A High Speed Processor for Binary Images

Robert W. Berger

CMU-RI-TR-84-2

Inspection Laboratory
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

February 1984

This research was supported by the Robotics Institute, Carnegie-Mellon University.

This paper appeared in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 83*, Washington, D. C., June 1983.

Table of Contents

1. Introduction 1
2. Background 1
3. Architecture of the Convolution Processor 1
4. Architecture of the Image Processing System 3
5. Examples 4
6. Conclusion 5

List of Figures

Figure 3-1: Data Path for One Row of the Operator Kernal 2
Figure 3-2: Data path for the Convolution Processor 3
Figure 3-3: Architecture of the Real-Time Vision System 4
Figure 5-1a: Character Recognition Example - Input 7
Figure 5-1b: Character Recognition Example - Mask 7
Figure 5-1c: Character Recognition Example - Correlation 8
Figure 5-1d: Character Recognition Example - Thresholded 8
Figure 5-2a: Printed Wiring Example - Input 9
Figure 5-2b: Printed Wiring Example - Mask 9
Figure 5-2c: Printed Wiring Example - Convolution 10
Figure 5-2d: Printed Wiring Example - Thresholded Convolution 10

Abstract

A video rate processor for binary images is described in this paper. The device convolves a binary image with a 3 level 64 by 64 mask. Images are processed at a rate of 10 million pixels per second, allowing real time processing of images obtained from a CCD camera.

1. Introduction

In this document we describe the design of an image processor for binary images. The device is used as a preprocessor in a computer vision system. The input image is obtained directly from a solid state camera and processed at video rates. The processing consists of convolving the input image with a user specified 64 by 64 operator kernel.

The convolution of a discrete two-dimensional image array $I(x,y)$ with a $2N+1$ by $2N+1$ operator kernel $K(i,j)$ yielding an output image $O(x,y)$ is given by

$$O(x,y) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} I(x-i,y-j)K(i,j) \qquad (1)$$

In the system described, the input image $I(x,y)$ is a binary image consisting of two levels, 1 and -1. The operator kernel $K(i,j)$ is a 64 by 64 arrayx whose elements are restricted to be 0's, 1's, and -1's. The input image is 2048 pixels wide and arbitrarily long. The image is obtained by a linear CCD array solid state camera. The device can process 10 million pixels per second.

2. Background

The convolution operation described by equation 1 can be expressed as a correlation with a mask $K'(i,j)$

$$O(x,y) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} I(x+i,y+j)K'(i,j) \qquad (2)$$

where the correlation mask $K'(i,j)$ is the convolution mask of equation 1 rotated by 180^{o}:

$$K'(i,j) = K(-i,-j)$$

The correlation operation can be used to detect translated instances of a given shape. This method, known as matched filtering,[1] is demonstrated in section 5; early hardware for performing the same task is described by Kovalevsky.[2] Another related type of image processor is the cellular logic computer.[3, 4] The operators in these processors are relatively small; 3 by 3 is common. The operator is specified by a boolean function of the 9 input pixels in the neighborhood of the output pixel. Larger operators are implemented by the cascaded application of the small operators.

3. Architecture of the Convolution Processor

A pipelined architecture is used to obtain the video rate processing. The data path for a single row of the operator kernel is shown in figure 3-1. The shift register contains 64 pixels from one row of the input image. The shift register contains a 1 wherever the corresponding input pixel is 1 and a 0 wherever the corresponding input pixel is -1. The sign register and magnitude register contain elements from one row of the operator kernel. The sign register contains a 1 wherever the corresponding element is positive; the magnitude register contains a 1 wherever the corresponding element is non-zero. The sign bits are EXCLUSIVE-OR'ed with the input pixels; the inversion of this result is then AND'ed with the magnitude bits. The number of one bits resulting from this operation is counted by a summation node, producing a 7 bit output for that row of the kernel. The circuit shown in figure 3-1 is commercially available on a single integrated circuit.[5]

The convolution processor contains 64 row processors, interconnected as shown in figure 3-2. One of the row processors is fed directly from the input pixel stream. The other row processors are fed from a series of

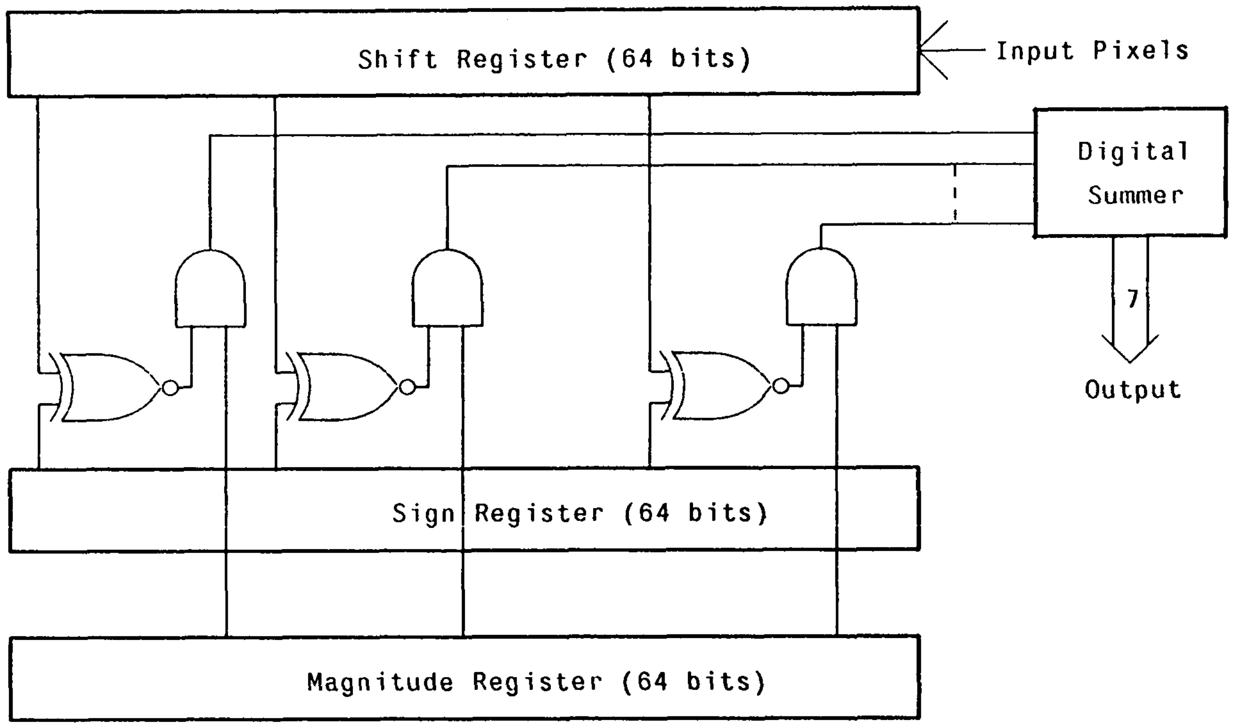


Figure 3-1: Data path for one row of the operator kernel

shift registers, each of which buffers 1 row of input pixels. The length of the shift registers are switchable between 1024 and 2048. In the normal mode of operation the shift registers are 2048 pixels long and buffer a full line of input pixels. The 1024 length mode is used with sampled images to approximate larger operators, as explained in section 4. The outputs of the 64 row processors are summed in an addition tree, producing a 13 bit output.

The output produced by the circuit described above is not equal to the desired result as shown in equation 1. The desired result, $O(x,y)$ can be obtained from the result produced by the circuit, T, by the equation

$$O(x,y) = 2T - m$$

where m is the number of non-zero elements in the operator kernel.

During each processing cycle, a new input pixel is shifted into the system, and an output value is generated. The high degree of parallelism in the pipelined design permits the system to process 10 million pixels per second.

The convolution processor is constructed on five MULTIBUS* boards. Four of the boards each contain 16 rows of the data path. A controller board contains a four-input addition tree and timing and interface circuitry. The division of the data path into four identical 16 row boards permits the controller board to implement several configurations in addition to the 64 row three level operator described above. One

*MULTIBUS is a trademark of Intel Corporation.

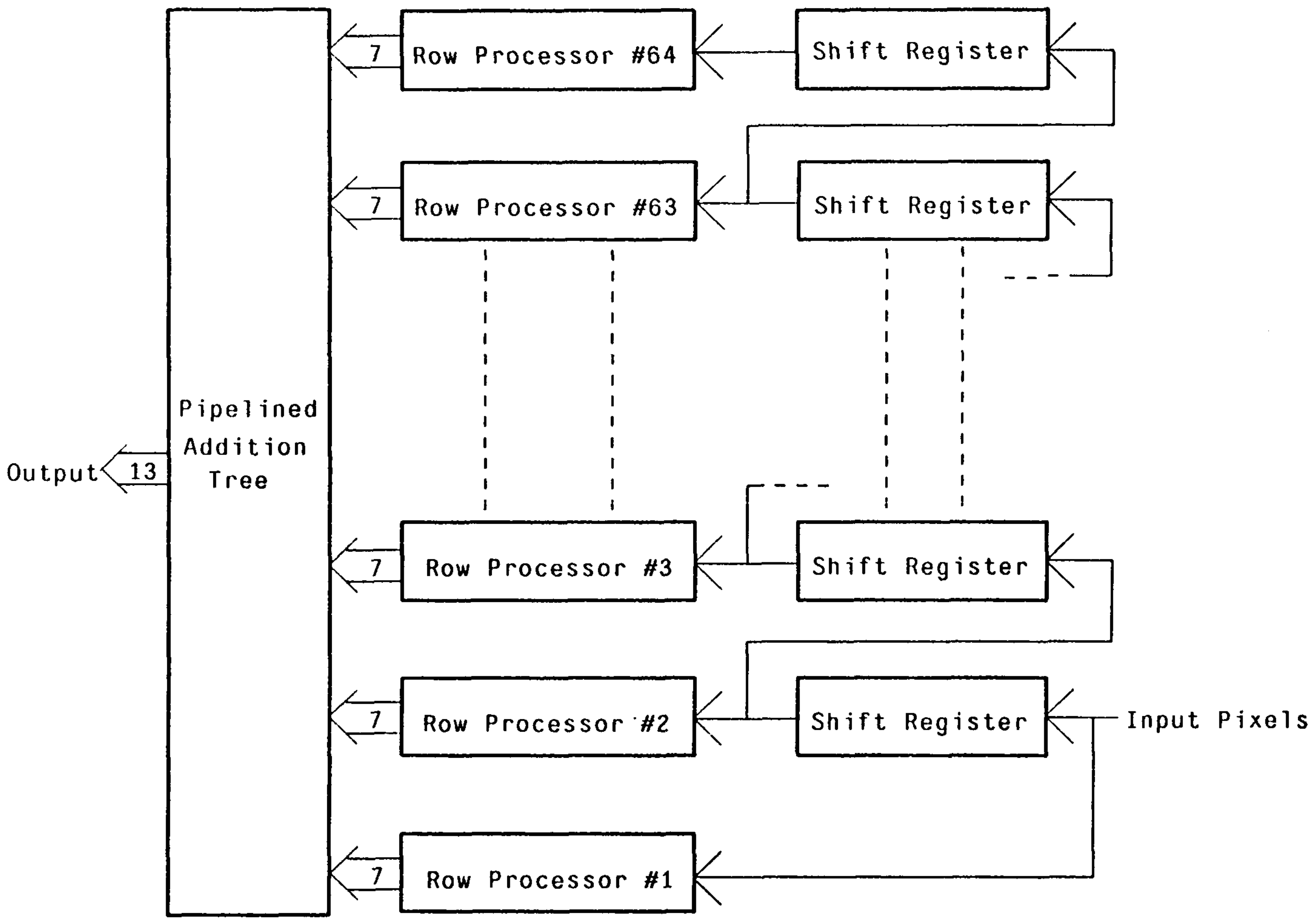


Figure 3-2: Data path for the convolution processor

variation is to segment the data path to allow multiple smaller operators, such as two 32 row operators or four 16 row operators. The other option is to combine the outputs from the data path boards with a binary weighted addition, allowing operators with more than three levels. The available configurations are a 32 row seven level operator and a 16 row thirty-one level operator.

4. Architecture of the Image Processing System

Figure 3-3 shows how the convolution processor is used as a preprocessor in a vision system. The charge-coupled device (CCD) camera produces a voltage for each pixel in the input image. The analog to digital converter translates this voltage into a 6 bit digital number. This number is compared with a threshold from the threshold memory to produce the binary image. The threshold memory provides a separate threshold for each of the 2048 column positions in the input image, allowing compensation for variations in the sensitivities of the individual photosensors and nonuniformities in the distribution of the scene illumination. The binary image can be fed directly into the convolution processor or through the smoothing and sampling circuit. The smoothing and sampling circuit performs 2:1 two-dimensional sampling of the input image, allowing 128 by 128 operator kernels to be approximated. When operated in a sampled mode, the lengths of the shift registers

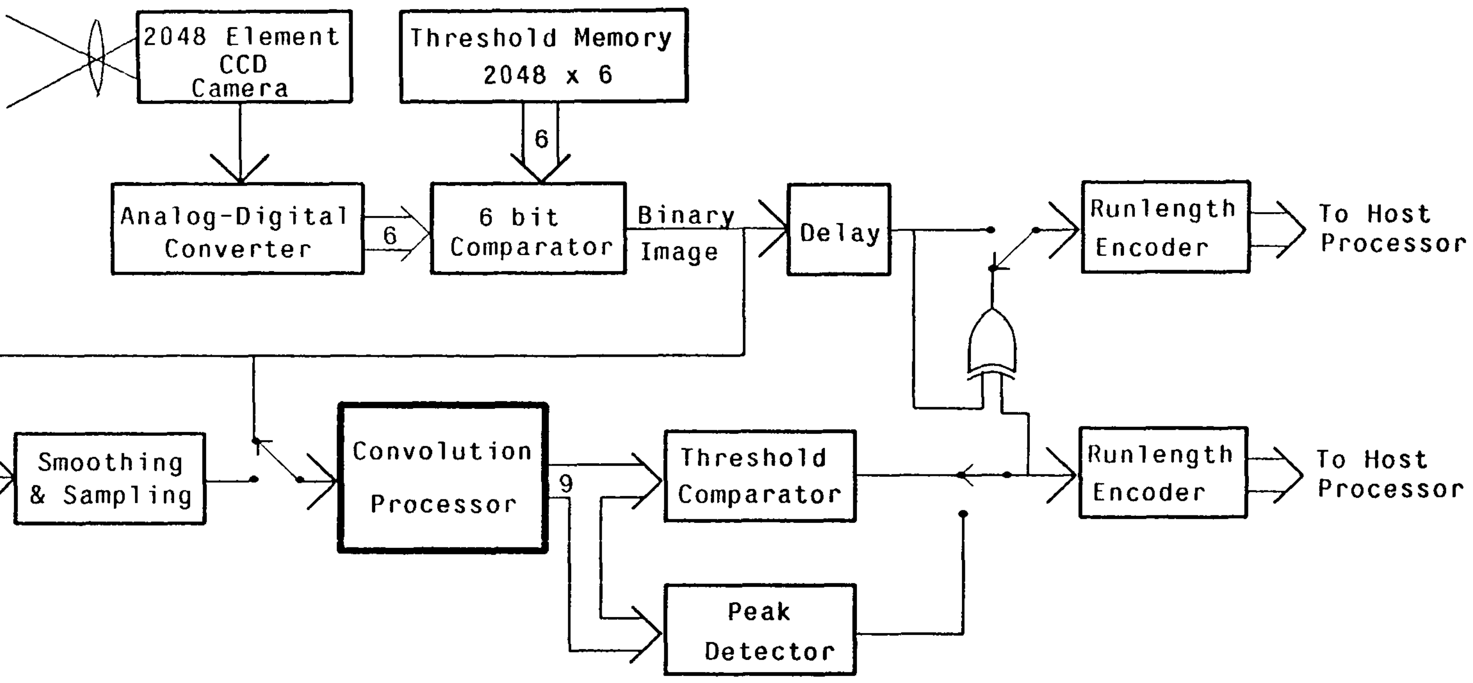


Figure 3-3: Architecture of the real-time vision system

in the convolution processor are reduced to 1024. The output of the convolution processor is 13 bits wide. This data is converted back into a binary image by either a threshold comparator or a two-dimensional peak detector. The resulting binary image is encoded by a runlength encoder and transmitted to the host processor. The runlength encoder encodes the transitions along each scan line, reducing the amount of data which must be transmitted to the host processor. A second runlength encoder encodes either the original unprocessed image or the difference between the original and unprocessed images.

5. Examples

Section 2 described how the convolution processor can be used in an optical character recognition system. Figure 5-1a is a digitized binary image consisting of positive (white) and negative (black) areas. Figure 5-1b is a correlation mask consisting of positive (white), negative (black), and neutral (gray) areas. Figure 5-1c is the correlation of 5-1a and 5-1b, as produced by the convolution processor. Figure 5-1d is the binary image resulting from thresholding 5-1c with an appropriate value. All 7 "r"'s in the original input image have been successfully detected.

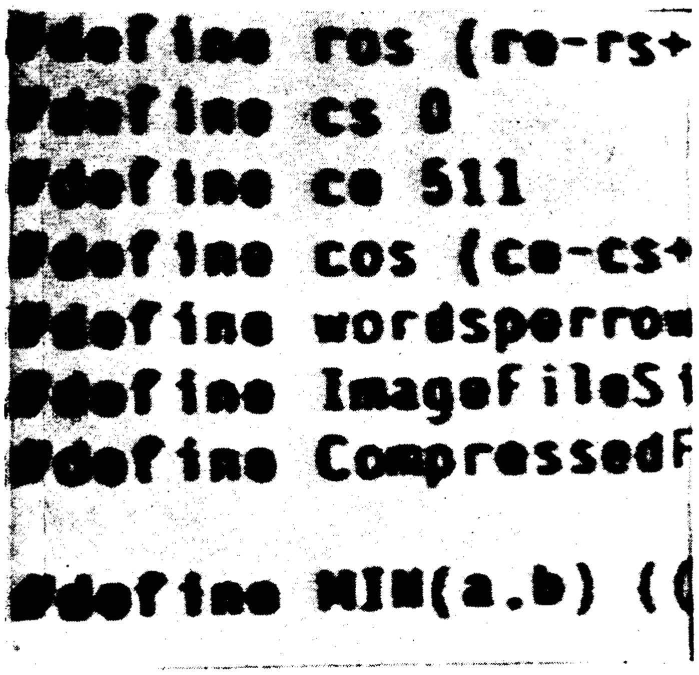
The second example is from the domain of automated inspection of printed wiring boards.[6, 7] The input image is figure 5-2a. The defect is the overwidth portion of the horizontal line near the center of the image. Figure 5-2b shows an operator kernel designed to detect this type of defect. The operator is a spatial bandpass filter consisting of a central positive response area surrounded by an outer negative response area. Such an operator has a maximum response for a feature whose width matches the diameter of the positive response area. The operator will detect defects of the type shown without falsely flagging normal lines and pads. Figure 5-2c shows the output of the convolution processor; figure 5-2d is the thresholded output.

6. Conclusion

A design for a binary convolution processor has been presented. Current technology allows cost effective implementation of video-rate convolution processors with large mask sizes. The large mask sizes allow complex template shapes and immunity from quantization noise. The algorithms demonstrated in section 5 are classical pattern recognition techniques which have been largely of theoretical interest due to the computational expense of implementing them. The technology described here makes these techniques attractive for use in practical real-time systems.

References

1. Rosenfeld, A., *Picture Processing by Computer,* Academic Press, New York, 1969.
2. Kovalevsky, V.A., *Image Pattern Recognition,* Springer-Verlag, New York, 1980.
3. Lougheed, R.M., McCubbrey, D.L., and Sternberg, S.R., "Cytocomputers: Architectures for Parallel Image Processing," *IEEE Workshop on Picture Data Description and Management,* August 1980, .
4. Preston, K., "Cellular Logic Computers for Pattern Recognition," *IEEE Computer,* January 1983, .
5. Eldon, John, "Digital Signal Processing Hits Stride with 64-bit Correlator IC," *Electronics,* July 14 1981, .
6. Thibadeau, R., "Printed Circuit Board Inspection," CMU-RI-TR-81-8, The Robotics Institute, 1981.
7. Thibadeau, R., Friedman, M. & Seto, J., "Automatic Inspection for Printed Wiring.," Tech. report TP-428, Institute for Interconnecting and Packaging Electronic Circuits, 1983.



```
#define ros (re-rs+
#define cs 0
#define ce 511
#define cos (ce-cs+
#define wordsperrow
#define ImageFileSi
#define CompressedF

#define MIN(a,b) ((
```

Figure 5-1a: Character Recognition Example - Input

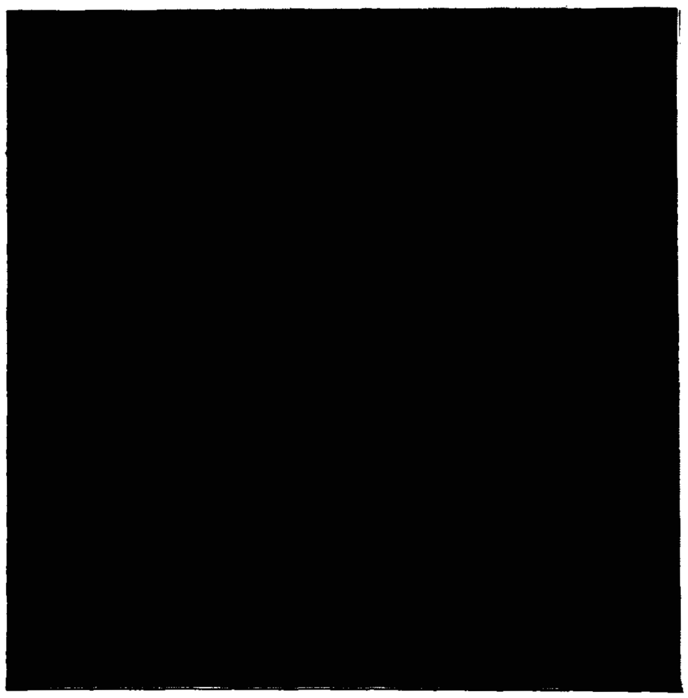


Figure 5-1b: Character Recognition Example - Mask

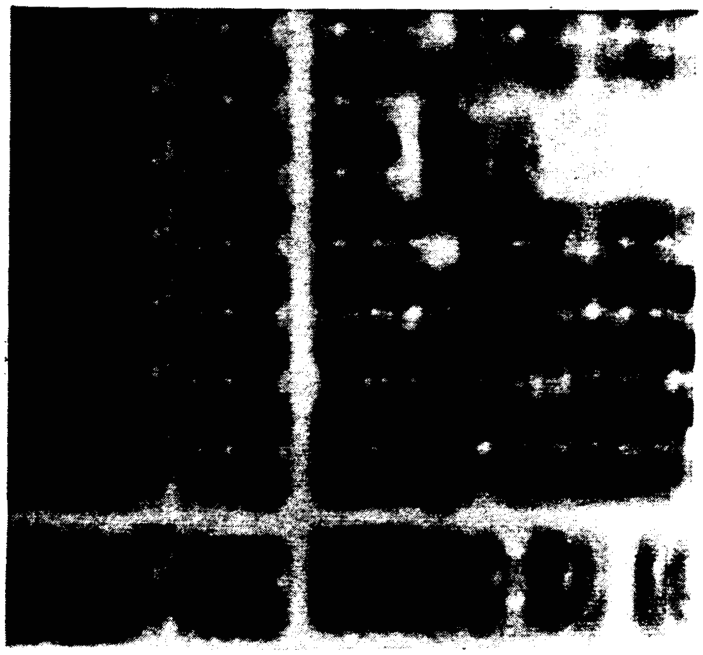


Figure 5-1c: Character Recognition Example - Correlation

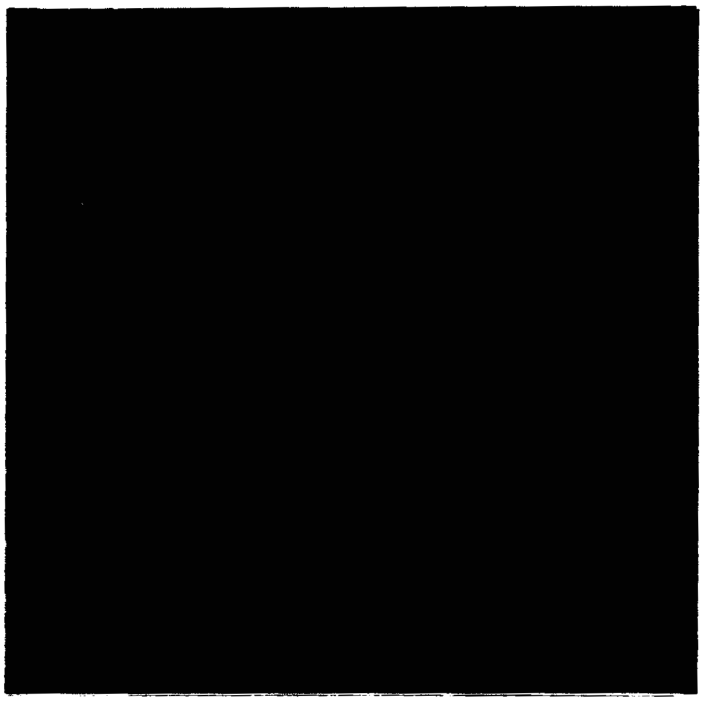


Figure 5-1d: Character Recognition Example - Thresholded Correlation

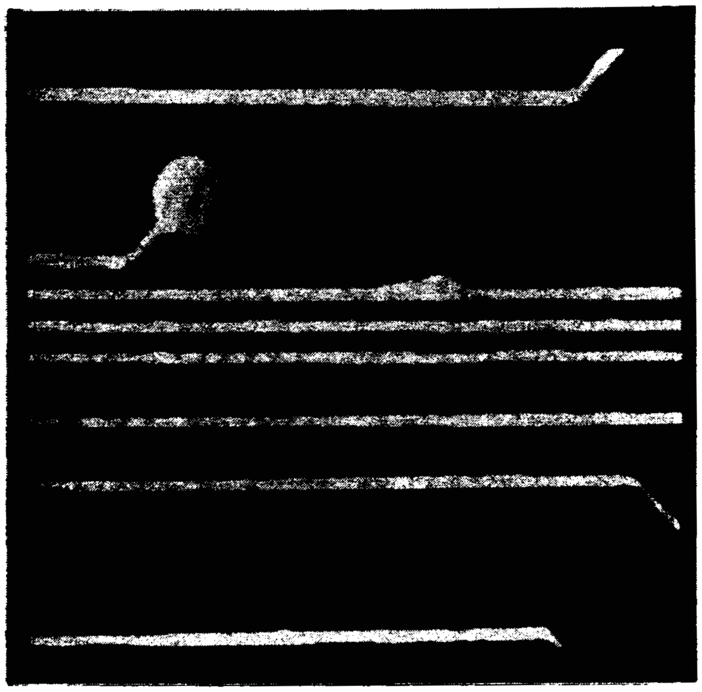


Figure 5-2a: Printed Wiring Example - Input

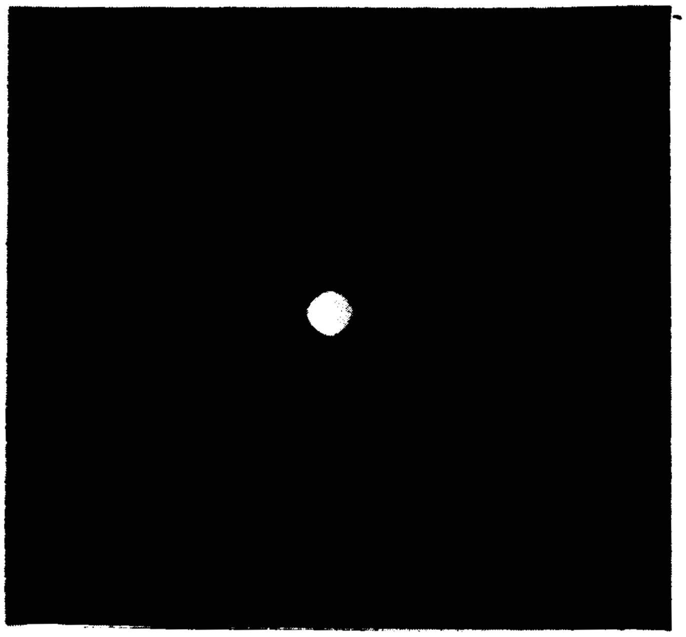


Figure 5-2b: Printed Wiring Example - Mask

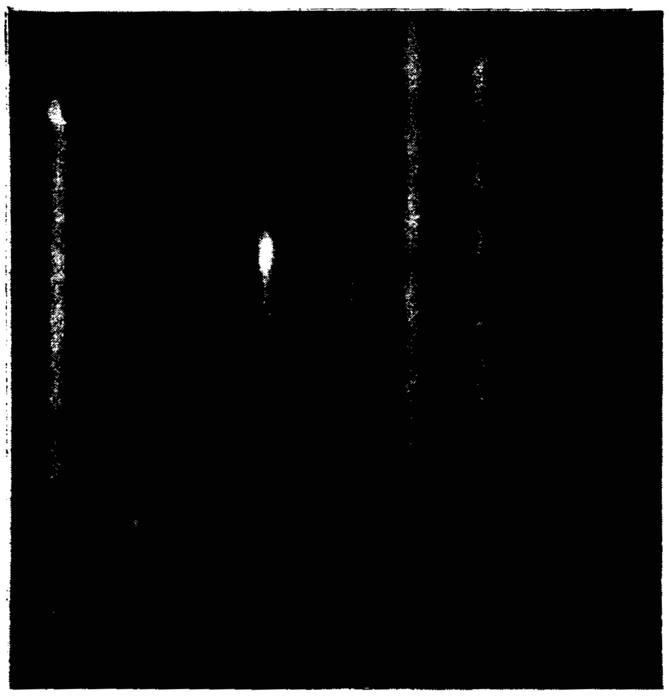


Figure 5-2c: Printed Wiring Example - Convolution

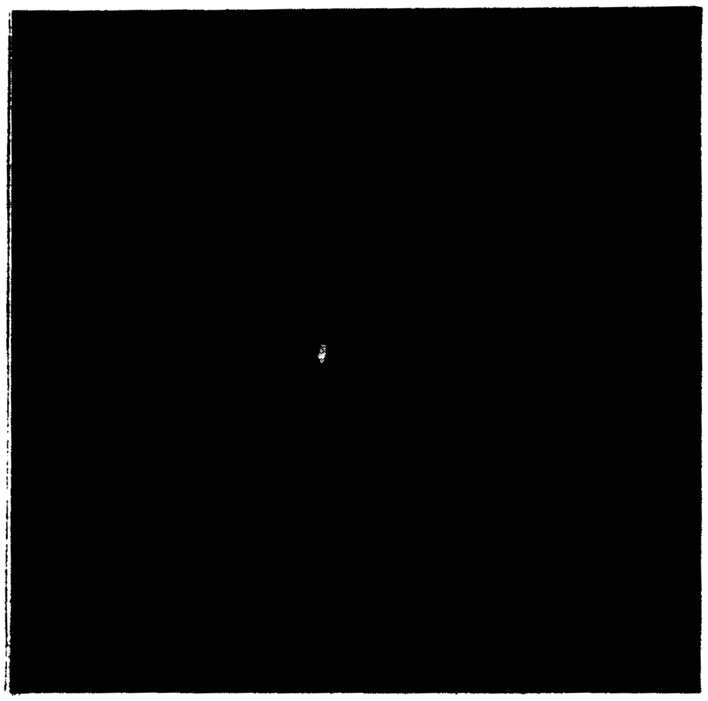


Figure 5-2d: Printed Wiring Example - Thresholded Convolution