Lehigh University
Lehigh Preserve

Theses and Dissertations

1987

A method for automatice alignment of ceramic IC packages during wire bond inspection /

Christine Ruth Hofmeister
Lehigh University

Follow this and additional works at: https://preserve.lehigh.edu/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation
Hofmeister, Christine Ruth, "A method for automatice alignment of ceramic IC packages during wire bond inspection /" (1987). *Theses and Dissertations*. 4758.
https://preserve.lehigh.edu/etd/4758

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

A METHOD FOR AUTOMATIC ALIGNMENT OF CERAMIC IC PACKAGES DURING WIRE BOND INSPECTION

by

Christine Ruth Hofmeister

A Thesis

Presented to the Graduate Committee

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

1987

This thesis is accepted and approved in partial fulfillment of the requirements for the Degree of Master of Science.

5/12/87
(date)

Professor in Charge

Chairman of Department

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor, Roger Nagel, for providing me with this research opportunity and the guidance that enabled me to complete this thesis. In addition I would like to thank Dean Fresonke for providing invaluable technical assistance, and Tsvi Goldenberg for providing the initial impetus for this project. Finally, I wish to thank my parents, family, and friends for their support and encouragement in this endeavor.

TABLE OF CONTENTS

Abstract 1

Chapter 1: Introduction 2

Chapter 2: Using Computer Vision during IC Assembly . . 4

2.1 Inspection of IC Devices 4

2.2 Computerizing the Vision Tasks 5

2.3 Classification of Current Techniques 7

Chapter 3: The Wire Bond Inspection System. 12

3.1 Goals of the System. 12

3.2 Implementing the Wire Bond Inspection. 14

3.2.1 Performing the Inspection 16

3.2.2 Providing Flexibility 17

3.2.3 Balancing Speed vs Cost 19

Chapter 4: An Improved Package Alignment Method 21

4.1 Goals of the Improvements. 21

4.2 Design of the Alignment Method 24

4.2.1 Choice of Reference Object. 24

4.2.2 Design of Setup and Alignment Phases. . . .27

4.3 Implementing the Alignment 32

4.3.1 Setup Phase 32

4.3.2 Alignment Phase 43

Chapter 5: Analysis 48
5.1 Hardware 48
5.1.1 The Mirror System 49
5.1.2 The 8086 / TARGA-8 Combination. 50
5.2 Hardware vs Software for Vision Processing . . .55
5.3 Choice of Language 56
5.4 Extending the Application of the Alignment . . .58

Chapter 6: Conclusions. 60

References 63
Vita 64

LIST OF FIGURES

3.1: Package with Broken Wires. 13

3.2: Hardware Components of the Wire Bond Inspection System. 15

3.3: Setting Scan Line Lengths. 18

4.1: Former Alignment Method. 22

4.2: Reference Object: Crossbar. 26

4.3: Unique Reference Object. 28

4.4: Frame Around Crossbar. 28

4.5: Alignment Windows. 30

4.6: Segmentation Using a Threshold 32

4.7: Edge Detection Templates 33

4.8: Applying Edge Detection Templates. 34

4.9: Locating Edges using Histograms. 36,37

4.10: Cross-Section Profile. 40,41

4.11: Tasks Performed during Setup Phase 42

4.12: Locating Vertical Bar. 44

4.13: Locating Horizontal Bar. 45

4.14: Tasks Performed during Alignment Phase 47

5.1: 8086 Memory Addressing 52

5.2: TARGA-8 Addressing Scheme 54

ABSTRACT

This thesis describes a method for automatic alignment of ceramic integrated circuit packages as part of a wire bond inspection system. The system for which this alignment method was developed provides automatic inspection of the wires connecting the IC chip to the package to determine whether they are broken, missing, or poorly bonded. Inspecting the wire bonds is part of the standard third optical inspection, which immediately follows the wire bonding step of the IC assembly process. It is cost effective to inspect the wires visually because at this stage devices with defective bonds can generally be fixed, whereas devices determined to be defective in the final electrical inspection, after molding or sealing the device, must usually be discarded. The wire bond inspection system outlined here successfully substitutes computer vision and automatic material handling for human vision and manual handling. The alignment portion of the inspection system, which is based on a template matching approach, is discussed in detail: the challenge here was to devise a reliable alignment technique which could be done in less than half a second with no special-purpose hardware other than the video digitizing board. The digitizer provided fast, memory-mapped access to the digitized image, but its use was hampered by a complicated addressing scheme inherited from the 8086 microprocessor used in the system. Because the vision processing was implemented solely in software, the choice of C as the primary language contributed to the ease and speed of code development, and to the execution speed of the system. The mirror system used for imaging also contributed to the execution speed, but the critical factor in attaining the desired two-second inspection cycle time was specializing the vision processing algorithms for this particular inspection task. To generalize the inspection and alignment algorithms while maintaining the two-second cycle time, some of the vision processing would have to be implemented in hardware.

Chapter 1

Introduction

Inspecting the wire bonds connecting an integrated circuit (IC) chip to its package is part of the third optical inspection, which immediately follows the wire bonding step in the IC assembly process. It is cost effective to inspect the wires visually because at this stage devices with defective bonds can generally be fixed, whereas devices determined to be defective in the final electrical inspection, after molding or sealing the device, must usually be discarded. As much of the IC assembly process is currently automated, it is also desirable to computerize the visual inspections: human inspectors are expensive, and are often subjective or inconsistent in inspecting the IC devices. At this time, several commercial systems are available for automatic alignment of the device, a task that is needed at several places in the assembly process and in the optical inspections themselves.

The wire bond inspection system outlined in Chapter 3 is designed to provide automatic inspection of the wires on ceramic packages to determine whether they are broken, missing, or poorly bonded. The computerized inspection system is 18-34% better at detecting faulty devices than human inspectors, depending on the device type, and

accomplishes the inspection in under two seconds per device. It also automates the material handling of the devices, thereby decreasing the losses due to contamination and mishandling.

However, because of the method used to align the devices at the inspection station, the prior version of this inspection system was limited in the types of ceramic devices which could be inspected. The package alignment approach described in detail in Chapter 4 addresses this problem with a more sophisticated and more reliable method. The alignment method is based on a template matching approach, an inherently reliable but slow approach.

The primary challenge in the developing the alignment method was to devise a reliable method which could be done in less than half a second with no special-purpose vision processing hardware. In addition to the design of the vision processing algorithms used in the alignment, the hardware and software components used in the wire bond inspection system also play an important role in meeting this challenge.

Chapter 2

Using Computer Vision during IC Assembly

2.1 Inspection of IC Devices

Inspection of integrated circuits has two components: visual and electrical inspection. The electrical inspection is critical for determining whether a device is functional, but it has been found that augmenting the electrical test with a visual inspection provides a more robust inspection. Visual inspection can detect defects which would affect the reliability of the device but do not show up in an electrical test [1].

This visual inspection of the chips is performed three times, at various stages during the assembly process. These inspections are known as the first, second, and third optical inspections. Using several inspections allows defective chips to be identified as soon as possible, thus avoiding further processing on these chips and providing rapid feedback for adjustment to the assembly process. The first optical inspection of wafers is performed before the electrical wafer probe test, usually on a sampling basis, with the goal of detecting obvious flaws. The second optical inspection is performed after wafer dicing. At this stage each diced chip, or die, is inspected thoroughly for

large defects such as cracks, blemishes, and defects caused by the dicing process, and for more subtle defects such as errors in the mask alignment. The third optical inspection takes place after the die is attached and the wires connecting the die to the package have been bonded. Here the inspector looks for missing wires, checks the wire bonds and loop height of the wires, and checks again for some of the same defects as in the second optical inspection. [1]

2.2 Computerizing the Vision Tasks

Visual input is needed not only for inspecting the IC but for aligning components at various stages during assembly. Automatic mechanical positioning is not accurate enough for positioning the wafer probe used in the electrical test, for positioning the die accurately on the package during die bonding, or for aligning the chip during wire bonding, all of which require visual alignment.

Although the optical inspection and alignment tasks have generally been done by people because of the difficulty of computerizing the vision processes involved, using human vision poses some problems. One problem is that people are susceptible to fatigue and boredom, thus producing inconsistent results, and that the judgments they must make are subjective. A computer could produce more consistent and more accurate results. Another problem with human

intervention is the inevitable loss due to contamination and mishandling, which could be prevented with a completely automated, closed system.

Computerization of the vision tasks has been slow because of the complexity and variability of the scene. Since the chips are not exactly identical, the computer vision algorithms used must be flexible enough to accomodate slight variations in the scene. Because the scene to be analyzed is complex, a gray-scale is generally used in order to preserve more detail than is contained in a binary image. Complete analysis of this detailed image with conventional techniques would be much too slow; a successful system must instead use intelligent localized searching and exploit particular properties of the image [2].

Inspection is a much more complex task than alignment: it generally involves alignment as a subtask, along with verifying patterns, looking for the ink dot which marks a defective chip, checking wire bonds, and analyzing wire placement. Today, many algorithms for performing automatic alignment exist, and computerized alignment is being used in die bonders, wire bonders, and for wafer probing. Much progress has also been made in automatic inspection at the first and second optical inspection stages. An overview of the techniques which have been successfully employed is provided in [3].

2.3 Classification of Current Techniques

Most pattern recognition algorithms can be classified under one of the following approaches: template matching, feature analysis, or structural and syntactic analysis. These approaches are distinguished by their choice of pattern representation and by their method of matching new scenes to this stored pattern. [4]

The template matching approach stores a template of the prototype scene and classifies new scenes by finding the best match to the template. The stored template is a two-dimensional pixel representation of the scene, and the correlation between the template and the scene is calculated pixel by pixel [5]. If the goal is to locate a particular scene within the image, the template is moved over the image searching for the highest correlation. If the goal is to classify the image, it is matched against a series of templates and is classified according to its correlation with these templates.

Template matching is very natural way to approach alignment and inspection tasks. In the alignment task, the template consists of a snapshot of some reference object. Alignment for a new scene is accomplished by locating the reference object within the scene and recalibrating the scene using the difference between the actual and expected locations of the reference object. Once the scene is

aligned, it can be inspected by computing its correlation with a template consisting of a master image. The scene passes inspection if it correlates well with the master image, and fails inspection if not.

If template matching is used for inspecting IC devices, the acceptable variations among the devices must be distinguished from defects. Because the devices are not expected to be exactly identical, the master image should be a good generalization of a number of samples, perhaps derived from an averaging of the samples. This generalized master image may not exactly match any of the samples; thus the matching criterion must allow slight variations. Inspection of wires using template matching would be impractical because the acceptable wire positions vary quite a bit. An averaging of images at the wires would give a blurred master image, which would correlate just as well with a broken wire as with a good one.

The template matching method, as the simplest of the three methods, has been the traditional approach for commercial alignment and inspection systems. The typical technique for die alignment on wire bonders is to use two small scenes specified by the operator to correct the rotational and translational alignment. The operator selects two areas in the image to serve as alignment templates. These areas can be somewhat arbitrary, but each must be unique. During alignment for wire bonding,

correlations are computed between the template and the image, with the highest correlation pinpointing the location of the corresponding scene in the image. Computing the correlation at all of these areas in the image would be extremely slow if implemented in software. These commercial alignment systems would not be viable without special hardware for implementing much of the vision processing. [4, 5]

In the feature analysis or decision-theoretic approach, the pattern is represented by its characteristic features, and classification is done by comparing features extracted from the image with the features in the pattern. One example is a commercial alignment system for wafer probers, the Teledyne TAC WARS II (Wafer Alignment Recognition System), which uses this feature analysis approach by extracting information about widths and lengths of streets on the wafer [5]. Because the process of feature extraction distills the information contained in the scene, this method is inherently less susceptible to minor variations or irregularities. On the other hand, it in not always easy to find features which adequately characterize complex scenes.

A drawback to the template approach is the fact that saving a two dimensional, pixel by pixel template requires a large amount of storage, whereas by extracting features from the scene, an equivalent amount of information can be stored much more succinctly. An approach which lies somewhere

between these two methods is to perform some kind of standard compression, such as a projection, on a two-dimensional template. The compressed template can be considered a feature characterizing the original template, and as long as no significant information is lost by this compression, this approach uses less storage, and does not involve a specialized feature extraction routine. The alignment method discussed in Chapter 4 uses this approach, as do the commercial wire bonder (Model 1419) and die bonder (Model 6300) from Kulicke & Soffa Industries, Inc. For the alignment in these two systems, the features extracted are digital signatures of the area of interest [5]. Because this feature extraction is not designed for a particular type of scene, it is more generic than the extraction of street edge features used in Teledyne's WARS II system.

Baird also uses the feature analysis method in his inspection of Darlington IC chip integrity. He identifies broken, cracked, undersized, fractured, or missing chips by inspecting the gray-level contrast across the four sides and corners of the chip, using the anticipated contrast as a feature. [6]

The structural and syntactic approach is also known as the non-reference method because in this method the prototype scene need not exist. The scene is represented by pattern primitives and a string, tree, or graph of their relations. Classification of images is accomplished by

segmenting the image to determine the pattern primitives and their relations, then analyzing the syntax to determine how close it is to the prototype structure. The success of this approach depends on whether adequate rules can be determined for the entire scene. [4]

The use of this third method has had some success in detecting defects in printed circuit boards and for inspecting mask patterns. The automatic inspection system for IC chips proposed by Hsieh and Fu [7,8] makes use of all three methods in an integrated, hierarchical approach.

Chapter 3

The Wire Bond Inspection System

3.1 Goals of the System

Most of the automatic IC assembly inspection systems available today automate only the material handling and data processing functions; a few are also capable of automating the second optical inspection, but none offer automatic third optical inspection. A major component of the third optical inspection is the visual inspection of the wires to determine whether wires are present, whether they follow the expected path, whether they are bonded at both ends, and, if the inspection allows three-dimensional viewing, whether the loop height is correct. However, conventional visual inspection cannot determine whether the wire bond is defective. The aim of this inspection system was to inspect for all of these defects except the wire loop height, and to improve upon the typical third optical wire inspection by checking the integrity of wire bonds.

The primary goal of the wire bond inspection system was to automate the material handling and inspection of wire bonds on ceramic devices. The specific goals of the inspection were to detect wires which are missing or are not bonded at both ends. The material handling goals were to have minimal operator interface during inspection, and to

automatically sort good and faulty devices.

The ceramic packages for which the inspection system is designed currently fail the third optical inspection primarily because of faulty wire bonds, missing wires, or broken wires. Figure 3.1 shows an example of a device with broken wires. If the inspection system could detect these defects, then, keeping in mind that all devices undergo an electrical test in the final stages of assembly, the other aspects of the third optical inspection such as loop height or chip damage would only need to be checked on a sampling basis.

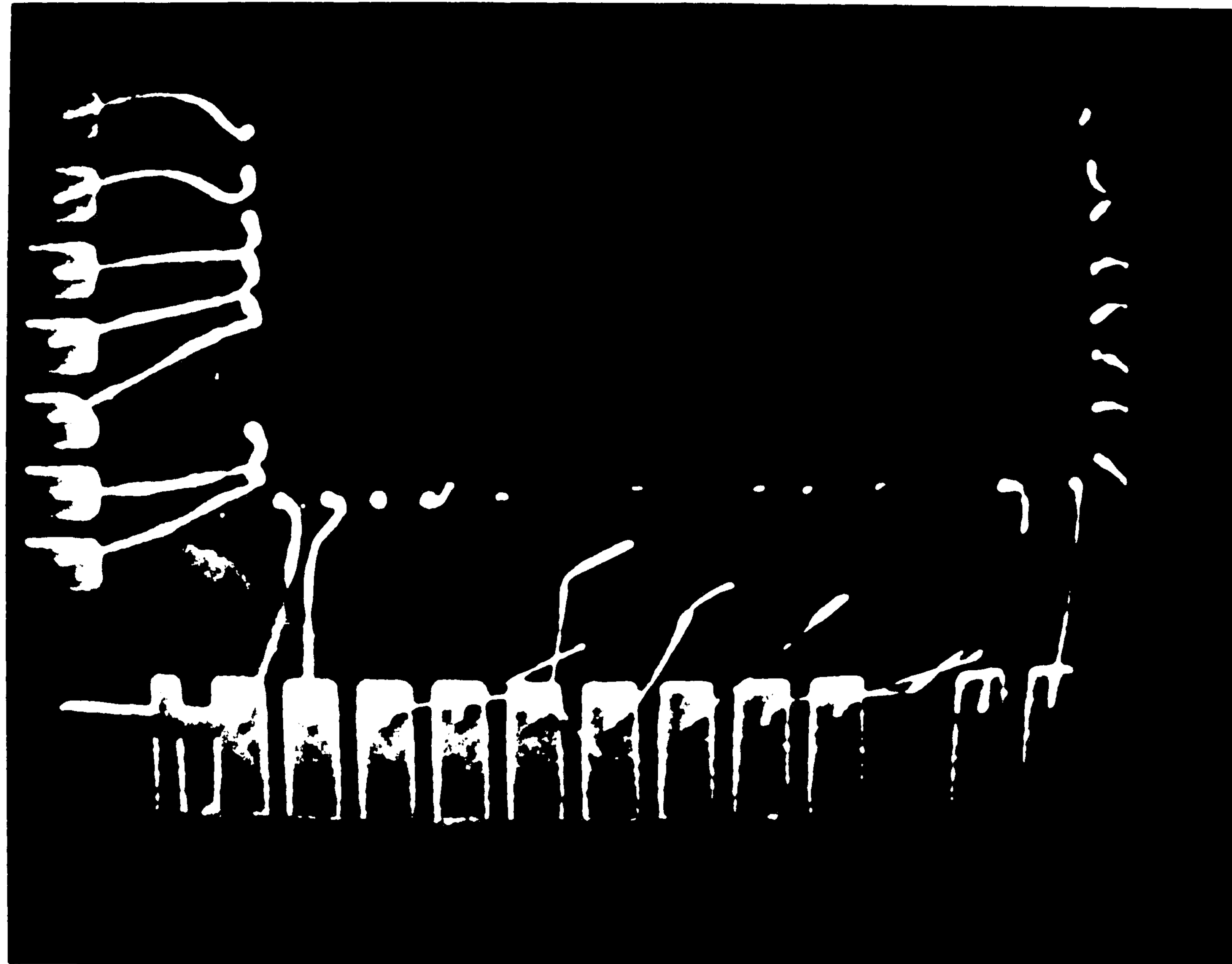


Figure 3.1: Package with Broken Wires

All devices rejected by the wire bond inspection system would be checked by human inspectors, to verify that

the device is defective, and to make a decision about reworking the device. No more than 1.57% of the devices passed by the inspection system should be defective, which is the rate achieved by human inspectors. The number of good devices rejected by the inspection system is somewhat less important because all rejections would be inspected manually; but if this percentage rose above approximately 20%, cost savings would be reduced because of the increased manual intervention.

Another goal was to make the system flexible enough to accept a large variety of ceramic devices but to keep the setup task for the operator simple. And finally, the system would have to cycle the devices through the inspection system within two to four seconds and be relatively inexpensive before its production could be justified.

3.2 Implementing the Wire Bond Inspection

The hardware components of the wire bond inspection system include:

1) video camera and optics
2) illumination system
3) monitor for digitized image
4) X/Y fine positioning: mirror system
5) video digitizer: AT&T TARGA-8
6) system control: AT&T 6300 microprocessor with hard drive, mouse, and printer
7) mechanical handler:
   - multi-stick input and output
   - automatic feed and sorting of devices
   - X and Y air blasts
   - stepping motor for coarse camera positioning

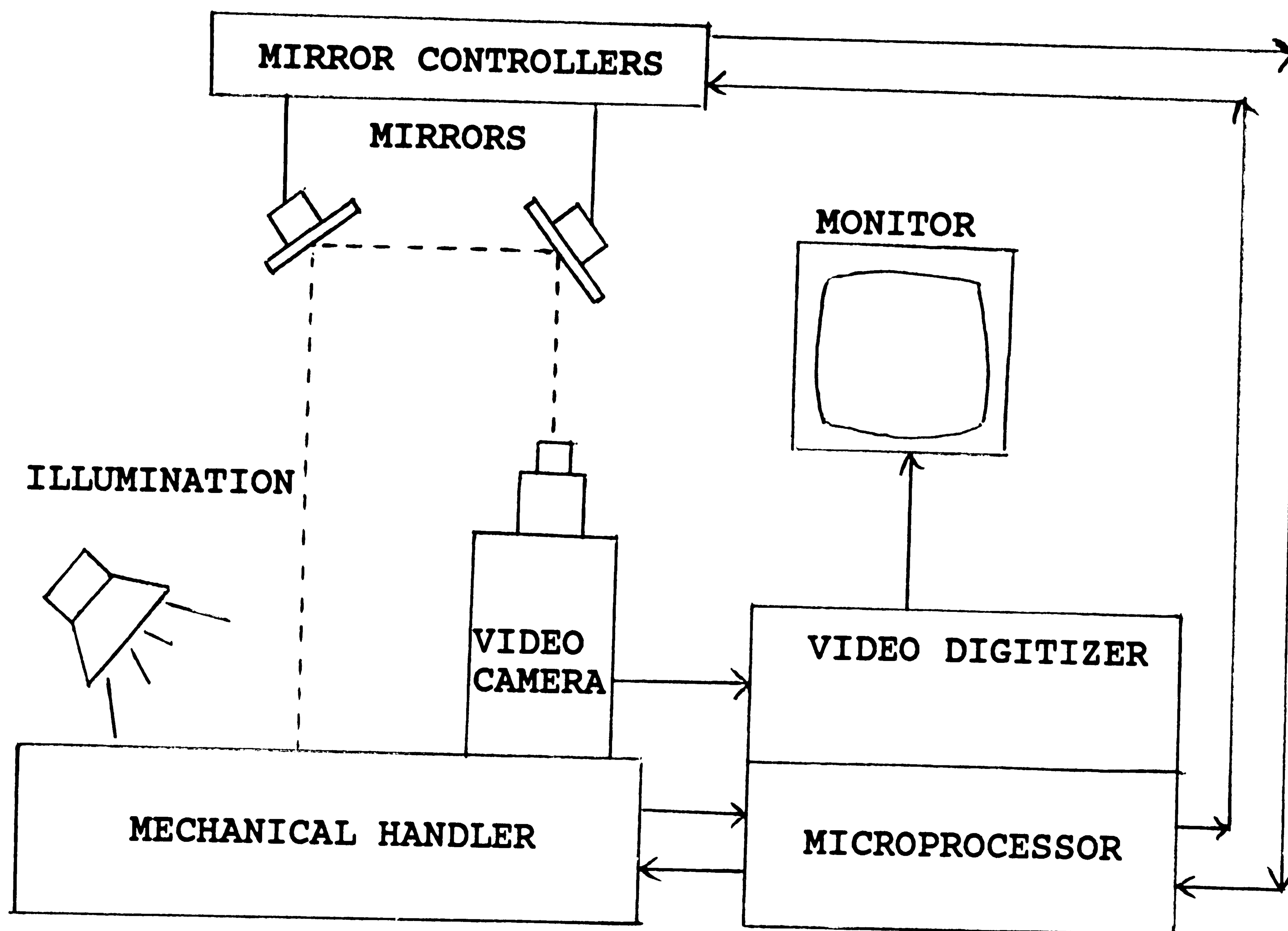


Figure 3.2: Hardware Components of the Wire Bond Inspection System

The relationship among these components is shown in Figure 3.2. These components are fairly typical of a generic inspection system; however the last four components, the fine-positioning system, the video digitizer, the system controller, and the mechanical handler, were significant in reaching the inspection system's goals. The contributions of these hardware components toward the goals are described along with a general description of the inspection system in the remainder of this chapter.

3.2.1 Performing the Inspection

Checking the wires in the wire bond inspection system is simplified by one of the features of the mechanical handler, the X and Y air blast. The handler directs air blasts at the wires on the device to dislodge broken wires and wires with defective bonds. Then the wire bond inspection is reduced to verifying that all wires are present and checking that the wire path is acceptable.

The handler has a multitude of tasks: it must single-thread devices from a cassette of sticks loaded by the operator through the air blast and inspection stations, shuttle the device into a stick in the good or bad cassette depending on the outcome of the inspection, and notify the operator when one of the cassettes gets full. This limits the operator's physical handling of the devices to the loading and unloading of cassettes, which minimizes possible mishandling or contamination.

Another major function of the mechanical handler is to provide coarse alignment of the device at the inspection station. The handler allows no more than ten mils (1000 mils = 1 inch) of translational displacement and no significant rotational displacement, thus simplifying the alignment task for the visual inspection software.

The alignment task is also simplified by the use of a mirror system which allows the image presented to the camera to be shifted without physically moving the device. The two

mirrors, one each for the x and y translations, are under computer control. Once the software has determined the misalignment of the device being inspected, the device alignment can be corrected by adjusting the mirrors, thereby repositioning the device within the image. Thus the alignment correction is done through hardware instead of through computations in the software.

The mirror system also allows a virtual image which is much larger than the image presented to the camera at any one time. By moving the mirrors and capturing a series of images of the device, a higher magnification can be used while maintaining the same field of view. The mirrors require only a 0.15 second settling time to shift the image, and their positioning accuracy is well beyond the requirements of the system. Without the mirror system, shifting the image would entail physically moving either the device or the camera.

3.2.2 Providing Flexibility

Although the operator involvement during inspection is minimal, if the system is to be flexible enough to inspect a variety of devices, the operator must teach the system how to inspect each device type. Thus the system has two modes of operation: a teach mode, in which the parameters needed for inspecting a particular type of device are established and stored on disk, and an inspect mode, in which those

parameters are retrieved and inspection on that device type is performed. The parameters taught are specific to each type of device; they include such things as number of wires per side, mirror voltages for positioning the image, zoom lens setting, and pixel/volt ratio.

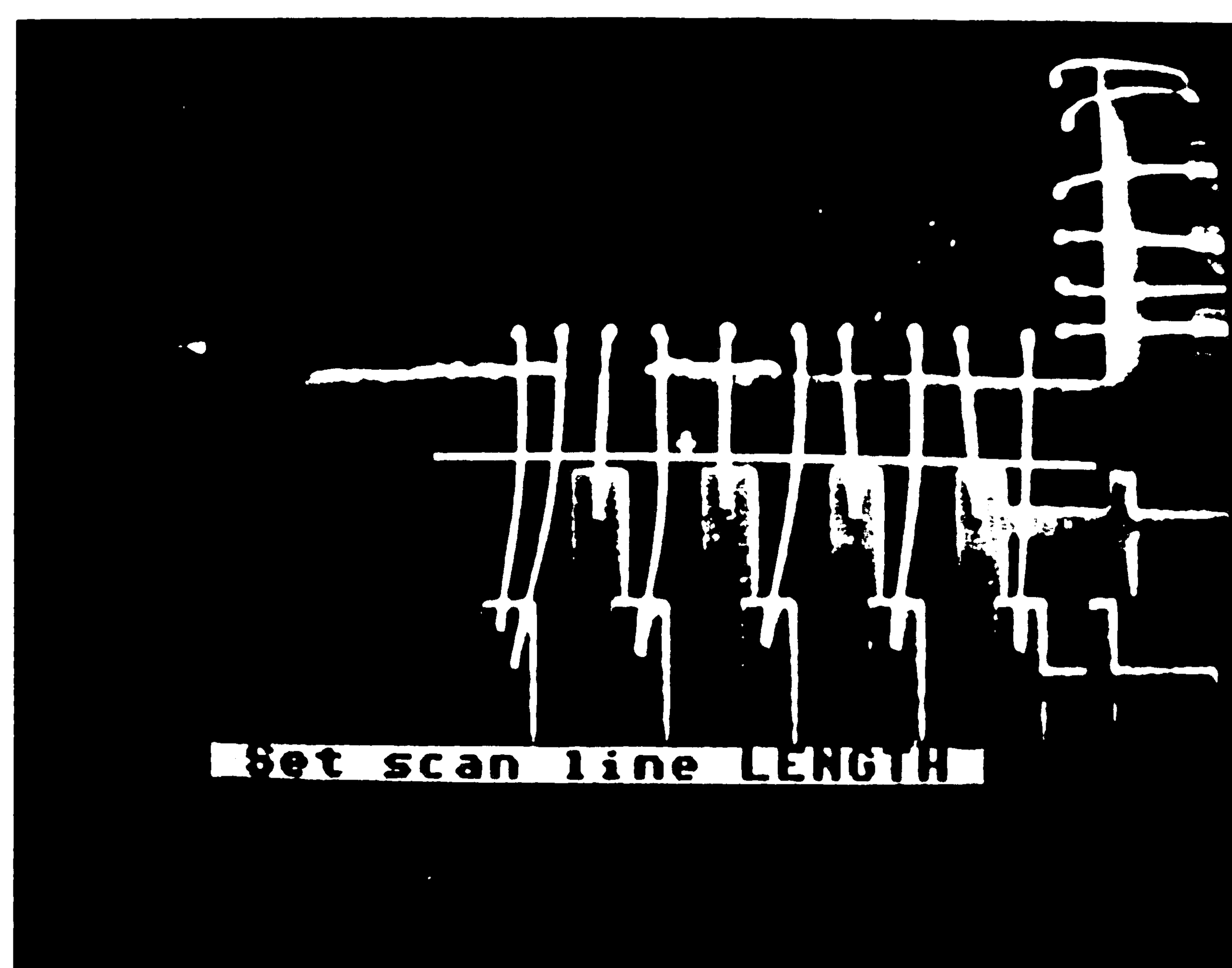


Figure 3.3: Setting Scan Line Lengths

The system is designed to simplify the operator interface during teach mode by facilitating setup of the parameters and by providing feedback for the parameters entered. The setup is nearly all mouse- and menu-driven, with menus appearing simultaneously on a video monitor and the microprocessor screen. The video monitor is used wherever possible to establish parameters graphically instead of numerically: scan line lengths, for example, are

"drawn" with the mouse directly onto the image on the video monitor, instead of being specified by pixel location using the keyboard (Figure 3.3).

3.2.3 Balancing Speed vs Cost

The final requirements for the inspection system were to make it fast and inexpensive, two factors which are often mutually exclusive. The system had to achieve an inspection cycle time of two seconds per device, and the cost had to be as low as possible while maintaining this cycle time.

To keep the total system inexpensive, an AT&T 6300 microprocessor is used for the vision processing and for the overall system control. The primary implication of this choice is that there is no special-purpose hardware for vision processing; all vision processing is implemented in software. However, when implemented solely in software, most standard image processing techniques are too slow to meet the time constraints of the system, requiring simplifications and modifications which exploit particular features of this inspection. For example, the implementation of the device alignment, described in detail in Chapter 4, relies heavily on certain characteristics of the ceramic devices for its alignment windows.

Since significant high-level software development was needed and the final software product had to be efficient, the choice of languages was important. C was chosen as the

primary language because, as a structured high-level language, it facilitates software development and simplifies the task of making modifications. But unlike most structured high-level languages, it produces quite efficient compiled code, a critical factor for this project. Assembly language was used in a few heavily used modules which were computationally expensive.

The speed of the inspection cycle is also improved by the use of the AT&T TARGA-8 video digitizer. This digitizer, which is one of a series providing various levels of color resolution, is well suited to this application with its 512 X 512 pixel resolution with 256 gray levels. The TARGA-8 digitizes and captures a video input signal in 1/30 second, and provides memory-mapped access to this captured image, so that the image processing software can get the image quickly.

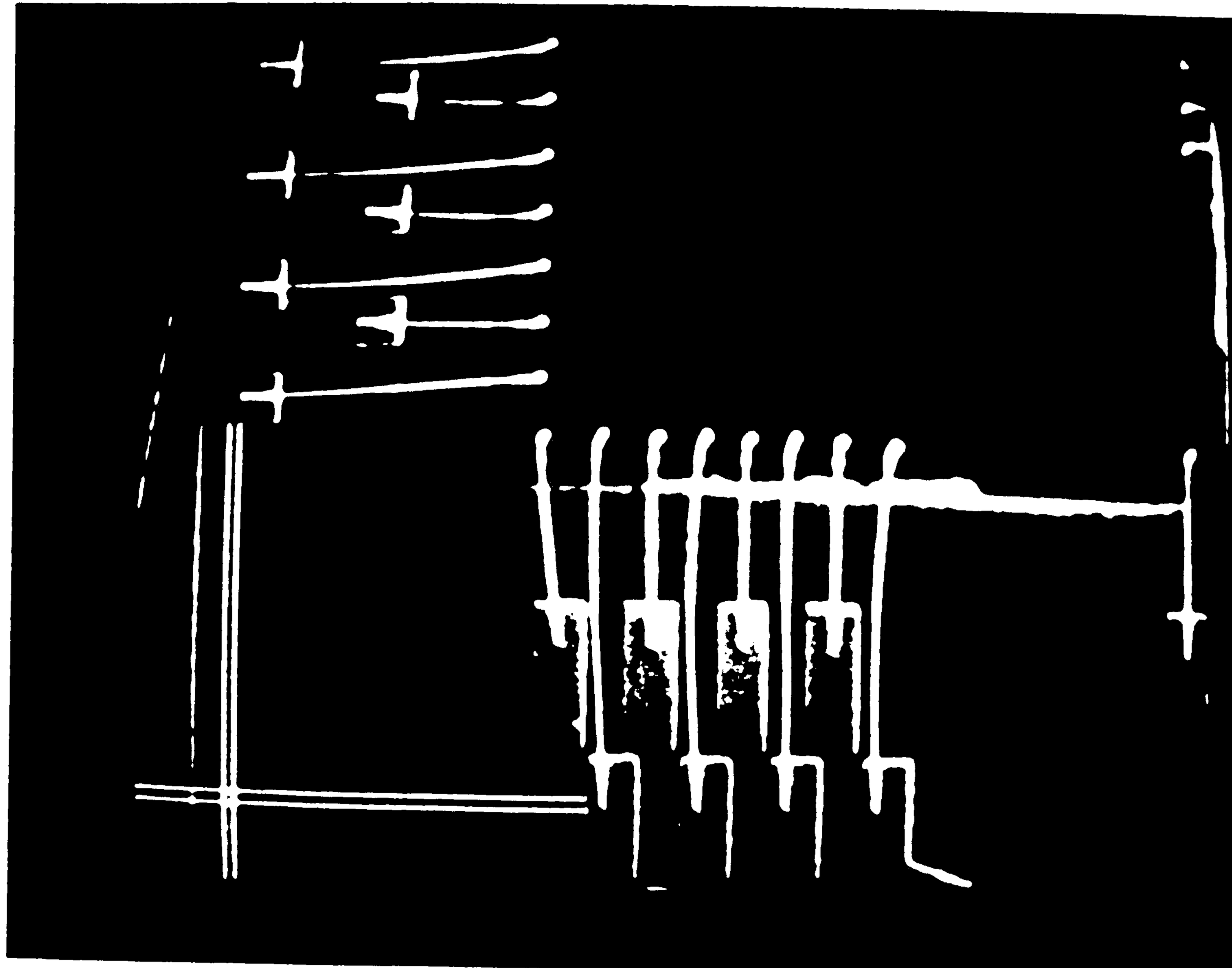
Chapter 4

An Improved Package Alignment Method

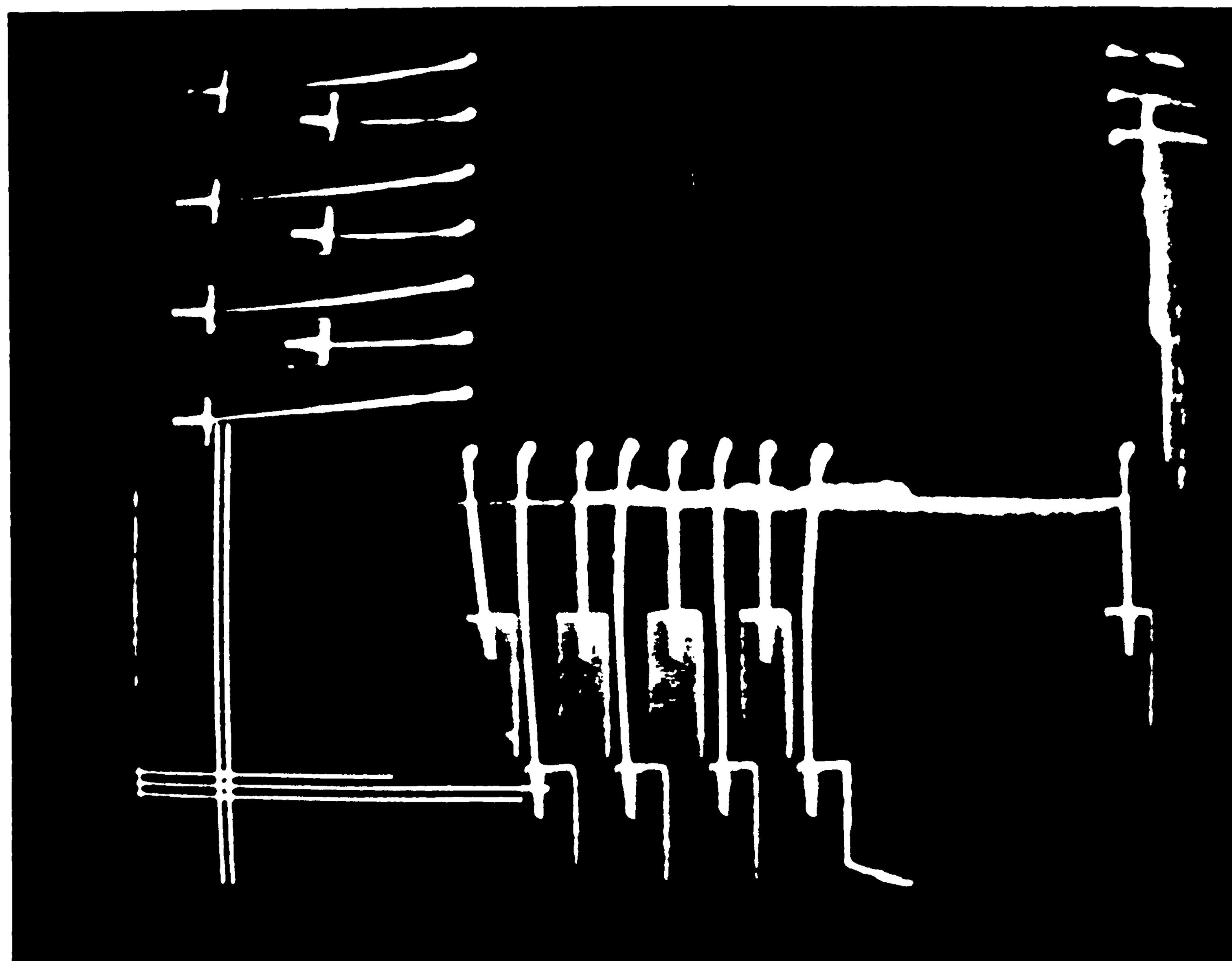
4.1 Goals of the Improvements

Although the positioning accuracy of the mechanical handler is within ten mils, the remaining displacement of the ceramic package must be compensated for by the software. After each package is positioned under the camera, the software must identify reference objects on the package and reposition the image based on the actual location of these objects with no more than a two mil error. In this system, rotational and vertical (z) displacement can be ignored because of the design of the mechanical handler, leaving only x and y displacement. Once the package displacement has been identified, the alignment is corrected by moving the mirrors so that the digitized image is positioned exactly the same as the image that was taught.

This alignment procedure was the weakest part of the previous generation wire bond inspection system. The reference objects used were the outermost package bond pads on the left and bottom sides of the ceramic package; more specifically, the edge of the bond pad closest to the corner. The algorithm used to search for these edges took two or more scans perpendicular to each edge, searching for



a) Successfully finds bond pads



b) Does not locate bond pads

Figure 4.1: Former Alignment Method

an increase in gray level which was above a threshold (Figure 4.1a). When two scans indicated an edge at approximately the same location, the bond pad was presumed to be at that location. Unfortunately, this approach would sometimes misidentify the bond pads, and would sometimes fail to find any edges at all. Figure 4.1b shows a case where the wire was misidentified as the bond pad at the left side, and the search failed altogether on the bottom side. The algorithm did not adjust its search pattern, only the search starting point, to accomodate different chip types, producing a search pattern which worked well for one chip type but was less effective for another. The more serious problem with the approach was it was not discriminating enough: other edges could be easily mistaken for the bond pad edge.

Thus the package alignment portion was the target for improvements to the wire bond inspection system. The two major goals of the improvements were to increase the reliability of the system, so that fewer devices would fail inspection simply because they could not be aligned; and to expand the applicability of the system to a wider variety of devices by using a more general scheme to locate reference objects. Additional goals were to maintain the two second per package cycle time, to allow a package displacement of up to ten mils, and to provide a simple user interface so that the operator could teach a new chip type without

knowing the details of the alignment algorithm.

4.2 Design of the Alignment Method

4.2.1 Choice of Reference Object

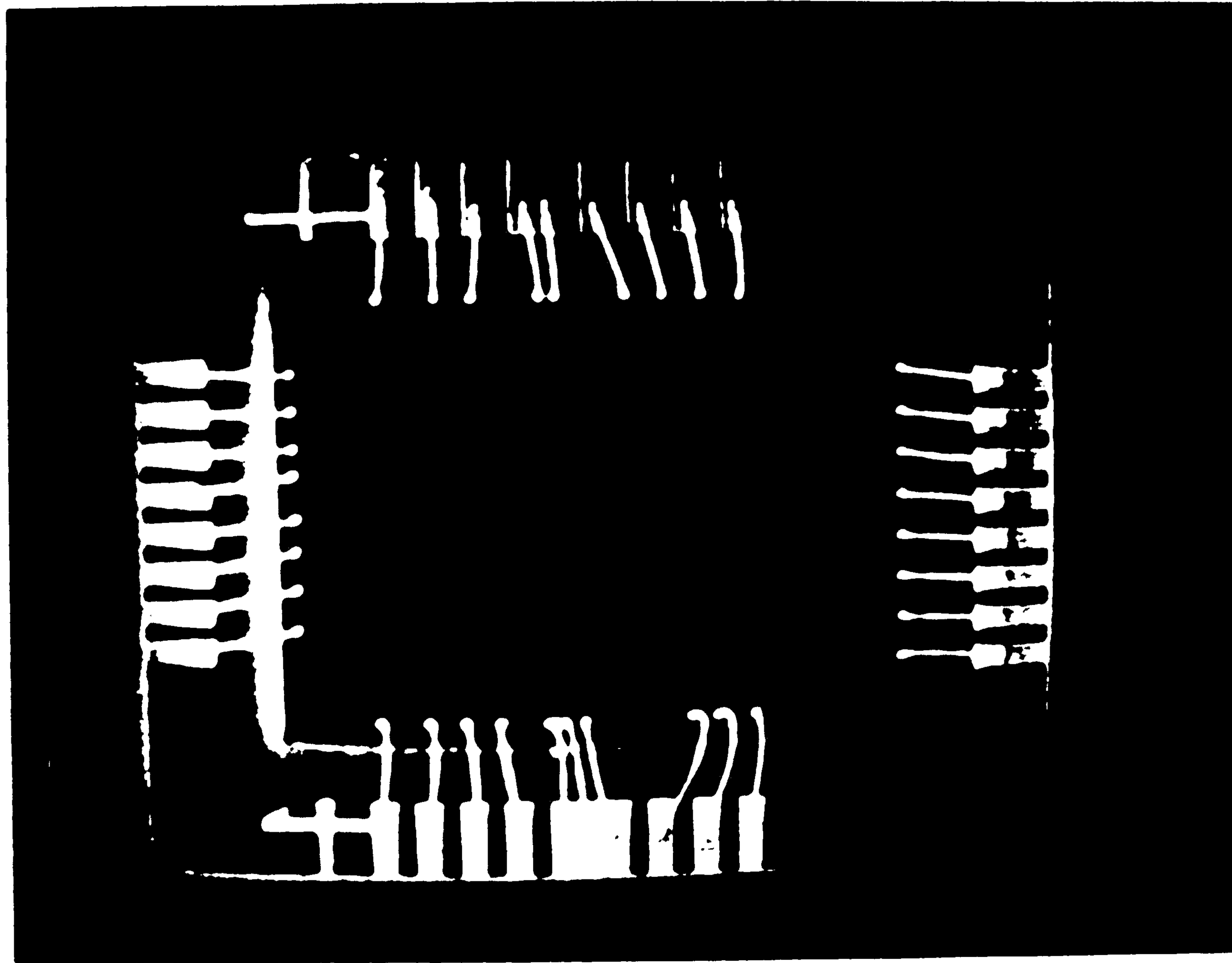
The overall approach to the package alignment remains the same, in that certain reference objects are automatically located and their displacement translated into changes in mirror positions to realign the package, but the choice of reference object and the algorithm used to locate it has changed. It is still preferable to use a reference object which is located on the package instead of on the die, because the lighting, magnification, and focus are set for the package, not the die. In addition, the die position is somewhat variable within the package, and the wire bond inspection algorithm must know the exact location of the package bond pads, but not of the die bond pads. However, using package bond pads as reference objects poses the difficulty of distinguishing one bond pad from adjacent similar or identical bond pads. This difficulty can be circumvented by choosing a unique object on the package as a reference object. This is the approach taken by the new alignment scheme.

Both alignment methods can be classified as feature analysis methods, but using a different reference object means that the features extracted during the teach phase

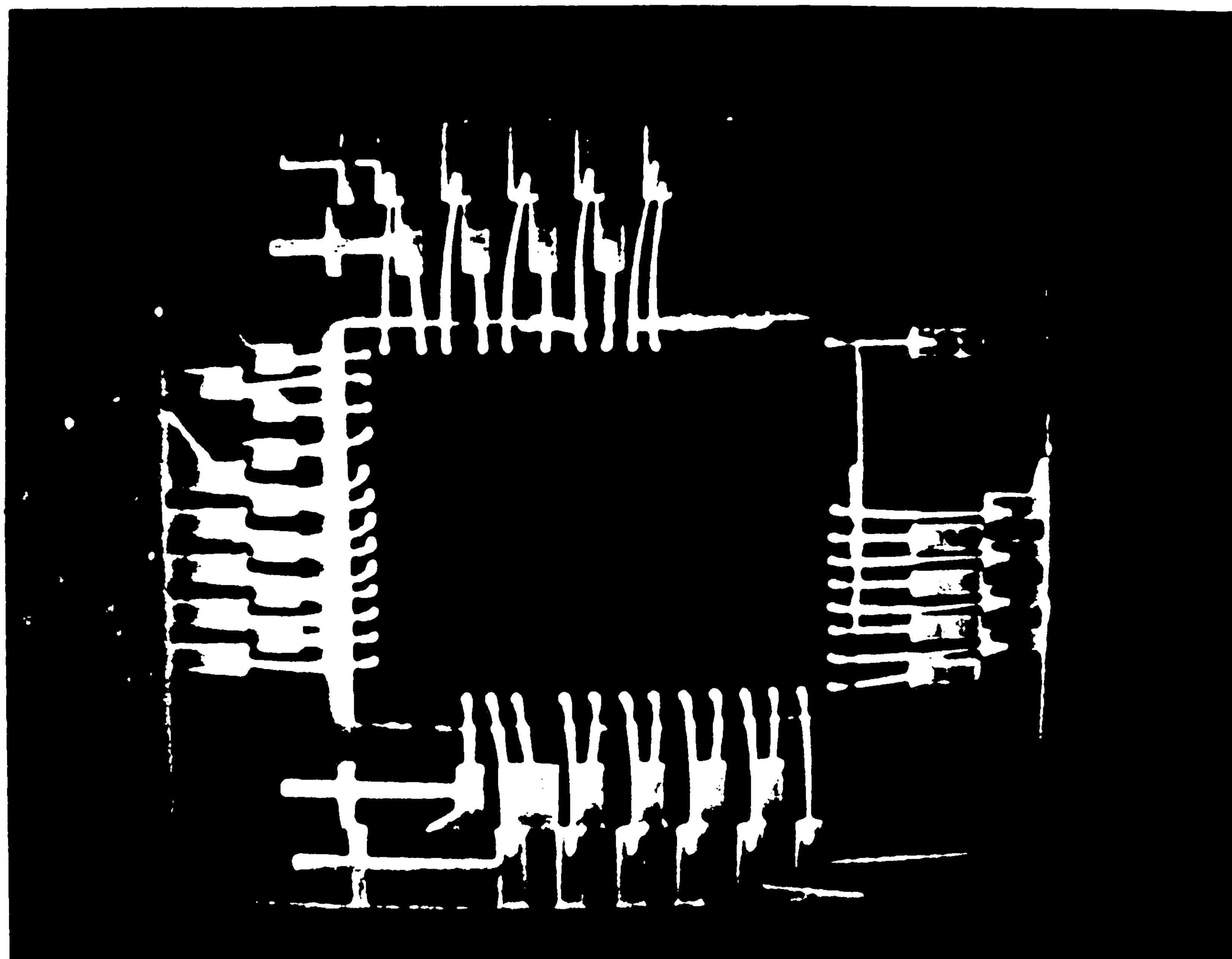
must change, and the algorithm used to find the best match for these features must change. Formerly, the features describing the package bond pads used as reference objects were the gray-level threshold indicating a change from background to bond pad and the expected location of these bond pads. The matching algorithm did not search for the best match to the features, but used the first two good matches found. The new alignment method extracts a much more detailed description of the reference object, and finds the best match to this object.

The unique reference object used in the new method is a crossbar shape and its surrounding background (Figure 4.2a). The crossbar, consisting of two intersecting vertical and horizontal bars, has the same reflectivity as the package bond pads, so that it also appears as a light object on a dark background. Like the package bond pads, it has straight, well-defined edges which are parallel to the x and y axes and a strong gray-level contrast between the object and the background. Using the crossbar as the reference object has two advantages over using the package bond pads: there are no wires bonded to it or crossing over it to introduce inconsistencies in the gray-level of the feature, and it forms a unique reference object on the chip.

This crossbar is present on all of the ceramic devices sampled, and for thirteen of the packages is the only crossbar in that area of the package. The remaining nine



a) One set of bond pads



b) Two sets of bond pads

Figure 4.2: Reference Object: Crossbar

packages have two sets of bond pads, inner and outer, with a crossbar for each set of bond pads, so that the two nearly identical crossbars are adjacent (Figure 4.2b). However, in all cases these crossbars can be distinguished by the amount of dark background surrounding them. For example, choosing the window shown in Figure 4.3, which includes the crossbar and as much of the background as possible, provides a unique reference object.

Although the reference object used is a crossbar, in fact the alignment algorithm uses only certain properties of the crossbar for the actual alignment. The setup task is simplified by using a crossbar, which is easy for an operator to identify, instead of requiring the operator to locate areas of the package which have certain abstract features. This means that the same alignment algorithm could be used with a new setup method for devices without the crossbar feature.

4.2.2 Design of Setup and Alignment Phases

The setup and the alignment steps needed for aligning devices correspond to the teach mode and inspect mode for the system. The setup task for alignment is part of the teach phase, in which parameters for a new device type are taught. Here the operator identifies the crossbar, and appropriate features are extracted from that crossbar. The actual alignment takes place during inspection, when the

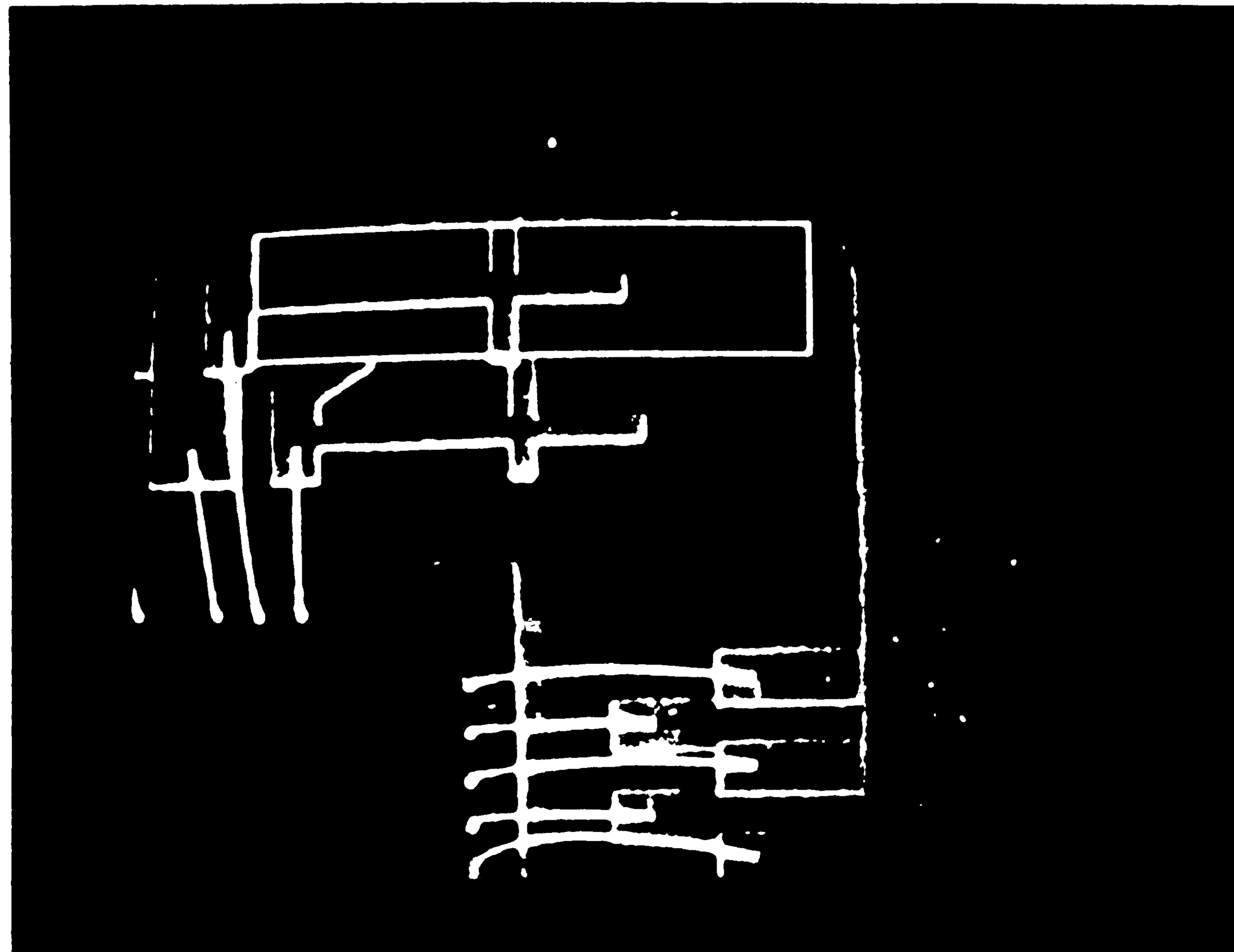


Figure 4.3: Unique Reference Object

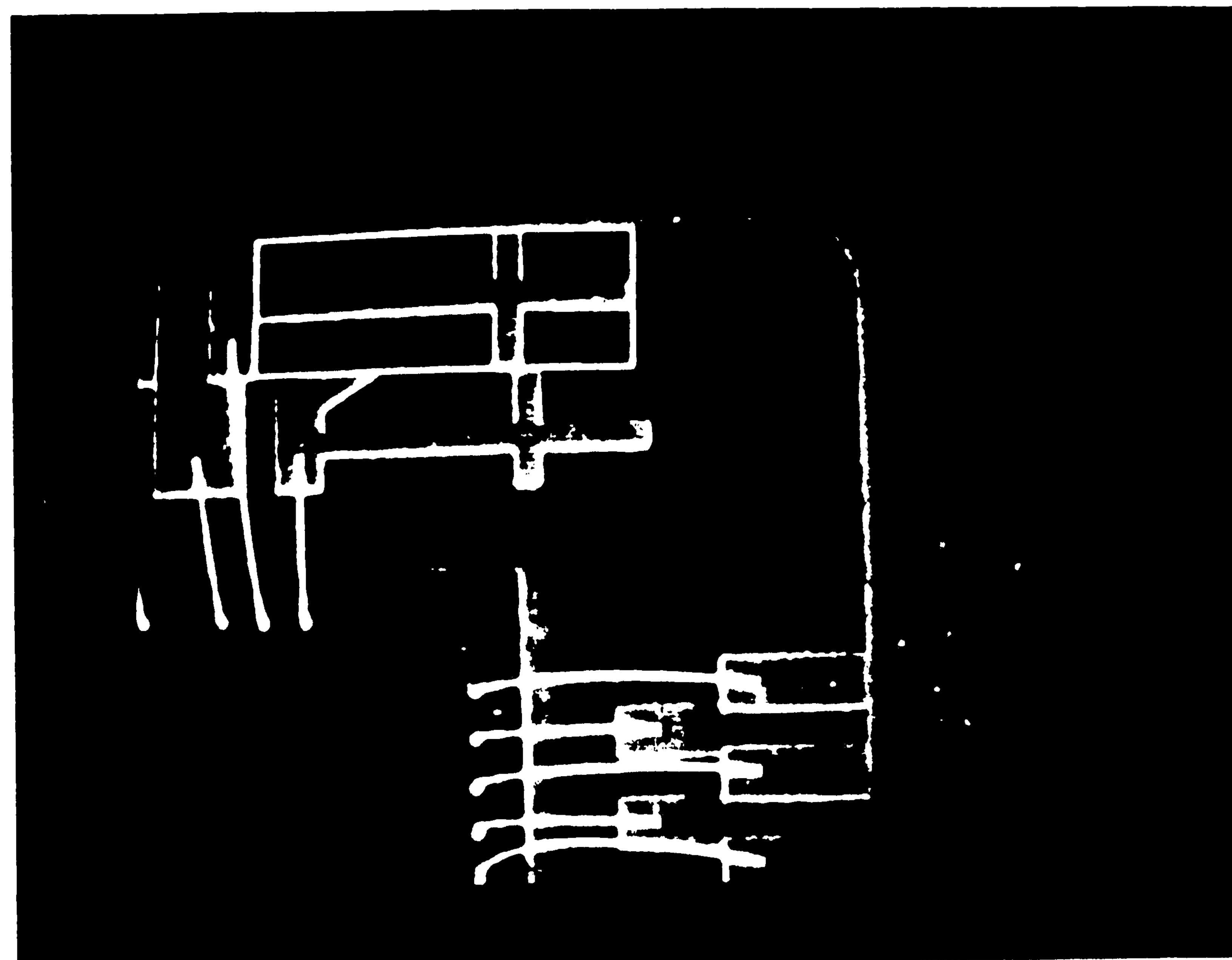
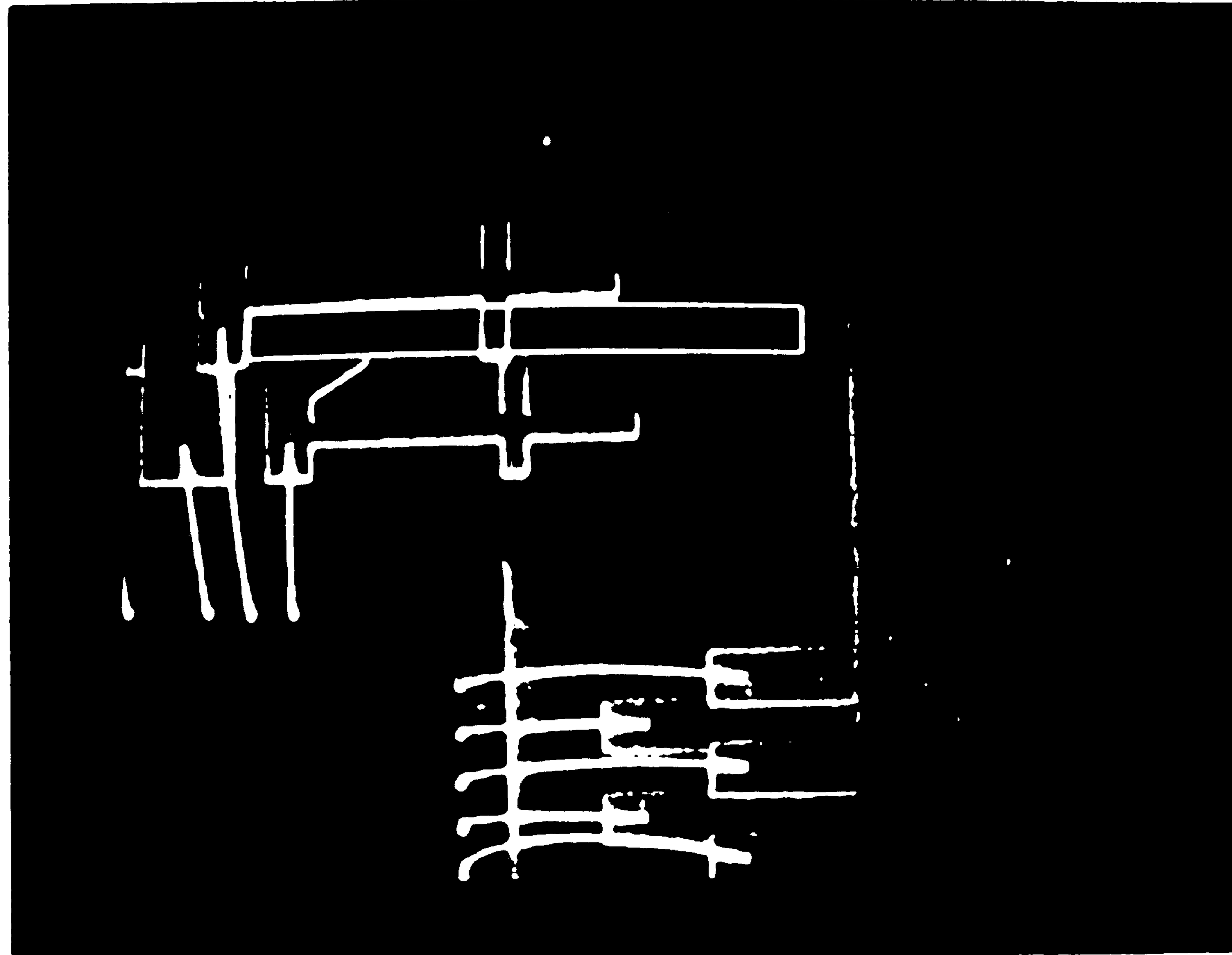


Figure 4.4: Frame Around Crossbar

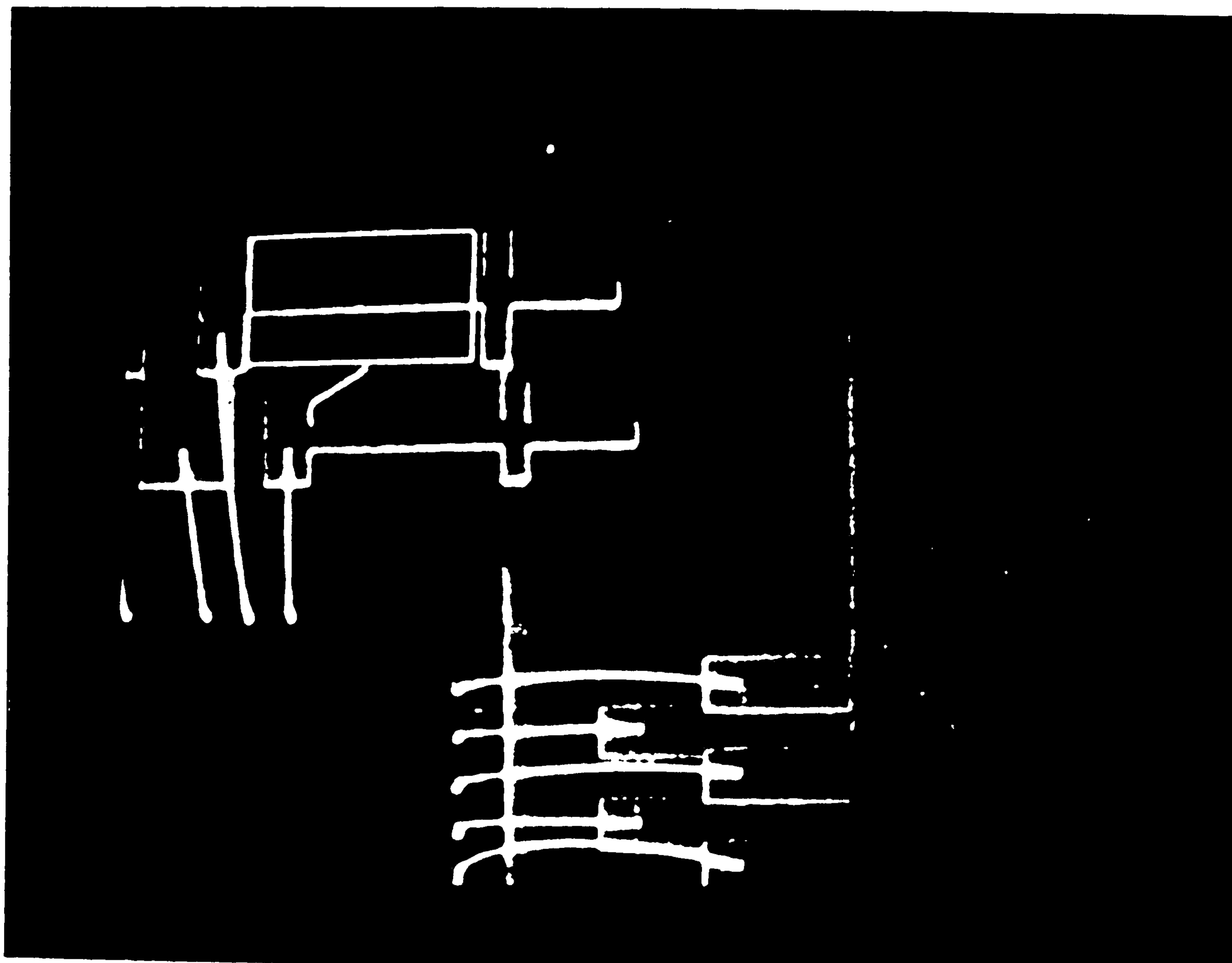
extracted features are retrieved and used to automatically align each device after it is placed under the inspection station.

During the teaching of new devices, the features needed for alignment must be extracted with minimum difficulty for the operator. Processing speed is not critical here, since the software need only be fast enough to interact smoothly with the operator. The operator identifies the crossbar to be used, and encloses it in a rectangular frame using the mouse (Figure 4.4). The software then analyzes the scene within the frame to determine the size, shape, and location of the cross. However, the scenes used for alignment consist of two windows, one each for the x and y alignment: one window contains solid background and a single vertical bar (Figure 4.5a), and the other contains background and a horizontal bar (Figure 4.5b). After the operator frames a second area which includes the crossbar and all of its surrounding background (Figure 4.3), the appropriate features can be extracted, and they are guaranteed to be unique.

Determining the alignment correction involves matching the reference object to the corresponding object in the image, which can be done in one of two ways. The first approach uses a search window the size of the reference object, and checks whether the object is within the search window. If the object is not found, the search window is



a) x alignment window



b) y alignment window

Figure 4.5: Alignment Windows

repositioned for another search, according to some heuristic. This process repeats until the object is found. The second approach uses a search window which is enlarged to cover the total possible displacement, so that the object is guaranteed to be within the search window. The speed of the first approach depends on how good the heuristic used to reposition the search window is, but when the heuristic predicts the object's position well, this approach can be much faster than the second, especially if the possible displacement is large. However, since the second approach uses an exhaustive search, it finds the best match, whereas the first approach stops its search after finding an adequate match. Thus the first approach is preferable only if a good heuristic can be developed for repositioning the search window, and if a reliable criterion can be determined for deciding when an adequate match has been found.

During the inspection phase, the alignment must be corrected in less than half a second to maintain the two second cycle time. The second approach to searching for the reference object, the exhaustive search, is used because of its simplicity and because and it finds the best match. To make this search fast enough, instead of searching pixel by pixel, the window is sampled at intervals that guarantee that the reference object will not be missed.

4.3 Implementating the Alignment

4.3.1 Setup Phase

The illumination of the device under the camera is designed to maximize the visibility of the wires, since they are the focus of the inspection, but this lighting is not ideal for illuminating the reference object, the crossbar. The two difficulties which arise are changes in light intensity across the area of interest, leaving some areas better lit than others; and the angle of the lighting, which highlights some edges of the object and leaves others dark.

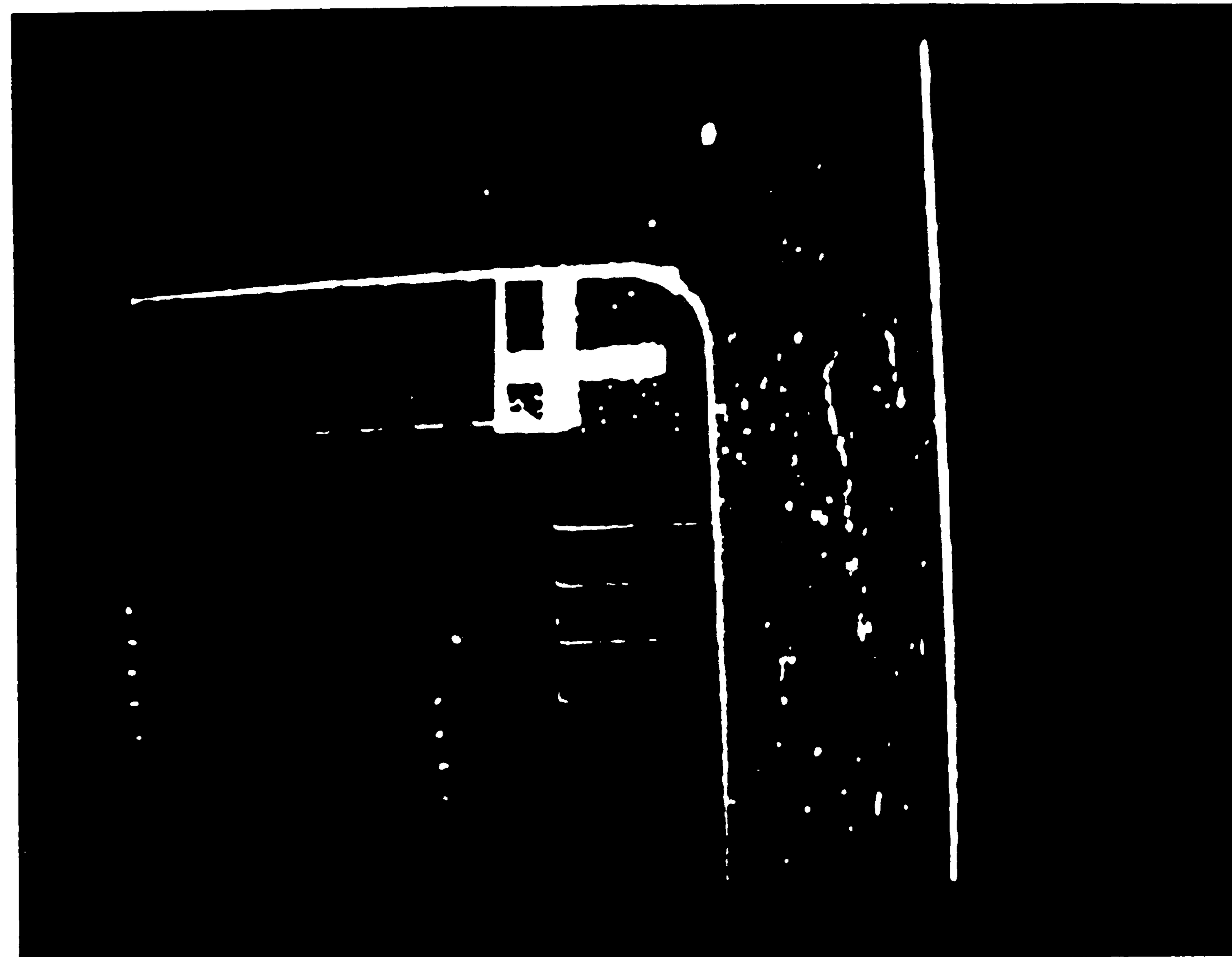


Figure 4.6: Segmentation Using a Threshold

Because the light intensity changes across the region, the image cannot be segmented by thresholding. Figure 4.6

shows a case where the gray-level threshold used to separate the object from the background in the upper right portion of the scene does not work in the lower left corner. In these circumstances, a segmentation method that uses a gradient instead of a threshold is more successful, because the gradient reflects the relative change in gray-level across the scene rather than the absolute change.

The 3x3 edge detection templates shown in Figure 4.7 use a gradient to locate edges. The template is positioned on a 3x3 pixel area and each pixel value is multiplied by its corresponding weighting factor in the template. The sum of the weighted pixel values is the central pixel's response to the template. Because the template weighting factors are 1's and -1's, the response essentially measures the gray-level change or gradient at that pixel. By using a two-dimensional template, the response depends also on the gradient of the rows above and below (or columns to the left and right), so that the response is highest when the template is positioned over an edge.

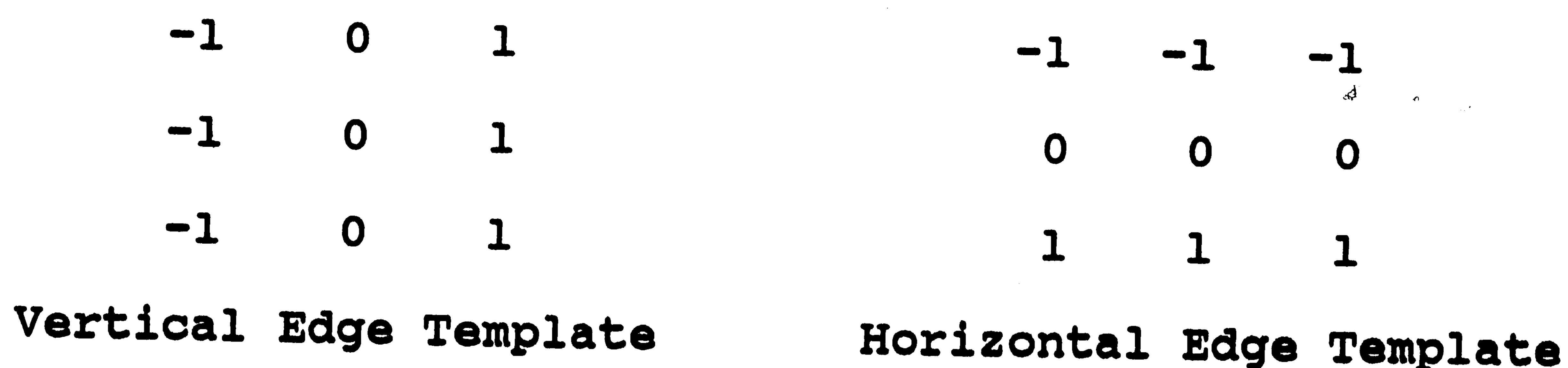


Figure 4.7: Edge Detection Templates

The edge templates are used to locate horizontal and vertical edges within the frame drawn by the operator, then these edges are used to segment the scene into a crossbar and background. The edge templates are applied to each pixel in the scene, and if the magnitude of the pixel's response to the template is above a set threshold, the pixel is on an edge.

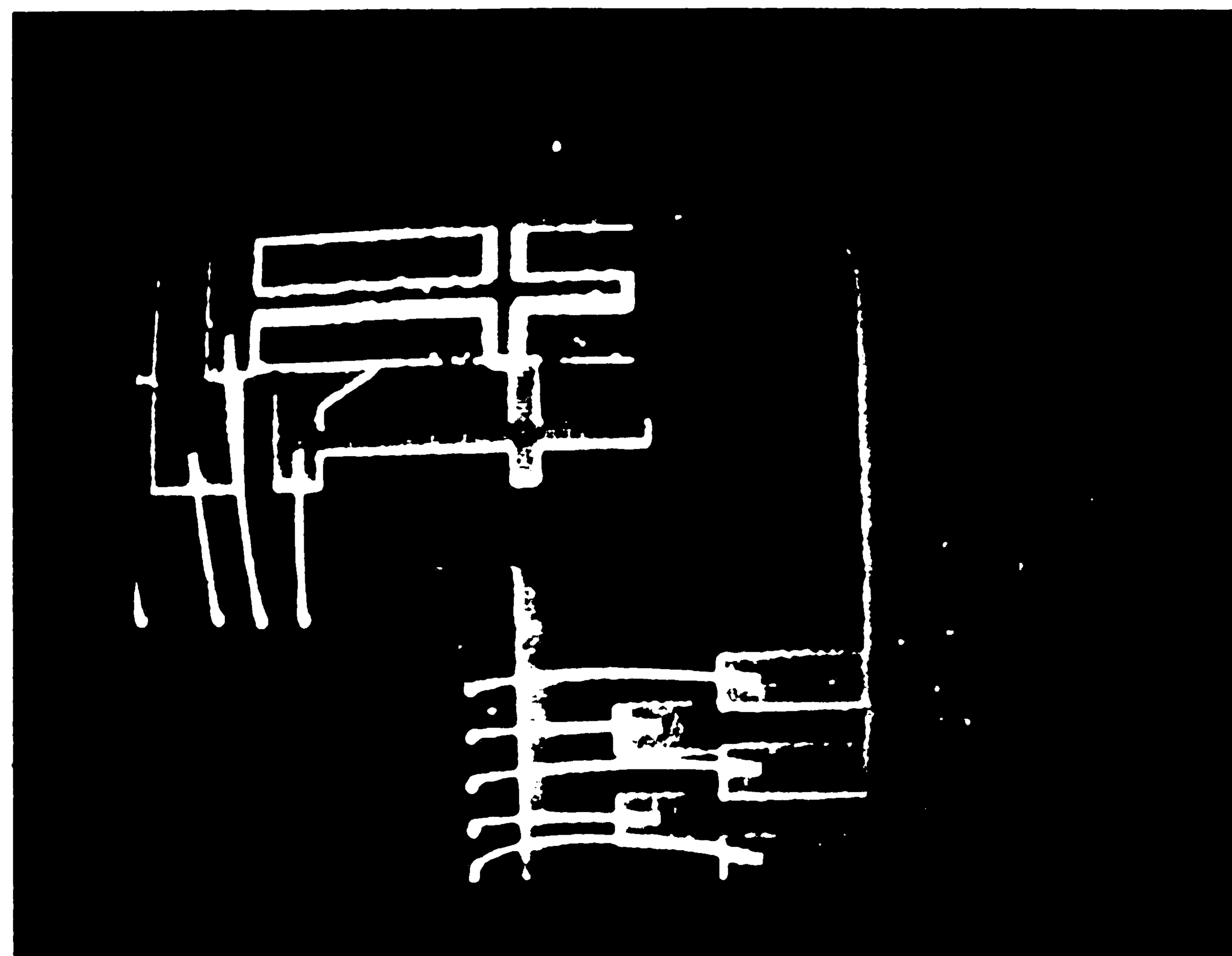


Figure 4.8: Applying Edge Detection Templates

The edge detection threshold is determined emperically, by choosing the highest (most discriminating) value that detects all edges, including the darkest. Figure 4.8 shows the scene after the edge detection templates have been applied. The pixels whose response to either template

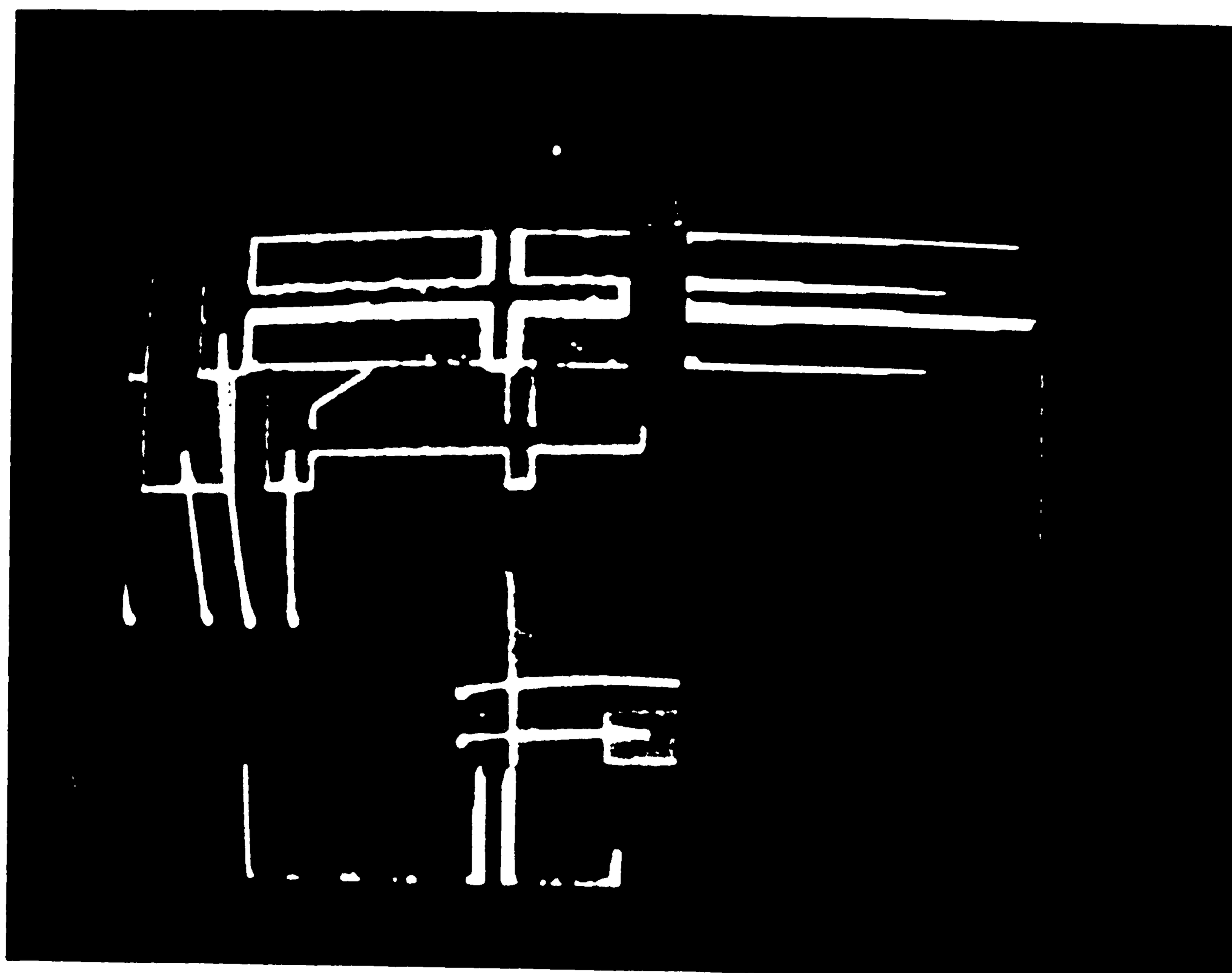
was above the threshold are shown as white pixels. The threshold value permits a trace of even the darkest edge, and because the lighting is controlled, this same threshold value works equally well for all twenty-two ceramic devices sampled.

Segmentation is accomplished by selecting the two vertical and two horizontal edges belonging to the crossbar. The ends of the crossbar are assumed to be at the frame boundaries. But before the crossbar edges can be selected, the edge pixels must be classified into horizontal and/or vertical edges.

Defining the vertical edges requires knowing how many vertical edge pixels are at each x location, which of these adjacent locations are part of the same vertical edge, and the span of y-coordinates each edge covers. The vertical edge pixels are the pixels whose response to the vertical edge template is above the threshold. The number of vertical edge pixels at a particular x location gives a measure of how strong the edge is, so that insignificant edges can be eliminated. The count of vertical edge pixels at each location is stored in a histogram, which is then used to determine the x location, width, and strength of vertical edges. A sample of the vertical edge histogram is shown in the lower portion of Figure 4.9a. The y location and length of the edge are determined by taking the minimum and maximum y-coordinates of all edge pixels making up that

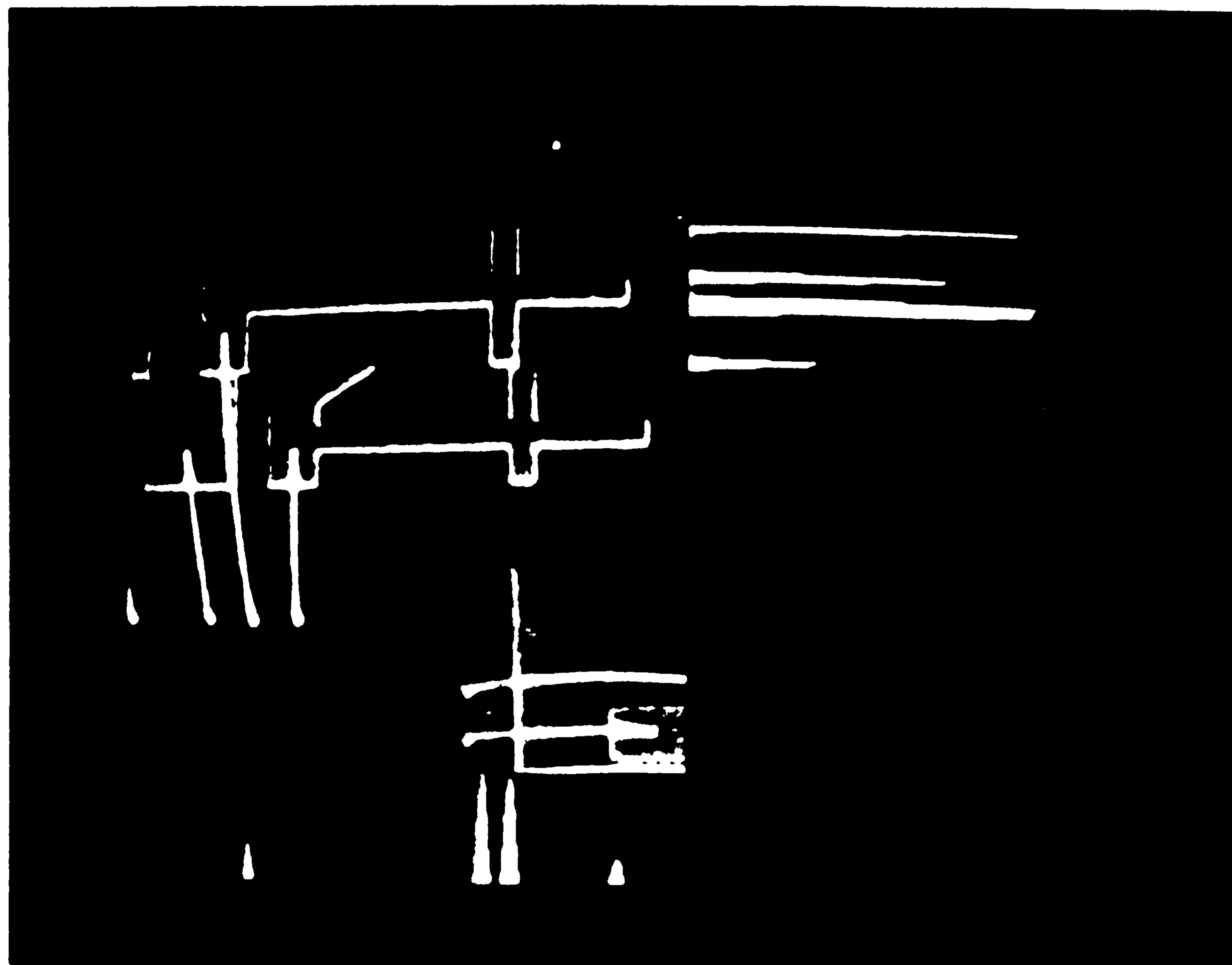
edge.

The number of vertical edges in the window corresponds to the number of peaks in the edge pixel histogram. The peaks in the histogram are found by first smoothing the histogram (Figure 4.9b), then taking the derivative of the histogram (Figure 4.9c). The derivative is positive wherever the histogram is rising, and negative where the histogram is falling. Thus a change from positive to negative in the derivative corresponds to a peak in the histogram. The start and end of each of these peaks can likewise be determined from the derivative.

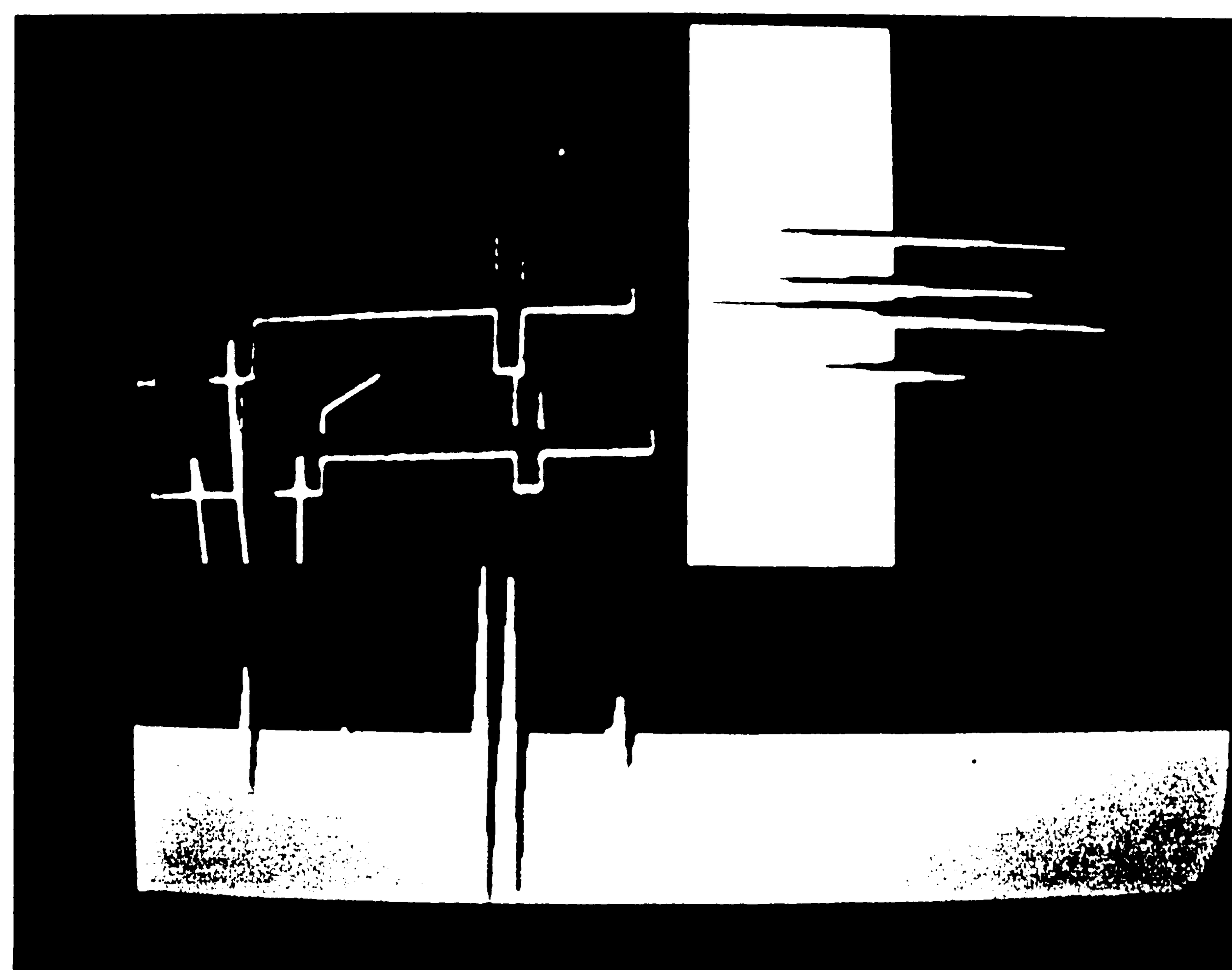


a) edge pixels histograms

Figure 4.9: Locating Edges using Histograms



b) smoothed edge pixel histograms



c) derivative of edge pixel histograms

Figure 4.9 continued

Before choosing the two vertical edges belonging to the crossbar, some compensation must be made for the inadequacies of the lighting. The angle of the lighting causes some edges of the crossbar to be highlighted with bright lines, which come through the edge detector as two edges instead of one. Double edges are corrected using a simple distance criterion: two edges are combined into one when they are very close, within four pixels of each other. In all twenty-two devices, the actual crossbar edges are at least ten pixels apart, and the double edges detected are never more than four pixels apart.

The vertical edges of the crossbar are selected from the strongest edges which span the frame from top to bottom. The edge containing the maximum value in the histogram is examined, and if it spans the frame from top to bottom and is not located at the right or left side of the frame, it is designated as a crossbar edge. This edge is eliminated from the histogram, and the search is repeated until the second edge is found.

The determination of the horizontal edges of the crossbar is done in the same fashion, using a histogram of the count of horizontal edge pixels at each y location. The horizontal edge histogram is displayed on its side, at the right edge of Figure 4.9. Since the crossbar is bounded by its two vertical edges, its two horizontal edges, and the operator's frame, the region within the frame has been

segmented, and the next step is to extract the features which will be used for alignment.

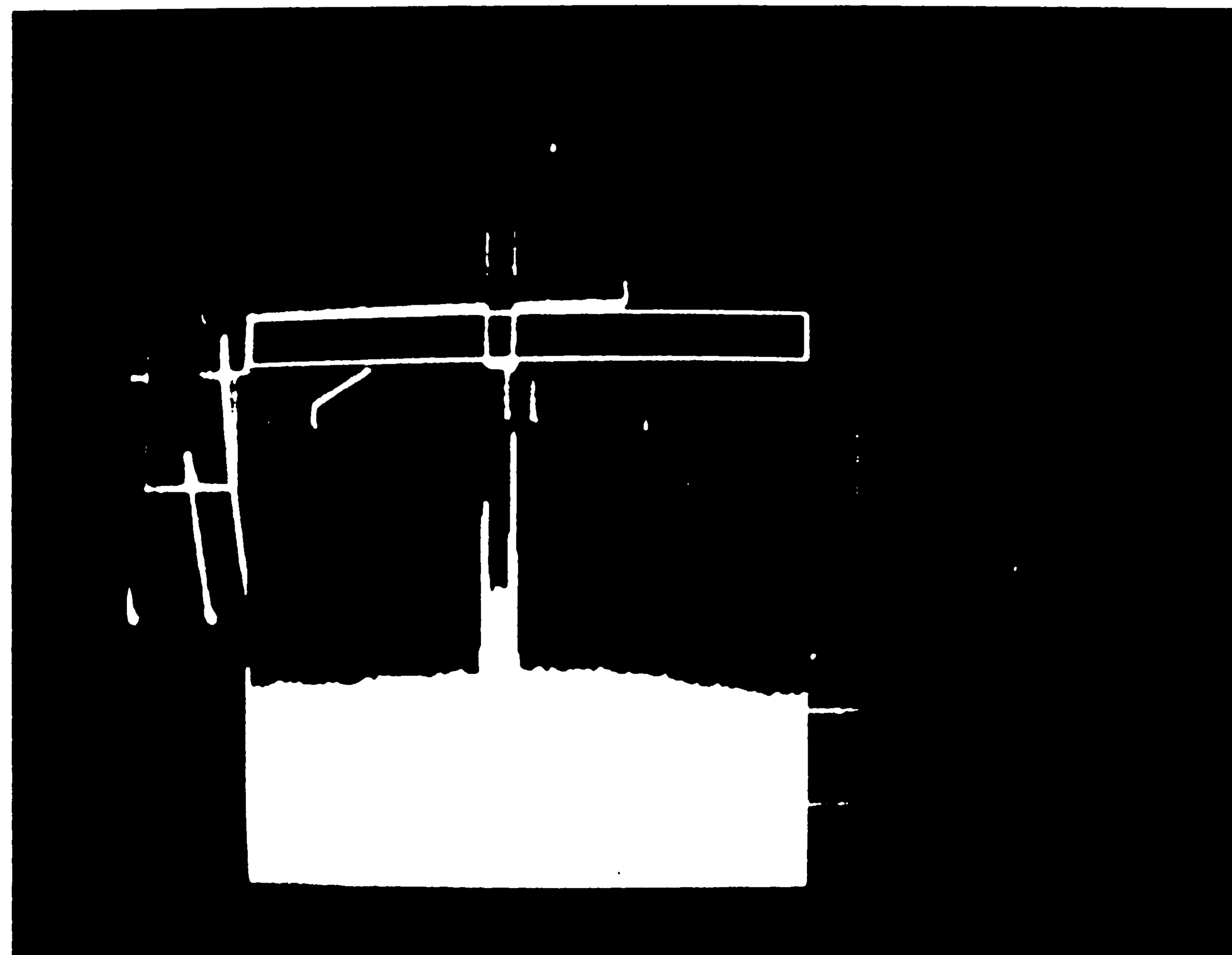
The features extracted from the two windows used for alignment (Figure 4.5) are the pixel locations of the four sides of the window, the two edges of the bar, the profile of a cross-section of the window, the mirror positions used to view that area of the device, and a boolean indicating which of the windows should be located first.

The cross-section profile, together with the bar edge locations, represents a compression of the pixel by pixel copy of the window that would be stored if the template matching approach were used to locate these alignment scenes. The alignment scenes are chosen so that they vary in only one direction: for the x alignment window (Figure 4.5a), all horizontal cross-sections are approximately the same, and for the y alignment window (Figure 4.5b), all vertical cross-sections are approximately the same. Thus each of these windows can be compressed to one dimension without losing significant information. The cross-section profile is composed of the average of all cross-sections in the window. Figure 4.10a shows a plot of this one-dimensional cross-section profile: the plot height indicates the relative gray-level at each pixel in the profile.

Another form of compression in done within the cross-section profile by averaging the background across the

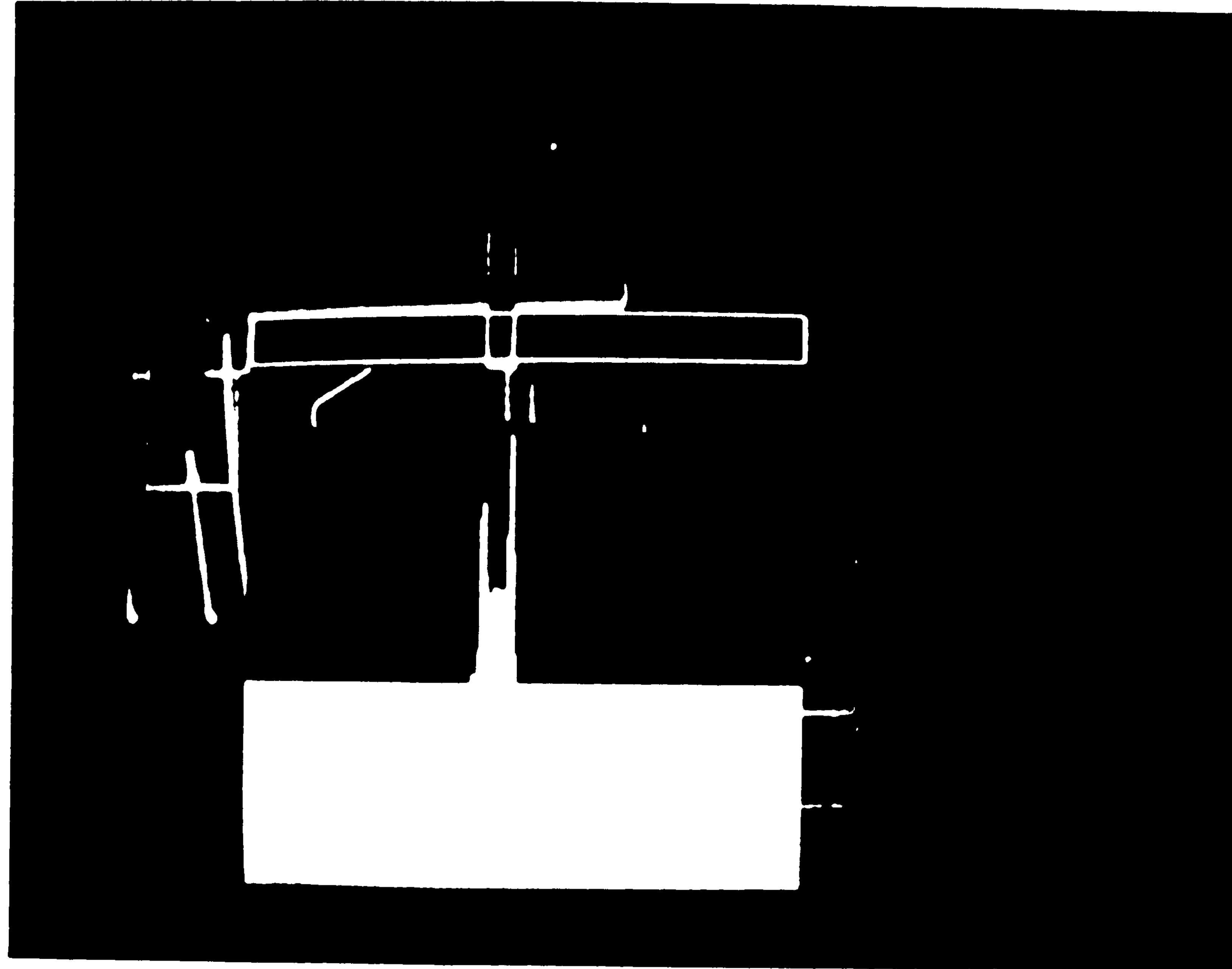
window and replacing the background in the cross-section profile by this average background (Figure 4.10b). The bar edge locations are used to separate the bar from the background. The compressions do not represent a significant loss of information, as is shown in Figure 4.10c, where the alignment window is recreated from the cross-section profile, and the compressions are necessary for locating the alignment scenes within the half-second time limit.

Figure 4.11 diagrams the tasks performed during the setup phase.

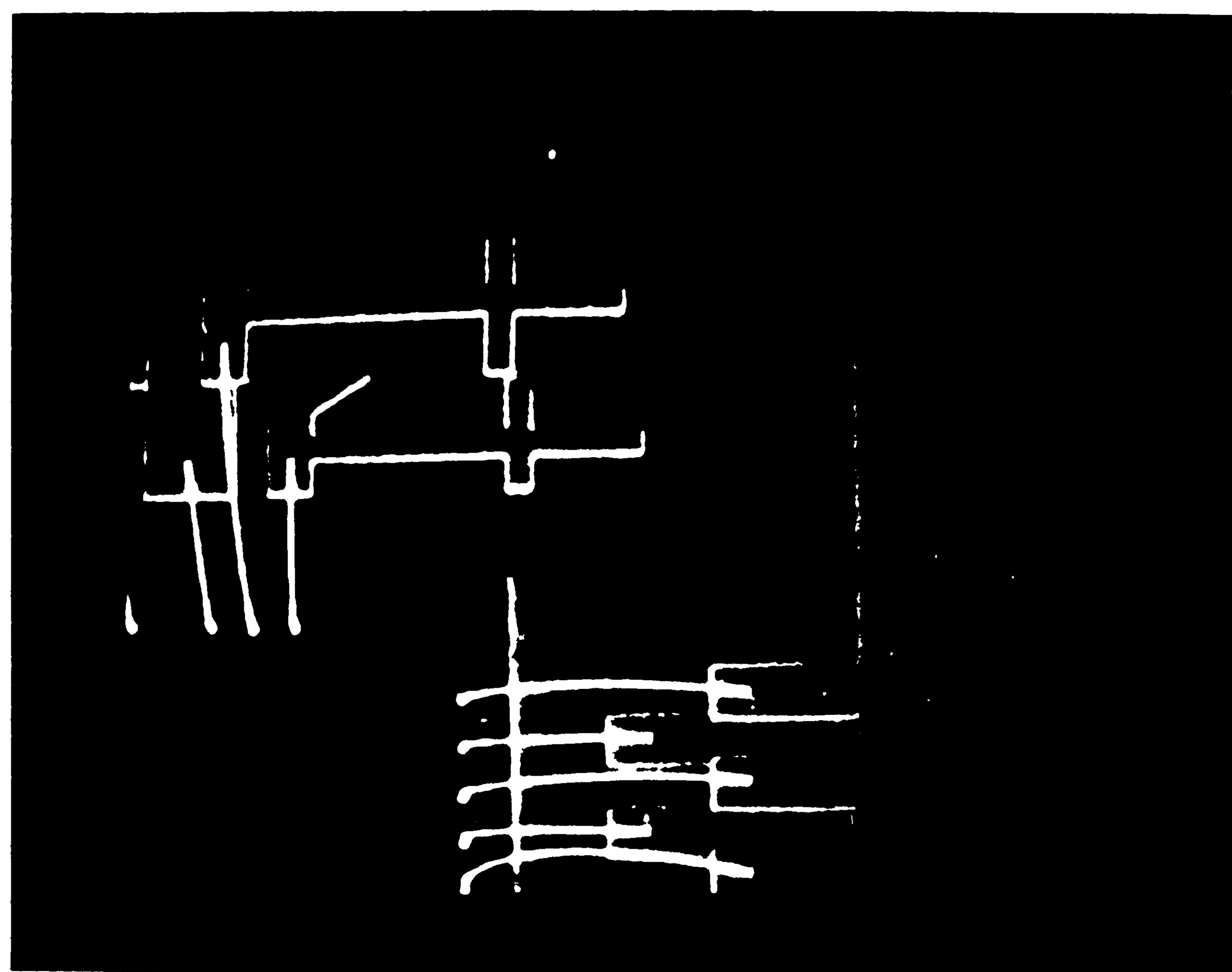


a) plot of averaged horizontal scans

Figure 4.10: Cross-Section Profile



b) plot of cross-section with averaged background



c) x alignment window re-created from cross-section profile

Figure 4.10 continued

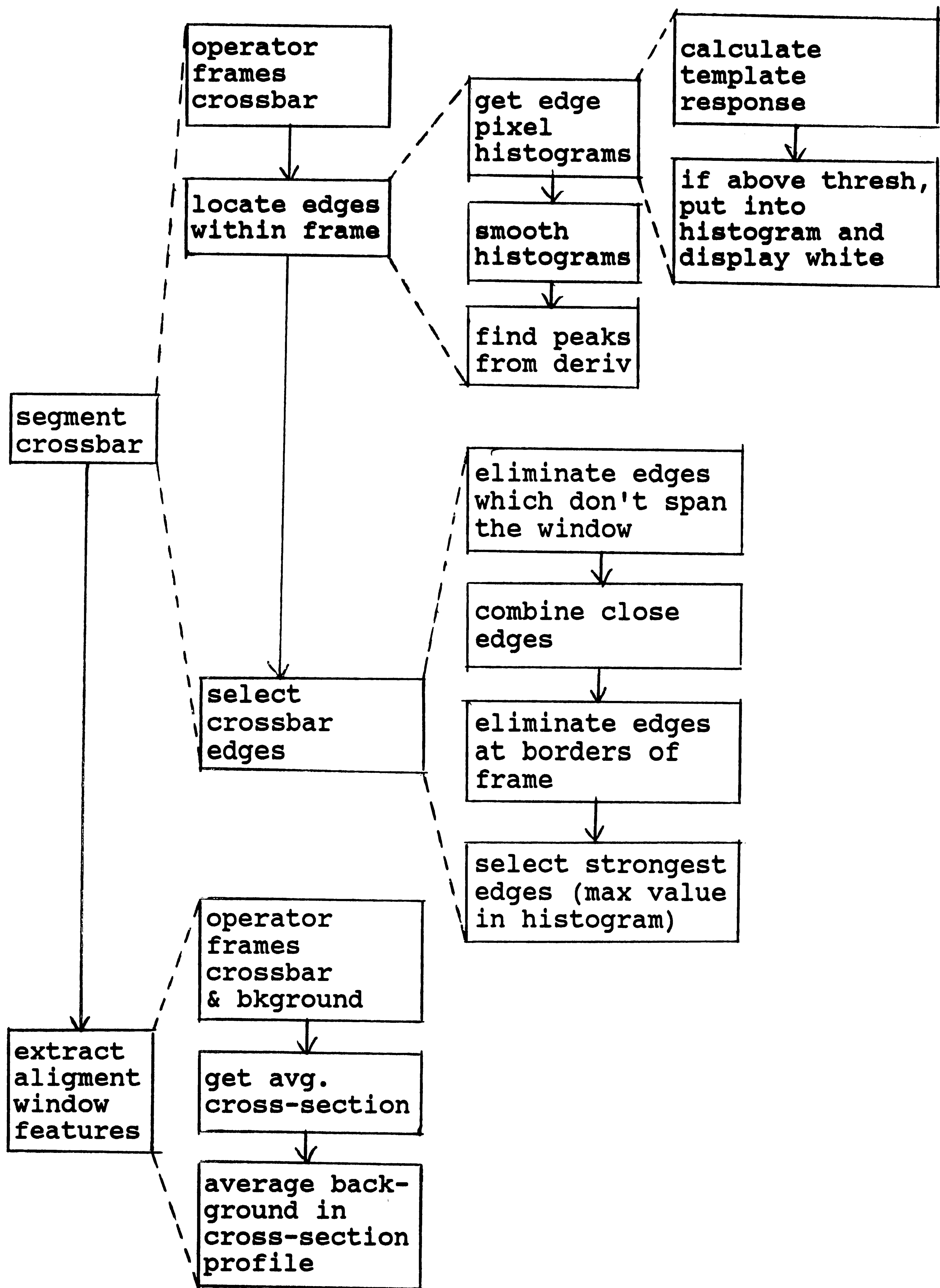


Figure 4.11: Tasks Performed during Setup Phase

4.3.2 Alignment Phase

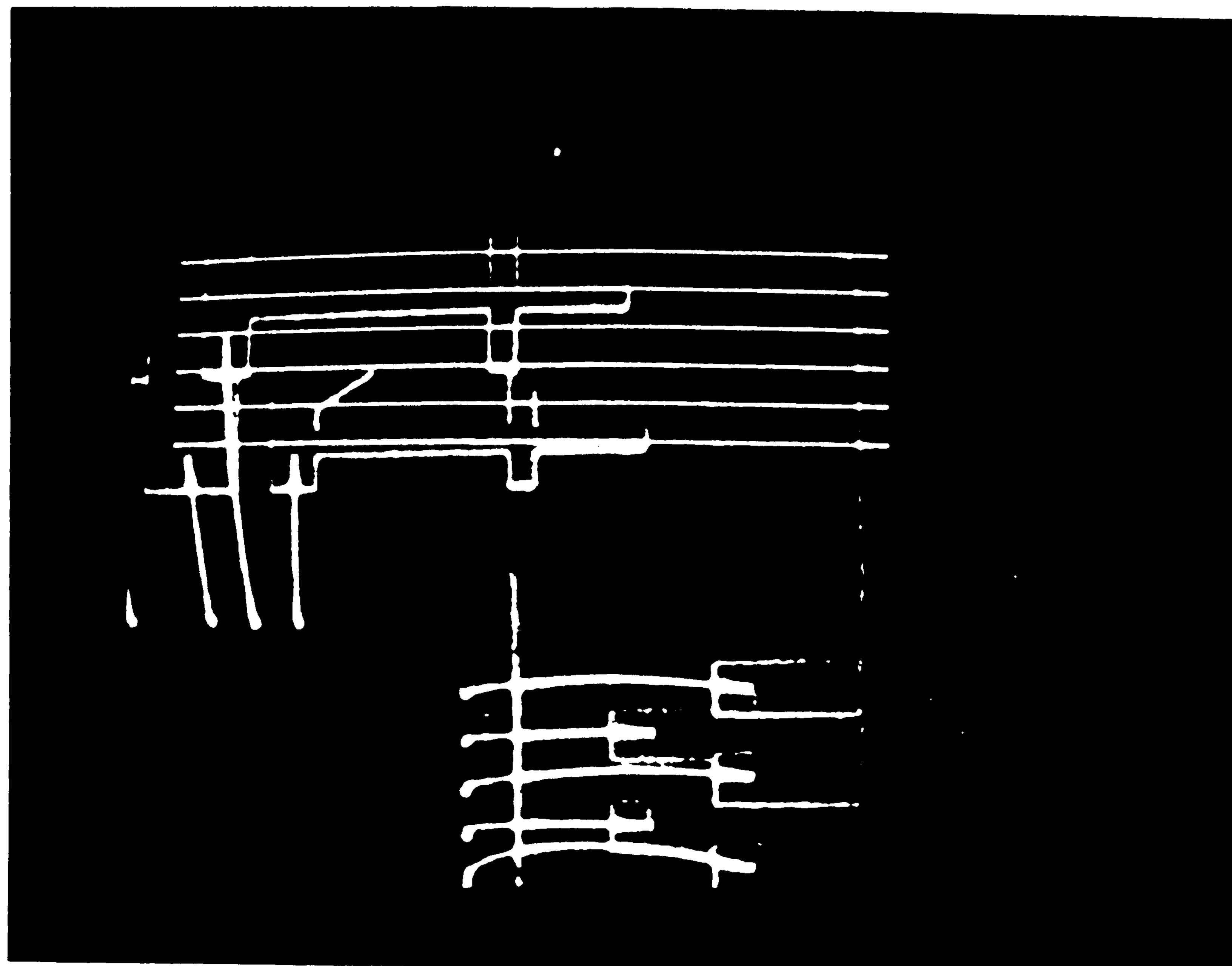
The edge templates used for teaching the reference objects (Figure 4.7) cannot be used during inspection for the same reason that the template matching approach cannot be used: both are much too slow without special hardware. In both cases the template operates on a block of pixels the size of the template, and the template is applied at each pixel in the area of interest. Thus the number of operations involved is approximately:

(number of entries in the template) X (number of pixels in in the search window)

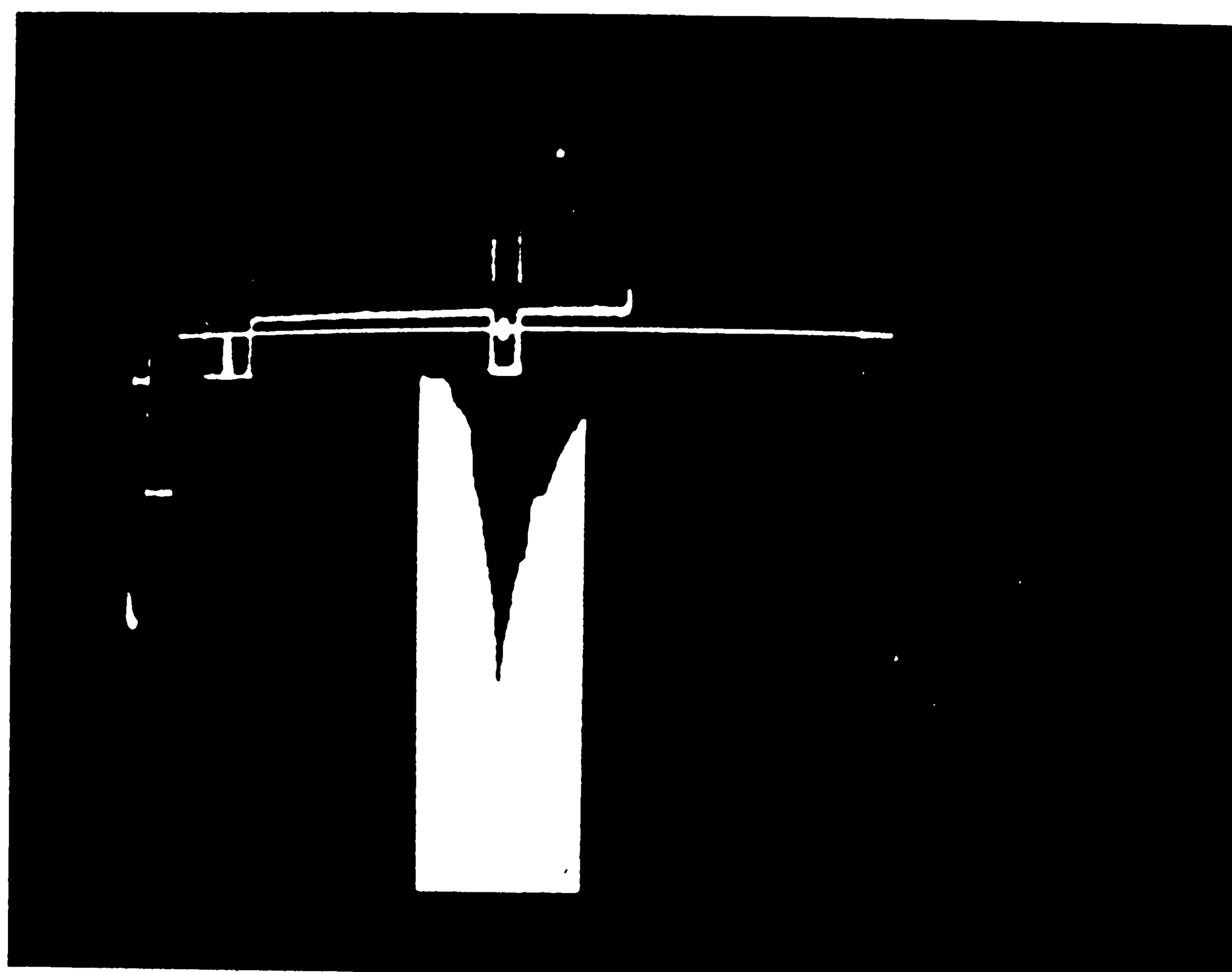
= (template width) X (template height) X (window width) X (window height).

The number of operations needed to find the best correlation with the template can be decreased by a factor when a one-dimensional template, the cross-section profile, is used instead of the usual two-dimensional template. Then the number of operations is: (template width) X (window width) X (window height). Even so, computing all correlations within the area of interest is too slow for this application: applying a one-dimensional template at each pixel in the search window takes minutes instead of seconds. The number of operations can be decreased by another factor by sampling the region.

Sampling the region can once again take advantage of the fact that the alignment scenes are essentially one dimensional. When searching for the vertical bar to correct

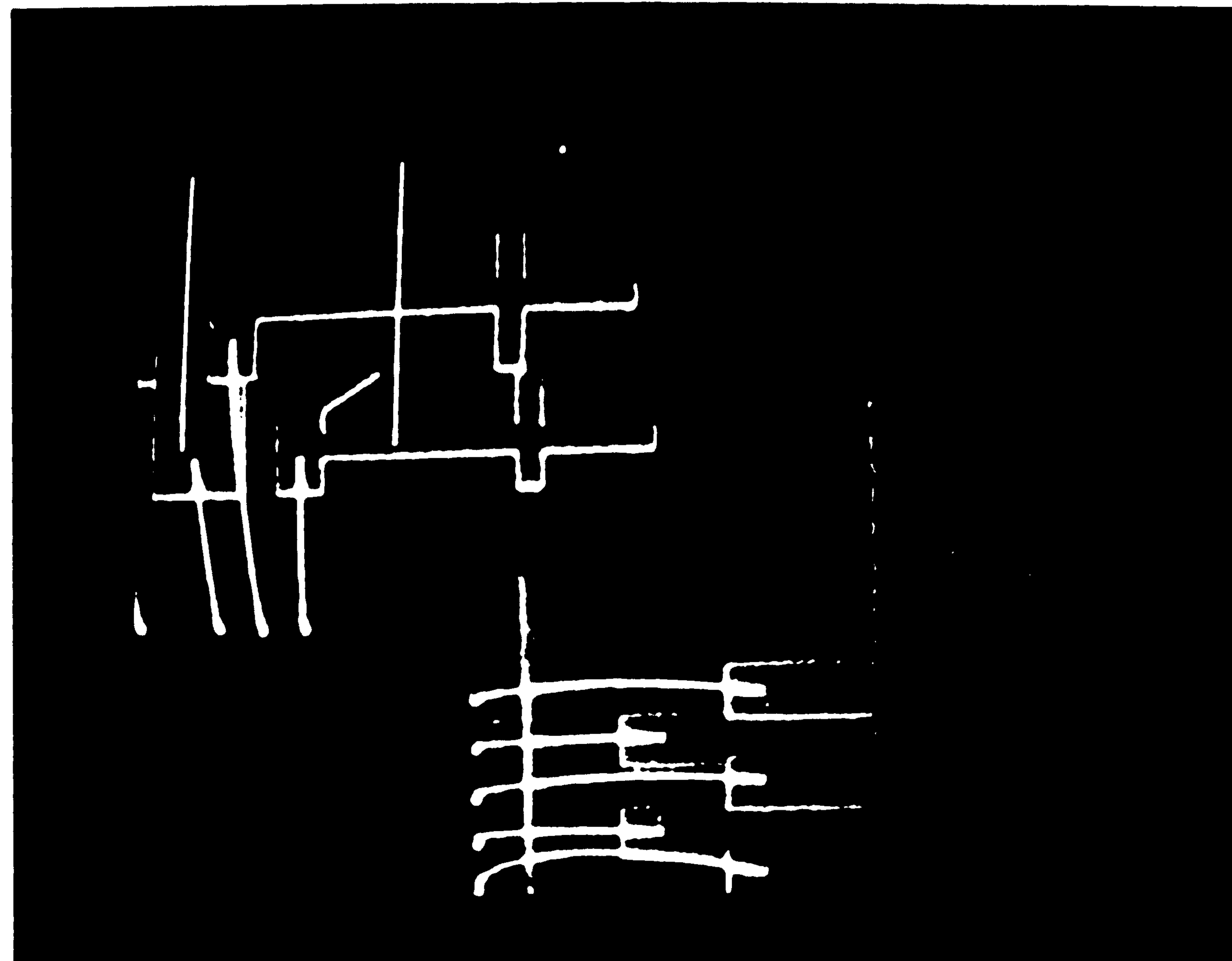


a) scanning the x alignment window

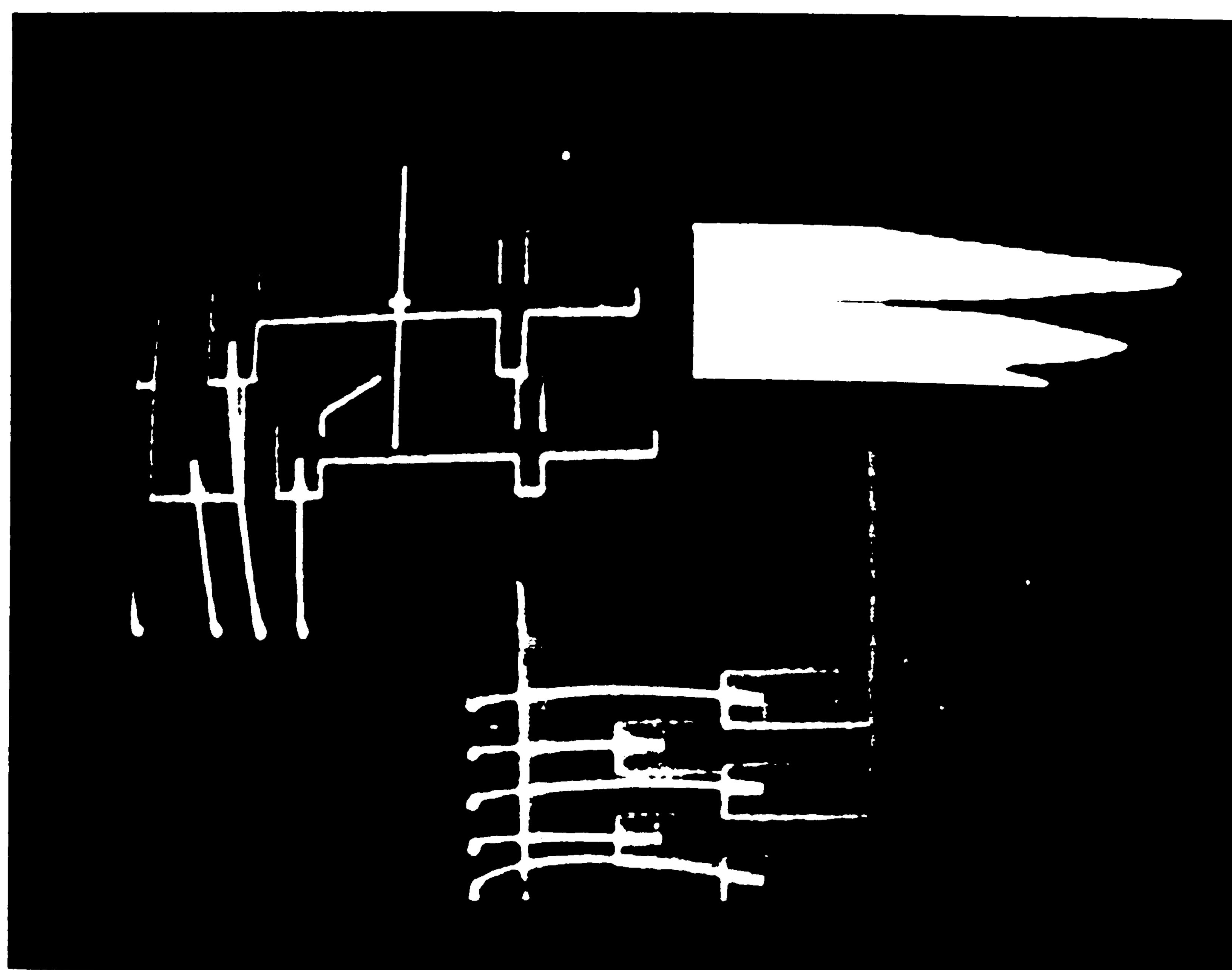


b) scan containing best correlation

Figure 4.12: Locating Vertical Bar



a) scanning the y alignment window



b) scan containing best correlation

Figure 4.13: Locating Horizontal Bar

the x alignment, the area is sampled by examining horizontal scans separated by a sampling interval which is slightly less than the height of the alignment scene (Figure 4.12a). The one-dimensional template is applied to each pixel on a horizontal scan, and since one of these horizontal scans crosses the vertical bar, the best correlation appears in that scan at the center of the vertical bar. In 4.12b, the scan containing the best correlation is marked, and the correlation at each pixel in that scan is plotted, with the plot height indicating the degree of correlation: the best correlation occurs where the plot height is at a minimum. The search for the horizontal bar is analogous, using vertical scans and a sampling interval just less than the width of the alignment scene (Figure 4.13).

The correlation between the one-dimensional template and the scan line is calculated by subtracting the template values from the pixels in the scan line, so that the best correlation occurs where the difference between the template and the scan line is smallest. However, when calculating the correlation in this way, the alignment correction for both directions takes about one and a half seconds, which is one second too long. The correlation calculation can be made faster by taking advantage of the fact that the background in the cross-section profile has a constant gray-level. The difference between the scan line and the average background is calculated only once, and is stored as the

background response. Then when calculating the correlation at a particular pixel, only the crossbar portion of the template is subtracted from the pixels in the scan line to find the crossbar response, and the appropriate background response is added in. Using these streamlined correlation computations, the alignment correction for both directions can be done within the half-second goal.

Figure 4.14 diagrams the tasks performed during the alignment phase.

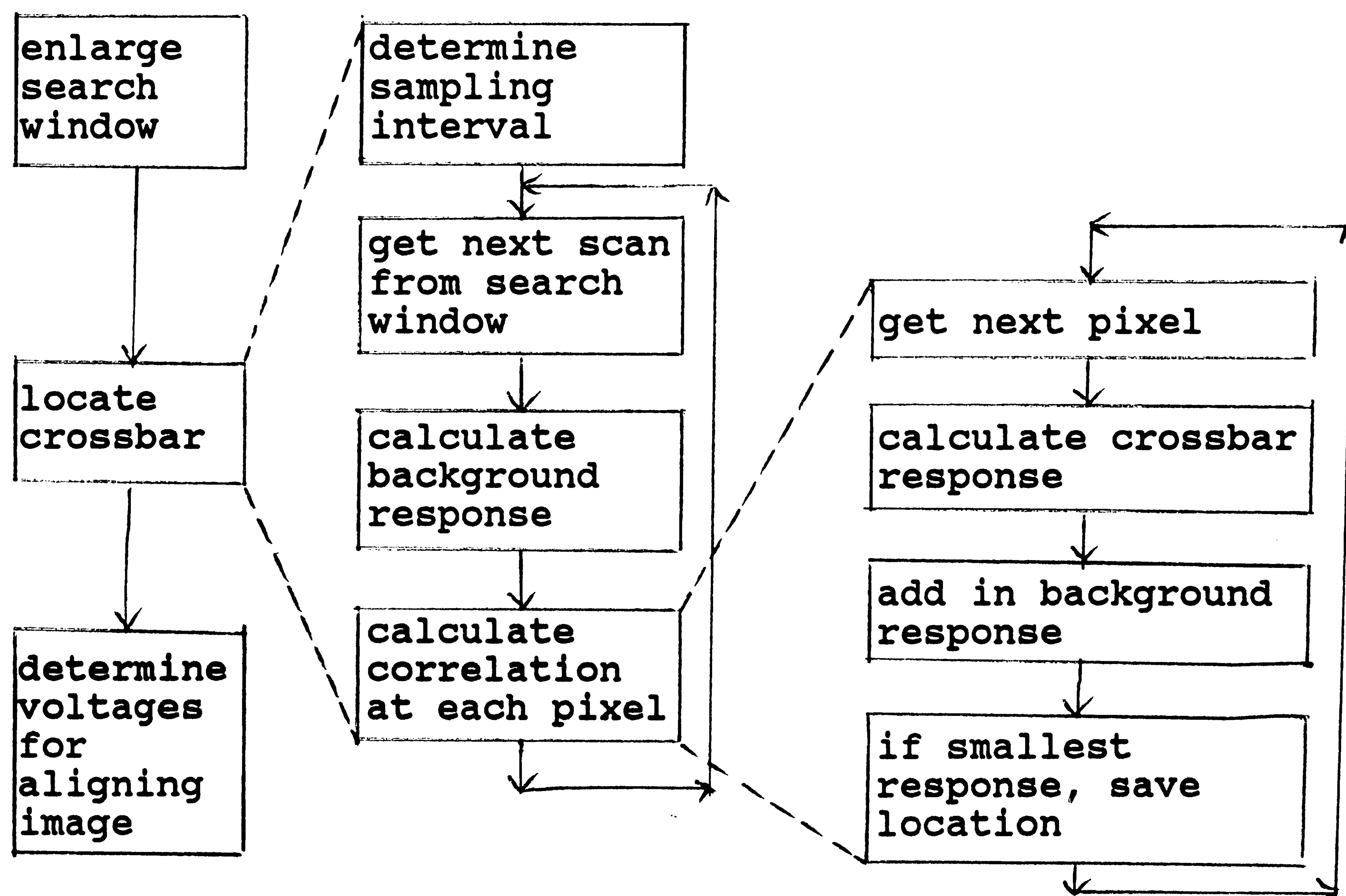


Figure 4.14: Tasks Performed during Alignment Phase

Chapter 5

Analysis

5.1 Hardware

The hardware and software components which have the biggest impact on the reliability, cost, and speed of the wire bond inspection system are discussed in this chapter. Because the vision processing is implemented solely in software, the choice of C as the primary language contributed to the ease and speed of code development, to the correctness of the code, and to the execution speed of the system. The mirror system used for imaging also contributes to the speed of the system, as does the TARGA-8 video digitizer. The TARGA board provides fast, memory-mapped access to the digitized image, but its use is hampered by a complicated addressing scheme imposed by the architecture of the 8086 microprocessor used in the AT&T 6300. However, the critical factor in attaining the desired two-second cycle time lies in the vision processing algorithms: they are specialized for this particular inspection task. To generalize the inspection and alignment algorithms while maintaining the cycle time, some of the vision processing would have to be implemented in hardware.

5.1.1 The Mirror System

One good implementation of hardware in the inspection system is the mirror system used for shifting the image under the camera. The mirror system consists of two mirrors, one each for x and y translation, and their corresponding motors and controllers (Figure 3.2). The settling time for the mirrors is proportional to the distance they must travel, but is less than 0.1 seconds for small adjustments, and less than 0.2 seconds for shifting to a new scene. Their positioning error is so small as to be undetectable, given the spacial resolution of the digital image, and their range covers more than two times the normal field of view.

The mirror controllers require additional boards for A/D and D/A conversion of signals between the microprocessor and controllers, but prevent the need to add expensive hardware to zoom in on the scene and to physically shift the device or the camera. The large virtual image available with the mirror system permits a high magnification without diminishing the field of view. The fast, accurate shifting of the image provides precise fine-positioning and a hardware correction for alignment.

The image shifting in this inspection system is limited to x and y translation, since rotational translation is not an issue, but a mirror system could be designed to rotate the image under the camera. Translations are

accomplished by mounting the mirrors perpendicular to each other, with the x-shift mirror rotating in the x plane and the y-shift mirror rotating in the y plane. If the pair of mirrors were rotated in the z plane, the image presented to the camera would be rotated. Translational alignment corrections are faster using the mirrors because the translation is done only once, by shifting the image to the correct position, instead of incorporating the translation correction into the image processing software. Correcting rotational alignment with the mirrors instead of through software would provide even more significant time savings, since rotation calculations are more complex.

5.1.2 The 8086 / TARGA-8 Combination

The 8086 microprocessor-based system is an attractive choice due to its low cost and compatibility with the inexpensive TARGA-8 board used for the image digitization and graphics display. However, because the TARGA boards were designed for the 8086 microprocessor, they inherit a complicated addressing scheme which hampers their use.

The TARGA-8 is a good tool for the vision processing portion of this inspection system because of its high 512 x 512 spatial resolution and 256 gray-levels, and its ability to digitize and capture the video input signal in real time (1/30 second). This TARGA board incorporates features which made it possible to create a good operator interface for the

setup portion of the system. Graphic displays can be created by capturing a digitized image in display memory; overlaying it with graphics such as menus, cursors, lines, and histograms; and redisplaying the image. A useful feature which the TARGA-8 board lacks is the ability to overlay computer-generated graphics on top of a live image. This capability is provided in some of the other more expensive color TARGA boards.

The only significant weakness in the TARGA board is its complicated addressing scheme, which is a direct result of the architecture of the 8086 microprocessor and affects the efficiency of vision processing and graphics generation to some extent. The Targa board uses memory-mapped I/O, allowing pixels in the digitized image to be accessed as bytes in memory. This feature, however, is not fully exploited by the wire-bond inspection system, largely because of the architecture of the 8086 microprocessor.

The 8086 is fundamentally a 16-bit machine: it is segmented into 64K-byte regions, and the full 20-bit address of a byte is composed of a segment number and a 16-bit offset for accessing the byte within the segment (Figure 5.1). If all memory accessing is contained within the same segment, the value in the segment register is fixed, and a new value need not be moved into the segment register for each access.

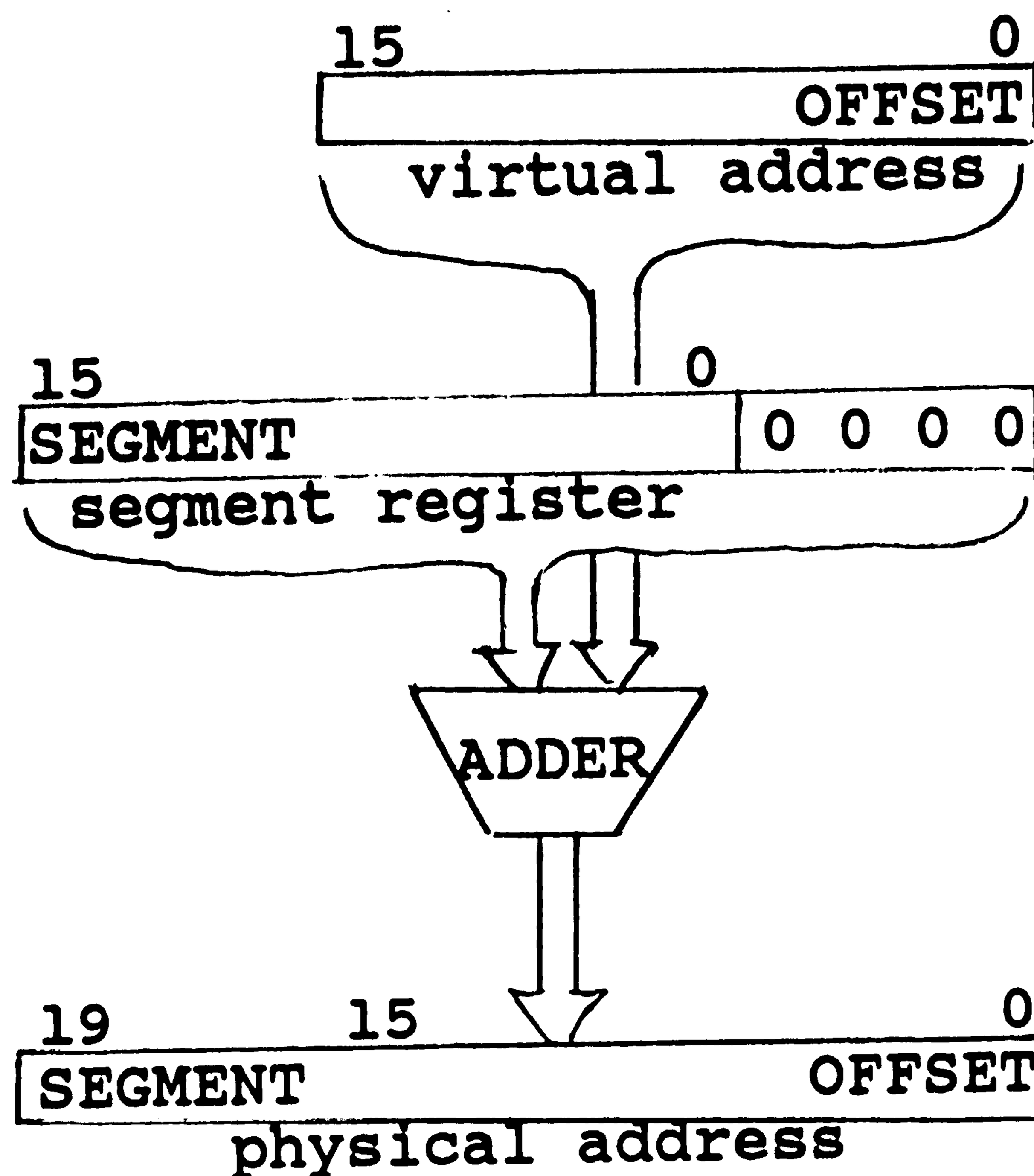


Figure 5.1: 8086 Memory Addressing

Since the digitized image, although it is memory-mapped, uses a segment of memory outside of the standard program address space, the TARGA segment address must be moved into the segment register before accessing pixels in the digitized image. The extra overhead associated with accessing TARGA memory, coupled with the fact that the 8086 can move a block of sequential bytes with less overhead per byte than moving one byte at a time, means that if a localized group of pixels will be accessed frequently, the block of pixels should be moved from the TARGA address space to the standard program address space before processing. This is in fact the case with most of the vision processing in the inspection system: access to the digitized image is

generally intensive and highly localized. The approach taken is to copy portions of the image into local buffers, so that all vision processing can be done in the standard program address space. Thus all transactions between the program and TARGA memory undergo an extra memory move, which be would unwarranted if the TARGA memory were more easily accessible.

Accessing the digitized image is further complicated by the fact that the TARGA memory itself cannot fit within one 64K memory segment. The 256 gray levels require 8 bits, or one byte, of memory for each pixel, so that the entire image requires 256K bytes of memory. The TARGA-8 contains 256K bytes of RARAM (Row Addressable Random Access Memory) divided into eight 32K byte pages, of which only two can be mapped to the 64K memory space allocated to the TARGA memory at any one time. The 18-bit address needed to access one pixel in the image is composed of a 3-bit page number for selecting one of the eight 32K byte pages, and a 15-bit offset into the page (Figure 5.2). The two page select registers, which dictate the mapping of two pages of TARGA memory into the 64K memory space, are set up as output ports. Thus accessing a pixel in the image requires the following steps:

1. Send 3-bit page number to one of the two page select registers

2. Set the segment register to point to the segment reserved for TARGA memory

3. Create the virtual address using the 15-bit offset into the page, set the high-order bit to select the correct page select register

4. Issue read/write for the appropriate number of bytes

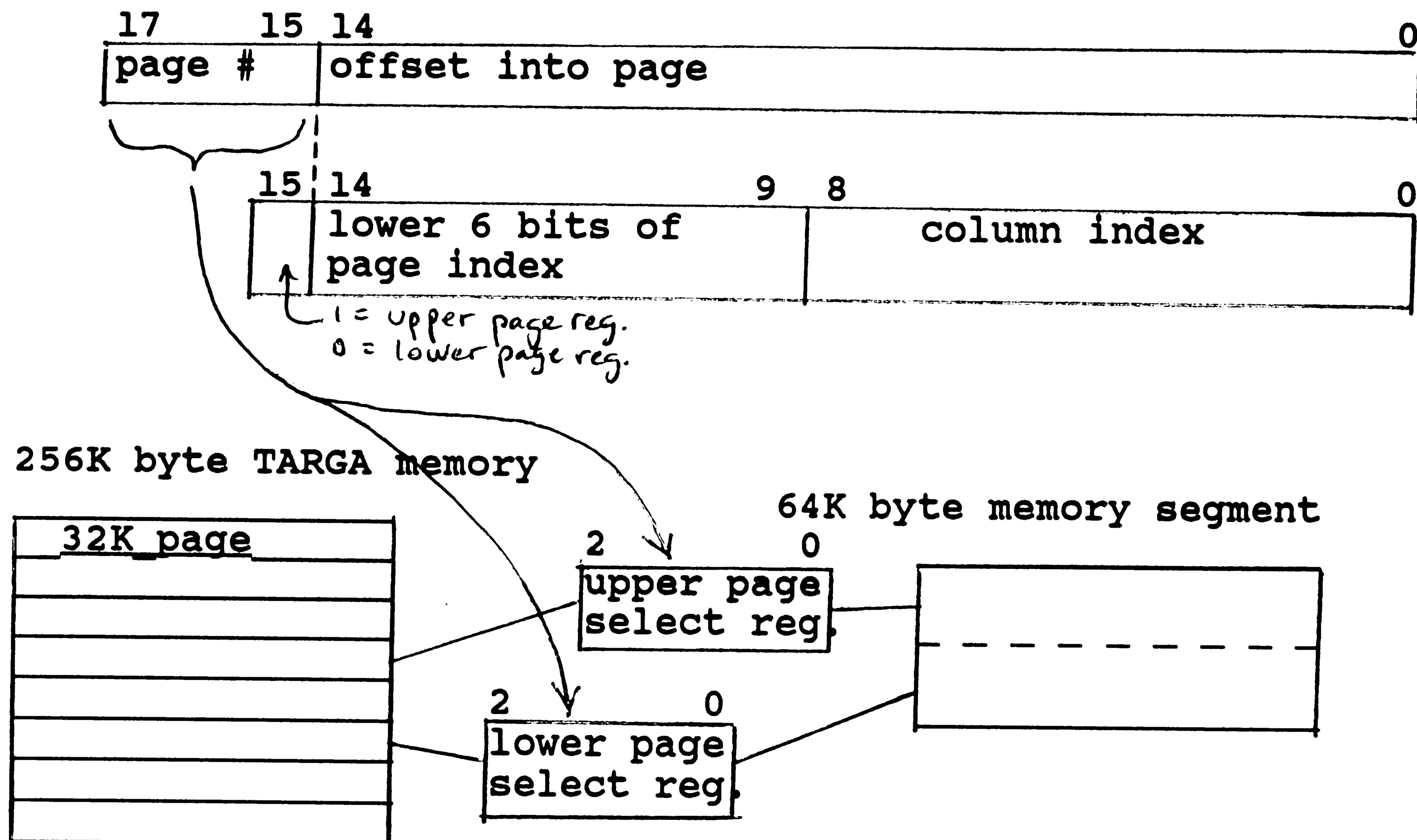


Figure 5.2: TARGA-8 Addressing Scheme

At each page boundary, a new page number must be sent to the page select register. If the microprocessor had a larger address space, all 256K bytes of TARGA memory could be mapped directly into the microprocessor address space and still leave plenty of memory space for the application. The larger address space, along with a larger segment size, would greatly simplify the addressing of TARGA memory. Crossing a segment boundary while accessing the TARGA memory would not be a concern if the segment size could contain all

256K bytes of TARGA memory within a single segment.

5.2 Hardware vs. Software for Vision Processing

In the wire bond inspection system, the TARGA board provides hardware for digitizing and capturing the image, but all analysis of the image is done through software rather than through faster but more expensive hardware processing. The reason for choosing a software implementation of the vision processing was to keep the cost of the system low. To implement the vision processing necessary for inspection and alignment within the two-second cycle time, the software solution relies on certain characteristics of this application which may not apply to other types of inspection systems. Thus although the inspection software developed works quite well within the bounds of this particular wire bond inspection system, it cannot necessarily be transfered to another type of inspection system with equal success.

In the alignment task, for example, the fundamental approach is to find the best correlation between a template of the alignment window and the image. To speed up the correlation calculations, constraints are made on the choice of alignment window: one window must contain a vertical bar on a dark, homogeneous background, and the other must contain a horizontal bar (Figure 4.5). The alignment algorithm can be applied only on scenes in which such

alignment windows exist. A more general alignment algorithm, one in which the alignment windows are chosen from any unique area in the scene, could be used if the correlation were computed using a full two-dimensional template at each pixel in the area of interest. However, computing the correlation in this fashion would take minutes, not 0.5 seconds, if implemented solely in software. This more general full correlation is realistic only when hardware is used for the correlation calculations.

5.3 Choice of Language

Because the vision processing is implemented in software, the choice of C as the primary language was important for in achieving the inspection cycle time goal of two seconds, in producing reliable, readable, modifiable code, and in meeting another goal not mentioned before, the system development time frame. A high-level language is the generally recommended tool for creating large software systems because most of the troublesome programming details are handled by the language and need not concern the programmer. A well-designed high-level language provides the programmer with data structures, operations, and control structures which are close to the logical structure of the problem. Thus the logical structure in the system is more visible with a high-level language, making it easier to develop the code, to verify its correctness, and to make

changes in the future.

For developing well-structured code, C occupies the middle ground between PASCAL and FORTRAN. PASCAL forces strict block control with its carefully defined control structures and the absence of a GOTO statement. FORTRAN at the other extreme provides few control structures, making it difficult to create structured code even for the experienced programmer. Although C provides all of the control structures needed to create good structured code, it also provides several variations on the GOTO statement. The richness of the language allows a great deal of flexibity, but increases the potential for abuse: an inexperienced programmer can easily create poorly structured, confusing code.

C, like PASCAL, facilitates the creation of new complex data types; but unlike PASCAL or FORTRAN, it provides operations for direct manipulation of bit data generally available only in assembly languages. C is weakly-typed, which is once again an advantage because of the flexibility it can provide, but a disadvantage in that a programmer can accidentally mix data types with unanticipated and unwanted results. The pixel data read from TARGA memory is stored as characters, but is interpreted as integer data. C allows the assignment of a character to an integer, but automatically provides sign extension when the high-order bit in the character is a 1.

The sign extension must be masked out to prevent pixels with gray levels above 127 from being interpreted as negative values.

Most implementations of C, including the Microsoft version used here, allow local variables such as loop counters or pointers to be allocated as registers instead of memory, which can enhance the execution speed of the code. In general, C produces faster code than most high-level languages: the C version of a sample module which sums an integer array ran 50% faster than the FORTRAN version of the same module. With C, it is rarely necessary to drop into assembly language either to increase execution speed or to manipulate bit data. The address calculations needed to access TARGA memory, for example, can be done easily in C.

5.4 Extending the Application of the Alignment

In all 12,672 trials using the twenty-two different types of ceramic packages, the alignment method in the wire bond inspection system successfully aligned the device within one mil, given an initial displacement of up to twenty mils. Because the alignment method correctly aligned the package 100% of the time, no verification of the correlation results was done within the alignment method. If, for example, the lighting were more intense during inspection than during setup, the one-dimensional template might correlate better with a scan line crossing over a

bright edge than with the crossbar. Thus if the same alignment method were to be used in another inspection application in a less controlled environment, some form of verification would be useful for ensuring that the best correlation found did correspond to the alignment window. Such a verification method could involve spot-checking portions of the scan lines which are expected to cross the edges of the crossbars.

The lighting is also an important factor in determining the two emperically-derived constants for the teach phase: the edge detection threshold and the maximum width of a crossbar edge. If the lighting is such that one of the edges of the crossbar is indistinct, it may escape detection using the edge templates and the default threshold. If, on the other hand, one of the edges is so bright that it appears wider than five pixels, it will be interpreted as two edges. In either of these cases, the segmentation of the crossbar from the background would fail. Thus a different lighting system than the one currently used in the inspection system would most likely require a change in these two constants. An enhancement to the setup phase could include some heuristics for determining the value of these two constants during setup, instead of apriori.

Chapter 6

Conclusions

Several things become clear from studying the implementation of this wire bond inspection system. The first is the importance of a controlled, consistent environment during the inspection. The lighting must be the same for each device inspected and the alignment of the device must be precise before inspection can be done. Because the fine alignment correction is built into this inspection system, the environment is somewhat more flexible and only the coarse alignment must be controlled. The same direction could be taken in decreasing the sensitivity of the inspection system to changes in the illumination, by using a more intelligent, hierarchical approach to analyzing the scene.

The approach used in teaching the alignment windows is a simple three-level hierarchy: at the top level, features for the two alignment windows are extracted. Before extracting these features, the next level down in the hierarchy must segment the scene framed by the operator into a crossbar and background. The lowest level is responsible for finding horizontal and vertical edges in the scene. It passes this edge information to the second level, which uses very simple heuristics for determining which edges belong to

the crossbar. The top level now knows where the crossbar is and can define its alignment windows accordingly. Although it does not incorporate sophisticated heuristics, this approach analyzes and extracts the features from a scene in which the general characteristics are known but the scene itself is unknown.

This hierarchical model would work well for building a system which is able to analyze a less predictable scene than those encountered by this wire bond inspection system. If the alignment phase of the inspection used the same hierarchical approach as in teaching the alignment windows, then the lighting during inspection would not have to be identical to the lighting during setup; the crossbars would be located by edge detection and segmentation instead of by matching the template created during setup. This could not be done in this system because the algorithms used for the setup phase are too slow for the inspection phase.

The second point which the implementation of this wire bond inspection system makes clear is the continual tradeoff between designing a general solution, one which could be transported to other inspection tasks, and one which can complete the task in a reasonable period of time. The algorithms used in this inspection system, including the alignment algorithms, are specialized: they use techniques such as sampling to increase the execution speed. The use of special hardware would greatly enhance the speed much of

the vision processing. This could be combined with a hierarchical approach by using hardware processing at the lower levels of the hierarchy, where most of the computationally intensive vision processing takes place.

REFERENCES

1. P. Burggraaf, "Inspection Trends in IC Assembly," Semiconductor International, June 1985, 76-81.

2. M. Negin, "Computer Vision Hierarchy for Pre-Seal (Third Optical) Inspection for Microelectronic Assembly," VISION '85 Conference Proceedings, March 25-28 1985, Machine Vision Association of SME.

3. A. E. Kayaalp and R. Jain, "A Knowledge Based Automatic On-line Wafer (IC) Inspection System," VISION '85 Conference Proceedings, March 25-28 1985, Machine Vision Association of SME.

4. K. S. Fu, "Pattern Recognition for Automatic Visual Inspection," Proceedings of SPIE, Vol. 336 Robot Vision (1982), 12-19.

5. P. Burggraaf, "Pattern Recognition on Bonders and Probers," Semiconductor International, February 1981, 53-70.

6. M. L. Baird, "SIGHT-I: A Computer Vision System for Automated IC Chip Manufacture," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-8, No. 2, February 1978, 133-139.

7. Y. Y. Hsieh and K. S. Fu, "An Automatic Visual Inspection System for Integrated Circuit Chips," Computer Graphics and Image Processing, 1980, 293-343.

8. Y. Y. Hsieh and K. S. Fu, "A Method for Automatic IC Chip Alignment and Wire Bonding," Proceedings of the 1979 IEEE Computer Society Conference on Pattern Recognition and Image Processing, August 6-8 1979, Chicago Illinois.

VITA

Christine R. Hofmeister, the daughter of Janet C. and Ralph H. Hofmeister, was born on December 30, 1959 in Boston, Massachusetts. She received her AB in Mathematics from Bryn Mawr College in May, 1981. From July, 1981 through August, 1985 she was employed as a Staff Mathematician, then Senior Mathematician at Pennsylvania Power & Light Company in Allentown, Pennsylvania. She left PP&L to pursue a Master's Degree in Computer Science at Lehigh University, where she has served as a Teaching Assistant and Research Assistant.

VITA

Christine R. Hofmeister, the daughter of Janet C. and Ralph H. Hofmeister, was born on December 30, 1959 in Boston, Massachusetts. She received her AB in Mathematics from Bryn Mawr College in May, 1981. From July, 1981 through August, 1985 she was employed as a Staff Mathematician, then Senior Mathematician at Pennsylvania Power & Light Company in Allentown, Pennsylvania. She left PP&L to pursue a Master's Degree in Computer Science at Lehigh University, where she has served as a Teaching Assistant and Research Assistant.