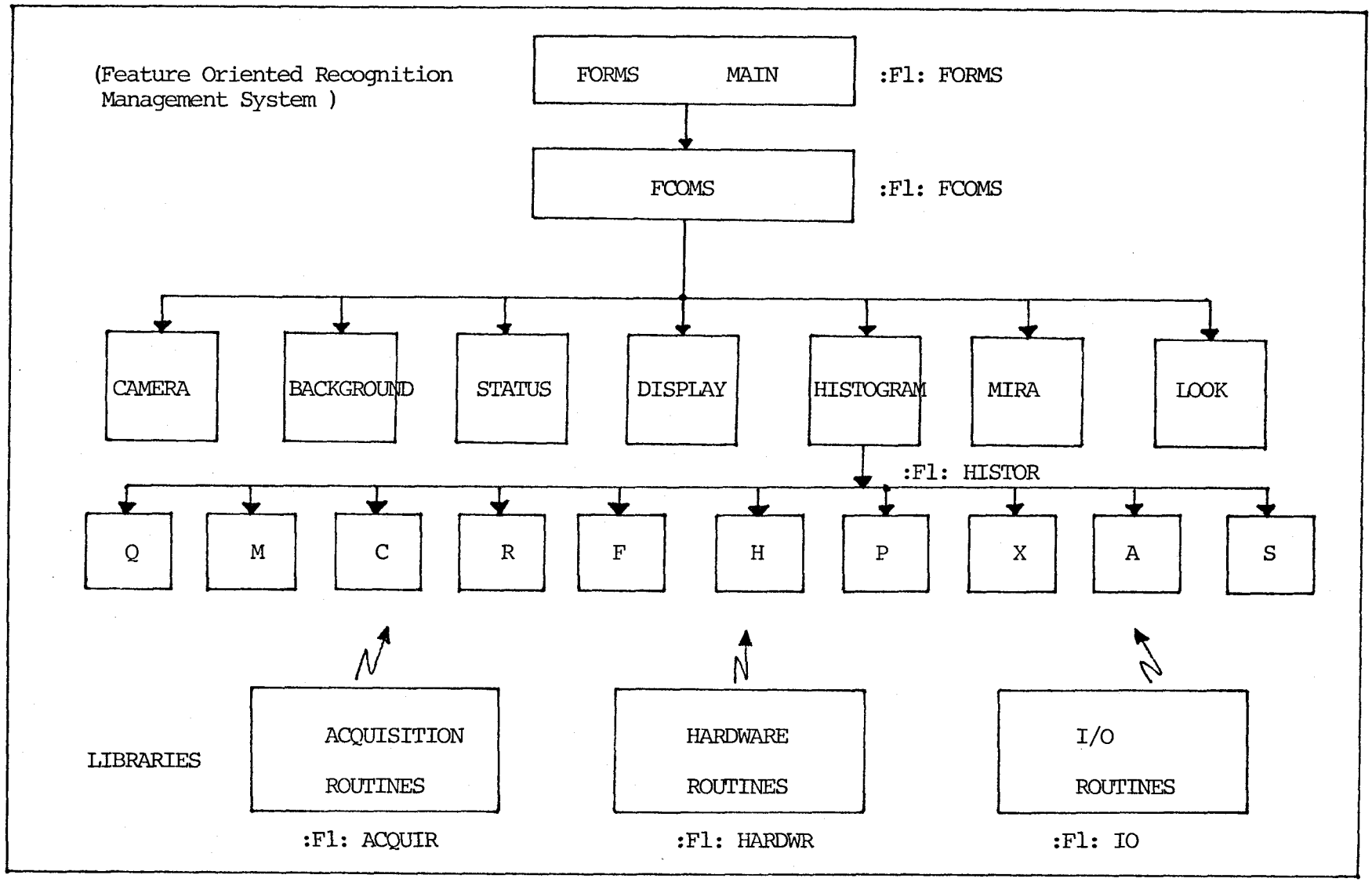


Figure 4.1 Software structure

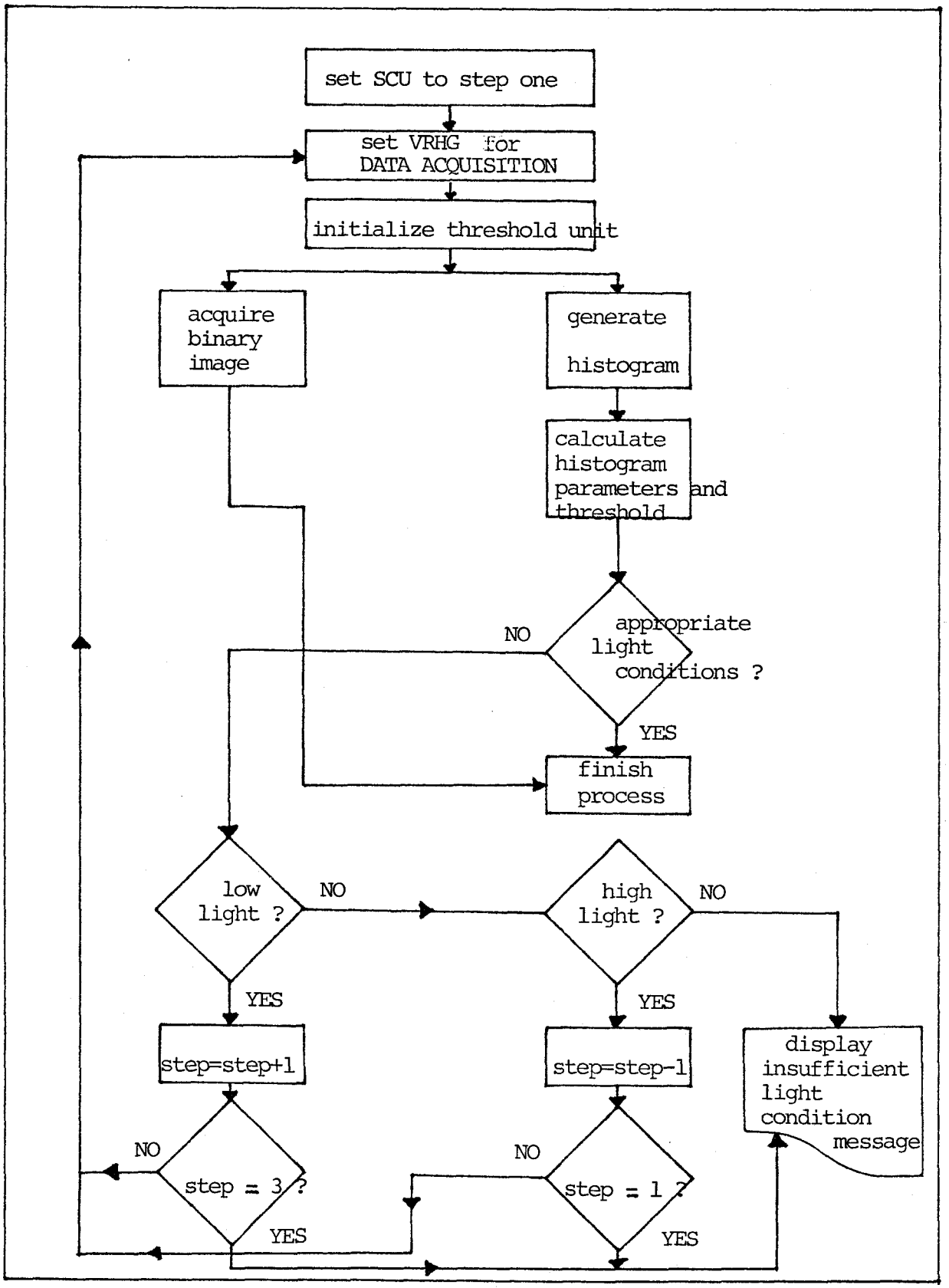


Figure 4.2 (ALS) adaptive process algorithm .

step one (normal operation of one frame time of light exposure). A frame request is issued by the microcomputer, and in parallel, a binary image is deposited in the on-board RAM (of the iAPX 86/20 microcomputer). Simultaneously,a histogram of grey-levels representing the distribution of light of the scene from which the silhouette was created, is generated and stored in the VRHG.

The histogram data is transferred to the microcomputer on-board RAM . Calculations on the histogram are performed and a decision is taken to either finish the process, or issue another frame request with a different sensitivity in the visual transducer.

The above process is repeated until a good image is acquired, or an insufficient light conditions message is issued to the user. This algorithm is shown in figure 4.2 , and a sequence of the histograms used for an adaptive process using all sensitivity steps, as well as calculated parameters is shown in chapter V .

4.3 Threshold adjustment

A typical bimodal histogram and calculated data are shown in figure 4.3 . Different parameters are defined in the histogram assuming a bimodal distribution of light is always obtained :

b_w ---------- black$width

Max_b ---------- maximum$black$value$grey$level

MIN ---------- minimum$grey$level

MAX ---------- maximum$grey$level

b_l ---------- black$lower$threshold

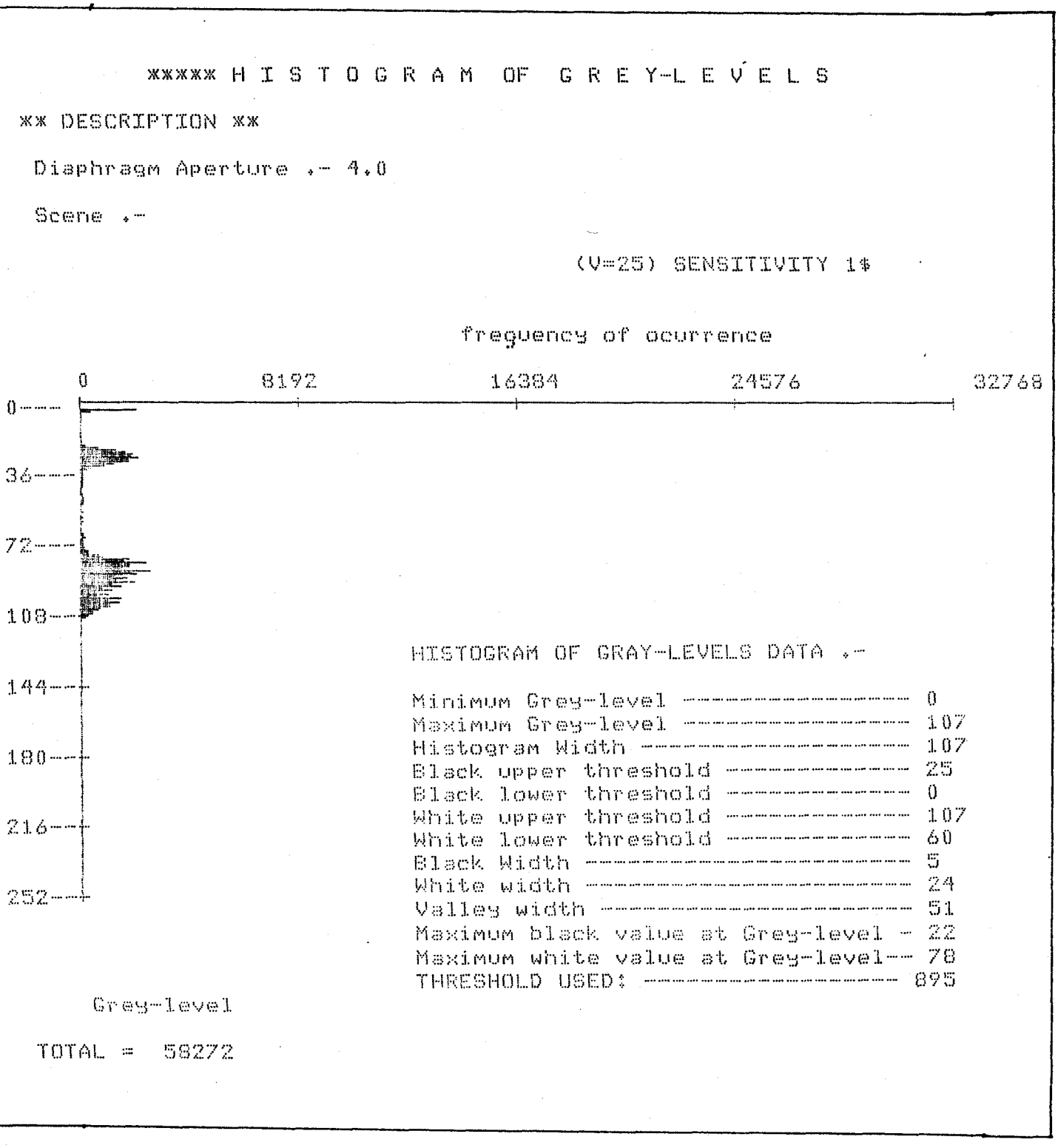


Figure 4.3 Bimodal histogram and calculated parameters.

b_u ---------- black$upper$threshold

W_h ---------- histogram$width

W_u ---------- white$upper$threshold

W_l ---------- white$lower$threshold

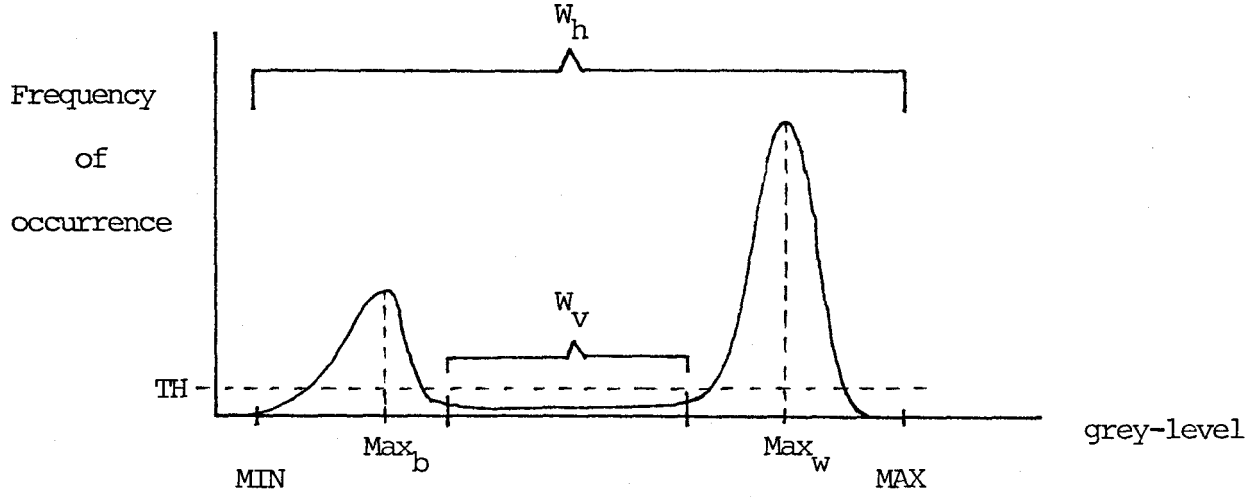
W_w ---------- white$width

W_v ---------- valley$width

Max_w ---------- maximum $white$value$grey$level .

These parameters establish the criteria for the automatic threshold adjustment and the sensitivity step. Although a bimodal distribution is shown every time, some variations on the shape of the histogram can occur due to instabilities in the ambient light.

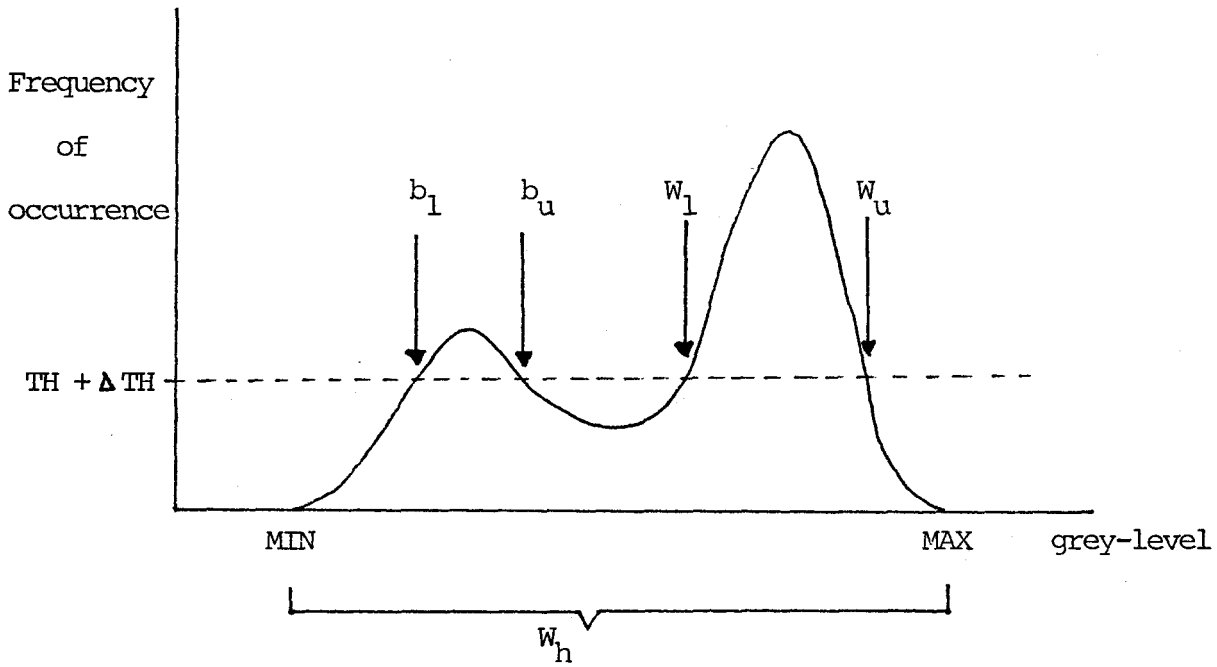
Suppose a histogram like the one below is being analyzed:



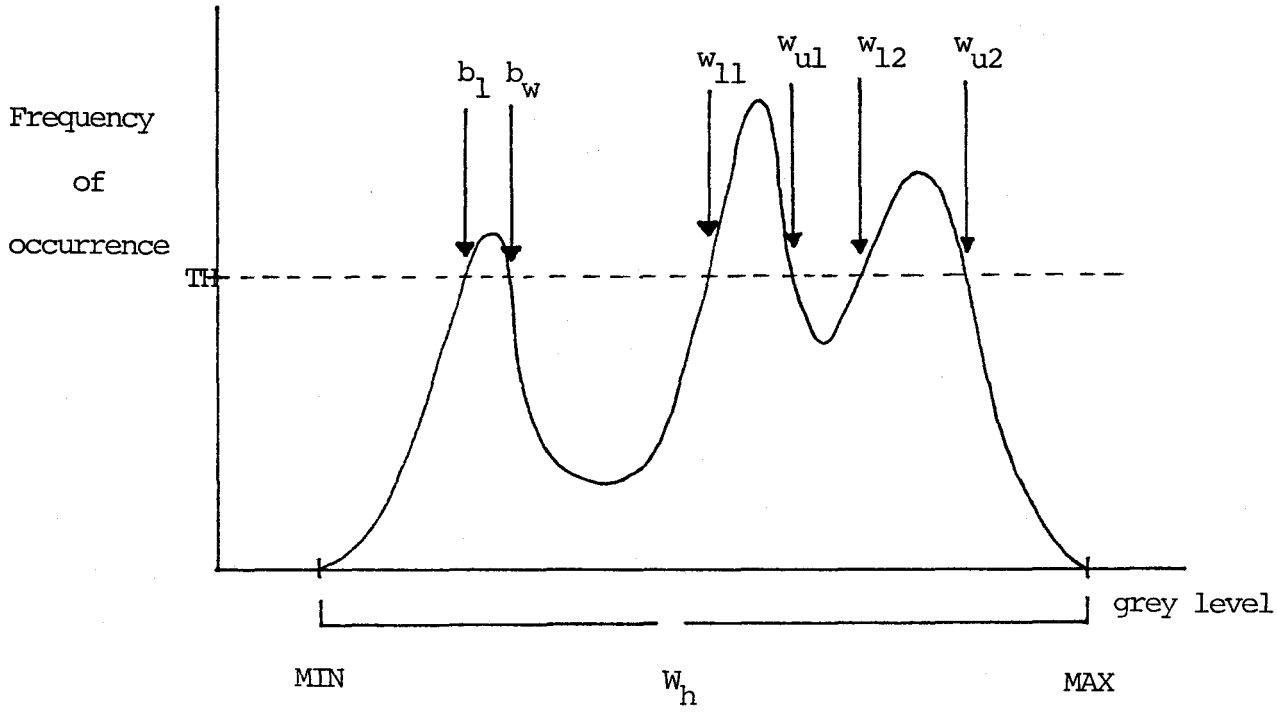
The two peaks are large enough to select the threshold midway between them, and the histogram and valley widths are big enough for easy setting of parameters boundaries.

A software threshold value (TH) is incorporated to improve the reliability of the calculations for setting the parameters boundaries.

For a histogram such as the one below, the threshold value has to be increased in order to obtain all of the parameters.



Sometimes the ambient light contains different sources of light, and the histogram shows two peaks in the white region as in the figure below . The system then can be confused, because parameters not defined appear in the histogram. These problems are avoided by using a modal technique, a dynamic software threshold (TH) and the appropriate algorithms. Assuming MIN and MAX are known values, calculated by scanning from the left to the right for MIN , and from right to the left for MAX until non-zero values are encountered,



the histogram width (W_h) is then calculated by :

$$W_h = MAX - MIN .$$

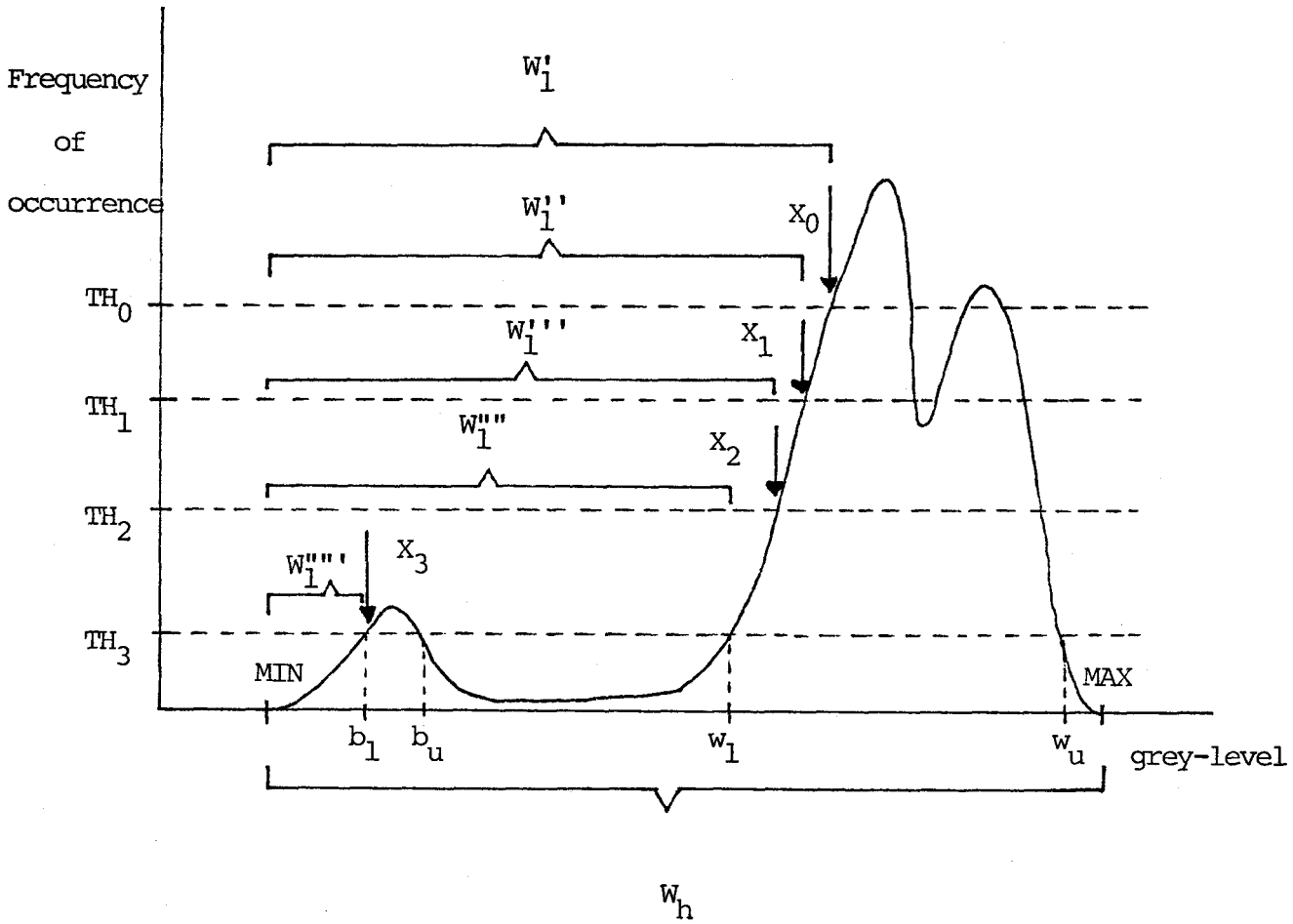
If W_h is larger than a pre-established value (chosen as 03CH by experimental results), the histogram shows a good bimodal shape, and the following algorithm is applied:

black lower:

set a threshold value TH_0 ,

beginning in MIN , scan from left to right until a value $X_n \geq TH_o$ is found, then make : lower black $(b_l) = X_n$.

Here a dummy parameter W_l is incorporated to adjust the TH software threshold to a value that guarantees that all parameters will be represented. The figure below shows this threshold adjustment for a typical histogram.



By iterative calculations of W_1 and W_h ,the threshold value used to find the remaining parameters is obtained as follows:

-calculate X_n ,

- do while ($W_1 \geq W_h/3$);

$TH_{n+1} = TH_n/2$;

$n = n + 1$;

calculate X_n ;

end .

Having obtained this value TH , the next algorithms are applied:

upper black:

- start scanning from b_1 to the right until a value

$X_2 < TH_n$ is found, then upper$black ($b_u$) = X_2 .

lower white:

- start scanning from b_u to the right until a value

$X_3 > TH_n$ is found, then lower$white = X_3

upper white:

- start scanning from MAX to the left until a value

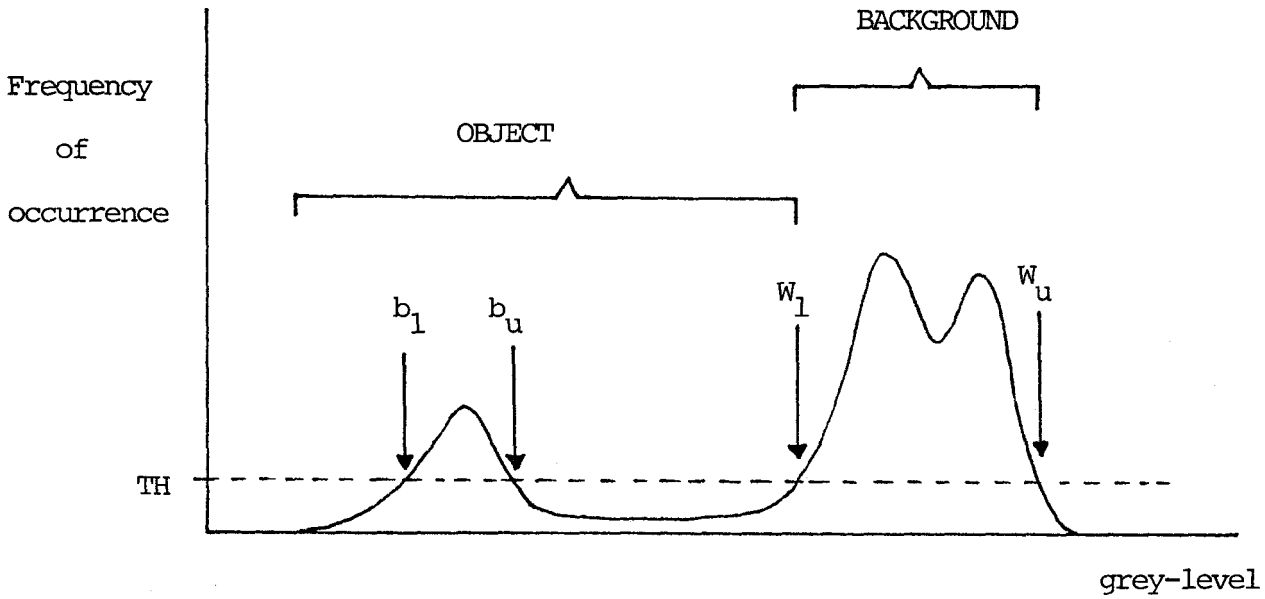
$X_4 > TH_n$ is found, then upper$white = X_4 .

These are the basic values needed to calculate all other parameters with algorithms shown in the software list presented in subsequent pages. Other routines are presented as well for printing out the histogram, smoothing of the histogram shape, sensitivity control of the visual transducer, display of parameters, implementation of the ALS adaptive algorithm shown in figure 4.3 , data acquisition processes and control of peripheral and digital devices within the system.

The ALS software calculates two threshold values from the parameters by establishing relations between the maximum and minimum values for the black and white regions as well as the valley width parameter defined before.

These calculated threshold values represent a band of grey-levels as shown in the figure below and are set into the threshold unit

described in section 3.1 to extract an object from its background.



For convenience in the vision system software, the lower threshold in the dual threshold unit becomes W_l and the upper threshold W_u.

4.4 Histogram package

This package was implemented for non-automatic operations performed with the ALS, as well as for giving the vision system analysis capability of its T.V. camera performance and lighting schema.

With this package, histograms of grey levels of the scene can be generated, erased, printed, smoothed , and analog displayed for different sensitivities of the visual transducer.

The implemented commands that accomplish these operations are:

M --- change sensitivity step in the visual transducer and page memory in the histogram buffer.

C --- clear function in the VRHG.

F --- histogram function in the VRHG.

R --- read function in the VRHG.

P --- print out the histogram with scale factor option.

X --- smooth the histogram .

H --- performs in one step the commands C, F, R, X and P .

A --- display analog histogram (an oscilloscope is needed).

S --- calculate all parameters.

Q --- return to FORMS (vision system control).

The sequence of operations for this commands is as follows:

M : Memory page

- read status register ,
- display current memory page and sensitivity step ,
- change current status by the desired one .

C : Clear

- output a clear pulse for the clear function ,
- wait until the histogram buffer has been erased .

F : Frame request

- input desired sensitivity step ,
- set sensitivity control unit for desired step ,
- output a frame request pulse for the histogram function .

R : Read

- select the VRHG for data acquisition ,
- programm the DMA board for histogram acquisition ,
- output a read pulse for the read function ,
- wait until data transfer has been completed.

P : Print histogram

- input comments to be printed out ,
- choose scale factor for the histogram graphic ,
- print histogram with desired scalation .

A : Analog histogram

- clear histogram ,
- generate histogram ,
- read histogram (digital/analog conversion) ,
- stop process if desired ,
- repeat the whole process .

For this operation (analog histogram) an oscilloscope is needed and two test pins are provided on the board to connect the oscilloscope probes. These pins are for the analog histogram output and synchronization pulse for oscilloscope synchronizing purposes . Labels are provided on the board to identify each one .

```
PL/M-86 COMPILER    HISTOGRAMROUTINES


ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE HISTOGRAMROUTINES
OBJECT MODULE PLACED IN :F1:HISTOR.OBJ
COMPILER INVOKED BY:  PLM86 :F1:HISTOR.SRC LARGE OPTIMIZE(3)


   1            HISTOGRAM$ROUTINES: DO;


                /* ************************************************
                   H I S T O G R A M S      R O U T I N E S

                   THIS MODULE CONTAINS THE STATISTICS CALCULA-
                   TIONS FOR THE HISTOGRAM, AS WELL AS PRINTING
                   AND AUTOMATIC THRESHOLD ROUTINES.


                       AUTHOR: MARIO PEÑA C.

                ************************************************/

                /* ------------------------------------------------
                   EXTERNAL   USAGE   DECLARATIONS
                     ------------------------------------------------*/


   2   1        PRINT$REAL: PROCEDURE(REAL$VAR,NUMBER$DEC$PLACES) EXTERNAL;
   3   2        DECLARE REAL$VAR REAL,
                        NUMBER$DEC$PLACES BYTE;
   4   2        END PRINT$REAL;

   5   1        CRT$REAL$OUT: PROCEDURE (REAL$VAR,NUMBER$DEC$PLACES) EXTERNAL;
   6   2        DECLARE REAL$VAR REAL,
                    NUMBER$DEC$PLACES BYTE;
   7   2        END CRT$REAL$OUT;

   8   1        CRTIN: PROCEDURE BYTE EXTERNAL;
   9   2        END CRTIN;

  10   1        CRT$STRING$OUT: PROCEDURE(PTR) EXTERNAL;
  11   2            DECLARE PTR POINTER;
  12   2        END CRT$STRING$OUT;

  13   1        PRINT$STRING: PROCEDURE(PTR)EXTERNAL;
  14   2        DECLARE PTR POINTER;
  15   2        END PRINT$STRING;

  16   1        PRINT$BYTE: PROCEDURE(CHAR) EXTERNAL;
  17   2            DECLARE CHAR BYTE;
  18   2        END PRINT$BYTE;

  19   1        PRINT$HEX: PROCEDURE(BITS) EXTERNAL;
  20   2            DECLARE BITS BYTE;
  21   2        END PRINT$HEX;

  22   1        CRT$BYTE$OUT: PROCEDURE (CHAR) EXTERNAL;
  23   2        DECLARE CHAR BYTE;
```

```
PL/M-86 COMPILER    HISTOGRAMROUTINES

 24  2     END CRT$BYTE$OUT;

 25  1     CRT$HEX$OUT : PROCEDURE (BIT$PATTERN) EXTERNAL ;
 26  2     DECLARE  BIT$PATTERN  BYTE;
 27  2     END CRT$HEX$OUT;

 28  1     CLEAR$HISTOGRAM$MEMORY: PROCEDURE EXTERNAL;
 29  2     END CLEAR$HISTOGRAM$MEMORY;


 30  1     FRAME$GRAB: PROCEDURE (TIME) EXTERNAL;
 31  2     DECLARE
              TIME BYTE;
 32  2     END FRAME$GRAB;

 33  1     GET$HISTOGRAM$DATA: PROCEDURE EXTERNAL;
 34  2     END GET$HISTOGRAM$DATA;

 35  1     SELECT$HISTOGRAM$PAGE: PROCEDURE ( PAGE$NUMBER ) EXTERNAL;
 36  2     DECLARE
              PAGE$NUMBER BYTE;
 37  2     END SELECT$HISTOGRAM$PAGE;

 38  1     SET$THRESHOLD : PROCEDURE (LOWER,UPPER) EXTERNAL ;
 39  2     DECLARE
              (LOWER,UPPER) BYTE ;
 40  2     END SET$THRESHOLD;

 41  1     CRT$CHAR$READY: PROCEDURE BYTE EXTERNAL;
 42  2     END CRT$CHAR$READY;

 43  1     CRT$CHECK$CONTROL: PROCEDURE BYTE EXTERNAL;
 44  2     END CRT$CHECK$CONTROL;

 45  1     READ$SYSTEM$STATUS$REGISTER: PROCEDURE BYTE EXTERNAL;
 46  2     END READ$SYSTEM$STATUS$REGISTER;

 47  1     CAMERA$TO$INTERLACED: PROCEDURE EXTERNAL;
 48  2     END CAMERA$TO$INTERLACED;

 49  1     DELAY: PROCEDURE (TIME) EXTERNAL;
 50  2     DECLARE TIME BYTE;
 51  2     END DELAY;


                /* ------------------------------------------------
                   GLOBAL            DECLARATIONS
                   ------------------------------------------------*/

 52  1        DECLARE
              CR LITERALLY '0DH',
              LF LITERALLY '0AH',
              ETX LITERALLY '03H',
              HISTOGRAM (256) WORD AT (7E00H),
              DESCRIPTION (200) BYTE,
              BOBEGA(12) BYTE,
              THRESHOLD WORD,
              SUMA WORD;
```

```
$EJECT


   /* ***************************************************
   @      P H R A S E
   @  THIS ROUTINE ACCEPT CHARACTERS FROM THE KEYBOARD @
   @  TO MAKE UP A PHRASE OF COMMENTS UP TO 200 CHARAC-@
   @  TERS , A ' $ ' MEANS THE END OF THE PHRASE      @
   ***************************************************/

53  1   PHRASE: PROCEDURE PUBLIC;

54  2   DECLARE N BYTE;

55  2   N=0;
56  2   DESCRIPTION(N)=0AH;
57  2   DO WHILE DESCRIPTION(N) <> 24H;
58  3   N=N+1;
59  3   DESCRIPTION(N)=CRTIN;
60  3   END;
61  2   DESCRIPTION(N+1)=0AH;
62  2   DESCRIPTION(N+2)=03H;
63  2   END PHRASE;

   /*************************************************************
   @      P R I N T       H I S T O G R A M       @
   @                        @
   @  THIS ROUTINE PRINTS A GRAPHIC OF THE HISTOGRAM    @
   @  IN MEMORY LOCATION (7E00H) , AS WELL AS THE
   @  SCALE , AXIS , TOTAL OF PIXELS ANALYZED ,DIAPHRAGM @
   @  APERTURE OF THE CAMERA AND COMMENTS OF THE SCENE  @
   @
   *************************************************************/


64  1   PRINT$HISTOGRAM : PROCEDURE PUBLIC ;


65  2   DECLARE
          HISTOGRAM(256) WORD AT (7E00H),
          SHISTO(256) WORD ,
          PRINTCODE BYTE,
          C(6) WORD,
          (K,L) BYTE ;
66  2   DECLARE
          SCALE$FACTOR BYTE,
          MAX$SCALE$FACTOR BYTE ,
          CMND BYTE,
          TOTAL WORD;
67  2   DECLARE
          APERTURE (7) BYTE,
          (MAX,LABEL1,LABEL2 ) WORD,
             N BYTE ;

68  2   DECLARE
          M1(*) BYTE DATA(LF,CR,'                              FREQUENCY OF OCURRENCE',LF,CR,ETX),
          M2(*) BYTE DATA(LF,CR,'ENTER SCALE FACTOR:( 0 TO 6 )',ETX),
```

```
                M3(*) BYTE DATA(LF,CR,'TYPE DESCRIPTION OF SCENE:( $ TO FINISH )',LF,CR,ETX);
 69   2     DECLARE
                M4(*) BYTE DATA(LF,CR,'          ***** H I S T O G R A M  OF  G R E Y-L E V E L S ',LF,CR,ETX);
 70   2     DECLARE
                M5(*) BYTE DATA(LF,CR,' ** DESCRIPTION ** ',LF,CR,ETX),
                M6(*) BYTE DATA(LF,CR,'      GREY-LEVEL   ',LF,CR,ETX),
                M7(*) BYTE DATA(LF,CR,'  TOTAL =  ',ETX);
 71   2     DECLARE
                M8(*) BYTE DATA(LF,CR,'  DIAPHRAGH APERTURE .- ',ETX),
                M9(*) BYTE DATA(LF,CR,'  SCENE .-  ',LF,CR,ETX);



              /*  PROCEDURE PRINTING THE HISTOGRAM */

 72   2           CALL CRT$STRING$OUT(@M3);
 73   2        CALL PHRASE;
 74   2        CALL CRT$STRING$OUT(@M8);

 75   2        DO N=0 TO 3;
 76   3        APERTURE(N)=CRTIN;
 77   3        END;
 78   2        APERTURE(4)=LF;
 79   2        APERTURE(5)=CR;
 80   2        APERTURE(6)=ETX;
 81   2        SCALE$FACTOR=0FH;
 82   2        DO WHILE SCALE$FACTOR ) 06H;
 83   3           CALL CRT$STRING$OUT(@M2);
 84   3           SCALE$FACTOR=CRTIN AND 0FH;
 85   3        END;
 86   2        MAX$SCALE$FACTOR= 7 - (SCALE$FACTOR);
 87   2        IF MAX$SCALE$FACTOR=0 THEN MAX= 0FFFFH;
                   ELSE
 89   2        MAX=( SHR(0FFFFH,MAX$SCALE$FACTOR))+1;
 90   2           LABEL1=SHR(MAX,2);

              /* BEGIN TO PRINT THE TITLES */

 91   2        CALL PRINTBYTE(1BH);
 92   2        CALL PRINTBYTE(13H);
 93   2        CALL PRINTSTRING(@M4);
 94   2        CALL PRINTSTRING(@M5);
 95   2        CALL PRINTSTRING(@M8);
 96   2        CALL PRINTSTRING(@APERTURE);
 97   2        CALL PRINTSTRING(@M9);
 98   2        CALL PRINTSTRING(@DESCRIPTION);
 99   2        CALL PRINTBYTE(LF);
100   2        CALL PRINTSTRING(@M1);
101   2        CALL PRINTBYTE(LF);
102   2        CALL PRINTBYTE(CR);
103   2        N=0H;
104   2        DO WHILE (N<5);
105   3        CALL PRINTBYTE(20H);
106   3        N=N+1;
107   3        END;
108   2        LABEL2=0H;
```

PL/M-86 COMPILER HISTOGRAMROUTINES

```
109  2          DO WHILE (LABEL2(MAX));
110  3             CALL PRINT$REAL(FLOAT(INT(LABEL2)),0H);
111  3             LABEL2=LABEL2 + LABEL1;
112  3                N=0H;
113  3                DO WHILE N(12;
114  4                   CALL PRINTBYTE(20H);
115  4                   N=N+1;
116  4                END;
117  3          END;
118  2          MAX=SHR(MAX,2);
119  2          CALL PRINT$REAL(FLOAT(INT(MAX))* 4.,0H);
120  2          MAX=SHL(MAX,2);
121  2          CALL PRINTBYTE(CR);
122  2          CALL PRINTBYTE(LF);


123  2          N=0H;
124  2          DO WHILE (N(8);
125  3             CALL PRINTBYTE(20H);
126  3             N=N+1;
127  3          END;

128  2          CALL PRINTBYTE(1BH);
129  2          CALL PRINTBYTE(25H);
130  2          CALL PRINTBYTE(30H);

131  2          LABEL2=0H;
132  2          DO WHILE (LABEL2(MAX);
133  3             N=0H;
134  3             CALL PRINTBYTE(5FH);
135  3             LABEL2=LABEL2+LABEL1;
136  3             DO WHILE N(114;
137  4                CALL PRINTBYTE(24H);
138  4                N=N+1;
139  4             END;
140  3          END;
141  2          CALL PRINTBYTE(5FH);

                   /* BEGINT TO PRINT THE VALUES OF THE HISTOGRAM */

142  2          TOTAL=0;

143  2          DO K=0 TO 255;
144  3          TOTAL=TOTAL+HISTOGRAM(K);
145  3          SHISTO(K)=SHR(HISTOGRAM(K),SCALE$FACTOR);
146  3          END;

147  2          DO L=0 TO 246 BY 6;

148  3          DO K=0 TO 5;
149  4          C(K)=0;
150  4          END;

151  3          CALL PRINTBYTE(0DH);
152  3          CALL PRINTBYTE(0AH);

153  3          N=0H;
```

```
PL/M-86 COMPILER    HISTOGRAMROUTINES

154   3         DO WHILE (N<40);
155   4            CALL PRINTBYTE(20H);
156   4            N=N+1;
157   4         END;

158   3         CALL PRINT$BYTE(5FH);

159   3         DO WHILE ( (C(0)<SHISTO(L)) OR (C(1)<SHISTO(L+1)) OR (C(2)<SHISTO(L+2))
                   OR (C(3)<SHISTO(L+3)) OR (C(4)<SHISTO(L+4)) OR (C(5)<SHISTO(L+5)) );

160   4         PRINTCODE=0;
161   4         DO K=0 TO 5;
162   5         IF C(K)<SHISTO(K+L) THEN DO;
164   6                                  C(K)=C(K)+1;
165   6                                  PRINTCODE=PRINTCODE OR SHL(01H,K);
166   6                                 END;
167   5         END;
168   4         CALL PRINTBYTE(PRINTCODE+20H);
169   4         END;
170   3         END;



171   2         N=0H;
172   2         DO WHILE N<42;
173   3            CALL PRINTBYTE (1BH);
174   3            CALL PRINTBYTE (1EH);
175   3            N=N+1;
176   3         END;

177   2         N=0H;
178   2         DO N=0 TO 256 BY 36 ;
179   3            CALL PRINTBYTE(CR);
180   3            CALL PRINTBYTE (1BH); /* TO MONOSPACED MODE */
181   3            CALL PRINTBYTE (13H);
182   3            CALL PRINT$REAL(FLOAT(INT(N)),0H);
183   3            CALL PRINTBYTE(2DH);
184   3            CALL PRINTBYTE(2DH);
185   3            CALL PRINTBYTE(2DH);
186   3            CALL PRINTBYTE (1BH); /* TO GRAPHICS MODE */
187   3            CALL PRINTBYTE (25H);
188   3            CALL PRINTBYTE (30H);
189   3            DO K= 0 TO 5;
190   4               CALL PRINTBYTE(00H);
191   4               CALL PRINTBYTE(0AH);
192   4               CALL PRINTBYTE(20H);
193   4            END;
194   3         END;



195   2         CALL PRINT$BYTE(0BH);
196   2         CALL PRINTBYTE(0AH);

197   2         CALL PRINTBYTE(1BH);
198   2         CALL PRINTBYTE(13H);
```

```
PL/M-80 COMPILER    HISTOGRAMROUTINES


 199   2          CALL PRINTSTRING(@M6);
 200   2          CALL PRINTSTRING(@M7);
 201   2          TOTAL=SHR(TOTAL,1);
 202   2          CALL PRINT$REAL((FLOAT(INT(TOTAL))* 2.),0);


 203   2          CALL PRINTBYTE(LF);
 204   2          CALL PRINTBYTE(LF);

 205   2          END PRINT$HISTOGRAM;
```

```
PL/M-80 COMPILER    HISTOGRAMROUTINES

          $ EJECT

              /*****************************************************************
              @  C O R R E C T   H I S T O G R A M
              @       THIS ROUTINE MAKES THE CORRECTION FOR THE CAMERA
              @  ERROR IN THE LEFT PART OF THE SCANNING
              @
              *****************************************************************/

 206   1      CORRECT$HISTOGRAM : PROCEDURE PUBLIC ;


 207   2          IF HISTOGRAM(0)>=  600H THEN HISTOGRAM(0)=HISTOGRAM(0)-600H;
                    ELSE
 209   2          HISTOGRAM(0)=0H;
 210   2          IF HISTOGRAM(1)>=  250H THEN HISTOGRAM(1)=HISTOGRAM(1)-250H;
                    ELSE
 212   2          HISTOGRAM(1)=0H;
 213   2          IF HISTOGRAM(2)>= 50H THEN HISTOGRAM(2)=HISTOGRAM(2)-50H;
                    ELSE
 215   2          HISTOGRAM(2)=0H;
 216   2          IF HISTOGRAM(3)>= 0AH THEN HISTOGRAM(3)=HISTOGRAM(3)-0AH;
                    ELSE
 218   2          HISTOGRAM(3)=0H;
 219   2          IF HISTOGRAM(4)>= 0AH THEN HISTOGRAM(4)=HISTOGRAM(4)-0AH;
                    ELSE
 221   2          HISTOGRAM(4)=0H;


 222   2      END CORRECT$HISTOGRAM;
```

```
PL/M-86 COMPILER    HISTOGRAMROUTINES

            $EJECT

                /*****************************************************************
                @       M A X I M U M   GREY-LEVEL
                @
                @   THIS ROUTINE RETURN THE VALUE OF THE MAXIMUM GREY LEVEL
                @   FOUND WITHIN THE WHOLE HISTOGRAM
                @
                *****************************************************************/

233   1     MAXIMUM$GREY$LEVEL : PROCEDURE  PUBLIC ;

234   2     DECLARE
                J BYTE;

235   2        J=254 ;
236   2        DO WHILE ((HISTOGRAM(J)(01FH)  AND (J)0H));
237   3           J=J-1;
238   3        END;
239   2        BODEGA(1)=J;


240   2     END MAXIMUM$GREY$LEVEL;


                /*****************************************************************
                @       H I S T O G R A M - W I D T H
                @       (STANDARD DEVIATION)
                @
                @   THIS ROUTINE RETURNS THE VALUE OF THE WHOLE
                @   HISTOGRAM WIDE
                @
                *****************************************************************/


241   1     HISTOGRAM$WIDE : PROCEDURE  PUBLIC ;

242   2        BODEGA(2)=(BODEGA(1) - BODEGA(0));

243   2     END HISTOGRAM$WIDE;


                  /*****************************************************************
                  @       M I N I M U M  GREY-LEVEL
                  @   THIS ROUTINE RETURNS THE VALUE OF THE MINIMUM FOUND
                  @   GREY-LEVEL IN THE WHOLE HISTOGRAM
                  @
                  *****************************************************************/

  223   1     MINIMUM$GREY$LEVEL : PROCEDURE  PUBLIC ;

  224   2        DECLARE
                   J BYTE;

  225   2           J=0H;
  226   2           DO WHILE HISTOGRAM(J)(01FH);
  227   3              J=J+1 ;
  228   3           END;
  229   2        IF J=0FFH THEN BODEGA(0)=0H;
                   ELSE
  231   2        BODEGA(0)=J;

  232   2     END MINIMUM$GREY$LEVEL;
```

PL/M-86 COMPILER HISTOGRAMROUTINES

```
            $EJECT

               /*****************************************************************
               @  B L A C K   L O W E R - T H R E S H O L D
               @  ASSUMES MAX Y MIN GREY-LEVELS HAVE BEEN CALCULATED AS WELL
               @  AS THE HISTOGRAM WIDTH .
               @*****************************************************************/

244  1     BLACK$LOWER$THRESHOLD: PROCEDURE PUBLIC;
245  2        DECLARE
                  J BYTE ,
                  ( W0 , W1)BYTE;

246  2        THRESHOLD=06FFH;
247  2        W1=255;
248  2        W0= 3 * (SHR(BODEGA(2),3));
249  2        DO WHILE ((W1 )= W0)  AND (THRESHOLD ) 0H));
250  3           J=BODEGA(0);
251  3           THRESHOLD= THRESHOLD / 2 ;
252  3              DO WHILE HISTOGRAM(J) ( THRESHOLD AND J (= BODEGA(1);
253  4                 J=J+1;
254  4              END;
255  3           W1= J - BODEGA(0);
256  3        END;
257  2        IF THRESHOLD=0H THEN BODEGA(3)=0H;
                  ELSE
259  2        BODEGA(3)=J;
260  2     END BLACK$LOWER$THRESHOLD;



               /*****************************************************************
               @  B L A C K  U P P E R  T H R E S H O L D
               @
               @  THIS ROUTINE RETURN THE VALUE OF THE BLACK UPPER
               @  THRESHOLD
               @
               *****************************************************************/

261  1     BLACK$UPPER$THRESHOLD : PROCEDURE  PUBLIC ;

262  2        DECLARE
                  J BYTE ;

263  2        IF THRESHOLD)0H THEN DO;
265  3           J= BODEGA(3)+5;
266  3           DO WHILE HISTOGRAM(J) ) THRESHOLD AND J (= BODEGA (3) ;
267  4              J=J+1;
268  4           END;
269  3           BODEGA(4)=J;
270  3           END;
                    ELSE
271  2           BODEGA(4)=0H;
272  2     END BLACK$UPPER$THRESHOLD;
```

```
$EJECT

   /*****************************************************************
   @  W H I T E   L O W E R   T H R E S H O L D

   @
   @  THIS ROUTINE RETURN THE VALUE OF WHITE LOWER THRES.
   @
   *****************************************************************/

273  1  WHITE$LOWER$THRESHOLD : PROCEDURE  PUBLIC;

274  2     DECLARE J BYTE;
275  2     IF THRESHOLD ) 0H THEN DO;
277  3        J=BODEGA(4)+3;
278  3        DO WHILE HISTOGRAM(J) ( THRESHOLD AND J (= BODEGA (1);
279  4           J=J+1;
280  4        END;
281  3     BODEGA (5)=J;
282  3     END;
           ELSE
283  2     BODEGA(5)=0H;

284  2  END WHITE$LOWER$THRESHOLD;


   /*****************************************************************
   @  B L A C K     W I D T H
   @
   @  THIS ROUTINE RETURNS THE WIDE OF THE BLACK DISTRI-
   @  BUTION  ( STANDARD DEVIATION )
   *****************************************************************/

285  1  BLACK$WIDE : PROCEDURE  PUBLIC;

286  2     DECLARE  J BYTE;

287  2        J=(BODEGA(4) - BODEGA(3));

288  2     BODEGA(6)=J;

289  2  END BLACK$WIDE;

   /*****************************************************************
   @  W H I T E       W I D T H
   @
   @  THIS ROUTINE RETURN THE VALUE OF THE WHITE WIDE
   *****************************************************************/

290  1  WHITE$WIDE : PROCEDURE  PUBLIC ;
291  2     DECLARE J BYTE;

292  2     J=(BODEGA(7) - BODEGA(5));

293  2     BODEGA(8)=J;

294  2  END WHITE$WIDE;
```

```
            $EJECT

                /*************************************************************
                @  W H I T E   U P P E R - T H R E S H O L D
                @
                *************************************************************/

275   1     WHITE$UPPER$THRESHOLD: PROCEDURE PUBLIC;

276   2         DECLARE
                    J BYTE;
277   2         IF THRESHOLD > 0H THEN DO;

279   3             J= BODEGA(1);
280   3             DO WHILE HISTOGRAM(J) < THRESHOLD AND J >= BODEGA(0);
281   4                 J=J-1;
282   4             END;
283   3         BODEGA(7)=J;
284   3         END;
                    ELSE
285   2         BODEGA(7)=0H;

286   2     END WHITE$UPPER$THRESHOLD;

                /*************************************************************
                @  V A L L E Y    W I D T H
                @
                @  THIS ROUTINE RETURNS THE VALUE OF THE WIDE VALLEY
                @  BETWEEN THE WHITE AND BLACK DISTRIBUTIONS
                *************************************************************/

287   1     VALLEY$WIDE : PROCEDURE  PUBLIC ;

288   2         DECLARE  J BYTE;
289   2             J=( BODEGA(5) - BODEGA(4));

290   2             BODEGA(9)=J;

291   2     END VALLEY$WIDE;

                /*************************************************************
                @  M A X I M U M   B L A C K    GREY-LEVEL
                @  RETURNS THE GREY-LEVEL IN WHICH THE MAXIMUM VALUE IS
                   *************************************************************/

292   1     MAX$BLACK$GREY$LEVEL: PROCEDURE PUBLIC;

293   2     DECLARE J BYTE,
                MAX WORD , K BYTE ;

294   2         J=BODEGA(3);
295   2         MAX=0H;
296   2         K=0H;
297   2         DO WHILE J <= BODEGA(4);
298   3             IF HISTOGRAM(J) > MAX THEN DO;
299   4                     MAX=HISTOGRAM(J);
300   4                     K=J;
```

```
PL/M-86 COMPILER    HISTOGRAMROUTINES

            $EJECT

               /*****************************************************************
               @          P A G E
               @   THIS ROUTINE SELECTS THE HISTOGRAM PAGE BY INPUT
               @   THE NUMBER FROM THE KEYBOARD
               @
               *****************************************************************/

344   1     PAGE : PROCEDURE PUBLIC;

345   2        DECLARE ( PAGE$NUMBER,STATUS )BYTE;
346   2        DECLARE M0(*) BYTE DATA(LF,CR,' TYPE PAGE NUMBER (0,1,2,3): ',ETX);
347   2        DECLARE M1(*) BYTE DATA(LF,CR,' CURRENT PAGE AND ENHANCE STEP : ',ETX);
348   2        STATUS=(READ$SYSTEM$STATUS$REGISTER AND 30H);
349   2        STATUS=((SHR(STATUS,4)) AND 0FH);
350   2        CALL CRT$STRING$OUT(@M1);
351   2        CALL CRT$HEX$OUT(STATUS);


352   2        CALL CRT$STRING$OUT (@M0);
353   2        PAGE$NUMBER=CRTIN;

354   2        IF PAGE$NUMBER= '0' THEN PAGE$NUMBER=0H;
356   2        IF PAGE$NUMBER= '1' THEN PAGE$NUMBER=01H;
358   2        IF PAGE$NUMBER= '2' THEN PAGE$NUMBER=02H;
360   2        IF PAGE$NUMBER= '3' THEN PAGE$NUMBER=03H;
362   2        CALL SELECT$HISTOGRAM$PAGE(PAGE$NUMBER);

363   2        RETURN;

364   2     END PAGE ;

               /*************************************************
               *        HISTOGRAM$BLANK
               *************************************************/

365   1     HISTOGRAM$BLANK: PROCEDURE PUBLIC;
366   2        DECLARE K BYTE;
367   2        SUMA=0H;
368   2        DO K=0 TO 255;
369   3           SUMA = SUMA + HISTOGRAM(K);
370   3        END;
371   2     END HISTOGRAM$BLANK;
```

PL/M-86 COMPILER HISTOGRAMROUTINES

```
                $EJECT
                   /*****************************************************************
                   @  H I S T O G R A M  S T A T I S T I C S
                   @       C A L C U L A T I O N S
                   @*****************************************************************/

372  1     HISTOGRAM$STATISTICS$CALCULATIONS: PROCEDURE PUBLIC;

373  2     CALL MINIMUM$GREY$LEVEL;  /* BODEGA 0 */

374  2     CALL MAXIMUM$GREY$LEVEL;  /* BODEGA 1 */

375  2     CALL HISTOGRAM$WIDE;      /* BODEGA 2 */

376  2     CALL BLACK$LOWER$THRESHOLD;  /* BODEGA 3 */

377  2     CALL BLACK$UPPER$THRESHOLD;  /* BODEGA 4 */

378  2     CALL WHITE$LOWER$THRESHOLD;  /* BODEGA 5 */

379  2     CALL BLACK$WIDE;  /* BODEGA 6 */

380  2     CALL WHITE$UPPER$THRESHOLD; /* BODEGA 7 */

381  2     CALL WHITE$WIDE;  /* BODEGA 8 */

382  2     CALL VALLEY$WIDE; /* BODEGA 9 */

383  2     CALL MAX$BLACK$GREY$LEVEL;  /* BODEGA 10 */

384  2     CALL MAX$WHITE$GREY$LEVEL;  /* BODEGA 11 */

385  2     BODEGA(5)=(BODEGA(5)-(BODEGA(9)/2))+(BODEGA(9)/8)+03H;

386  2     BODEGA(7)=(BODEGA(7)+(BODEGA(9)/2))+(BODEGA(9)/8)+2FH;

387  2     END HISTOGRAM$STATISTICS$CALCULATIONS;
```

```
                $EJECT

                   /*****************************************************************
                   @  H I S T O G R A M     S T A T I S T I C S
                                            D I S P L A Y
                   @   THIS ROUTINE CALCULATE ALL PARAMETERS DEFINED FOR THE
                   @   HISTOGRAM.
                   @
                   *****************************************************************/

386   1      HISTOGRAM$STATISTICS$DISPLAY  : PROCEDURE  PUBLIC ;

387   2      DECLARE
                  CMND BYTE ,
                  M0(*) BYTE DATA(LF,CR,'BLACK WIDTH ------------------------ ',ETX),
                  M1(*) BYTE DATA(LF,CR,'MAXIMUM BLACK VALUE AT GREY-LEVEL - ',ETX),
                  M2(*) BYTE DATA(LF,CR,'MINIMUM GREY-LEVEL ----------------- ',ETX),
                  M3(*) BYTE DATA(LF,CR,'MAXIMUM GREY-LEVEL ----------------- ',ETX),
                  M4(*) BYTE DATA(LF,CR,'BLACK LOWER THRESHOLD -------------- ',ETX),
                  M5(*) BYTE DATA(LF,CR,'BLACK UPPER THRESHOLD -------------- ',ETX),
                  M6(*) BYTE DATA(LF,CR,'HISTOGRAM WIDTH -------------------- ',ETX),
                  M7(*) BYTE DATA(LF,CR,'WHITE UPPER THRESHOLD -------------- ',ETX);

390   2      DECLARE
                  M8(*) BYTE DATA(LF,CR,'WHITE LOWER THRESHOLD -------------- ',ETX),
                  M9(*) BYTE DATA(LF,CR,'WHITE WIDTH ------------------------ ',ETX),
                  M10(*)BYTE DATA(LF,CR,'VALLEY WIDTH ----------------------- ',ETX),
                  M11(*)BYTE DATA(LF,CR,'MAXIMUM WHITE VALUE AT GREY-LEVEL-- ',ETX),
                  M12(*)BYTE DATA(LF,LF,CR,'** PRINT OUT ? (Y/N) **',ETX),
                  M13(*)BYTE DATA(LF,CR,'HISTOGRAM OF GRAY-LEVELS DATA .-',LF,CR,ETX);
391   2      DECLARE
                  M14(*)BYTE DATA(LF,CR,' **** LOW LIGHT CONDITION ***',LF,CR,
                           '---- OPEN DIAPHRAGM ONE STEP ---',LF,CR,ETX),
                  M15(*)BYTE DATA(LF,CR,' **** HIGH LIGHT CONDITION ***',LF,CR,
                           '--- CLOSE DIAPHRAGH ONE STEP --',LF,CR,ETX),
                  M16(*)BYTE DATA(LF,CR,'THRESHOLD USED: -------------------- ',ETX);

392   2      DECLARE
               ( B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11 ) BYTE;

393   2      B0=BODEGA(0);
394   2      B1=BODEGA(1);
395   2      B2=BODEGA(2);
396   2      B3=BODEGA(4);
397   2      B4=BODEGA(5);
398   2      B5=BODEGA(6);
399   2      B6=BODEGA(8);
400   2      B7=BODEGA(9);
401   2      B8=BODEGA(10);
402   2      B9=BODEGA(11);
403   2      B10=BODEGA(3);
404   2      B11=BODEGA(7);


405   2      IF BODEGA(2)<01FH THEN CALL CRT$STRING$OUT(@M14);
407   2      IF BODEGA(2)> 0C8H THEN CALL CRT$STRING$OUT(@M15);
```

```
409  2      CALL CRT$STRING$OUT(@M13);

410  2      CALL CRT$STRING$OUT(@M2);
411  2      CALL CRT$REAL$OUT(FLOAT(INT(B0)),0);

412  2      CALL CRT$STRING$OUT(@M3);
413  2      CALL CRT$REAL$OUT(FLOAT(INT(B1)),0);

414  2      CALL CRT$STRING$OUT(@M6);
415  2      CALL CRT$REAL$OUT(FLOAT(INT(B2)),0);

416  2      CALL CRT$STRING$OUT(@M5);
417  2      CALL CRT$REAL$OUT(FLOAT(INT(B3)),0);

418  2      CALL CRT$STRING$OUT(@M4);
419  2      CALL CRT$REAL$OUT(FLOAT(INT(B10)),0);

420  2      CALL CRT$STRING$OUT(@M7);
421  2      CALL CRT$REAL$OUT(FLOAT(INT(B11)),0);

422  2      CALL CRT$STRING$OUT(@M8);
423  2      CALL CRT$REAL$OUT(FLOAT(INT(B4)),0);

424  2      CALL CRT$STRING$OUT(@M9);
425  2      CALL CRT$REAL$OUT(FLOAT(INT(B5)),0);

426  2      CALL CRT$STRING$OUT(@M9);
427  2      CALL CRT$REAL$OUT(FLOAT(INT(B6)),0);

428  2      CALL CRT$STRING$OUT(@M10);
429  2      CALL CRT$REAL$OUT(FLOAT(INT(B7)),0);

430  2      CALL CRT$STRING$OUT(@M1);
431  2      CALL CRT$REAL$OUT(FLOAT(INT(B8)),0);

432  2      CALL CRT$STRING$OUT(@M11);
433  2      CALL CRT$REAL$OUT(FLOAT(INT(B9)),0);

434  2      CALL CRT$STRING$OUT(@M16);
435  2      CALL CRT$REAL$OUT(FLOAT(INT(THRESHOLD)),0);

436  2      CALL CRT$STRING$OUT(@M12);
437  2      CMND=CRTIN;
438  2      IF CMND='Y' THEN DO;
440  3             CALL PRINT$STRING(@M13);

441  3             CALL PRINT$STRING(@M2);
442  3             CALL PRINT$REAL(FLOAT(INT(B0)),0);
443  3             CALL PRINT$STRING(@M3);
444  3             CALL PRINT$REAL(FLOAT(INT(B1)),0);
445  3             CALL PRINT$STRING(@M6);
446  3             CALL PRINT$REAL(FLOAT(INT(B2)),0);
447  3             CALL PRINT$STRING(@M5);
448  3             CALL PRINT$REAL(FLOAT(INT(B3)),0);
449  3             CALL PRINT$STRING(@M4);
450  3             CALL PRINT$REAL(FLOAT(INT(B0)),0);
```

```
451  3            CALL PRINT$STRING(@M7);
452  3            CALL PRINT$REAL(FLOAT(INT(B1)),0);
453  3            CALL PRINT$STRING(@M8);
454  3            CALL PRINT$REAL(FLOAT(INT(B4)),0);
455  3            CALL PRINT$STRING(@M9);
456  3            CALL PRINT$REAL(FLOAT(INT(B5)),0);
457  3            CALL PRINT$STRING(@M9);
458  3            CALL PRINT$REAL(FLOAT(INT(B6)),0);
459  3            CALL PRINT$STRING(@M10);
460  3            CALL PRINT$REAL(FLOAT(INT(B7)),0);
461  3            CALL PRINT$STRING(@M1);
462  3            CALL PRINT$REAL(FLOAT(INT(B8)),0);
463  3            CALL PRINT$STRING(@M11);
464  3            CALL PRINT$REAL(FLOAT(INT(B9)),0);
465  3            CALL PRINT$STRING(@M16);
466  3            CALL PRINT$REAL(FLOAT(INT(THRESHOLD)),0);
467  3            CALL PRINT$BYTE(LF);
468  3            CALL PRINT$BYTE(CR);
469  3            CALL PRINT$BYTE(LF);
470  3            CALL PRINT$BYTE(LF);
471  3         END;


472  2      END HISTOGRAM$STATISTICS$DISPLAY;

            /*************************************************
            @         E N H A N C E
            *************************************************/

473  1      ENHANCE: PROCEDURE (STEP) PUBLIC;
474  2         DECLARE
               STEP BYTE;
475  2         STEP=(STEP AND 03H);
476  2         CALL SELECT$HISTOGRAM$PAGE(STEP);
477  2         CALL TIME(100);
478  2      END ENHANCE;

            /*************************************************************
            @   H I S T O G R A M          P R O C E S S
            *************************************************************/

479  1      HISTOGRAM$PROCESS: PROCEDURE  (STEP,N) PUBLIC ;
480  2      DECLARE
            (STEP,N) BYTE;
481  2         CALL ENHANCE(STEP);
482  2         CALL CLEAR$HISTOGRAM$MEMORY;
483  2         CALL FRAME$GRAB(N);
484  2         CALL GET$HISTOGRAM$DATA;
485  2         CALL HISTOGRAM$BLANK;
486  2         DO WHILE SUMA=0 ;
487  3            CALL CLEAR$HISTOGRAM$MEMORY;
488  3            CALL FRAME$GRAB(N);
489  3            CALL GET$HISTOGRAM$DATA;
490  3            CALL HISTOGRAM$BLANK;
491  3         END;
492  2         CALL CORRECT$HISTOGRAM;
493  2      END HISTOGRAM$PROCESS;
```

```
PL/M-86 COMPILER    HISTOGRAMROUTINES

            $EJECT


                /*************************************************************
                @  A U T O M A T I C        T H R E S H O L D
                @      SET THE THRESHOLD VALUES IN THE REGISTERS
                *************************************************************/

494   1     AUTOMATIC$THRESHOLD: PROCEDURE  BYTE  PUBLIC;

495   2     DECLARE
                (STEP,N) BYTE;
496   2       CALL ENHANCE(0H);
497   2       CALL CLEAR$HISTOGRAM$MEMORY;
498   2       CALL ENHANCE(01H);
499   2       CALL CLEAR$HISTOGRAM$MEMORY;
500   2       CALL ENHANCE(02H);
501   2       CALL CLEAR$HISTOGRAM$MEMORY;
502   2       CALL ENHANCE(03H);
503   2       CALL CLEAR$HISTOGRAM$MEMORY;
504   2       STEP=0;
505   2       N=2;

506   2     CALL HISTOGRAM$PROCESS(STEP,N);
507   2     CALL HISTOGRAM$STATISTICS$CALCULATIONS;
508   2     DO  WHILE BODEGA(2) < (030H) ;
509   3         SUMA=0H;
510   3         STEP=STEP+1;
511   3         N= (2*N);
512   3         CALL HISTOGRAM$PROCESS(STEP,N);
513   3         CALL HISTOGRAM$STATISTICS$CALCULATIONS;
514   3             IF STEP> 03H THEN DO;
516   4                   CALL CAMERA$TO$INTERLACED;
517   4                   CALL DELAY(03H);
518   4                   RETURN STEP;
519   4                     END;
520   3     END;

521   2     CALL SET$THRESHOLD (BODEGA(5),BODEGA(7));
522   2     CALL CAMERA$TO$INTERLACED;
523   2     CALL DELAY(03H);
524   2     RETURN STEP;

525   2     END AUTOMATIC$THRESHOLD;



                /*************************************************************
                @  D I S P A T C H    H I S T O G R A M
                @
                @  THIS ROUTINE SELECTS THE DIFFERENT ACTIONS TAKEN
                @  WITH THE HISTOGRAM NUMERICAL VALUES IN MEMORY
                @
                *************************************************************/

526   1     DISPATCH$HISTOGRAM : PROCEDURE PUBLIC;
```

PL/M-86 COMPILER HISTOGRAMROUTINES

```
527  2     DECLARE
              M0(*) BYTE DATA (LF,CR,' * HISTOGRAM * ',LF,CR,                    '-  COMMANDS  -',LF,CR,
          -                    'M= PAGE IN MEMORY',LF,CR,ETX),
              M1(*) BYTE DATA ('C= CLEAR',LF,CR,                          'F= FRAME',LF,CR,
          -       'R= READ',LF,CR,ETX),
              M2(*) BYTE DATA ('S= STATISTICS',LF,CR,                       'P= PRINT HISTOGRAM',LF,CR,
          -             'Q= RETURN TO " FORMS "',LF,CR,ETX),
              M3(*) BYTE DATA ('A= ANALOG HISTOGRAM ( OSCILLOSCOPE )',LF,CR,          'H= ALL PROCESS  ',LF,
          -  CR,ETX),
              M4(*) BYTE DATA ('X= CORRECT HISTOGRAM',LF,CR,ETX);
528  2     DECLARE
              M5(*) BYTE DATA ('T= AUTOMATIC THRESHOLD',LF,CR,ETX),
                 M8(*) BYTE DATA (LF,CR,'* STEP  ',ETX);
529  2     DECLARE M9(*) BYTE DATA (LF,CR,'* DELAY ',ETX);

530  2     DECLARE
              ( CMND,J,K) BYTE;

531  2        DO WHILE 1;
532  3           CALL CRT$STRING$OUT(@M0);
533  3           CALL CRT$STRING$OUT(@M1);
534  3           CALL CRT$STRING$OUT(@M2);
535  3           CALL CRT$STRING$OUT(@M3);
536  3           CALL CRT$STRING$OUT(@M4);
537  3           CALL CRT$STRING$OUT(@M5);


538  3           CMND=CRTIN;

539  3           IF CMND= 'M' THEN DO;
541  4                 CALL CRT$STRING$OUT(@M8);
542  4                 J=CRTIN;
543  4                 CALL CRT$STRING$OUT(@M9);
544  4                 K=CRTIN;
545  4                 CALL HISTOGRAM$PROCESS(J,K);
546  4                 CALL CAMERA$TO$INTERLACED;
547  4                 CALL DELAY(0);
548  4                 CALL PRINT$HISTOGRAM;
549  4                       END;
                    ELSE
550  3           IF CMND= 'M' THEN CALL PAGE;
                    ELSE
553  3           IF CMND= 'C' THEN CALL CLEAR$HISTOGRAM$MEMORY;
                    ELSE
554  3           IF CMND= 'F' THEN DO;
556  4                    CALL CRT$STRING$OUT(@M7);
557  4                       K=CRTIN;
558  4                    CALL FRAME$GRAB(K);
559  4                    CALL CAMERA$TO$INTERLACED;
560  4                    CALL DELAY(0);
561  4                   END;
                    ELSE
562  3           IF CMND= 'S' THEN CALL GET$HISTOGRAM$DATA;
                    ELSE
564  3           IF CMND= 'P' THEN CALL PRINT$HISTOGRAM;
                    ELSE
566  3           IF CMND= 'A' THEN DO;
```

```
PL/M-86 COMPILER    HISTOGRAM$ROUTINES


 568   4                    CALL CRT$STRING$OUT(@M8);
 569   4                    J=CRTIN;
 570   4                    CALL CRT$STRING$OUT(@M9);
 571   4                    K=CRTIN;
 572   4              DO WHILE 1;
 573   5              IF CRT$CHAR$READY THEN
 574   5                    IF CRT$CHECK$CONTROL THEN DO;
 576   6                          CALL CAMERA$TO$INTERLACED;
 577   6                          CALL DELAY(03H);
 578   6                          RETURN;
 579   6                              END;
 580   5                       CALL HISTOGRAM$PROCESS(J,K);
 581   5              END;
 582   4              END;
                        ELSE
 583   3              IF CMND= 'S' THEN DO;
 585   4                          CALL HISTOGRAM$STATISTICS$CALCULATIONS;
 586   4                          CALL HISTOGRAM$STATISTICS$DISPLAY;
 587   4                          END;

                        ELSE
 588   3              IF CMND= 'X' THEN CALL CORRECT$HISTOGRAM;
                        ELSE
 590   3              IF CMND= 'T' THEN DO;
 592   4                          J= AUTOMATIC$THRESHOLD;
 593   4                          J=(J+30H);
 594   4                          CALL CRT$STRING$OUT(@M8);
 595   4                          CALL CRT$BYTE$OUT(J);
 596   4                       END;
                        ELSE
 597   3              RETURN;
 598   3          END;
 599   2      END DISPATCH$HISTOGRAM;

 600   1      END HISTOGRAM$ROUTINES;


MODULE INFORMATION:

     CODE AREA SIZE     = 1625H   5669D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 032DH    813D
     MAXIMUM STACK SIZE = 0028H     40D
     1064 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
1          FORMS$COMMANDS: DO;


                  /*************************************************************
                   *                                                           *
                   *                  F O R M S    C O M M A N D S             *
                   *                                                           *
                   *     THESE ROUTINES IMPLEMENT THE COMMAND SET OF THE       *
                   *    FORMS SYSTEM.                                          *
                   *                                                           *
                   *************************************************************/

                /*  ---------------------------------------------------------------------
                          E X T E R N A L   U S A G E   D E C L A R A T I O N S
                    --------------------------------------------------------------------- */
  2   1           CRT$STRING$OUT: PROCEDURE(PTR) EXTERNAL;
  3   2                     DECLARE PTR POINTER;
  4   2           END CRT$STRING$OUT;

  5   1           CRTIN: PROCEDURE BYTE EXTERNAL;
  6   2           END CRTIN;

  7   1           CRT$HEX$OUT: PROCEDURE(BIT$PATTERN) EXTERNAL;
  8   2                     DECLARE BIT$PATTERN BYTE;
  9   2           END CRT$HEX$OUT;

 10   1           CRT$CHAR$READY: PROCEDURE BYTE EXTERNAL;
 11   2           END CRT$CHAR$READY;

 12   1           CRT$BYTE$OUT: PROCEDURE(CHAR) EXTERNAL;
 13   2                     DECLARE CHAR BYTE;
 14   2           END CRT$BYTE$OUT;

 15   1           CRT$CHECK$CONTROL: PROCEDURE BYTE EXTERNAL;
 16   2           END CRT$CHECK$CONTROL;

 17   1           CRTIN$NO$ECHO: PROCEDURE BYTE EXTERNAL;
 18   2           END CRTIN$NO$ECHO;

 19   1           CRT$REAL$IN: PROCEDURE REAL EXTERNAL;
 20   2           END CRT$REAL$IN;

 21   1           CRT$REAL$OUT: PROCEDURE (REAL$VAR, NUMBER$DEC$PLACES) EXTERNAL;
 22   2                   DECLARE REAL$VAR REAL, NUMBER$DEC$PLACES BYTE;
 23   2           END CRT$REAL$OUT;

 24   1           ERROR$MESSAGE: PROCEDURE (ERROR$NUMBER) EXTERNAL;
 25   2                   DECLARE ERROR$NUMBER BYTE;
 26   2           END ERROR$MESSAGE;

 27   1           PRINT$STRING: PROCEDURE(PTR) EXTERNAL;
 28   2                     DECLARE PTR POINTER;
 29   2           END PRINT$STRING;

 30   1           PRINT$BYTE: PROCEDURE(ASCII$CHAR) EXTERNAL;
 31   2                     DECLARE ASCII$CHAR BYTE;
 32   2           END PRINT$BYTE;

 33   1           PRINT$HEX: PROCEDURE(BIT$PATTERN) EXTERNAL;
 34   2                     DECLARE BIT$PATTERN BYTE;
 35   2           END PRINT$HEX;

 36   1           PRINTER$READY: PROCEDURE BYTE EXTERNAL;
 37   2           END PRINTER$READY;
```

```
PL/M-86 COMPILER    FORMSCOMMANDS

            $EJECT

 38  1          PRINT$REAL: PROCEDURE (REAL$VAR,NUMBER$DEC$PLACES) EXTERNAL;
 39  2                  DECLARE
                       REAL$VAR REAL,
                       NUMBER$DEC$PLACES BYTE;
 40  2          END PRINT$REAL;

 41  1          FRAME$REQUEST: PROCEDURE EXTERNAL;
 42  2          END FRAME$REQUEST;

 43  1          MODE$SELECT: PROCEDURE (MODE) EXTERNAL;
 44  2              DECLARE MODE BYTE;
 45  2          END MODE$SELECT;

 46  1          READ$MODE: PROCEDURE BYTE EXTERNAL;
 47  2          END READ$MODE;

 48  1          READ$SYSTEM$STATUS$REGISTER: PROCEDURE BYTE EXTERNAL;
 49  2          END READ$SYSTEM$STATUS$REGISTER;


 50  1          SET$SYSTEM$STATUS$REGISTER: PROCEDURE(STATUS) EXTERNAL;
 51  2              DECLARE STATUS BYTE;
 52  2          END SET$SYSTEM$STATUS$REGISTER;

 53  1          CLEAR$HISTOGRAM$MEMORY: PROCEDURE EXTERNAL;
 54  2          END CLEAR$HISTOGRAM$MEMORY;

 55  1          SELECT$HISTOGRAM$PAGE: PROCEDURE(PAGE$NUMBER) EXTERNAL;
 56  2              DECLARE PAGE$NUMBER BYTE;
 57  2          END SELECT$HISTOGRAM$PAGE;

 58  1          READ$HISTOGRAM$MEMORY: PROCEDURE EXTERNAL;
 59  2          END READ$HISTOGRAM$MEMORY;

 60  1          SET$THRESHOLD: PROCEDURE( LOWER, UPPER) EXTERNAL;
 61  2              DECLARE  (LOWER, UPPER) BYTE;
 62  2          END SET$THRESHOLD;

 63  1          READ$LOWER$THRESHOLD: PROCEDURE BYTE EXTERNAL;
 64  2          END READ$LOWER$THRESHOLD;

 65  1          READ$UPPER$THRESHOLD: PROCEDURE BYTE EXTERNAL;
 66  2          END READ$UPPER$THRESHOLD;

 67  1          GET$HISTOGRAM$DATA: PROCEDURE EXTERNAL;
 68  2          END GET$HISTOGRAM$DATA;

 69  1          DELAY: PROCEDURE (TIME) EXTERNAL;
 70  2          DECLARE TIME BYTE;
 71  2          END DELAY;

 72  1          CAMERA$TO$SEQUENTIAL: PROCEDURE EXTERNAL;
 73  2          END CAMERA$TO$SEQUENTIAL;

 74  1          CAMERA$TO$INTERLACED: PROCEDURE EXTERNAL;
```

```
PL/M-86 COMPILER    FORMSCOMMANDS

 75   2          END CAMERA$TO$INTERLACED;

 76   1          GET$PACKED$BINARY$DATA: PROCEDURE (TIME) EXTERNAL;
 77   2          DECLARE
                     TIME BYTE;
 78   2          END GET$PACKED$BINARY$DATA;

 79   1          AUTOMATIC$THRESHOLD: PROCEDURE BYTE EXTERNAL;
 80   2          END AUTOMATIC$THRESHOLD;
```

```
PL/M-86 COMPILER    FORMSCOMMANDS

            $EJECT

 81   1          DECLARE

                   TRUE        LITERALLY '0FFH',
                   FALSE       LITERALLY '0',
                   CR          LITERALLY '0DH',
                   LF          LITERALLY '0AH',
                   ETX         LITERALLY '03H',
                   BS          LITERALLY '08H',
                   HISTOGRAM(256) WORD AT (7E00H),
                   CHEC WORD,
                   RLF         LITERALLY '0EH',
                   CLEAR$EOL   LITERALLY '0FH',
                   BELL        LITERALLY '07H',
                   FOREVER     LITERALLY 'WHILE 1';

 82   1          DECLARE

                  M(*) BYTE DATA (CR,LF,LF,'  FORMS SYSTEM STATUS:',03H),
                  UNDERLINE(*) BYTE DATA(CR,LF,'_____________________________',03H),
                  M1(*) BYTE DATA (CR,LF,LF,'NO. OBJECTS LEARNED:    ',03H),
                  M2(*) BYTE DATA (CR,LF,'NO. OBJECTS RECOGNIZED: ',03H),
                  M3(*) BYTE DATA (CR,LF,'NO. OBJECTS REJECTED:   ',03H),
                  M4(*) BYTE DATA (CR,LF,'FEATURES ENABLED:       ',03H),
                  M5(*) BYTE DATA (CR,LF,'LOWER THRESHOLD: ',03H),
                  M6(*) BYTE DATA (CR,LF,'UPPER THRESHOLD: ',03H),
                  M7(*) BYTE DATA (CR,LF,LF,'DATA TYPE: ',03H),
                  M8(*) BYTE DATA (CR,LF,'SYSTEM STATUS REGISTER: ',03H),
                  M9(*) BYTE DATA (CR,LF,'BACKGROUND: ',03H);

 83   1          DECLARE

                  CHANGE(*) BYTE DATA (CR,LF,'CHANGE (Y/N)? ',03H),
                  BLACK(*) BYTE DATA ('BLACK (1)',03H),
                  WHITE(*) BYTE DATA ('WHITE (0)',03H);
```

```
PL/M-86 COMPILER    FORMSCOMMANDS

            $EJECT


               /*************************************************************
               *          P U L S O            *
               *                               *
               *   THIS ROUTINE CHECK IF THE FRAME REQUEST HAS     *
               *   BEEN APPLIED TO VRHD AND BPP .           . *
               *************************************************************/

 84   1        PULSO:PROCEDURE PUBLIC;
 85   2        DECLARE K BYTE;

 86   2        CHEC=0H;
 87   2        DO K=0 TO 255;
 88   3           CHEC= CHEC + HISTOGRAM(K);
 89   3        END;
 90   2        END PULSO;

            /* ------------------------------------------------------
               -                                                    -
               -                    L O O K                         -
               -                                                    -
               -    THIS ROUTINE ACQUIRED IN PARALLEL THE           -
               -    HISTOGRAM AND THE BINARY IMAGE IN THE           -
               -    VISION SYSTEM.                                  -
               -                                                    -
               -                                                    -
               -                                                    -
               -                                                    -
               -                                                    -
               -                                                    -
               ------------------------------------------------------ */

 91   1     LOOK: PROCEDURE PUBLIC;

 92   2     DECLARE
               M BYTE ,
               MO(*) BYTE DATA(LF,CR,' NOT ENOUGH LIGHT DATA WILL BE ERRONEOUS',ETX);

 93   2           M=AUTOMATIC$THRESHOLD;
 94   2        CHEC=0H;
 95   2           IF M=        CALL GET$PACKED$BINARY$DATA(02H);
 97   2           IF M=01H  THEN DO;
 99   3                  CALL CLEAR$HISTOGRAM$MEMORY;
100   3                  DO WHILE CHEC=0H;
101   4                     CALL GET$PACKED$BINARY$DATA(04H);
102   4                     CALL GET$HISTOGRAM$DATA;
103   4                     CALL PULSO;
104   4                  END;
105   3                     END;
106   2           IF M=02H  THEN DO;
108   3                  CALL CLEAR$HISTOGRAM$MEMORY;
109   3                  DO WHILE CHEC=0H;
110   4                     CALL GET$PACKED$BINARY$DATA(08H);
111   4                     CALL GET$HISTOGRAM$DATA;
112   4                     CALL PULSO;
```

```
113  4                  END;
114  3                    END;
115  2             IF M=03H  THEN DO;
117  3                    CALL CLEAR$HISTOGRAM$MEMORY;
118  3                    DO WHILE CHEC=0H;
119  4                        CALL GET$PACKED$BINARY$DATA(0FH);
120  4                        CALL GET$HISTOGRAM$DATA;
121  4                        CALL PULSD;
122  4                    END;
123  3                    END;
124  2         IF M)03H THEN CALL CRT$STRING$OUT(@M0);
126  2         CALL CAMERA$TO$INTERLACED;
127  2         CALL DELAY(03H);

128  2      END LOOK;

             /*************************************************************
             *                      M I R A                              *
             *************************************************************/

129  1      MIRA: PROCEDURE PUBLIC;

130  2         DECLARE J BYTE;
131  2         DECLARE M0(*) BYTE DATA(LF,CR,' DELAY ? --',ETX);

132  2         CALL CRT$STRING$OUT(@M0);
133  2         J=CRTIN;
134  2         CALL GET$PACKED$BINARY$DATA(J);
135  2         CALL CAMERA$TO$INTERLACED;
136  2         CALL DELAY(03H);

137  2      END MIRA;
```

```
$EJECT

/*-------------------------------------------------------
  -                                                     -
  -                  D I S P L A Y                      -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -------------------------------------------------------*/

130  1    DISPLAY: PROCEDURE PUBLIC;

139  2      DECLARE
              M(*) BYTE DATA('Output device (C/P)? ',03H),
              DEVICE BYTE,
              IR(240) STRUCTURE( IC(39) BYTE ) AT (4074H),
              BLANKS(*) BYTE DATA ('    ',03H),
                  X BYTE,
                  Y BYTE;

140  2      CALL CRT$STRINGOUT(@M);
141  2      DEVICE=CRTIN;

142  2      DO Y=0 TO 239;

143  3        IF DEVICE='P' THEN DO;
145  4                           CALL PRINTBYTE(CR);
146  4                           CALL PRINTBYTE(LF);
147  4                          END;
148  3                    ELSE DO;
149  4                           CALL CRT$BYTE$OUT(CR);
150  4                           CALL CRT$BYTE$OUT(LF);
151  4                          END;

152  3         DO X=0 TO 39;
153  4          IF CRT$CHAR$READY THEN IF CRT$CHECK$CONTROL THEN RETURN;
156  4          IF DEVICE='P' THEN CALL PRINT$HEX  ( IR(Y).IC(X) );
158  4                        ELSE CALL CRT$HEX$OUT( IR(Y).IC(X) );
159  4         END;

160  3         IF DEVICE='P' THEN DO;
162  4                           CALL PRINT$STRING(@BLANKS);
163  4                           CALL PRINT$REAL( FLOAT(INT(Y)) ,0 );
164  4                          END;
165  3                    ELSE DO;
166  4                           CALL CRT$STRING$OUT(@BLANKS);
167  4                           CALL CRT$REAL$OUT( FLOAT(INT(Y)) ,0 );
168  4                          END;

169  3        END;

170  2    END DISPLAY;
```

```
$EJECT

/*-------------------------------------------------------
  -                                                     -
  -                  S T A T U S                        -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  -                                                     -
  ------------------------------------------------------- */

171  1    STATUS: PROCEDURE PUBLIC;

172  2       CALL CRT$STRINGOUT(@M);
173  2       CALL CRT$STRINGOUT(@UNDERLINE);
174  2       CALL CRT$STRINGOUT(@M1);
175  2       CALL CRT$STRINGOUT(@M2);
176  2       CALL CRT$STRINGOUT(@M3);
177  2       CALL CRT$STRINGOUT(@M4);

178  2       CALL CRT$BYTE$OUT(LF);
179  2       CALL CRT$STRINGOUT(@M5);
180  2       CALL CRT$REALOUT( FLOAT(INT(READ$LOWER$THRESHOLD)) , 0 );
182  2       CALL CRT$REALOUT( FLOAT(INT(READ$UPPER$THRESHOLD)) , 0 );

183  2       CALL CRT$STRINGOUT(@M7);
184  2       CALL CRT$HEXOUT( READ$MODE );

185  2       CALL CRT$STRINGOUT(@M8);
186  2       CALL CRT$HEXOUT( READ$SYSTEM$STATUS$REGISTER );

187  2       CALL CRT$STRINGOUT(@M9);
188  2       IF (( READ$SYSTEM$STATUS$REGISTER AND 80H) = 0 )
                    THEN CALL CRT$STRINGOUT(@BLACK);
190  2                  ELSE CALL CRT$STRINGOUT(@WHITE);

191  2    END STATUS;
```

```
PL/M-86 COMPILER    FORMSCOMMANDS

            $EJECT

            /*--------------------------------------------------------
              -                                                      -
              -                  T H R E S H O L D                   -
              -                                                      -
              -                                                      -
              -                                                      -
              -                                                      -
              -                                                      -
              -                                                      -
              -                                                      -
              -                                                      -
              -                                                      -
              -                                                      -
              -                                                      -
              --------------------------------------------------------  */

192   1     THRESHOLD: PROCEDURE PUBLIC;

193   2       DECLARE
                (L,H) BYTE,
                T(*) BYTE DATA (CR,LF,'Type new values (0 to 255) ...',03H);

194   2       CALL CRT$BYTEOUT(RLF);
195   2       CALL CRT$STRINGOUT(@M5);
196   2       CALL CRT$REALOUT( FLOAT(INT(READ$LOWER$THRESHOLD)) , 0 );
197   2       CALL CRT$STRINGOUT(@M6);
198   2       CALL CRT$REALOUT( FLOAT(INT(READ$UPPER$THRESHOLD)) , 0 );

199   2       CALL CRT$STRINGOUT(@CHANGE);
200   2       L=CRTIN;

201   2       IF L='Y' THEN
202   2        DO;
203   3         CALL CRT$STRINGOUT(@T);
204   3         CALL CRT$STRINGOUT(@M5);
205   3         L=LOW( UNSIGN( FIX( CRT$REAL$IN )));
206   3         CALL CRT$STRINGOUT(@M6);
207   3         H=LOW( UNSIGN( FIX( CRT$REAL$IN )));
208   3         CALL SET$THRESHOLD( L,H );
209   3        END;

210   2     END THRESHOLD;
```

```
PL/M-86 COMPILER    FORMSCOMMANDS

            $EJECT

            /*-----------------------------------------------------
              -                                                   -
              -                  B A C K G R O U N D              -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              -                                                   -
              ----------------------------------------------------- */

  211   1     BACKGROUND: PROCEDURE PUBLIC;

  212   2       DECLARE
                 T BYTE,
                 B(*) BYTE DATA (CR,LF,'Select background color (B/W) ...',03H);

  213   2       CALL CRT$BYTEOUT(@LF);
  214   2       CALL CRT$STRINGOUT(@M9);
  215   2       IF (( READ$SYSTEM$STATUS$REGISTER AND 80H) = 0 )
                    THEN CALL CRT$STRINGOUT(@BLACK);
  217   2            ELSE CALL CRT$STRINGOUT(@WHITE);

  218   2       CALL CRT$STRINGOUT(@B);
  219   2       T=CRTIN;
  220   2       IF T='B' THEN
  221   2         CALL SET$SYSTEM$STATUS$REGISTER(7FH AND READ$SYSTEM$STATUS$REGISTER);
  222   2       IF T='W' THEN
  223   2         CALL SET$SYSTEM$STATUS$REGISTER(80H OR READ$SYSTEM$STATUS$REGISTER);

  224   2     END BACKGROUND;
```

```
PL/M-86 COMPILER    FORMSCOMMANDS

            $EJECT

            /*-----------------------------------------------
              -                                             -
              -                 C A M E R A                 -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              -                                             -
              ----------------------------------------------- */

 225   1      CAMERA: PROCEDURE PUBLIC;

 226   2          DECLARE
                   T BYTE,
                   C(*) BYTE DATA(CR,'TYPE CAMERA SCAN MODE (I/S)....',03H);

 227   2          CALL CRT$STRINGOUT(@C);
 228   2          T=CRTIN;
 229   2          IF T='S' THEN CALL CAMERA$TO$SEQUENTIAL;
 231   2          IF T='I' THEN CALL CAMERA$TO$INTERLACED;

 233   2      END CAMERA;

 234   1      END FORMS$COMMANDS;


MODULE INFORMATION:

    CODE AREA SIZE     = 0680H   1664D
    CONSTANT AREA SIZE = 0000H      0D
    VARIABLE AREA SIZE = 000CH     12D
    MAXIMUM STACK SIZE = 000CH     12D
    481 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
PL/M-86 COMPILER    ACQUISITIONROUTINES

ISIS-II PL/M-86 V2.1 COMPILATION OF MODULE ACQUISITIONROUTINES
OBJECT MODULE PLACED IN :F1:ACQUIR.OBJ
COMPILER INVOKED BY:  PLM86 :F1:ACQUIR.SRC LARGE OPTIMIZE(3)


  1          ACQUISITION$ROUTINES: DO;



             /*************************************************************
              *                                                           *
              *      A C Q U I S I T I O N       R O U T I N E S          *
              *                                                           *
              *       Routines included:                                  *
              *       ------------------                                  *
              *                                                           *
              *             FRAME$GRAB                                    *
              *             GET$HISTOGRAM$DATA                            *
              *             GET$PACKED$BINARY$DATA                        *
              *             GET$GREY$LEVEL$DATA                           *
              *                                                           *
              *************************************************************/



             /*  ------------------------------------------------------------
                       E X T E R N A L   U S A G E   D E C L A R A T I O N S
                 ------------------------------------------------------------  */

  2  1          FRAME$REQUEST: PROCEDURE EXTERNAL;
  3  2          END FRAME$REQUEST;

  4  1          MODE$SELECT: PROCEDURE (MODE) EXTERNAL;
  5  2              DECLARE MODE BYTE;
  6  2          END MODE$SELECT;

  7  1          READ$SYSTEM$STATUS$REGISTER: PROCEDURE BYTE EXTERNAL;
  8  2          END READ$SYSTEM$STATUS$REGISTER;

  9  1          READ$HISTOGRAM$MEMORY: PROCEDURE EXTERNAL;
 10  2          END READ$HISTOGRAM$MEMORY;

 11  1          CAMERA$TO$SEQUENTIAL: PROCEDURE EXTERNAL;
 12  2          END CAMERA$TO$SEQUENTIAL;

 13  1          CAMERA$TO$INTERLACED: PROCEDURE EXTERNAL;
 14  2          END CAMERA$TO$INTERLACED;

 15  1          DELAY: PROCEDURE (TIME) EXTERNAL;
 16  2          DECLARE TIME BYTE ;
 17  2          END DELAY;


 18  1          DECLARE

                   TRUE       LITERALLY '0FFH',
                   FALSE      LITERALLY '0';
```

PL/M-86 COMPILER ACQUISITIONROUTINES

```
$EJECT

/*-------------------------------------------------------
  -                                                     -
  -            F R A M E   G R A B                      -
  -                                                     -
  -   THIS ROUTINE FIRST CHECKS IF THE TM2500 CAMERA    -
  -  IS SCANNING IN SEQUENTIAL MODE, AND IF NOT,        -
  -  CHANGES THE SYSTEM STATUS REGISTER (BIT 1) TO DO   -
  -  SO.  A FRAME REQ PULSE IS THEN ISSUED TO THE       -
  -  SYSTEM FOLLOWED BY A TIME DELAY DURING WHICH THE   -
  -  DATA IS COLLECTED.  FINALLY, IF THE CAMERA WAS     -
  -  SCANNING IN THE INTERLACED MODE, THEN IT IS        -
  -  RESTORED TO THAT MODE.                            -
  -                                                     -
  ------------------------------------------------------- */

19  1   FRAME$GRAB: PROCEDURE (TIME) PUBLIC;

20  2     DECLARE TIME BYTE ;

21  2     CALL MODE$SELECT(0FFH);
22  2     TIME=(TIME AND 0FH);
23  2     IF ((READ$SYSTEM$STATUS$REGISTER AND 40H) <> 0)
                    THEN
24  2                   DO;
25  3                     CALL CAMERA$TO$SEQUENTIAL;
26  3                     CALL DELAY(03H);
27  3                   END;
28  2     CALL FRAME$REQUEST;
29  2     CALL DELAY (TIME);

30  2   END FRAME$GRAB;
```

```
$EJECT

/*-------------------------------------------------------
  -                                                     -
  -    G E T    H I S T O G R A M    D A T A           -
  -                                                     -
  -  THIS ROUTINE SELECTS HISTOGRAM DATA ON THE DMA     -
  -  BUS AND PROGRAMS THE DMA BOARD TO TRANSFER THE     -
  -  DATA INTO THE 86/12 RAM STARTING AT LOCATION       -
  -  7E00H.                                            -
  -                                                     -
  -------------------------------------------------------*/

 31  1    GET$HISTOGRAM$DATA: PROCEDURE PUBLIC;

 32  2        CALL MODE$SELECT(0F8H);

 33  2        OUTPUT(0FEH)=0FFH;
 34  2        OUTPUT(0F0H)=01H;          /*      LENGTH        */
 35  2        OUTPUT(0FAH)=07H;          /*      CONTROL       */
 36  2        OUTPUT(0FEH)=0;
 37  2        OUTPUT(0FFH)=0BEH;         /*  FIRST ADDRESS     */

 38  2        CALL READ$HISTOGRAM$MEMORY;
 39  2        CALL TIME(40);

 40  2        OUTPUT(0F8H)=0;            /*      RESET         */

 41  2    END GET$HISTOGRAM$DATA;
```

```
$EJECT

/*-------------------------------------------------------------
  -                                                           -
  -  G E T   P A C K E D   B I N A R Y   D A T A              -
  -                                                           -
  -   THIS ROUTINE SELECTS THE PACKED BINARY DATA            -
  - TYPE AND PROGRAMS THE iSBC 501 DMA CONTROLLER            -
  - TO TRANSFER THE DATA INTO THE 86/12 RAM STARTING         -
  - AT LOCATION 4000H.                                       -
  -                                                           -
  ------------------------------------------------------------- */

42  1   GET$PACKED$BINARY$DATA: PROCEDURE (TIME) PUBLIC;

43  2   DECLARE TIME BYTE;
44  2     CALL MODE$SELECT(0FFH);

45  2     OUTPUT(0FCH)=0AAH;
46  2     OUTPUT(0FDH)=1DH;        /*      LENGTH       */
47  2     OUTPUT(0FAH)=07H;        /*      CONTROL      */
48  2     OUTPUT(0FEH)=0;
49  2     OUTPUT(0FFH)=0A0H;       /*  FIRST ADDRESS    */

50  2     CALL FRAME$GRAB(TIME);

51  2     OUTPUT(0F9H)=0;          /*       RESET       */

52  2   END GET$PACKED$BINARY$DATA;
```

```
PL/M-86 COMPILER    ACQUISITIONROUTINES

            $EJECT

            /*--------------------------------------------------------
              -                                                      -
              -    G E T    G R E Y    L E V E L    D A T A         -
              -                                                      -
              -  THIS ROUTINE SELECTS GREY LEVEL DATA ON THE DMA     -
              - BUS AND PROGRAMS THE DMA BOARD TO TRANSFER THE       -
              - DATA INTO THE 86/12 RAM STARTING AT LOCATION         -
              - 4000H.                                               -
              -                                                      -
              -------------------------------------------------------- */

 53  1      GET$GREY$LEVEL$DATA: PROCEDURE (TIME) PUBLIC;

 54  2          DECLARE  TIME  BYTE ;

 55  2          CALL MODE$SELECT(0F7H);

 56  2          OUTPUT(0FCH)=0C5H;
 57  2          OUTPUT(0FDH)=038H;          /*       LENGTH        */
 58  2          OUTPUT(0FAH)=07H;           /*       CONTROL       */
 59  2          OUTPUT(0FEH)=0;
 60  2          OUTPUT(0FFH)=0A0H;          /*  FIRST ADDRESS      */

 61  2          CALL FRAME$GRAB(TIME);

 62  2          OUTPUT(0F9H)=0;             /*       RESET         */

 63  2      END GET$GREY$LEVEL$DATA;



 64  1      END ACQUISITION$ROUTINES;


MODULE INFORMATION:

     CODE AREA SIZE     = 00ECH    236D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0016H     22D
     186 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

CHAPTER V

MEASUREMENTS

This chapter contains measurements on the ALS and the TN 2500 G.E. camera . By using the available facilities provided by the histogram package (software) described in chapter IV, four types of measurements were carried out : the VRHG performance, dynamic range and uniformity on the sensitivity of the sensor transducer for different spatial locations, adaptive range of the ALS, and an adaptive process for a typical operation of the ALS under poor light conditions.

5.1 VRHG performance

In order to verify the performance of the VRHG, histograms of grey-levels were taken for different scenes and under equal light conditions. Diaphragm aperture and sensitivity step in the camera were set to f=1.8 and normal operation (step 0) respectively.

The histograms were printed without any smoothing software stage. The total number of pixels analyzed for each picture was always 58 528 (spatial resolution used by the VRHG) as expected.

The number of peaks and the white and black magnitude relation were obtained as expected as well. The magnitude for each area

of grey-level on the scene was established previously by using two dominant grey-levels:

- white backlighted acr lic (background)
- N number of " blocks " (object) .

A " block " is defined to be a constant square area of grey-level G (10 % of the camera's field of view). In order to present different magnitudes relations between the white and black peaks of the histogram, use was made of a different number of blocks for each scene. Figures 5.1 to 5.4 show these histograms.

A second test was done to verify the VRHG performance by taking histograms of an 18 % reflectance grey-card [5-1] covering the total area of the camera's field of view. The diaphragm aperture was set all the time to f=4.0 and different illumination was presented each time by way of a dimmer control in the light source.

The table 5.4a shows the equivalence among the BL/V number, EV number and foot Lambert units .

Figures 5.5 to 5.7 show these results with only one major peak of grey level as expected.

The total number of pixels analyzed are now different due to the smoothing algorithms applied to the histograms.

5.2 TN 2500 G.E. camera measurements

In chapter III, the operation of the camera was described. In this section , measurements on the particular camera used by the vision system have been done on its dynamic range (number of grey-levels used for the grey scale pictures) and uniformity of sensitivity

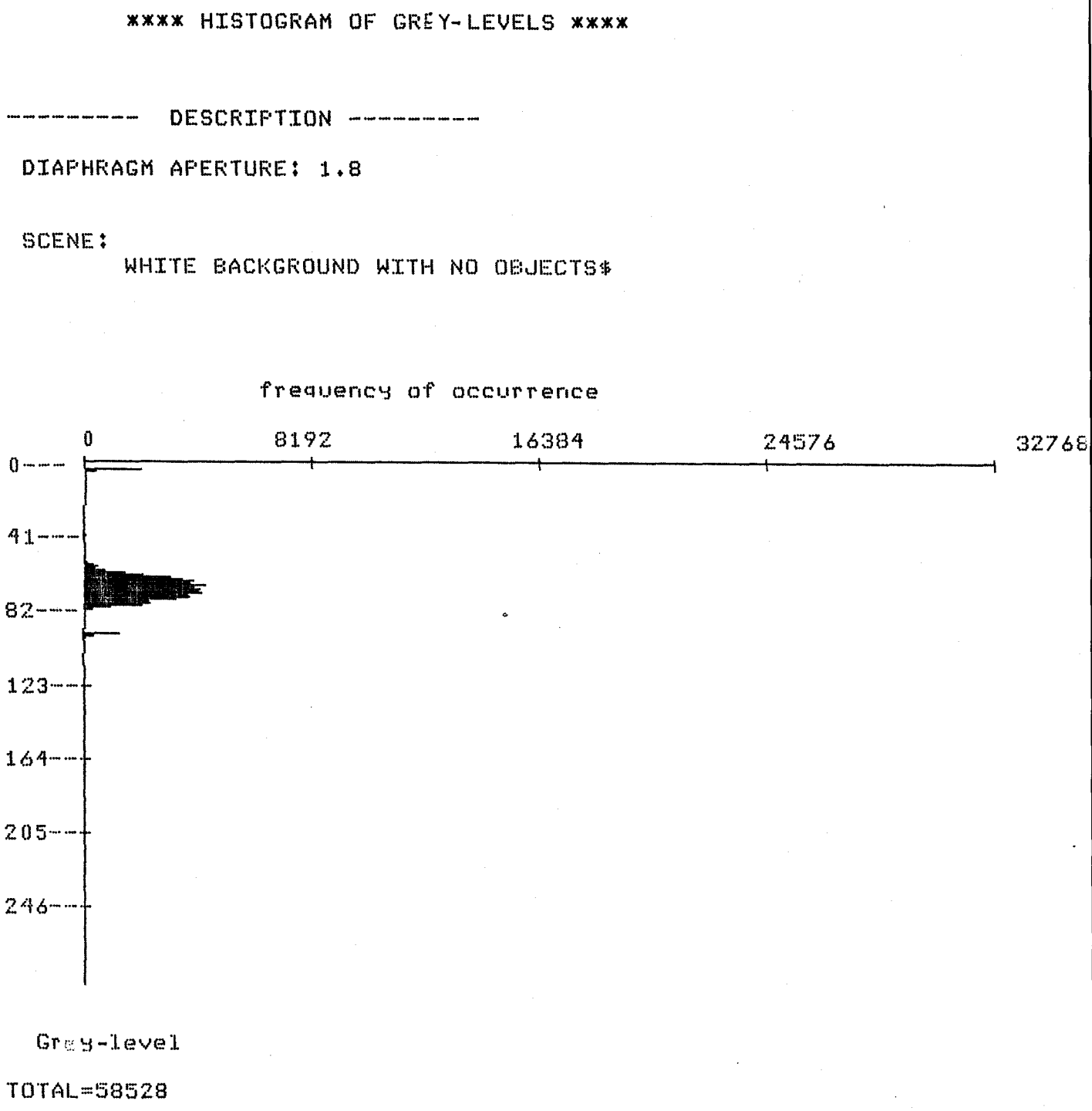


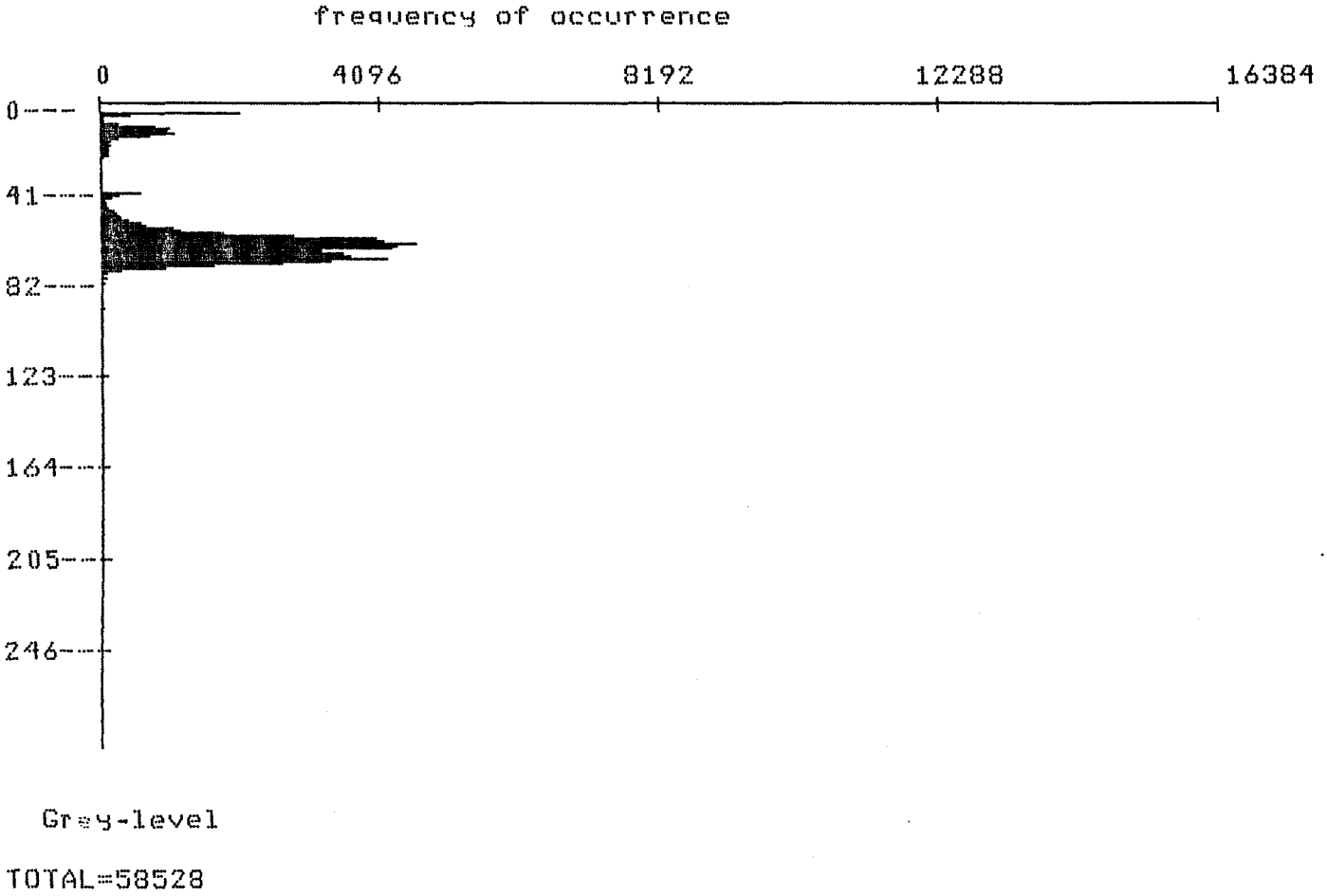
Figure 5.1 VRHG/ performance . (scheme 1)

**** HISTOGRAM OF GREY-LEVELS ****

---------- DESCRIPTION ----------

DIAPHRAGM APERTURE: 1.8

SCENE:
ONE OBJECT IN WHITE BACKGROUND $



TOTAL=58528

Figure 5.2 VRHG / performance (scheme 2)

ILLUMINATION SCALE

BL/V number	EV number	Foot-Lambert
100	7	5.2
90	6.8	4.8
80	6.5	4.2
70	5.6	2.8
65	5.2	2.3
60	4.6	1.7
55	4.1	1.1
50	3.7	0.8
45	3.0	0.5
40	2.3	0.35
35	1.6	0.22
30	1.4	0.185
25	1.3	0.160
20	1.0	0.125
15	0.9	0.100
10	0.7	0.080

Table 5.4a Illumination scale used in the measurements (reproduced from [5-2]) .

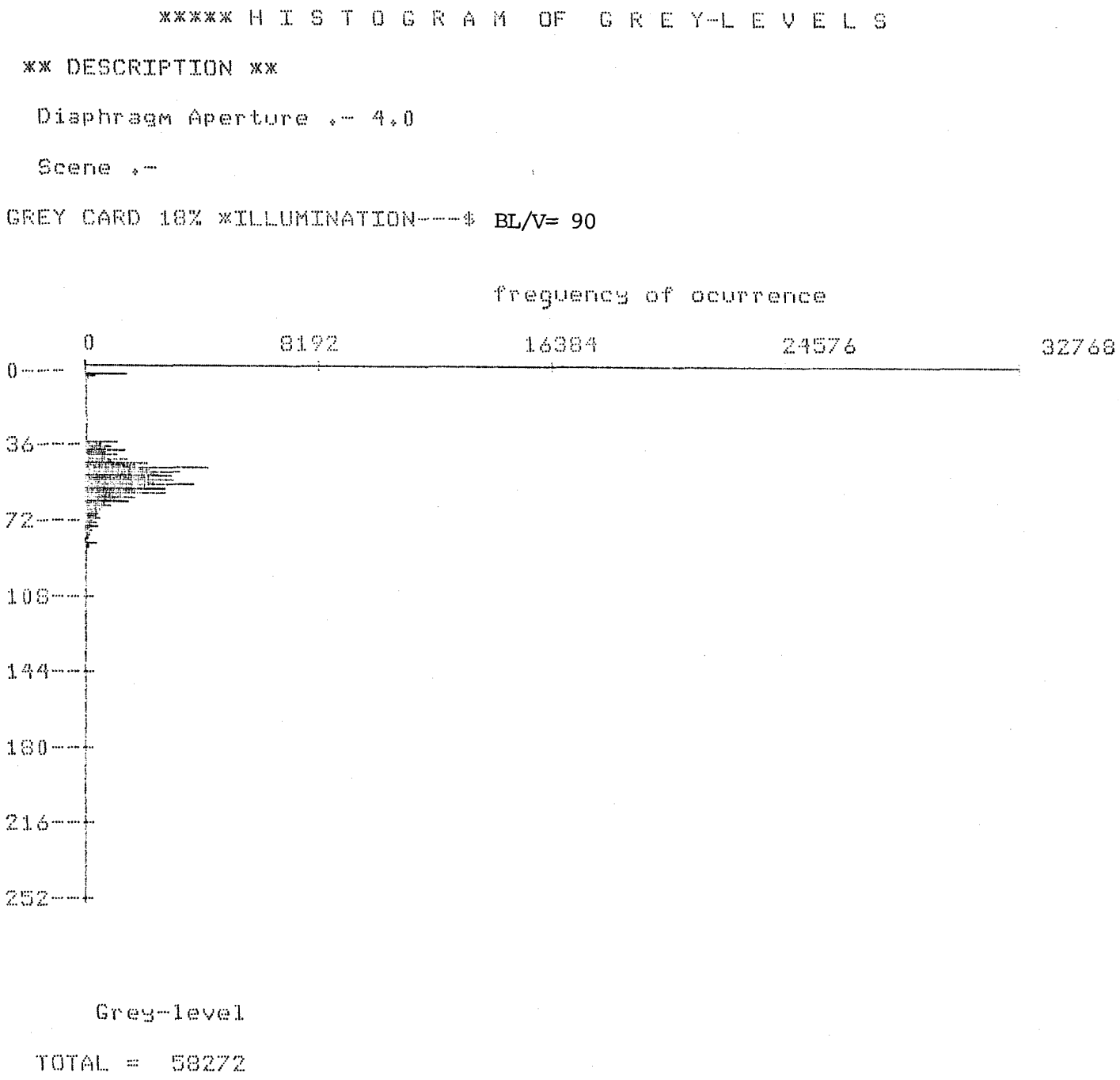


Figure 5.5 VRHG performance (scheme 5).

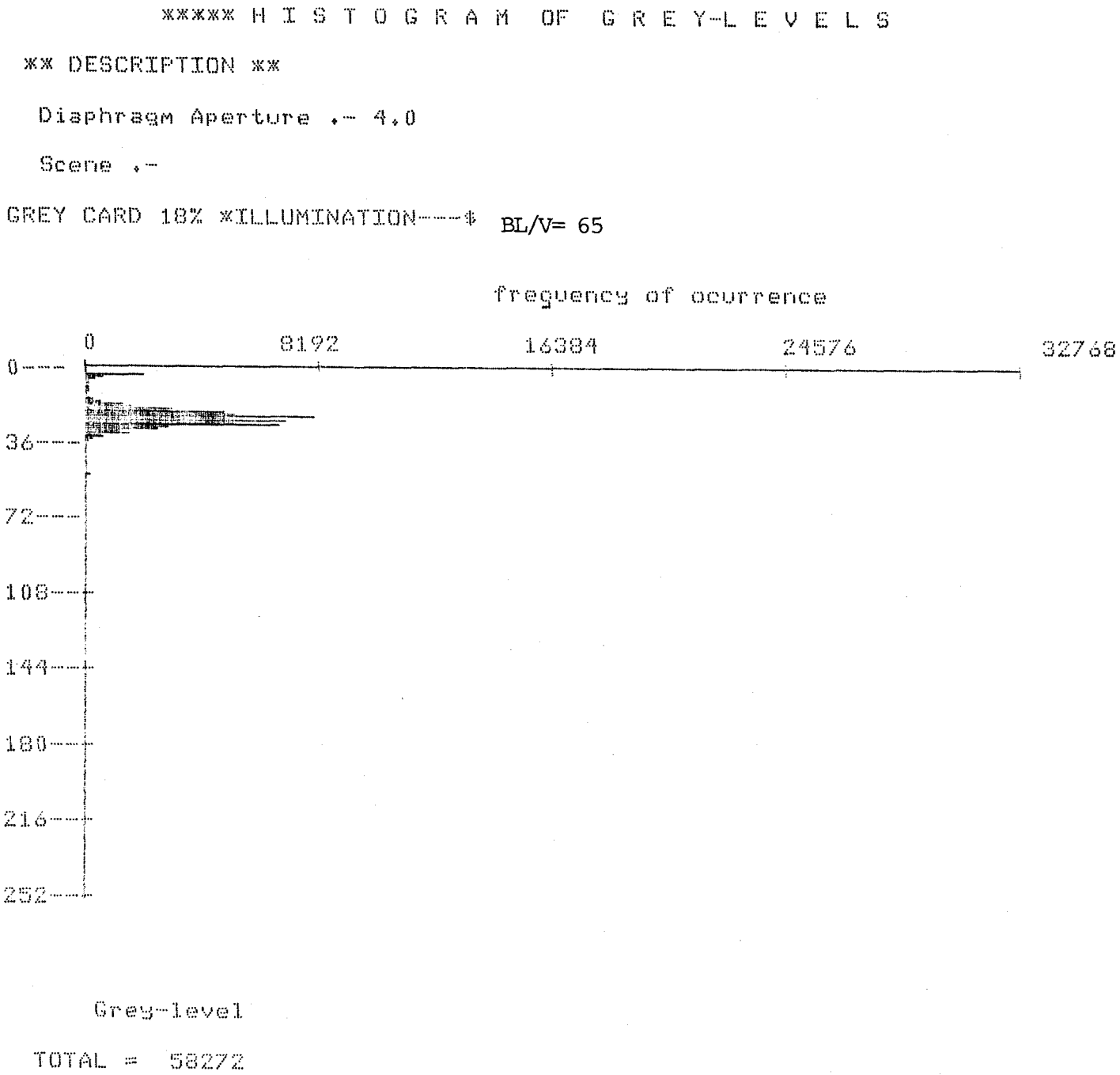


Figure 5.6 VRHG performance (scheme 6) .

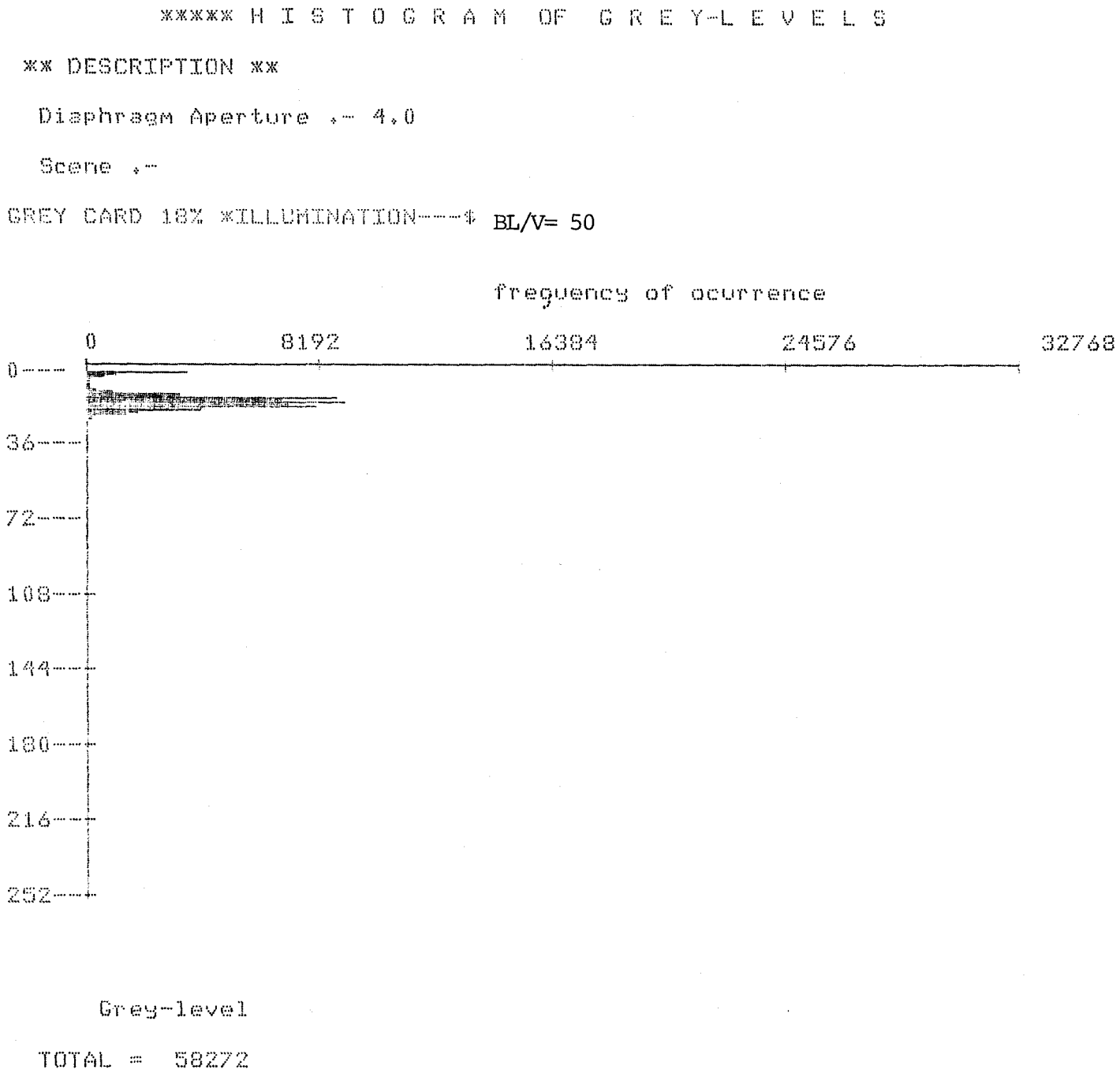


Figure 5.7 VRHG performance (scheme 7) .

throughout the matrix array visual sensor.

The dynamic range was measured by taking histograms of grey-levels of the same scene and ambient light for all diaphragm apertures in the camera. Normal sensitivity operation of the ALS was used all the time and ambient light conditions were that of a conventional office room (EV= 5.5.).

Care was taken to keep these light conditions stable during the time of the measurements. Eight diaphragm apertures (f number) are available in the camera: 1.8, 2.8, 4.0, 5.6, 8.0, 11, 16, and 22 .

Scaled graphs of these results are shown in figures 5.8 to 5.15. The smaller the " f " number the larger the amount of light entering the camera. From the graphs it can be observed that the less light, the more narrow the histogram width obtained and the smaller the number of grey-levels to represent the scene used.

In any case a grey-level greater than 128 was never obtained, which indicates that only a seven bit grey scale is used by the camera.

Figures 5.5 to 5.7 illustrate that a constant number of grey-levels is always present in the most black region of the grey scale. This is due to a physical failure in the sensor array of the camera which can also be observed by monitoring the analog video signal.

To test the sensitivity throughout the matrix array of the sensor transducer, a scheme like the one shown in the figure 5.16 was used.

One hundred histograms of an object (1 % of the total work area) on a white background were taken and for one hundred

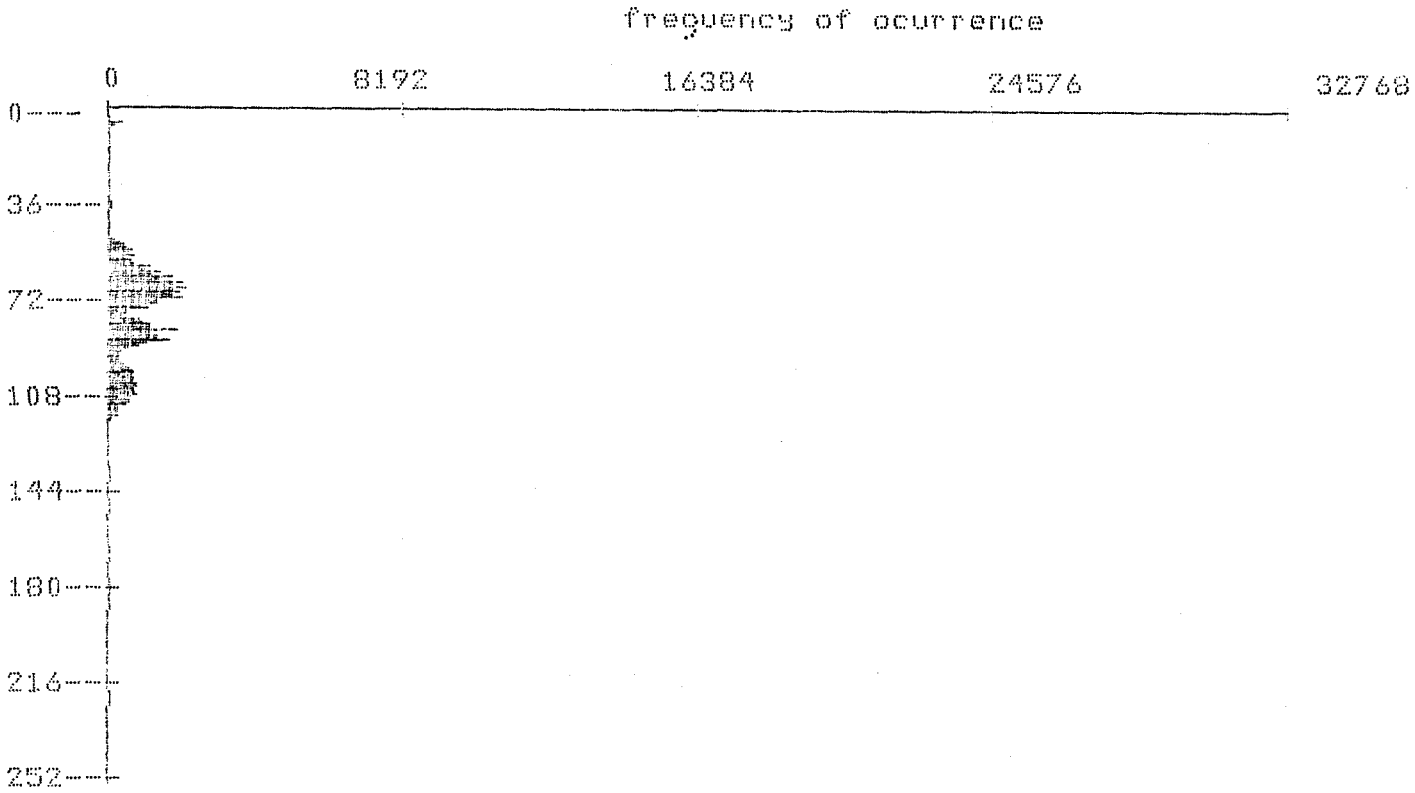
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture .- 1.8

Scene .-

BACK-LIGHTED WHITE ACRYLIC AND TWO BLOCKS *ILLUMINATION--$ BL/V=90



TOTAL = 57030

Figure 5.8 Dynamic range f=1.8

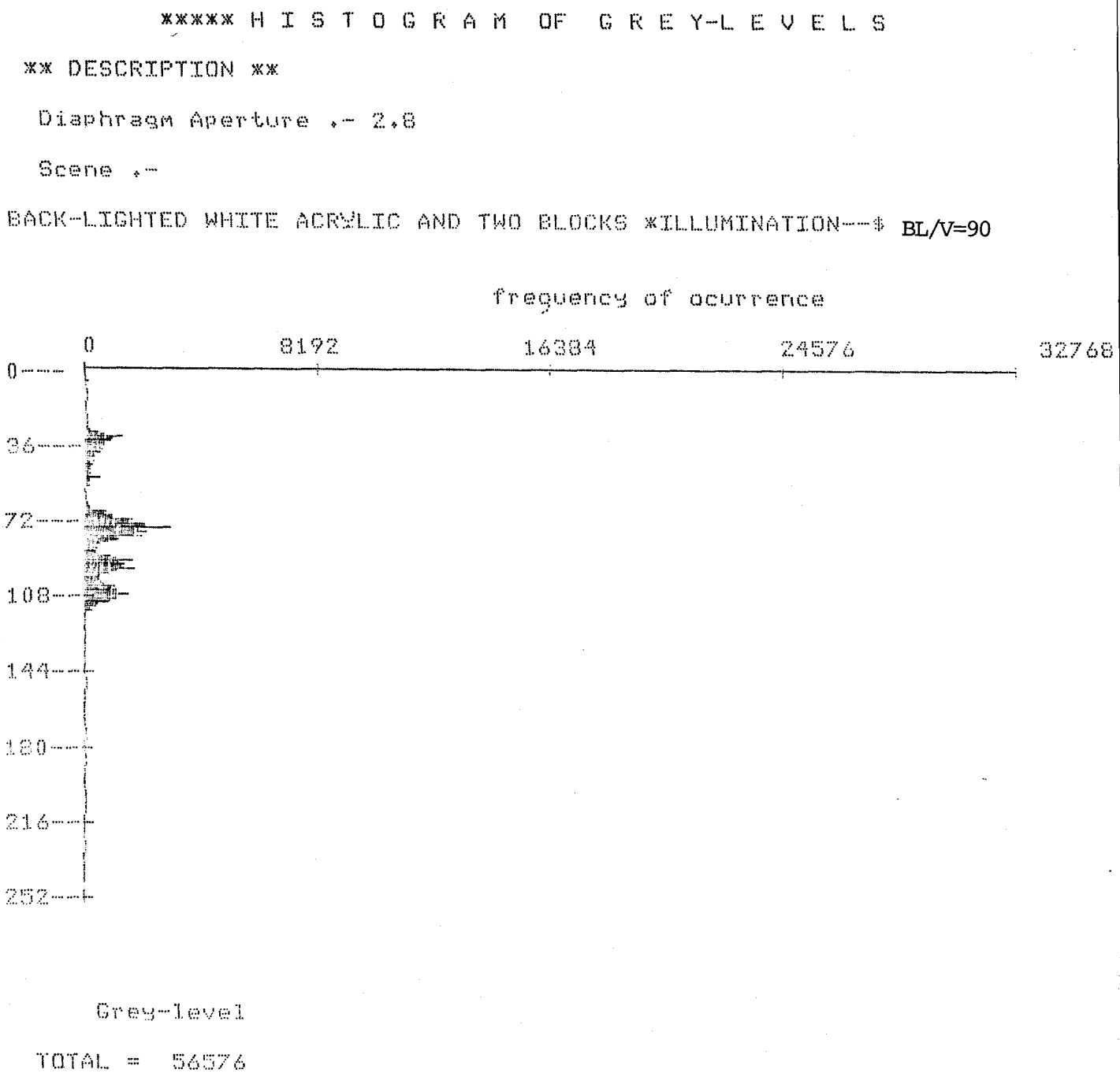


Figure 5.9 Dynamic range f=2.8

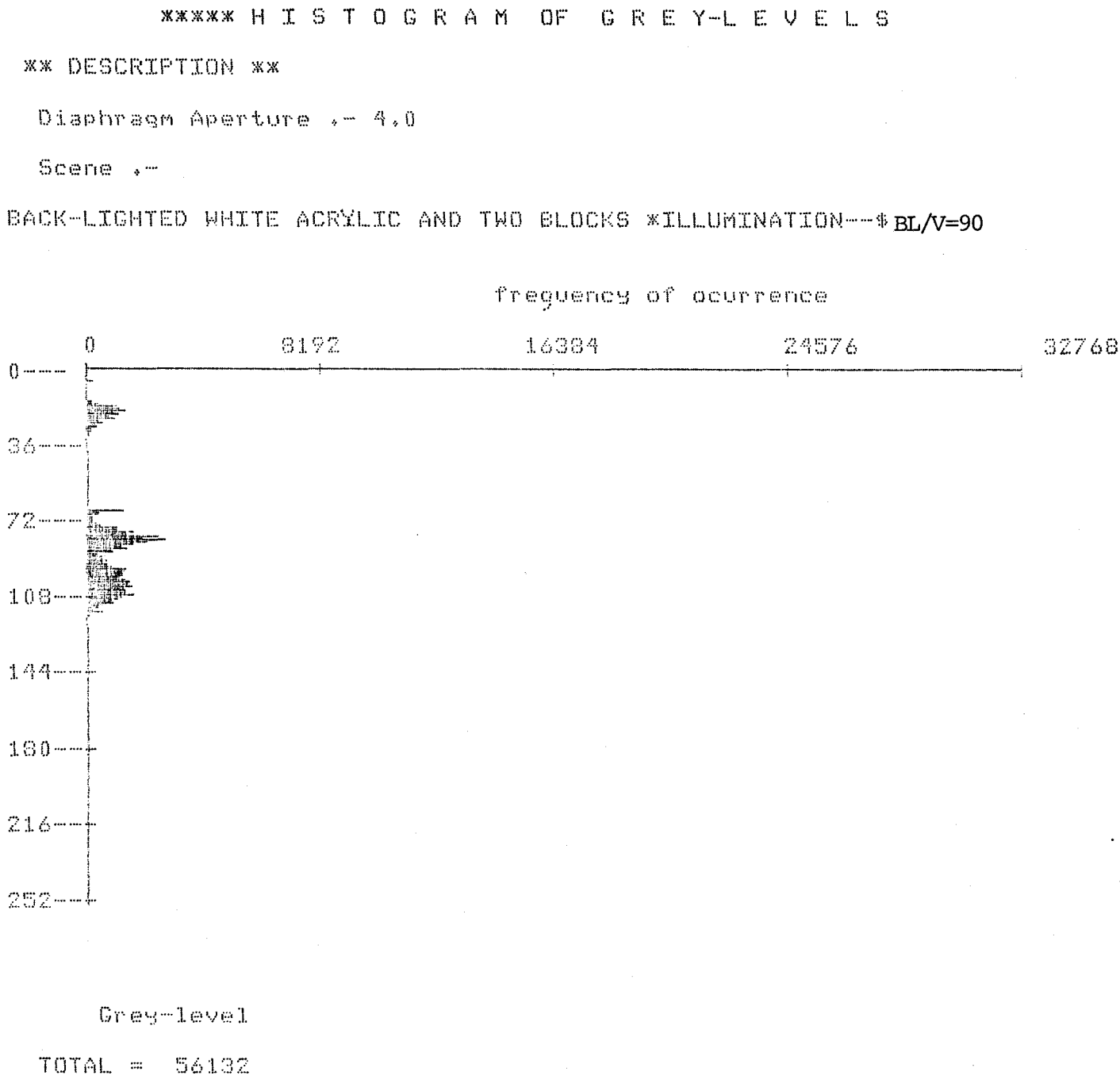


Figure 5.10 Dynamic range f=4.0

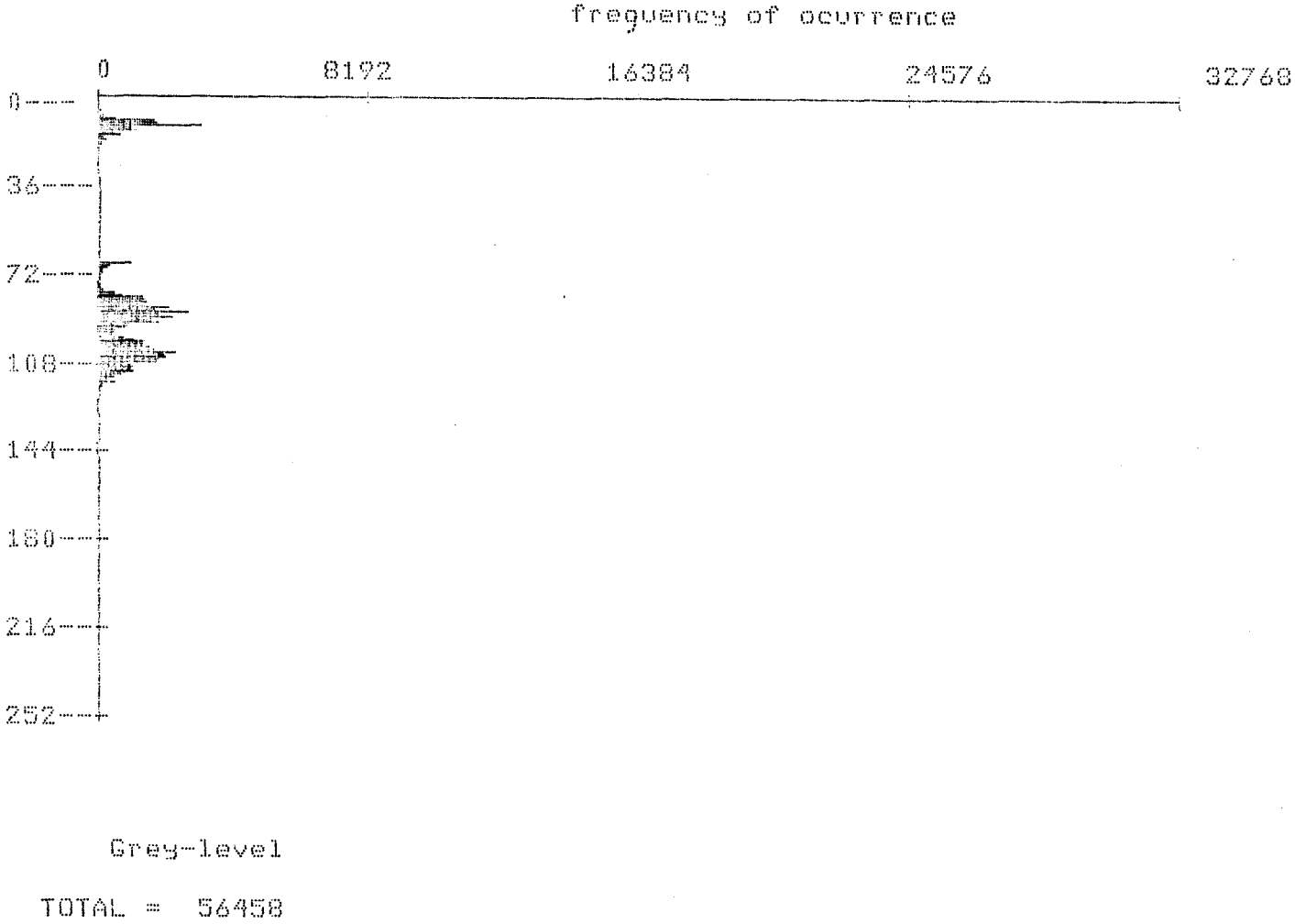
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture ,- 5.6

Scene ,-

BACK-LIGHTED WHITE ACRYLIC AND TWO BLOCKS *ILLUMINATION----$ BL/V=90



TOTAL = 56458

Figure 5.11 Dynamic range f=5.6

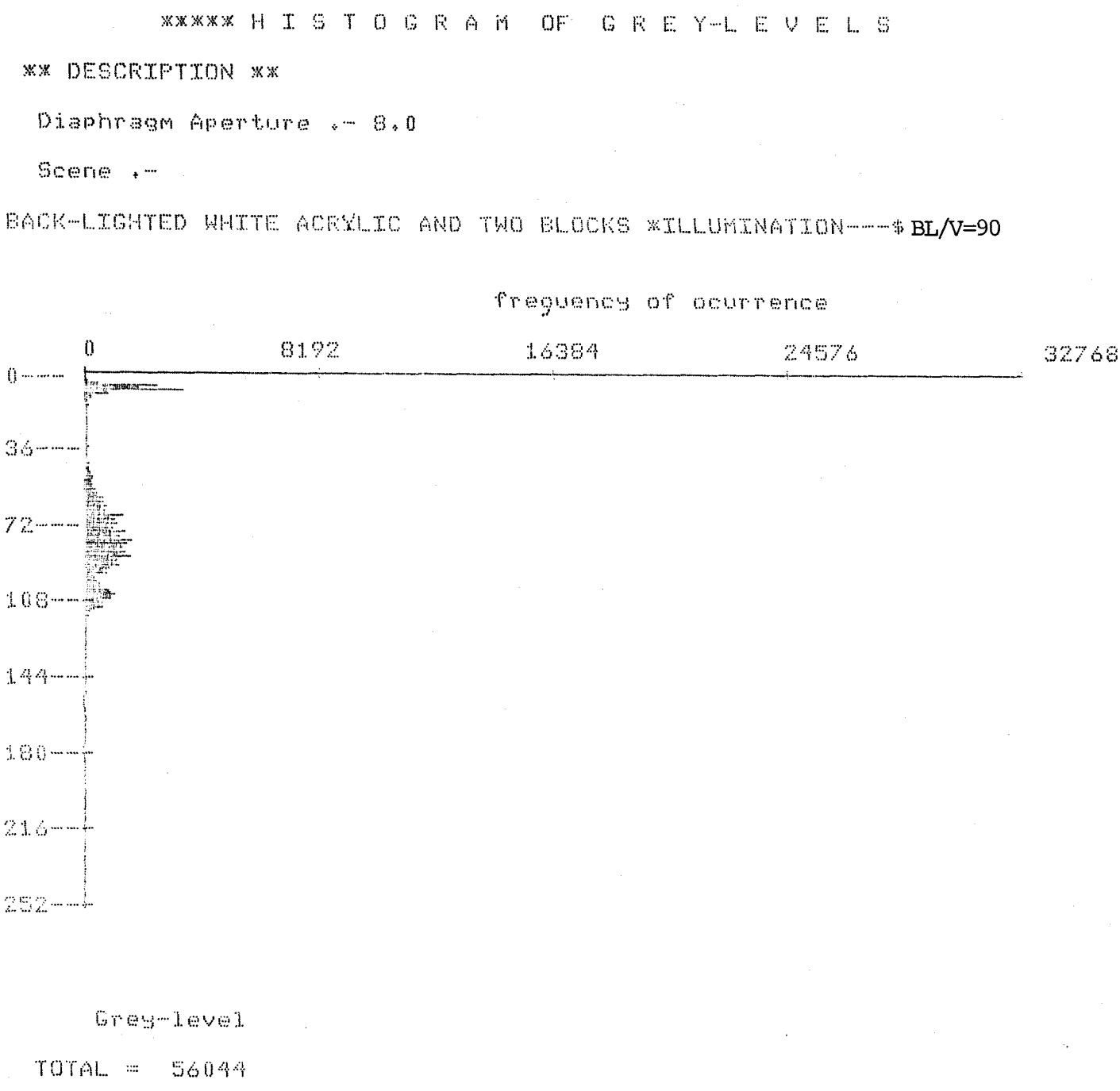


Figure 5.12 Dynamic range f=8.0

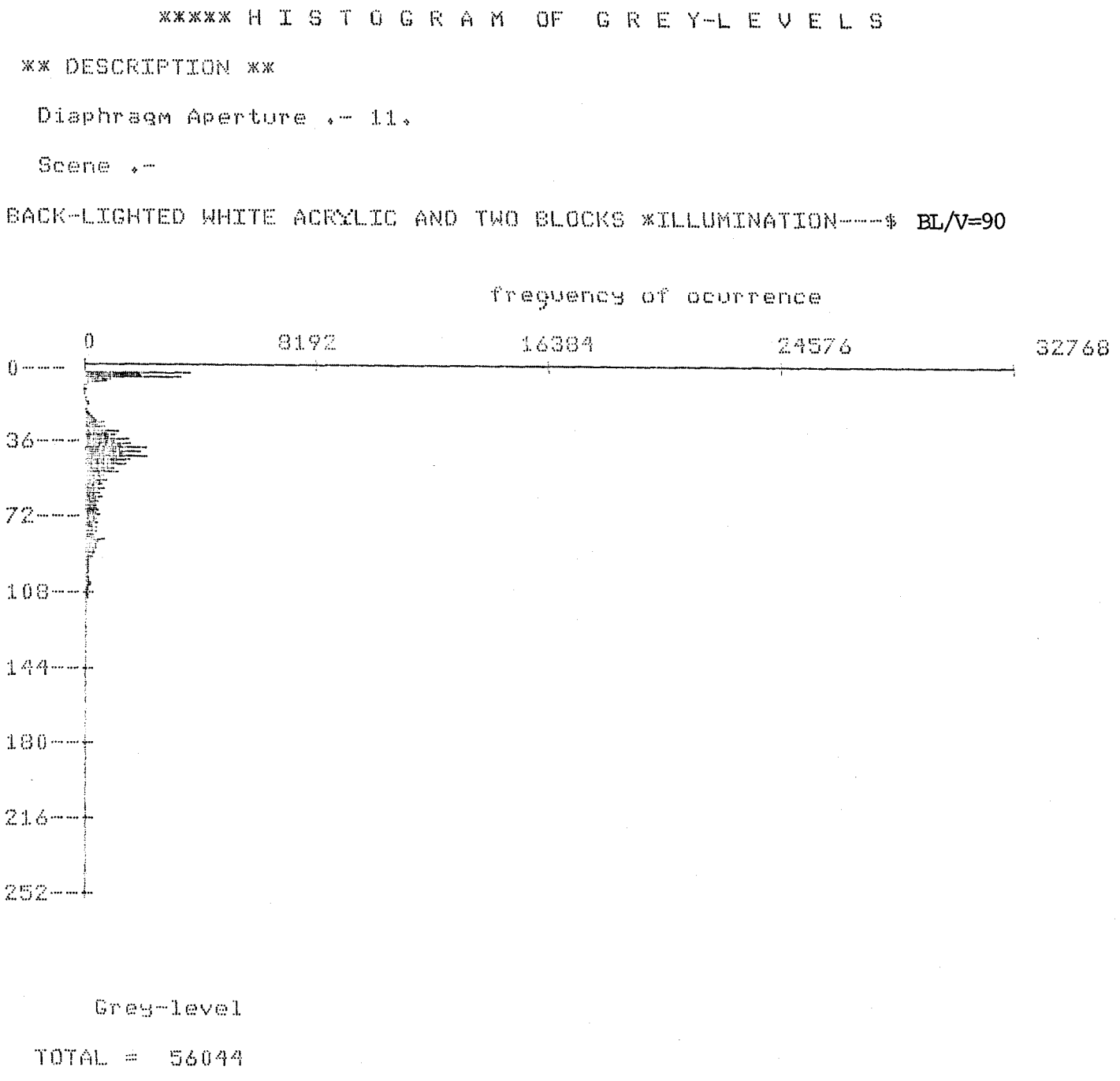


Figure 5.13 Dynamic range f=11.0

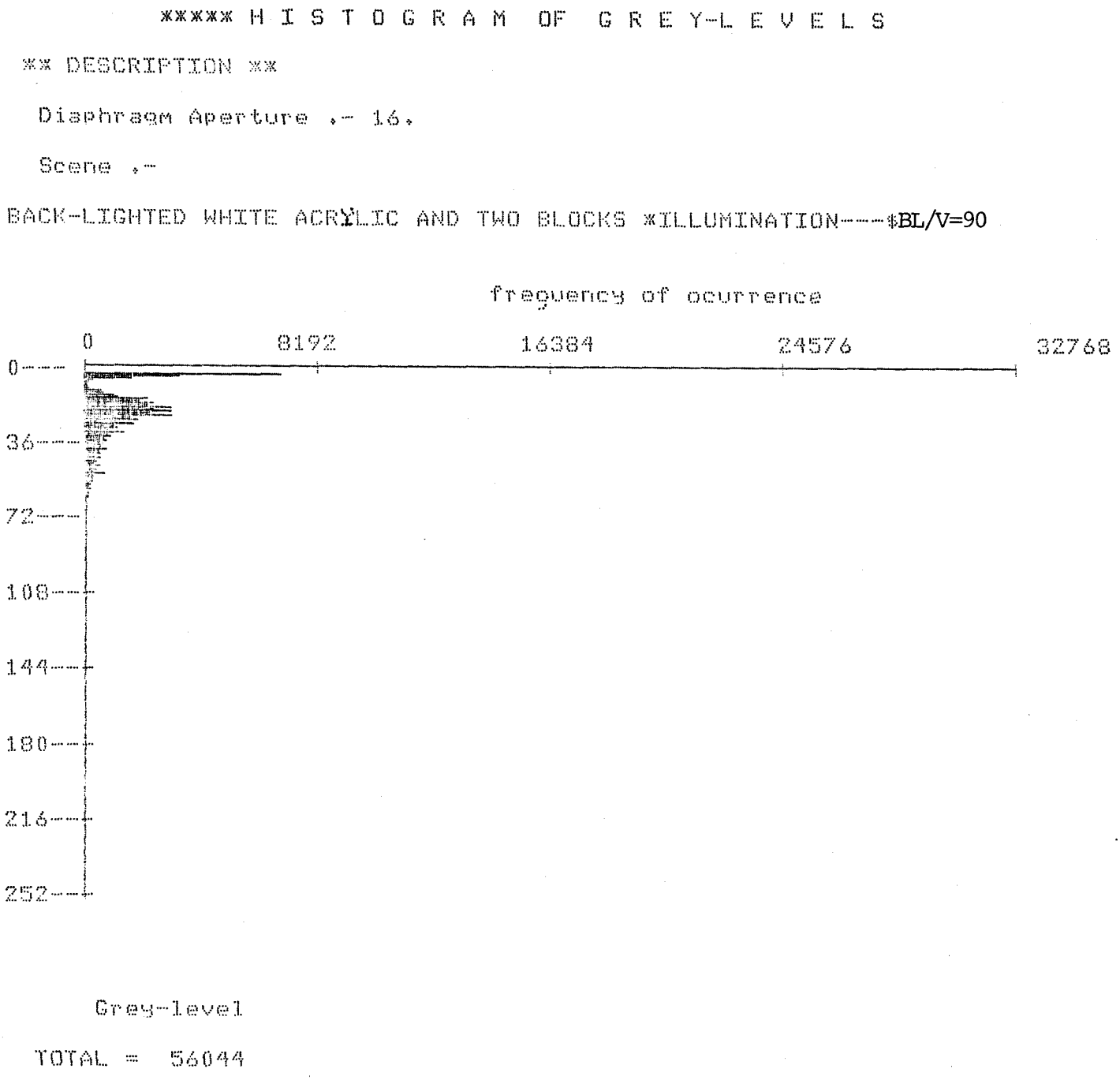


Figure 5.14 Dynamic range f=16.0

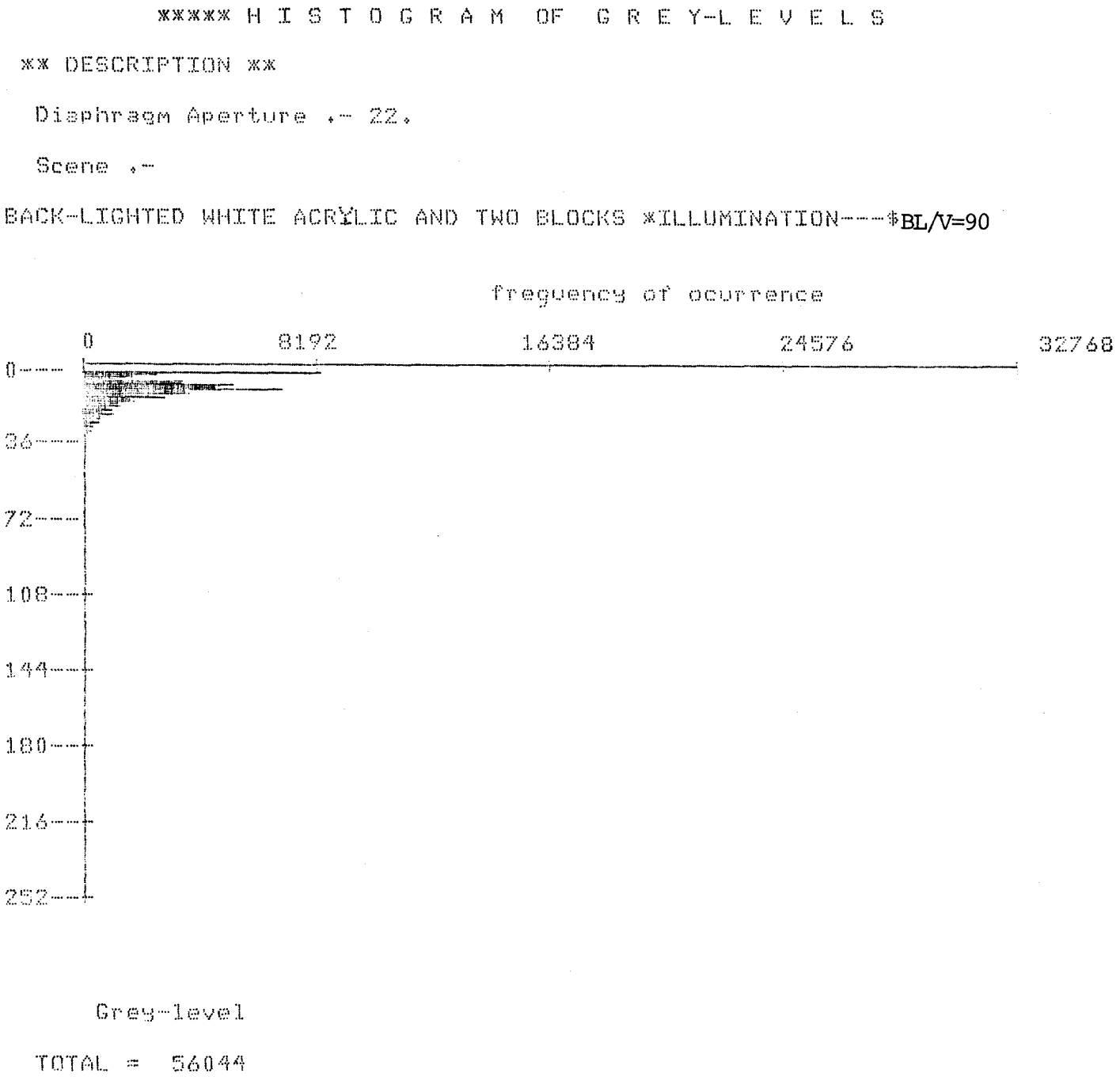


Figure 5.15 Dynamic range f=22.0

HISTOGRAM OF GRAY-LEVELS DATA :-

Minimum Grey-level	0
Maximum Grey-level	114
Histogram Width	114
Black upper threshold	15
Black lower threshold	0
White upper threshold	114
White lower threshold	20
Black Width	5
White width	86
Valley width	3
Maximum black value at Grey-level	12
Maximum white value at Grey-level	100
THRESHOLD USED:	895

$$Mn = \left[\frac{W_w}{B_w} - V_w \right] + H_w$$

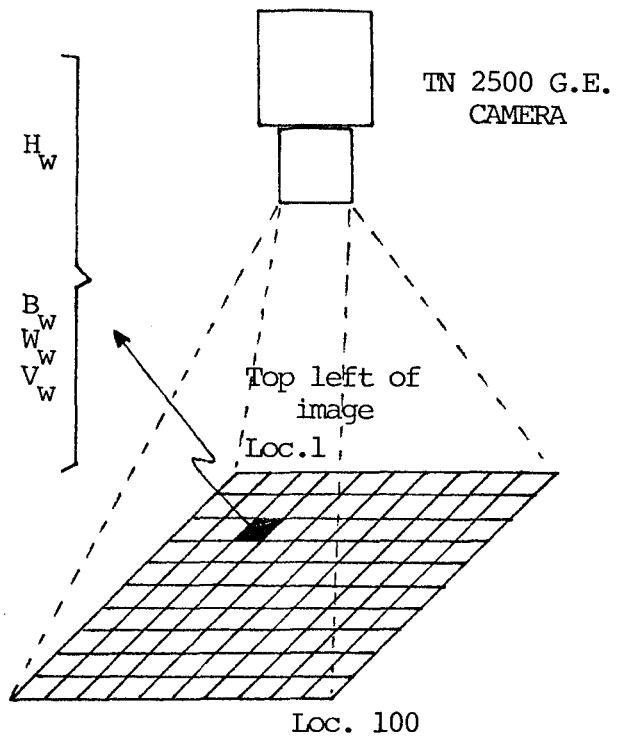


Figure 5.16 Scheme to test uniformity of the visual transducer

different locations. For each histogram the parameters defined in chapter IV were related as:

$$\left[\frac{\text{white width}}{\text{black width}} - \text{valley width} \right] + \text{histogram width}$$

to obtain a merit number of each scene. Results are shown in figure 5.17.

5.3 Adaptive range of the ALS

The ALS can provide the vision system with adaptive lighting capability within a specified range of illumination. This range is covered by the four steps of sensitivity (steps 1, 2 , 3 and 4).

Figures 5.18 to 5.33 show the particular range of each sensitivity step, and the total adaptive range is shown in figure 5.34 .

Each figure shows the illumination conditions and the description of the analyzed scene. The BL/V= X number indicates: back-lighted scene and illumination of it as perceived by the camera.

5.4 Adaptive process of the ALS

A complete adaptive process for a low light condition scene (within the specified range) was recorded. Figures 5.35 to 5.38 show the four different distributions of light for the four different sensitivity steps. In the adaptive process the histogram parameters were calculated from the histogram of sensitivity step 3, where a substantial improvement in the histogram shape can be observed in reference with the histogram of sensitivity step 1.

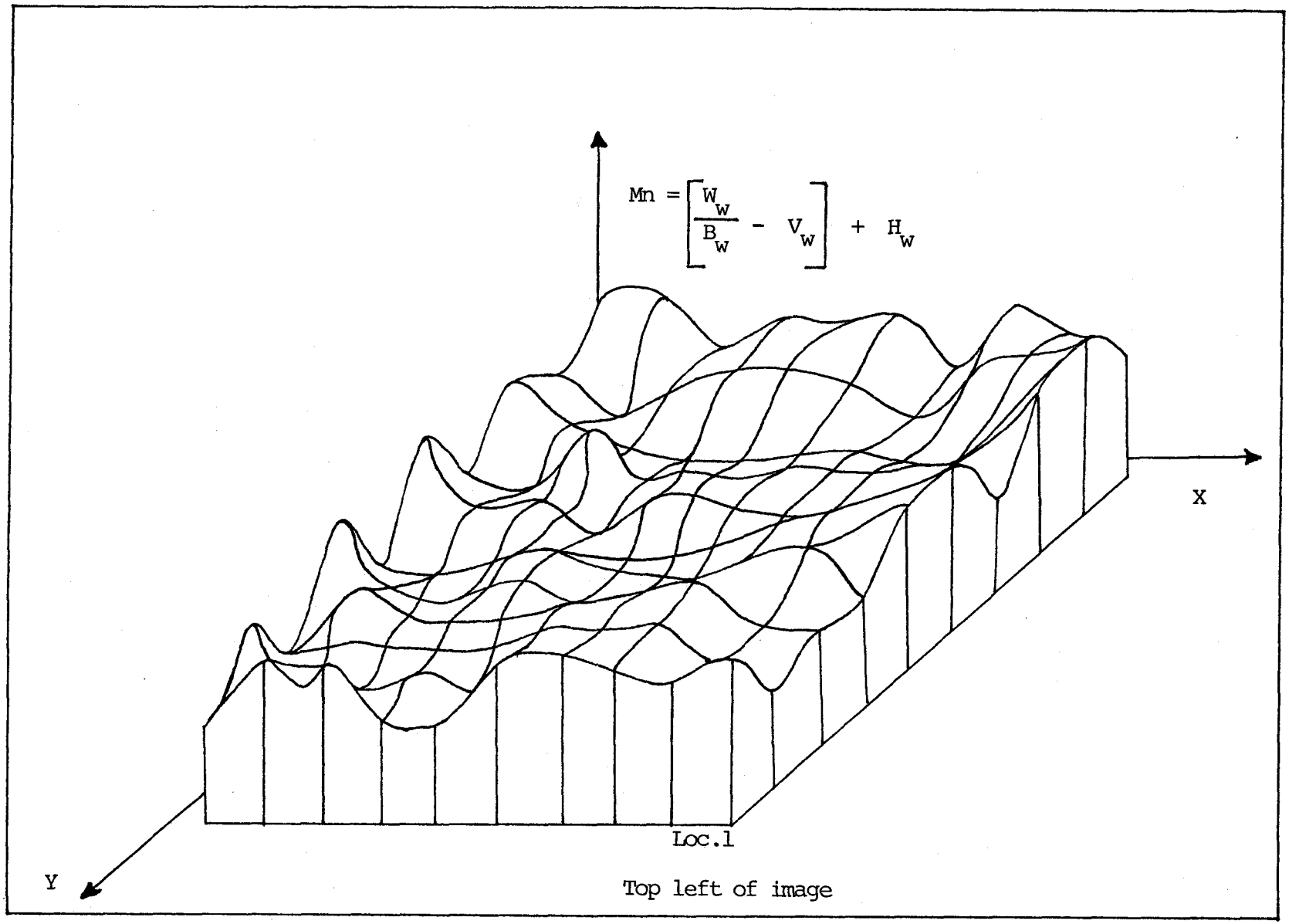


Figure 5.17 Uniformity of the visual transducer

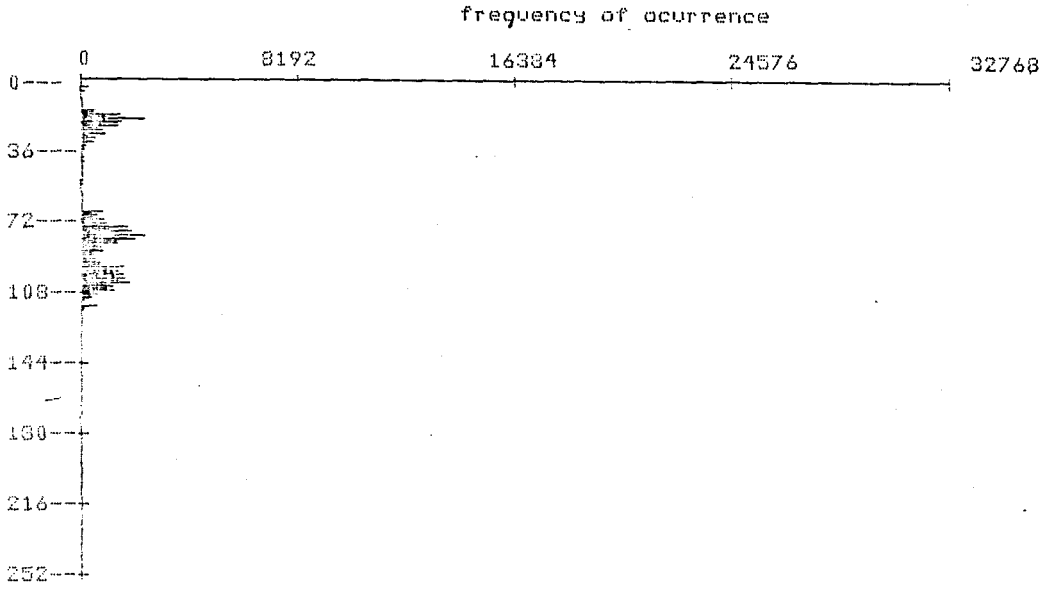
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture .- 4.0

Scene .-

BL/V=100 ILLUMINATION---$ Sensitivity 0



TOTAL = 56260

HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	114
Histogram Width	114
Black upper threshold	19
Black lower threshold	0
White upper threshold	114
White lower threshold	55
Black Width	5
White width	32
Valley width	53
Maximum black value at Grey-level	16
Maximum white value at Grey-level	76
THRESHOLD USED:	075

Figure 5.18 Upper range sensitivity 0 .

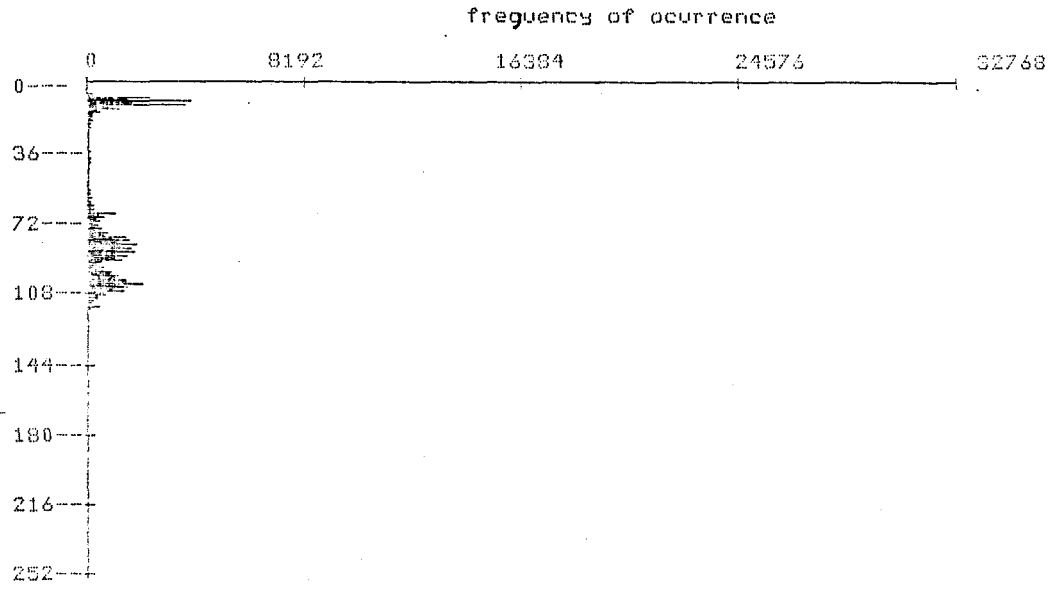
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture .- 4.0

Scene .-

BL/V=70 ILLUMINATION ---$ Sensitivity 0



TOTAL = 19834

HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	2
Maximum Grey-level	114
Histogram Width	112
Black upper threshold	9
Black lower threshold	2
White upper threshold	114
White lower threshold	46
Black Width	5
White width	40
Valley width	55
Maximum black value at Grey-level	6
Maximum white value at Grey-level	100
THRESHOLD USED:	895

Figure 5.19 Medium range/sensitivity 0

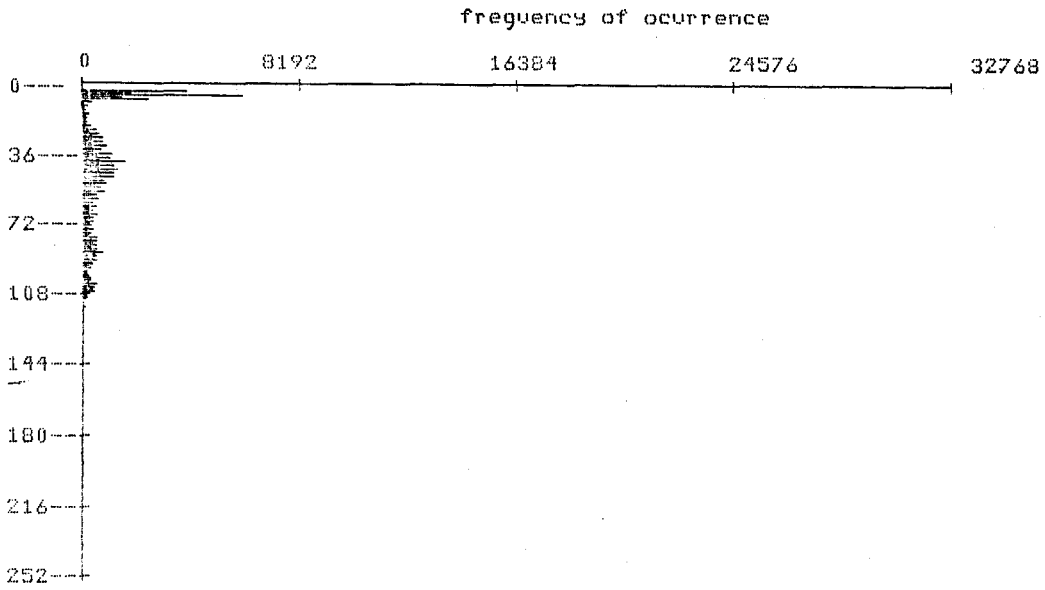
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture .- 4.0

Scene .-

BL/V=50 ILLUMINATION --- SENSITIVITY 0 $



TOTAL = 19436

HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	112
Histogram Width	112
Black upper threshold	5
Black lower threshold	0
White upper threshold	112
White lower threshold	25
Black Width	5
White width	12
Valley width	27
Maximum black value at Grey-level	2
Maximum white value at Grey-level	36
THRESHOLD USED:	895

Figure 5.20 Lower range/sensitivity 0

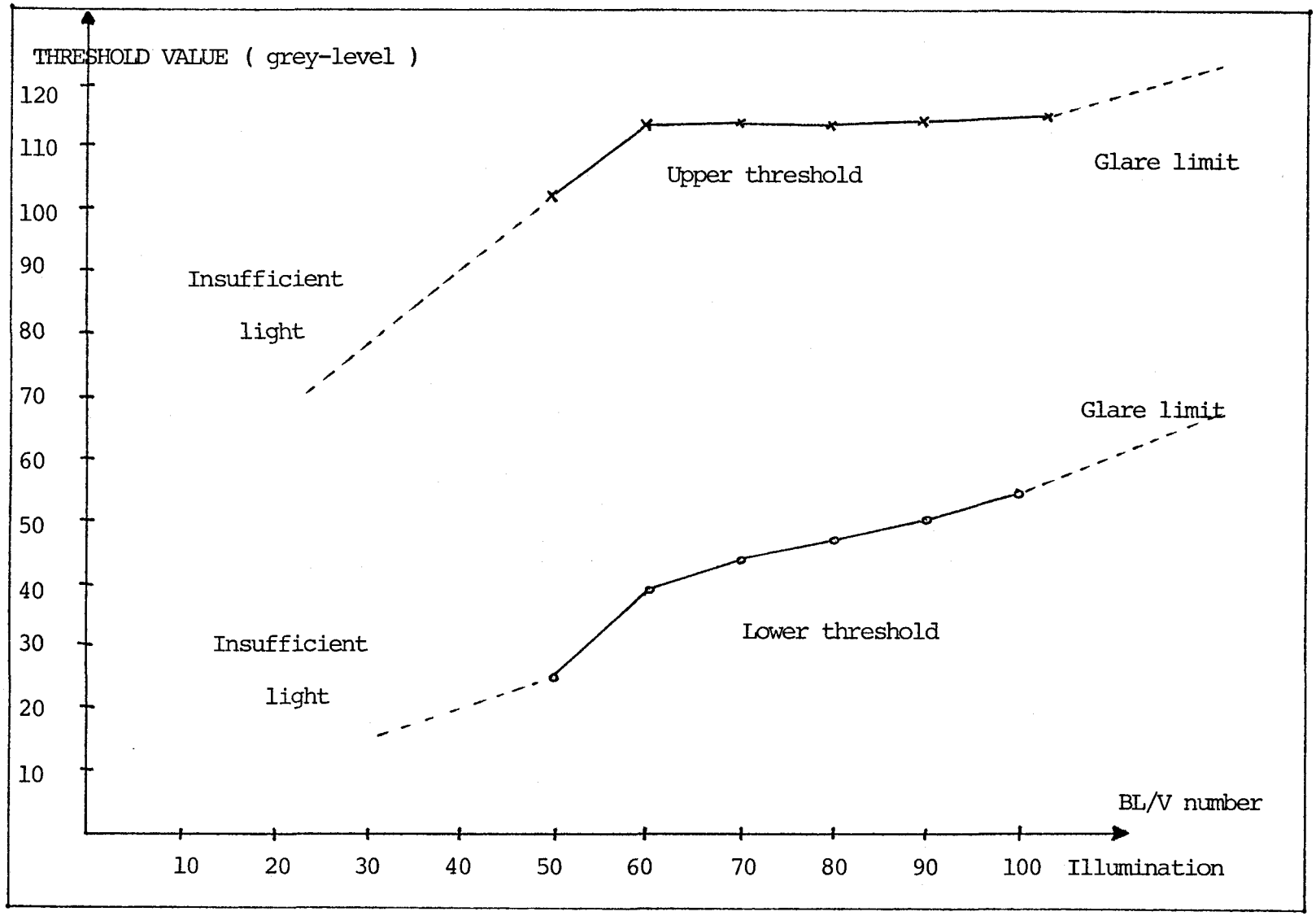


Figure 5.21 Adaptive range/ sensitivity 0

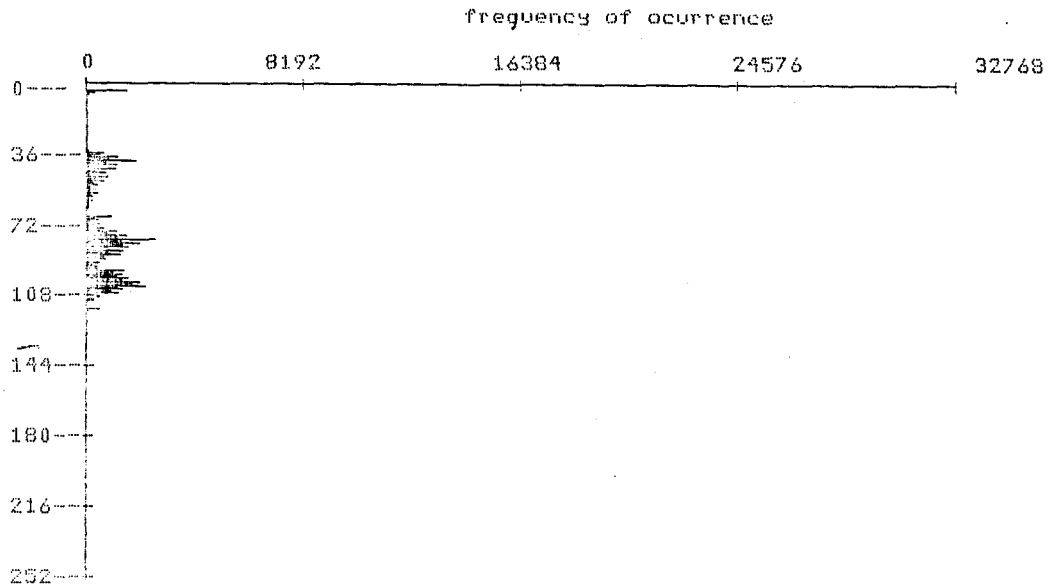
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture .- 4.0

Scene .-

BL/V=90 ILLUMINATION --- SENSITIVITY 1 $



TOTAL = 21664

HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	114
Histogram Width	114
Black upper threshold	5
Black lower threshold	0
White upper threshold	114
White lower threshold	26
Black Width	5
White width	70
Valley width	29
Maximum black value at Grey-level	0
Maximum white value at Grey-level	76
THRESHOLD USED:	895

Figure 5.22 Upper range/sensitivity 1

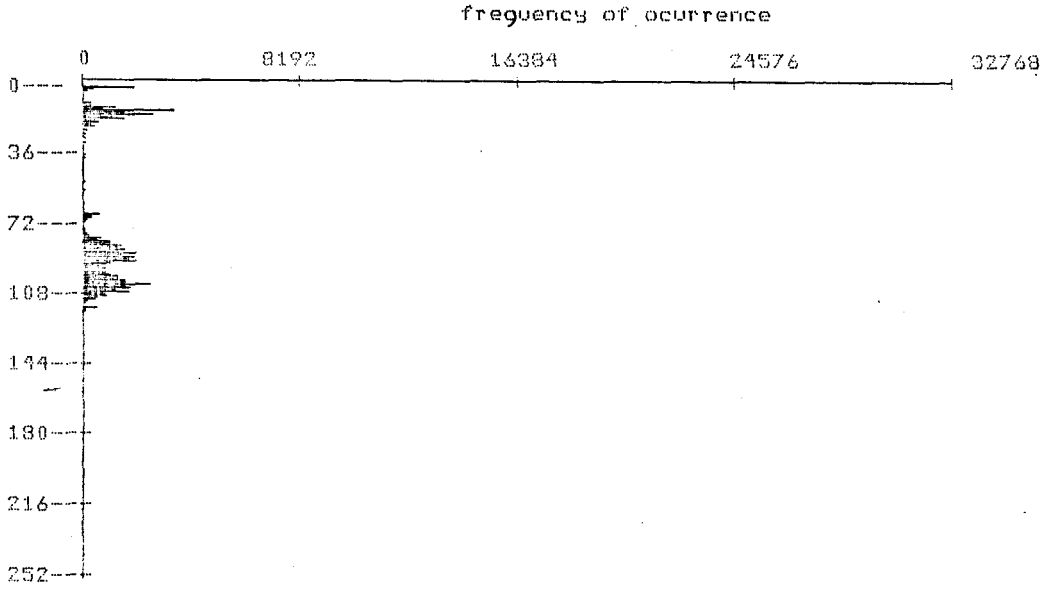
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture .- 4.0

Scene .-

BL/V=60 ILUUMINATION ---- SENSITIVITY 1 $



TOTAL = 21664

HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	114
Histogram Width	114
Black upper threshold	5
Black lower threshold	0
White upper threshold	114
White lower threshold	11
Black Width	5
White width	94
Valley width	5
Maximum black value at Grey-level	0
Maximum white value at Grey-level	12
THRESHOLD USED:	895

Figure 5.23 Medium range / sensitivity 1

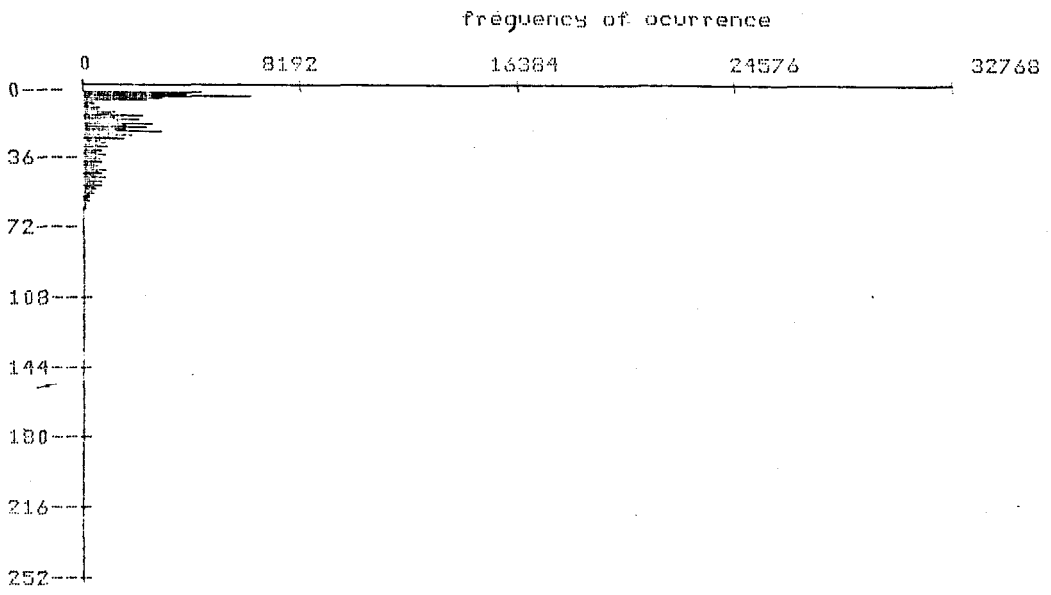
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture .- 4.0

Scene .-

BL/V=30 ILLUMINATION ---- SENSITIVITY 1 $



TOTAL = 21664

HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	61
Histogram Width	61
Black upper threshold	5
Black lower threshold	0
White upper threshold	61
White lower threshold	11
Black Width	5
White width	14
Valley width	5
Maximum black value at Grey-level	2
Maximum white value at Grey-level	20
THRESHOLD USED:	895

Figure 5.24 Lower range/sensitivity 1

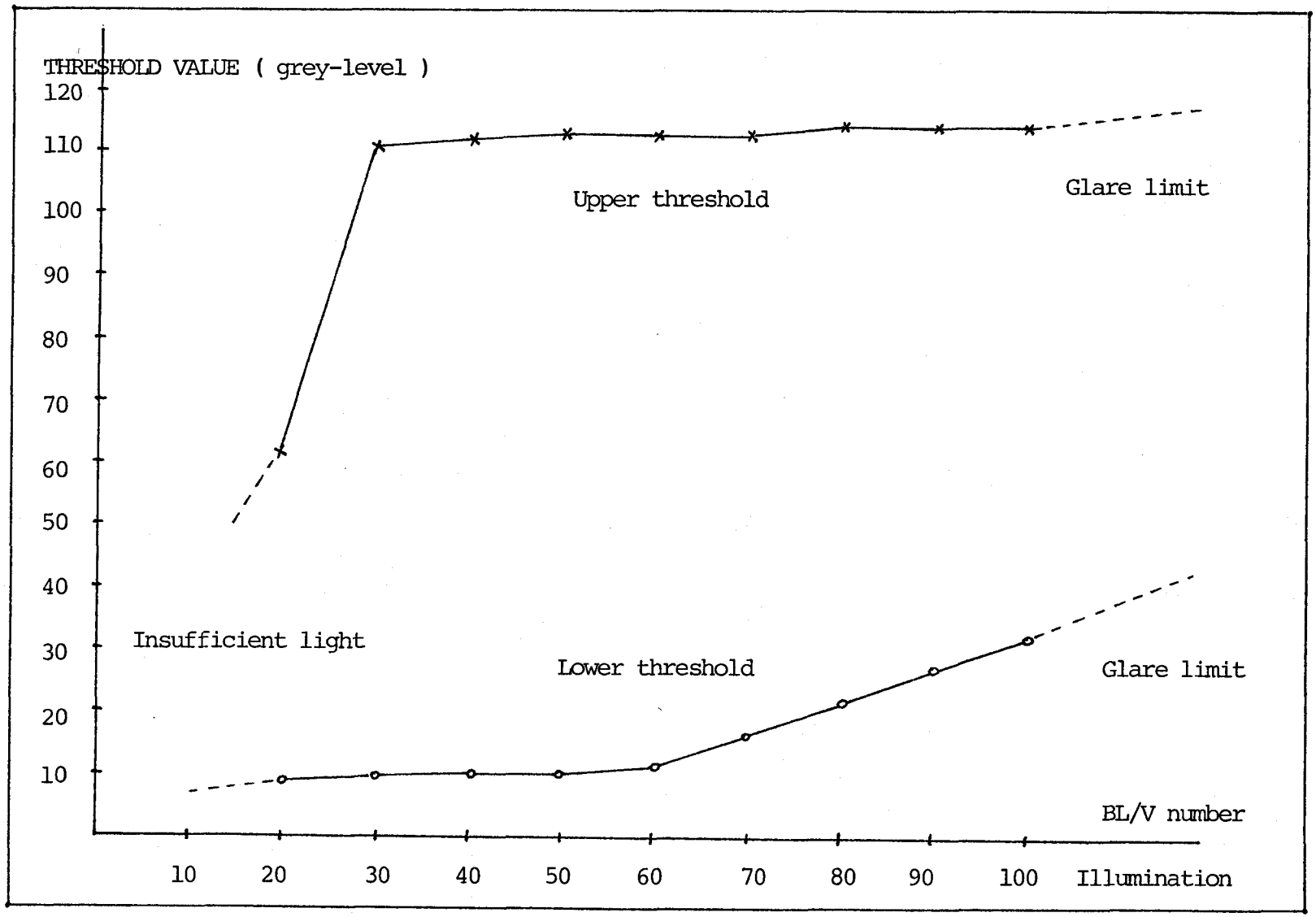
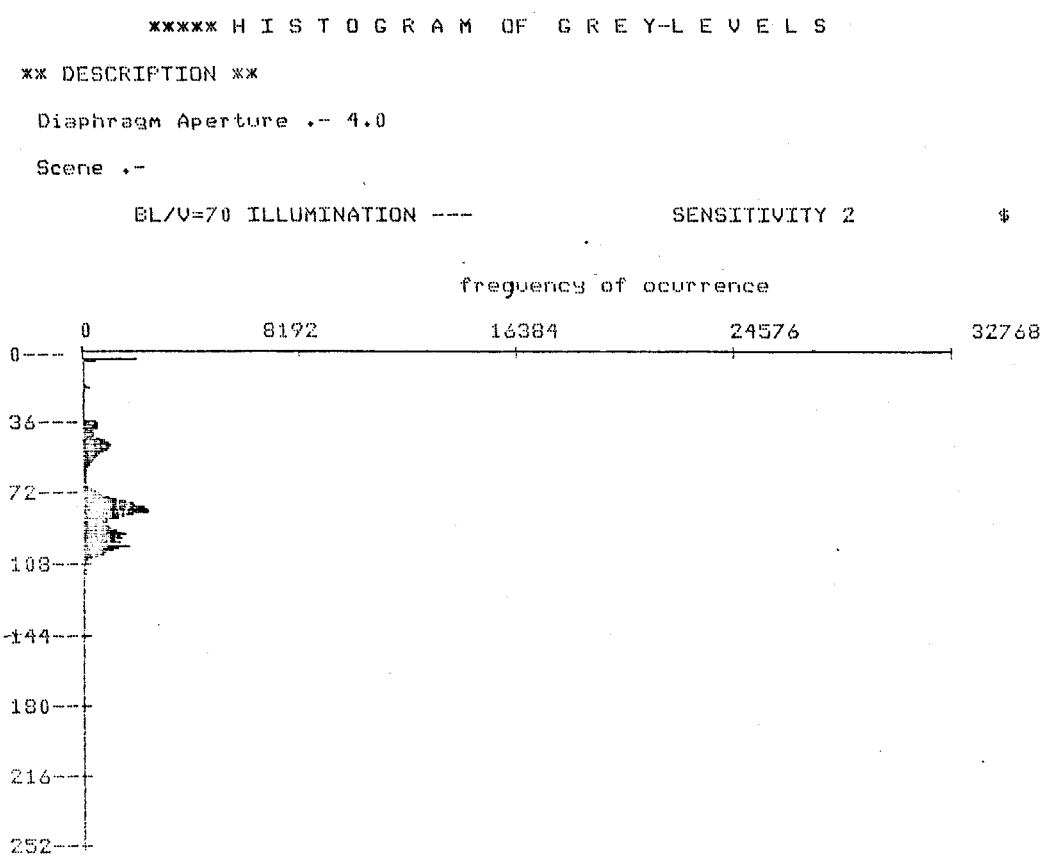


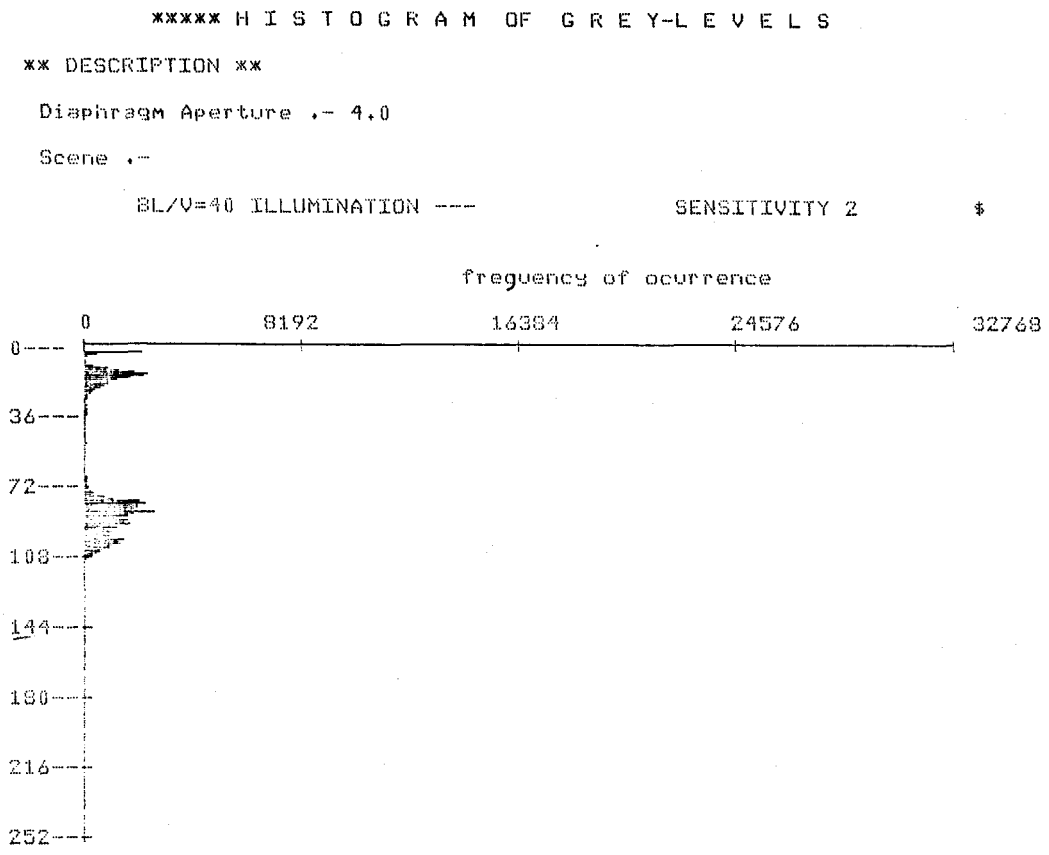
Figure 5.25 Adaptive range/ sensitivity 1



HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	110
Histogram Width	110
Black upper threshold	5
Black lower threshold	0
White upper threshold	110
White lower threshold	32
Black Width	5
White width	54
Valley width	39
Maximum black value at Grey-level	0
Maximum white value at Grey-level	79
THRESHOLD USED:	895

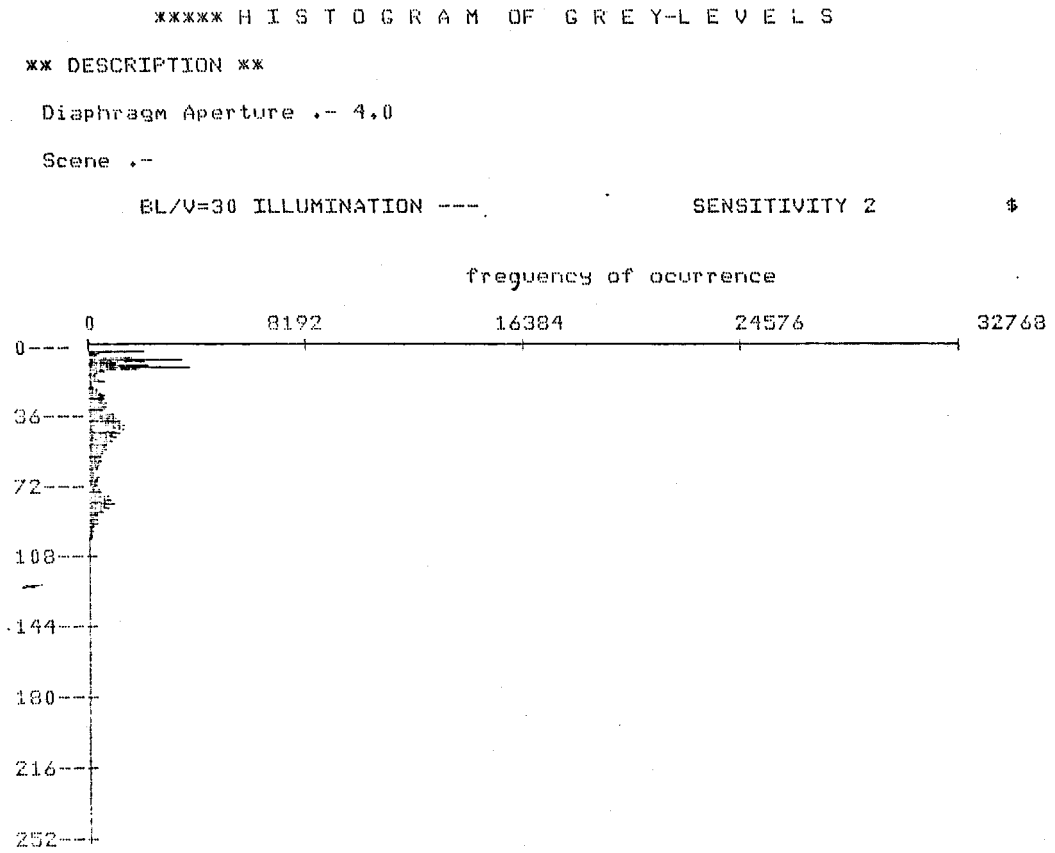
Figure 5.26 Upper range / sensitivity 2



HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	106
Histogram Width	106
Black upper threshold	5
Black lower threshold	0
White upper threshold	106
White lower threshold	10
Black Width	5
White width	90
Valley width	4
Maximum black value at Grey-level	0
Maximum white value at Grey-level	82
THRESHOLD USED:	895

Figure 5.27 Medium range / sensitivity 2



HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	98
Histogram Width	98
Black upper threshold	5
Black lower threshold	0
White upper threshold	98
White lower threshold	10
Black Width	5
White width	36
Valley width	3
Maximum black value at Grey-level	4
Maximum white value at Grey-level	8
THRESHOLD USED:	895

Figure 5.28 Lower range / sensitivity 2

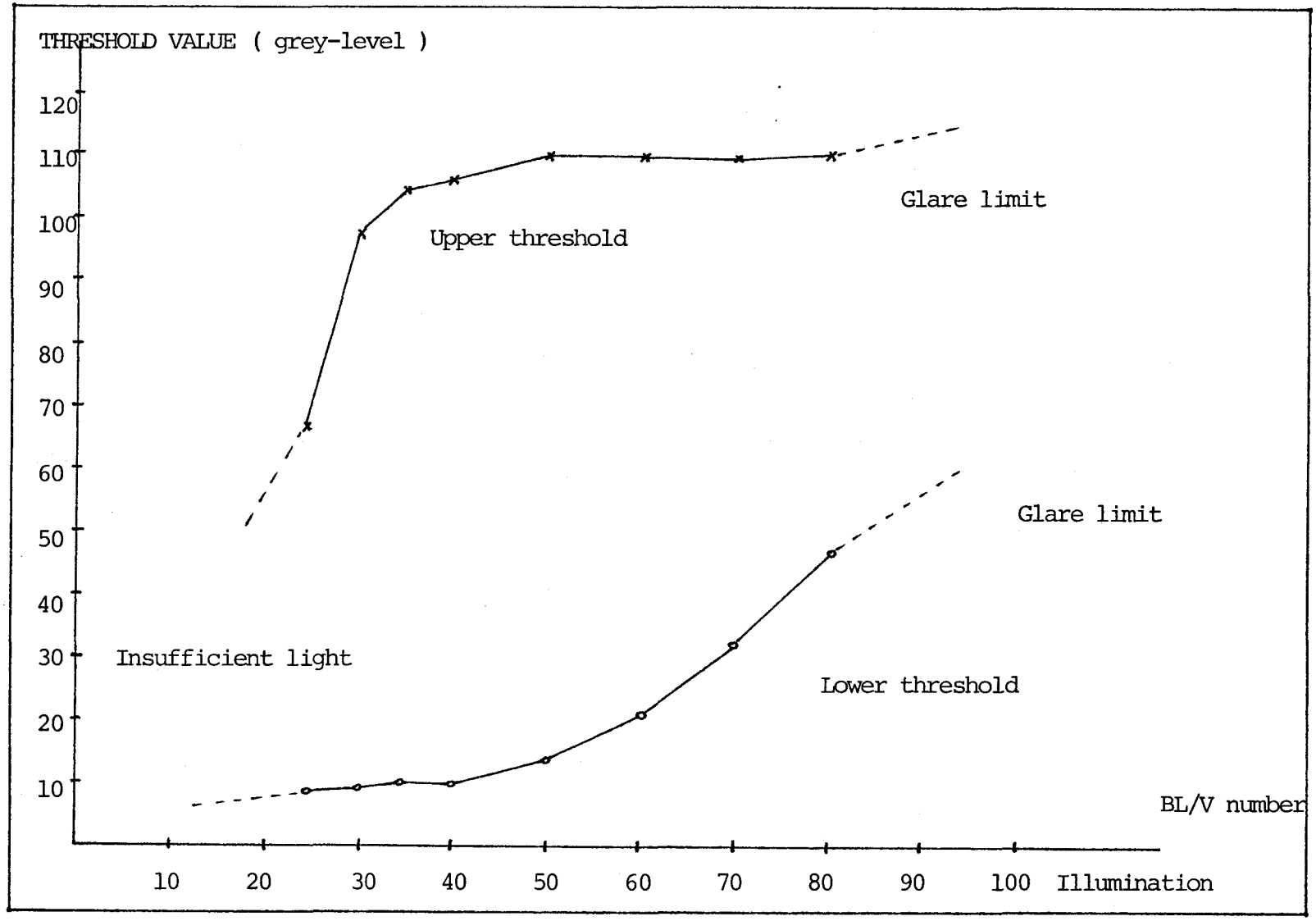


Figure 5.29 Adaptive range / sensitivity 2

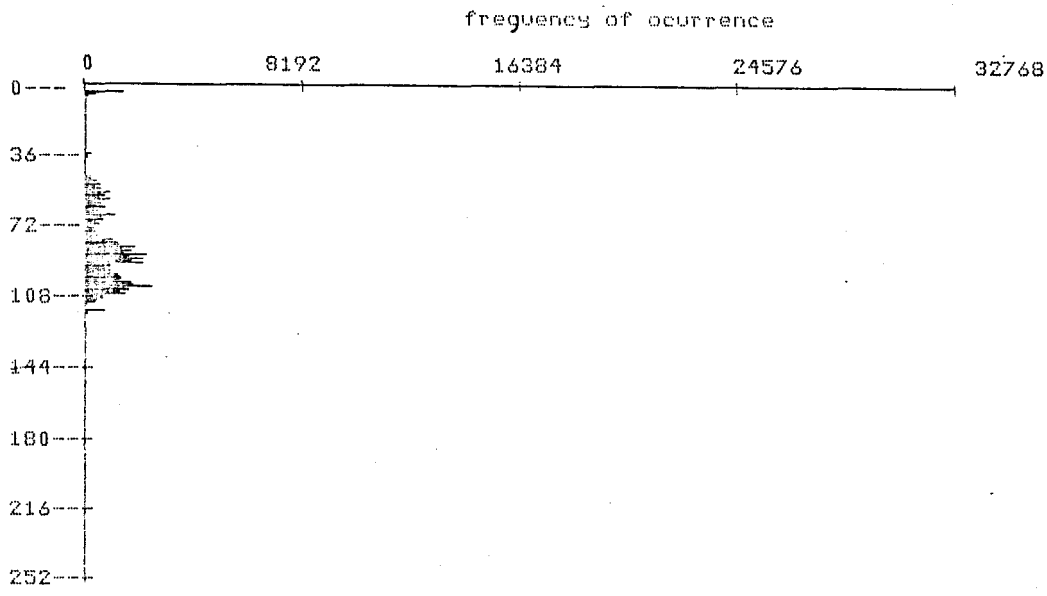
***** H I S T O G R A M OF G R E Y-L E V E L S

** DESCRIPTION **

Diaphragm Aperture .- 4.0

Scene .-

BL/V=60 ILLUMINATION --- SENSITIVITY 3 $



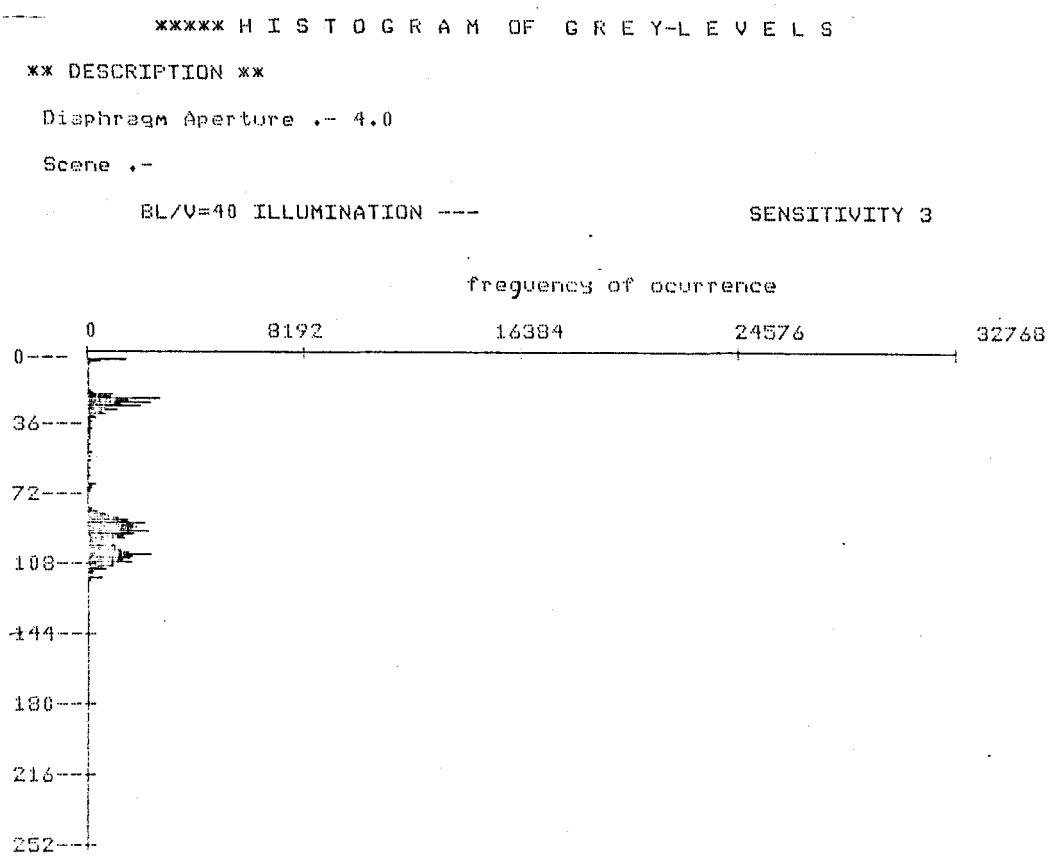
Grey-level

TOTAL = 21664

HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level ---------- 0
Maximum Grey-level ---------- 114
Histogram Width ---------- 114
Black upper threshold ---------- 5
Black lower threshold ---------- 0
White upper threshold ---------- 114
White lower threshold ---------- 45
Black Width ---------- 5
White width ---------- 40
Valley width ---------- 59
Maximum black value at Grey-level - 0
Maximum white value at Grey-level-- 100
THRESHOLD USED: ---------- 895

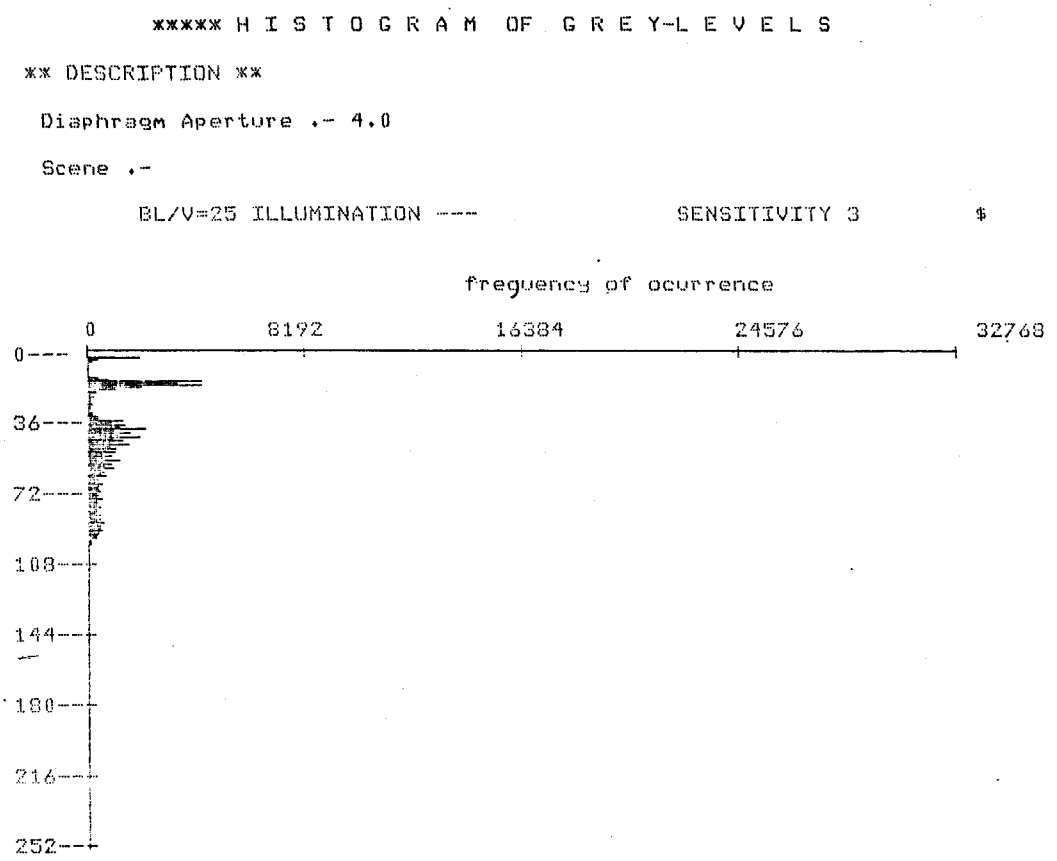
Figure 5.30 Upper range / sensitivity 3



HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	114
Histogram Width	114
Black upper threshold	5
Black lower threshold	0
White upper threshold	114
White lower threshold	17
Black Width	5
White width	84
Valley width	15
Maximum black value at Grey-level	0
Maximum white value at Grey-level	20
THRESHOLD USED:	895

Figure 5.31 Medium range / sensitivity 3



HISTOGRAM OF GRAY-LEVELS DATA .-

Minimum Grey-level	0
Maximum Grey-level	112
Histogram Width	112
Black upper threshold	5
Black lower threshold	0
White upper threshold	112
White lower threshold	12
Black Width	5
White width	40
Valley width	7
Maximum black value at Grey-level	0
Maximum white value at Grey-level	14
THRESHOLD USED:	095

Figure 5.32 Lower range / sensitivity 3

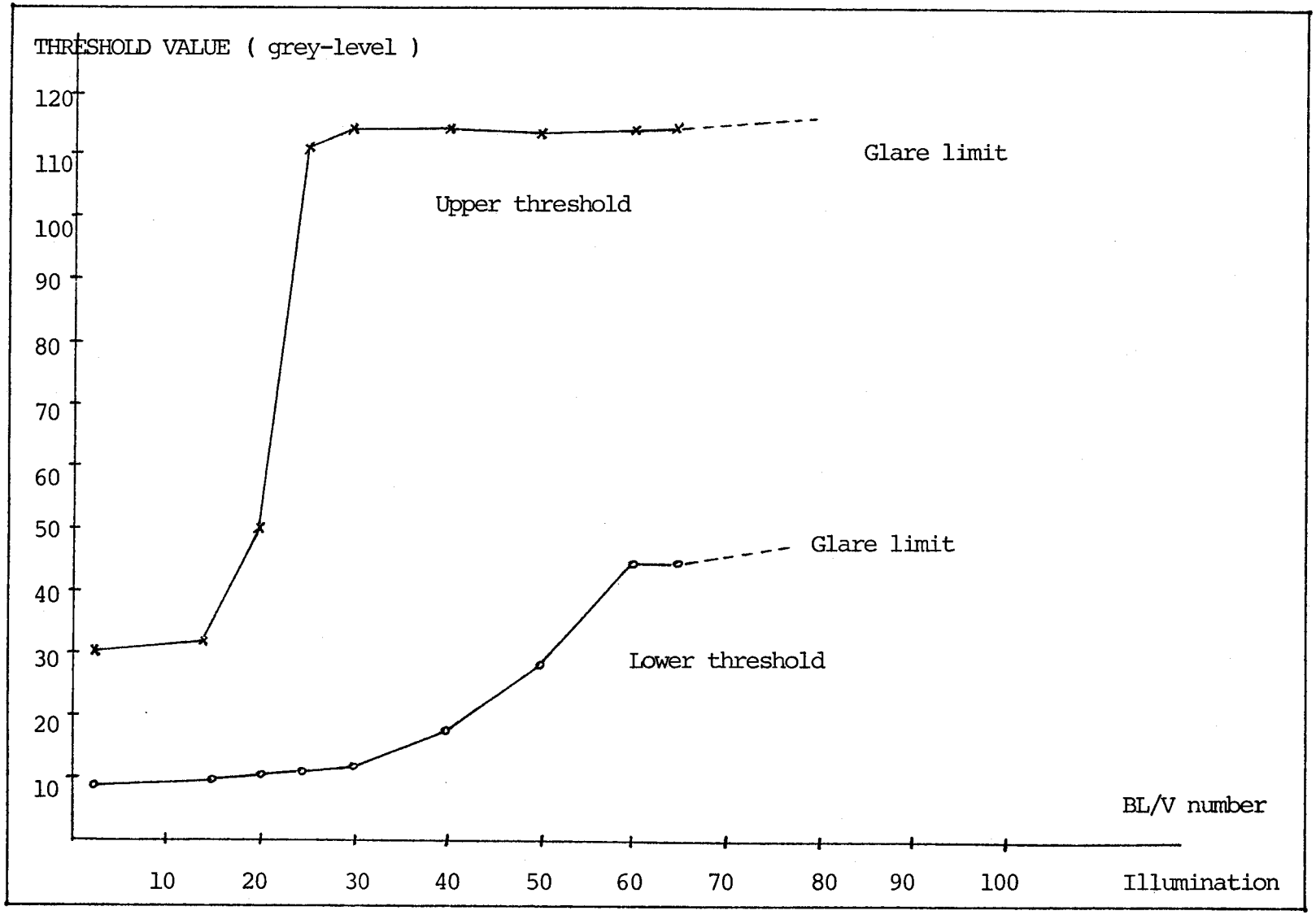


Figure 5.33 Adaptive range / sensitivity 3

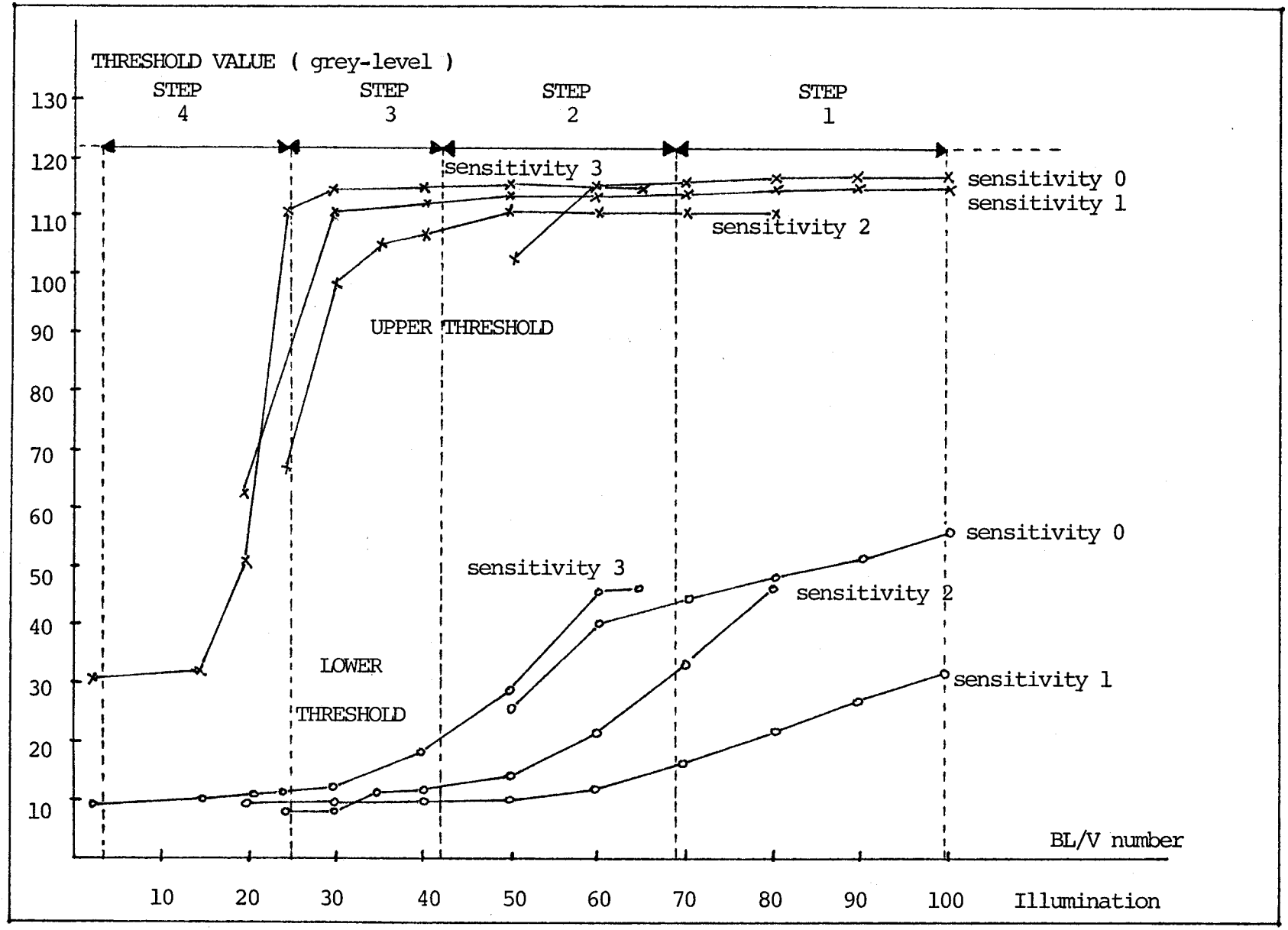


Figure 5.34 Adaptive range of the ALS.

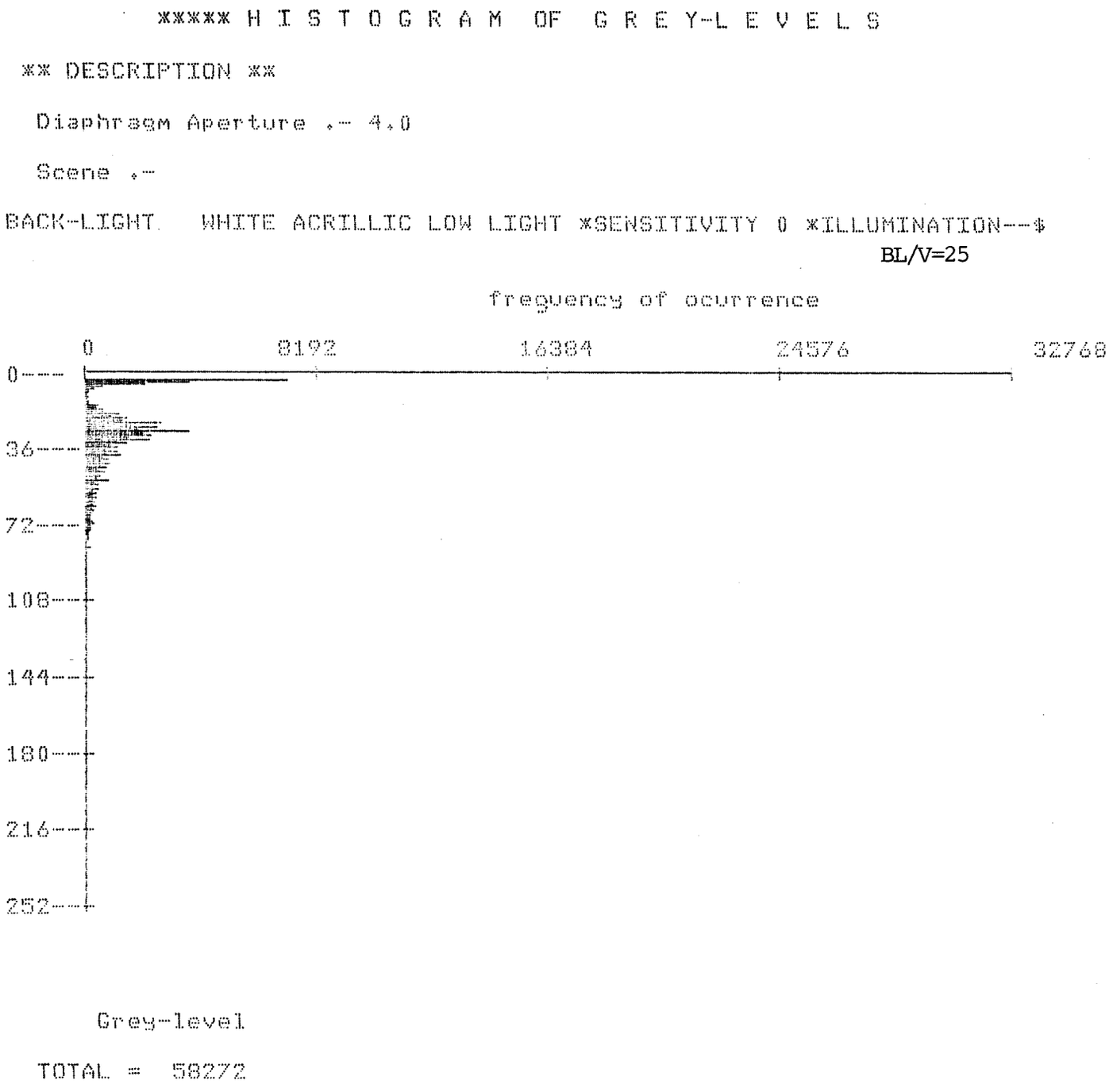


Figure 5.35 Adaptive process first iteration .
(step 1)

Figure 5.37 Adaptive process / third iteration.
(step 3)

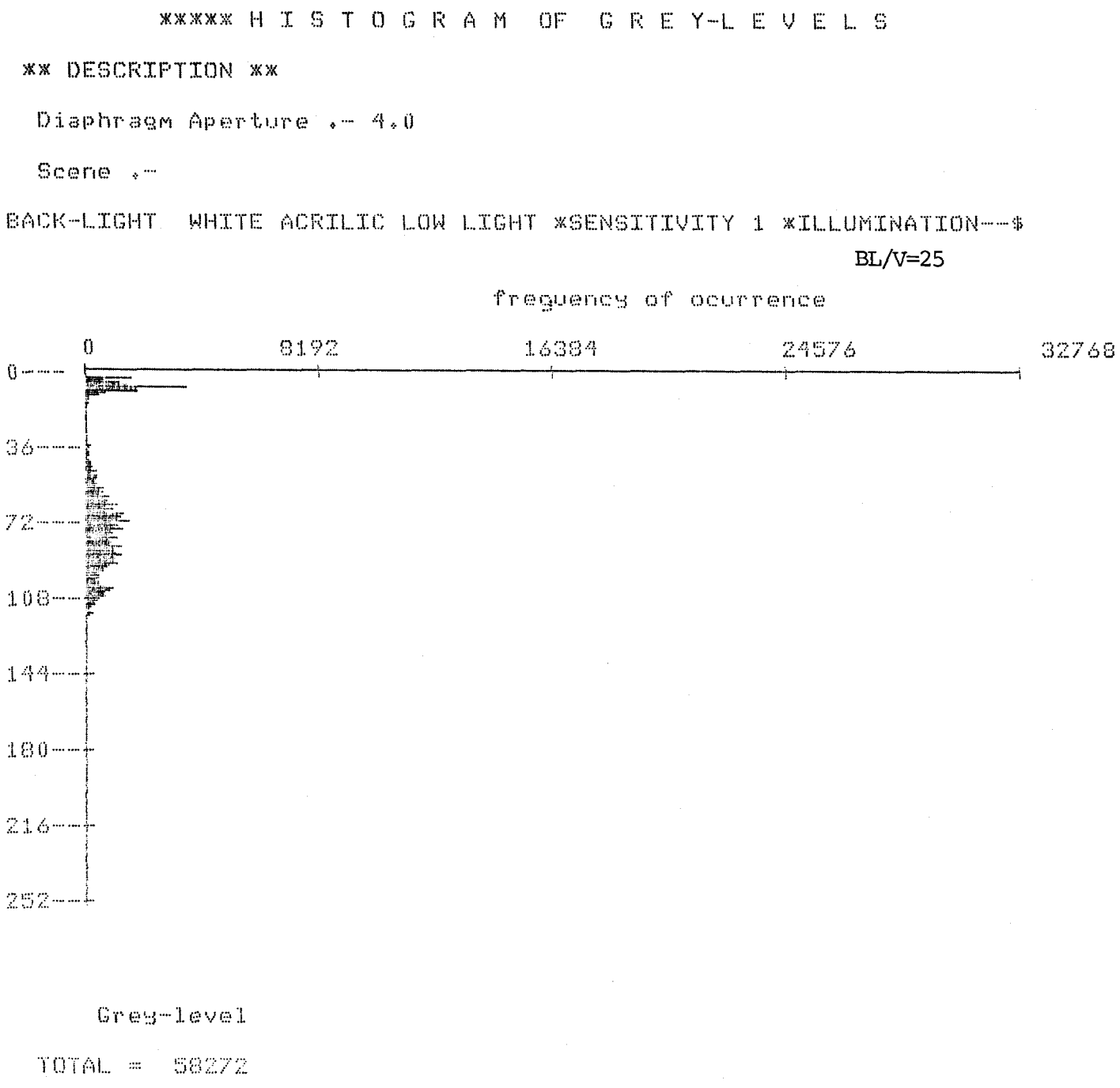


Figure 5.36 Adaptive process /second iteration
(step 2)

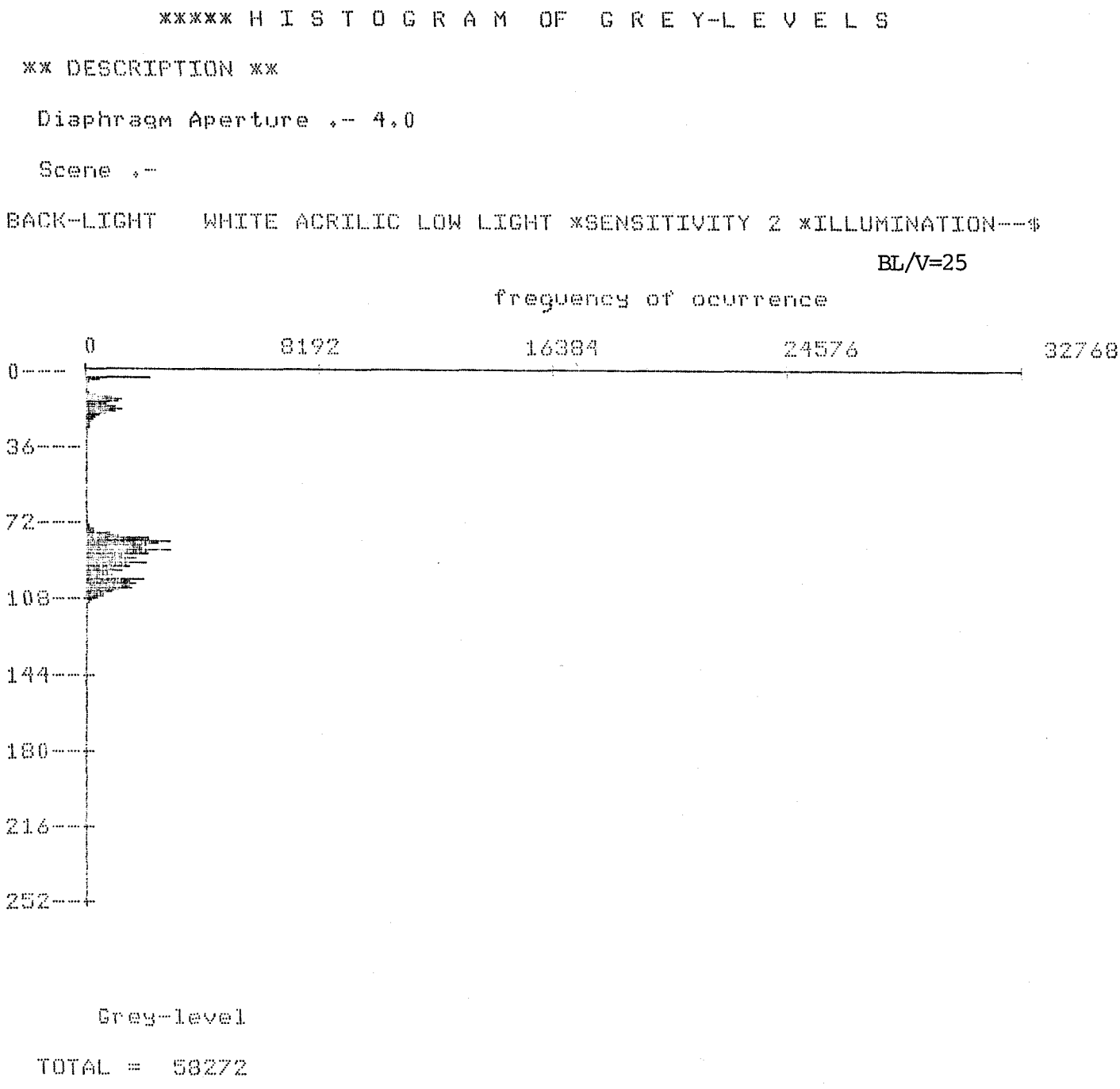


Figure 5.37 Adaptive process / third iteration.
(step 3)

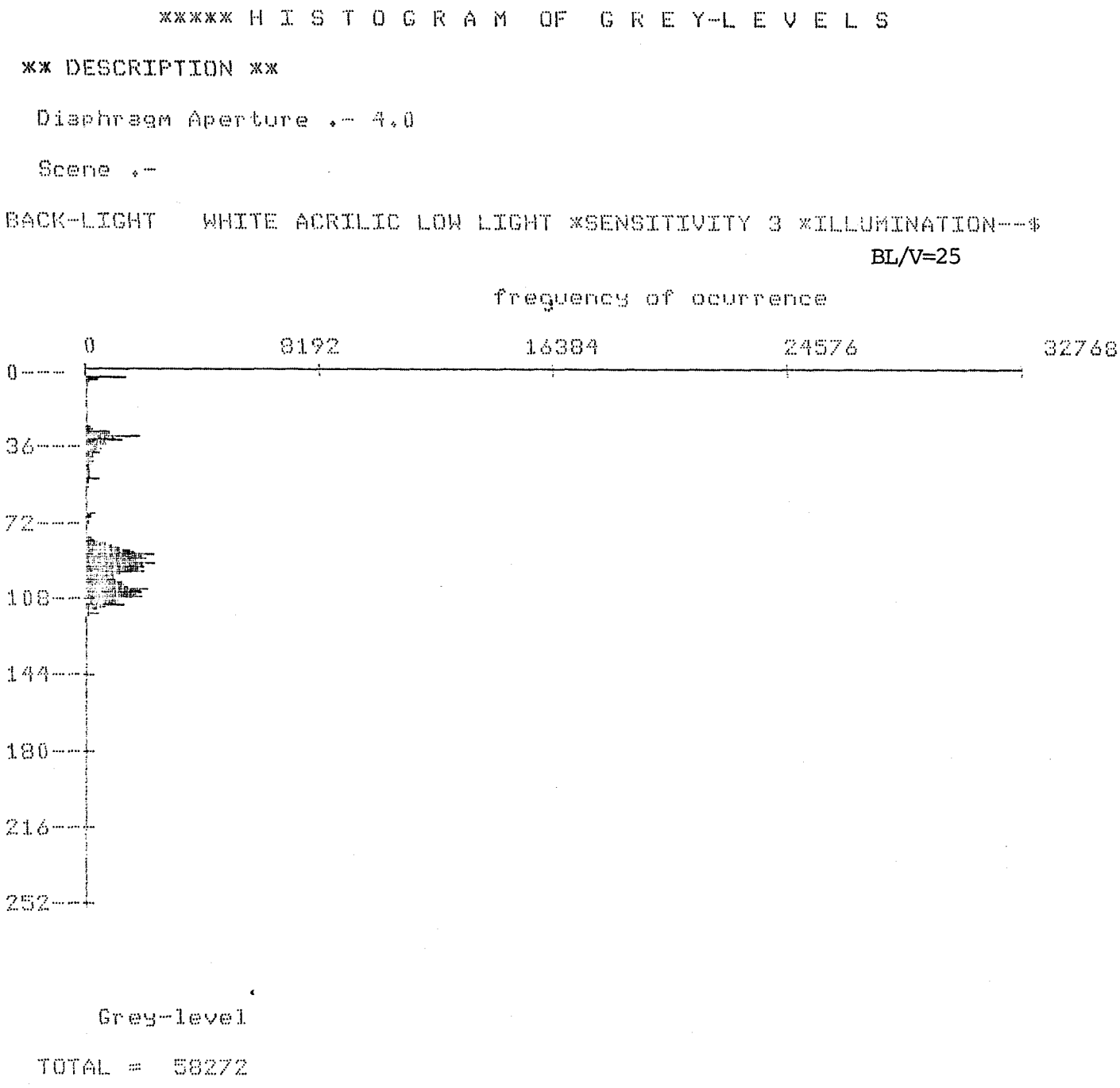


Figure 5.38 Adaptive process / fourth iteration (step 4).

CHAPTER VI

DISCUSSION

Vision systems have evolved to the point of being used on real time industrial applications to achieve particular and limited tasks. Much work has still to be done to reach the time of using general purpose vision systems.

Insufficient knowledge of the way the human eye processes visual information has limited the vision system's designers in imitating the eye's performance and looking for novel, easy and more applicable implementation approaches.

Basically the designer seeks methods which reduce the amount of data to be processed. Digital electronic devices today are still not fast enough so that sophisticated pattern recognition techniques can be applied in real time problems.

Silhouette vision systems solve, to some extent, this constraint, but the information extracted from a silhouette is very limited; they do not completely characterize a scene as human eyes do it. It seems that the vision system designer has to direct his approaches to parallel processing techniques to match the required speed in such image processing applications.

This report has attempted to contribute to the improvement of such systems by implementing the ALS . Lighting adaptability is certainly a desired problem to be solved, but many other problems concerning light have still to be solved as mentioned throughout this report. To this point, it seems that grey-level processing for scene analysis is the way to follow for future implementations that attempt to perform more like the human eye. In this way, the VRHG is a powerful tool for grey-level processing.

The implementation of the addressable RAM technique described in chapter III shows the importance of dedicated hardware to achieve primary tasks in industrial applied image processing.

This technique can be modified and used to capture, at high speed, primitive features commonly used in silhouette vision systems. In this way, the host processor of the system is released from such tasks and freed for other purposes.

REFERENCES

Chapter I

[1-1] H. Lechtman, N. Nagel. E. Plomann , " Connecting the PUMA Robot with the MIC VS-100 vision system and other sensors", ROBOTS VI Conference Proceedings , March 1982 , Detroit , Michigan USA , pp. 447-466 .

[1-2] D. Capson, M.Pena , R. Kitai, " A Silhouette Vision System for Robots", First Canadian Conference on Robotics, September 1982 , Missassauga , Canada .

Chapter II

[2-1] H. Davson " The Eye " Vol 1, Vegetative Physiology and Biochemistry Academic Press, New York 1962 , pp. 5-6 .

[2-2] R. Gonzalez, P. Wintz , " Digital Image Processing ", Addisson Wessley . P.C. 1977 , pp. 14 .

[2-3] H. Davson " The Physiology of the Eye " , Academic Press, New York , 1972 (third edition) , pp. 11 .

[2-4] H. Davson " The Physiology of the Eye ", Academic Press , New York, 1972 (third edition) , pp. 125 .

[2-5] H. Davson " The Physiology of the Eye ", Academic Press , New York (third edition) , 1972 , pp. 87-89 .

[2-6] H. Davson " The Physiology of the Eye " , Academic Press , New York (third edition) , 1972 , pp. 131 .

[2-7] R. Gonzalez. P. Wintz , " Digital Image Processing ", Addisson Wesley P.C. , 1977 , pp. 16 .

[2-8] H. Davson, " The Eye " , Vol.2 , Academic Press , New York 1962 , pp. 40-56 .
H. Davson " The Physiology of the Eye " , Academic Press ,1972 New York , pp. 209 .

[2-9] R. Gonzalez , P. Wintz, " Digital Image Processing " , Addisson Wesley P.C. , 1977 , pp. 17 .

[2-10] R. Gonzalez and P. Wintz ," Digital Image Processing ", Addisson Wesley P.C. , 1977 , pp. 16-20 .

[2-11] C. Braccini, G. Gambardella , G. Sandini , " Borrowing from the Eyes to Create Robot Vision Algorithms" Robotics Today Winter 1980-1981 , pp. 74-78 .

[2-12] A.H. Longford, G. Reticon " Intelligent Vision for Industrial Control ", Sensor Review Vol.2, No.2, April 1982 , pp. 8-14.

[2-13] R. Bosch , " Vision Research to Practice ". Sensor Review, Vol.2,No.2, April 1982 , pp. 68-70 .

[2-14] R.L. Gregory, " The Intelligent eye " , Mc Graw Hill Paperbacks, New York , 1970 .

[2-15] I. Rock , " Introduction to Perception " , Macmillan, New York, 1975 .

[2-16] H. Davson , " The Physiology of the Eye ", Academic Press , New York , (third edition) , 1972 , pp. 132 .

[2-17] R. Gonzalez , P. Wintz , " Digital Image Processing ", Addisson Wesley P.C. 1977 . pp. 25-27 .

[2-18] R.E. Woods and R. Gonzalez , " Real time Digital Image Enhancement " , Proc. of IEEE , Vol.69,No.5 , May 1981 , pp. 634-654 .

[2-19] Gleason J. and Agin G. " The Vision Module sets its sight on sensor-controlled manipulation and inspection ", Robotics Today , Winter 1980-1981 , pp. 36-40 .

[2-20] W. Doyle, " Operations useful for similarity-invariant Pattern Recognition " , J . Assoc. Computer Mach. 9 , 1962 , pp.259-267 .

[2-21] J.M.S. Prewitt and M.L. Mendelsohn , " The analysis of Cell Images " , Ann N.Y. Acad. Sci. 128 , 1966 , pp. 1035-1053 .

[2-22] R.J. Wall, A. Klinger and K.R. Castleman " Analysis of image Histograms " , Second International Joint Conference on Pattern Recognition (IEEE Publ. 74CH-0885-4C) Copenhagen, August 1974 , pp. 341-344 .

[2-23] D. Mason, I. Lauder , D. Rutovitz and G.Spowart , " Measurements of C-bands in human chromosomes ,(preprint).

[2-24] J.S. Weszka . R.N. Nagel and A. Rosenfeld , " A Threshold Selection Technique " , IEEE Trans. Computers 23 , 1974 , pp. 1322-1326 .

[2-25] J. Weszka and A. Rosenfeld " Threshold Evaluation Techniques", IEEE Trans. SMC Vol SMC-8 , No.8 , August 1978 , pp. 622-629 .

[2-26] John F. Jarvis , Bell Laboratories " Research directions in industrial Machine Vision " , Computer Magazine , December 1982 - pp. 55-61 .

[2-27] R. Gonzalez and R. Safabakhsh , " Computer Vision Techniques for Industrial Applications and Robot Control " , Computer M. December 1982 , pp. 17-32 .

[2-28] Y. Shirai and S. Tsuji , " Extraction of the Line Drawing of Three-dimensional objects by Sequentiall Illumination from several directions " , Pat. Recog. , Vol.4 , 1972 , pp. 343-351.

[2-29] J.F. Jarvis , " Automatic Visual Inspection of Glass Metal Seals " , Proc . Fourth Int'l. Joint Conf. Pat. Recog. , 1978, pp. 961-965 .

[2-30] S.W. Holland , L. Rossol and M.R. Ward , " CONSIGHT-I ; A vision controlled Robot system for transferring parts from belt conveyors", Computer Vision and Sensor-based Robots , G.G. Dodd and L. Rossol, eds. , Plenum Press , New York , N.Y. , 1979 , pp. 81-97 .

[2-31] H. Toda and I. Masaki , " Kawasaki Vision System Model 79-A ", Proc. Tenth Int'l . Symp. Industrial Robots, 1980 , pp. 163-174.

Chapter III

[3-1] D. Capson," Techniques for the Recognition of Silhouettes", M. Eng. Thesis , McMaster University , Hamilton , Ontario , 1981 , Canada .

[3-2] General Electric , " TN 2500 Solid State video/digital camera OPERATING MANUAL " , Optoelectronics Systems Operation, April 1980 , Syracuse , N.Y. , pp. 15-22 .

Chapter IV

[4-1] Intel Corporation " PL/M-86 Programming Manual ", Santa Clara, California 1979 .

[4-2] Intel Corporation , " MCS-86 Software Development Utilities operating instructions for ISIS-II users", Santa Clara , California 1979 .

Chapter V

[5-1] C. Shipman , " SLR Photographers Handbook " , H.P. Books , Tucson , Arizona 1977 , pp. 58 .

[5-2] Ashai , " Pentax SPOTMETER V-FL " , Pentax of Canada LTD. , Vancouver , B.C. 1982 .