# A LARGE-SCALE OPTICAL CHARACTER RECOGNITION SYSTEM SIMULATION

David P. Himmel and David Peasner

Recognition Equipment Inc.

## ABSTRACT

A simulation is described which provides a vehicle for the synthesis and subsequent performance analysis of optical character recognition (OCR) algorithms aimed at solving some of the most difficult problems encountered in OCR.

The simulation treats the problems of linking complex interactive algorithms together and processing large real-world data files under economic constraints.

The simulation is written in Fortran V and is installed on University Computing Company's Dallas 1108 facility. It is comprised of a main program, four major subroutines, 29 supporting routines, and the system library routines. It presently requires 104,500 words of memory (418,000 bytes). Results of the computer simulation on real-world handprint character data are presented.

## I. INTRODUCTION

In this paper we will describe a program which was used as a vehicle for the design, development, and testing of several algorithms used for Optical Character Recognition (OCR) of handprinted characters. Points of general interest, both from a programming and an OCR point of view, will be discussed, and a few specific details about programming aspects will also be given.

The technique of simulating algorithms, heuristics, or math models for OCR in the design stage has been used at Recognition Equipment for about 10 years with great success. I will take just a moment to outline our design process. The goal of an OCR simulation of this sort is to complete the design and prove the performance of an OCR technique; once this is accomplished the simulation provides a basis for hardware design. First, essential parts of the basic algorithm are programmed, and the program operates on some limited set of data samples (handprinted character images in this case). From this initial simulation, any large unforeseen problems are made visible, and obvious improvements are made. Many times the algorithm is scrapped at this point. If the algorithm proves viable, the simulation is filled out by programming the complete algorithm in all its detail, including supporting pre/and post-processing algorithms; then more extensive simulations are conducted to improve the design, using even larger data bases.

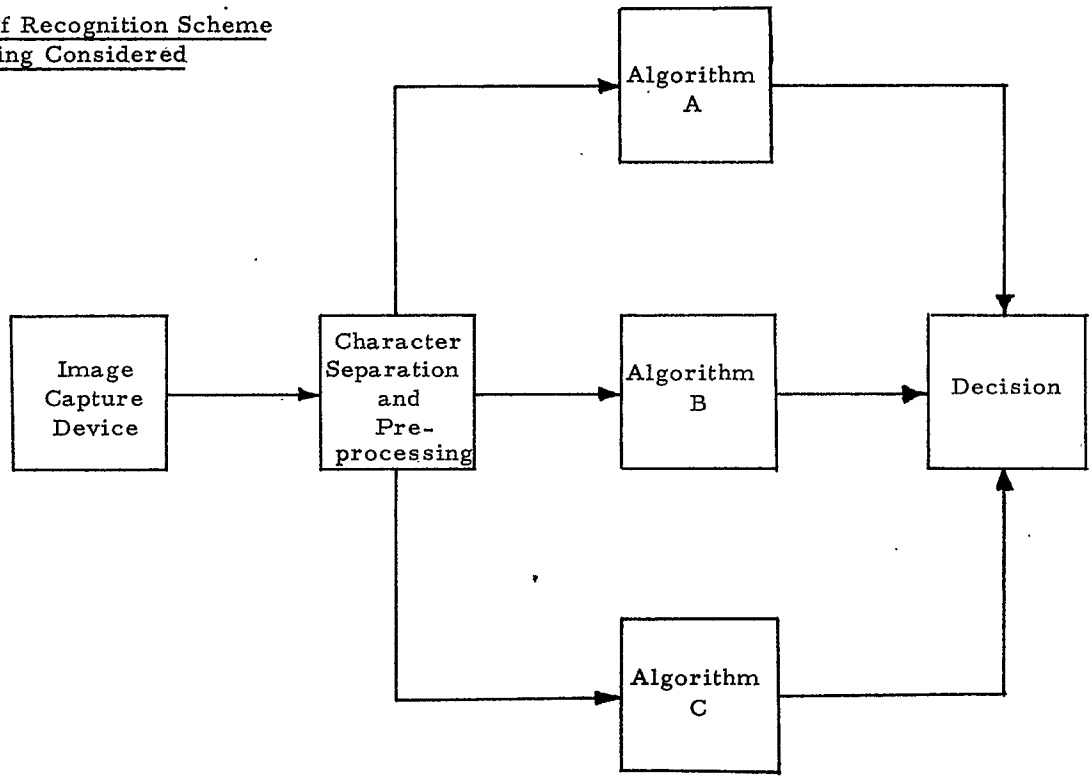
During the entire simulation process the design engineer interacts with the programmer, and together they translate the designer's ideas into an operating program. Once the design is complete and proven, further interaction occurs among the design engineer, the programmer, and a logic-circuit designer to implement the algorithm in hardware.

The core of this simulation is three algorithms, each of which performs character recognition itself. Illustration 1 shows a block diagram of the major algorithms being simulated. It will not be necessary to go into details of the algorithms. Grossly, each of the algorithms (A, B, and C) scans the character image and detects the presence of certain "features" based on the occurrence of black cells in proper relation to the scanning. Recognition then is performed by applying a decision algorithm to the results of feature detection. The A, B, and C algorithms are necessarily preceded by some auxiliary character separation and preprocessing algorithms and are followed by a decision scheme that combines the results of the three algorithms into a single decision. This configuration of recognition algorithms performs essentially like the keypunch verification process, where one algorithm checks and verifies the output of another.

The three algorithms were developed independently by three different engineers interacting with the same programmer over a period of about one year before being integrated into a single simulation. Although the algorithms were developed

ILLUSTRATION 1

Outline of Recognition Scheme Being Considered




independently, they were designed to be complementary in the sense that one algorithm would perform well on handprint samples that another could not handle.

## II. THE OCR PROBLEM - HANDPRINTED NUMERALS

The function of the algorithms and of the simulation described here is to read handprinted numeric characters on special forms such as can be generated by clerks, telephone operators, etc. Illustrations 2 and 3 show some examples of handprinted numerals that have been processed by this simulation. Some of the problems encountered by a handprint reader are illustrated by these examples. One of the most striking characteristics of handprinted data is, of course, the almost unlimited variation in shapes of characters. However, the shapes for each character can be categorized according to certain characteristics of the strokes which make the feature recognition approach feasible. In addition to shape variation, there can be a large degree of noise in the image due to dirt and smudges or foreign markings on the document. Character stroke fading, especially with ball point pen introduces breaks in the character stroke; this is sometimes compounded by digitizing error due to the analog-to-digital conversion process. It is the job of the preprocessing algorithms to eliminate extraneous noise and to fill in broken strokes. Another problem encountered with handprint is size variation: the characters commonly exhibit a 3:1 size range so that the algorithms must be prepared to recognize small as well as large character images. Character separation is a non-trivial problem in the handprint environment, although some good solutions do exist. When the characters overlap or even touch one another, and are at the same time variable in width, it can become very tricky for the machine to separate them prior to recognition.

## III. THE SIMULATION PROGRAMS

To implement a program of this size, the basic system parameters are the first items to identify. Program structural factors, such as record identification formats, data array sizes, method of transmission between programs/subroutines, and working and scratch array locations, were seen to be critical to the entire research activity and were decided upon at the very beginning of this project. As new algorithms were programmed, each was made to be compatible with the others -- at least with respect to the data organization and communications. This initial approach would turn out to be valuable when we

ILLUSTRATION 2

Handprint Character Samples

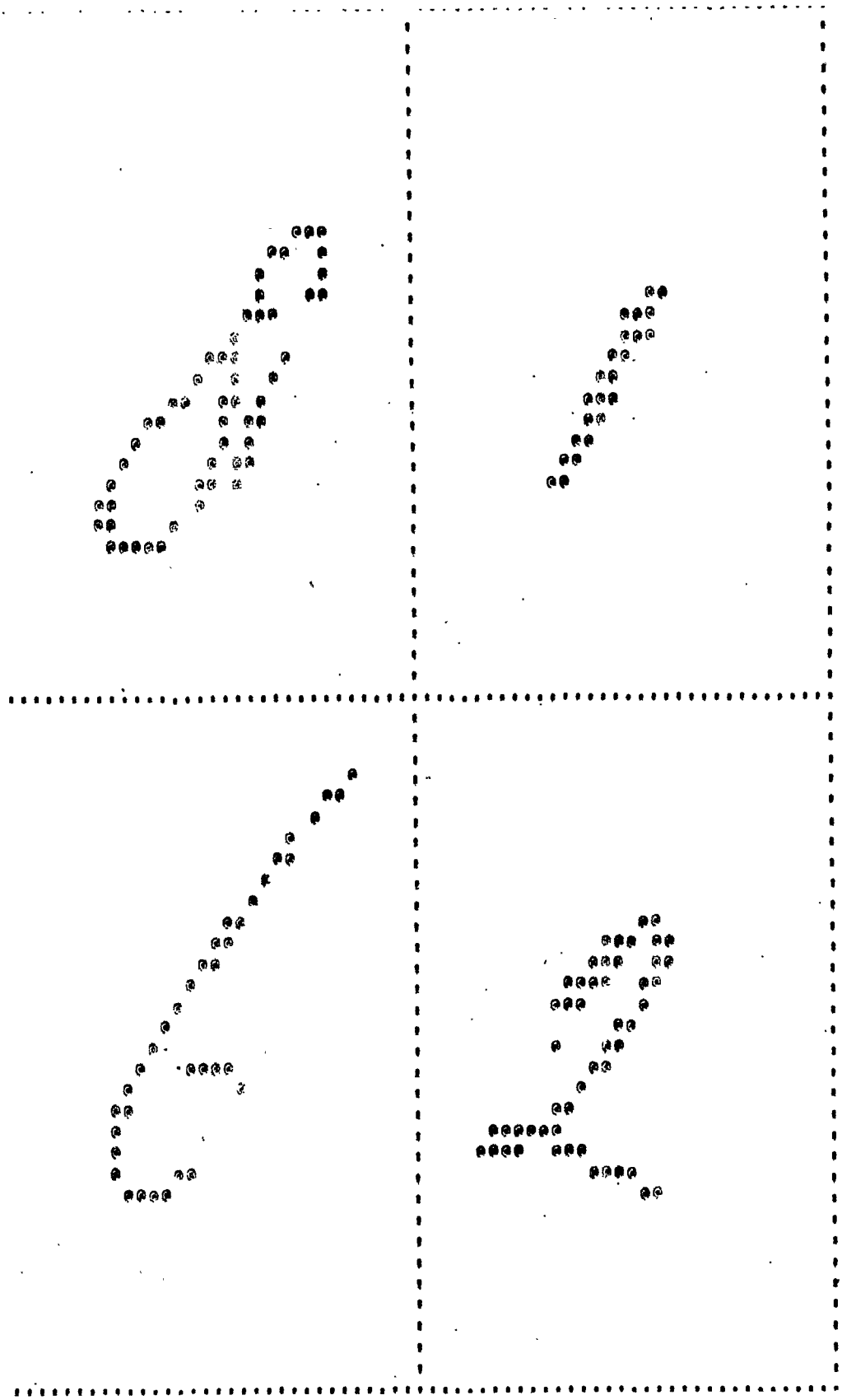
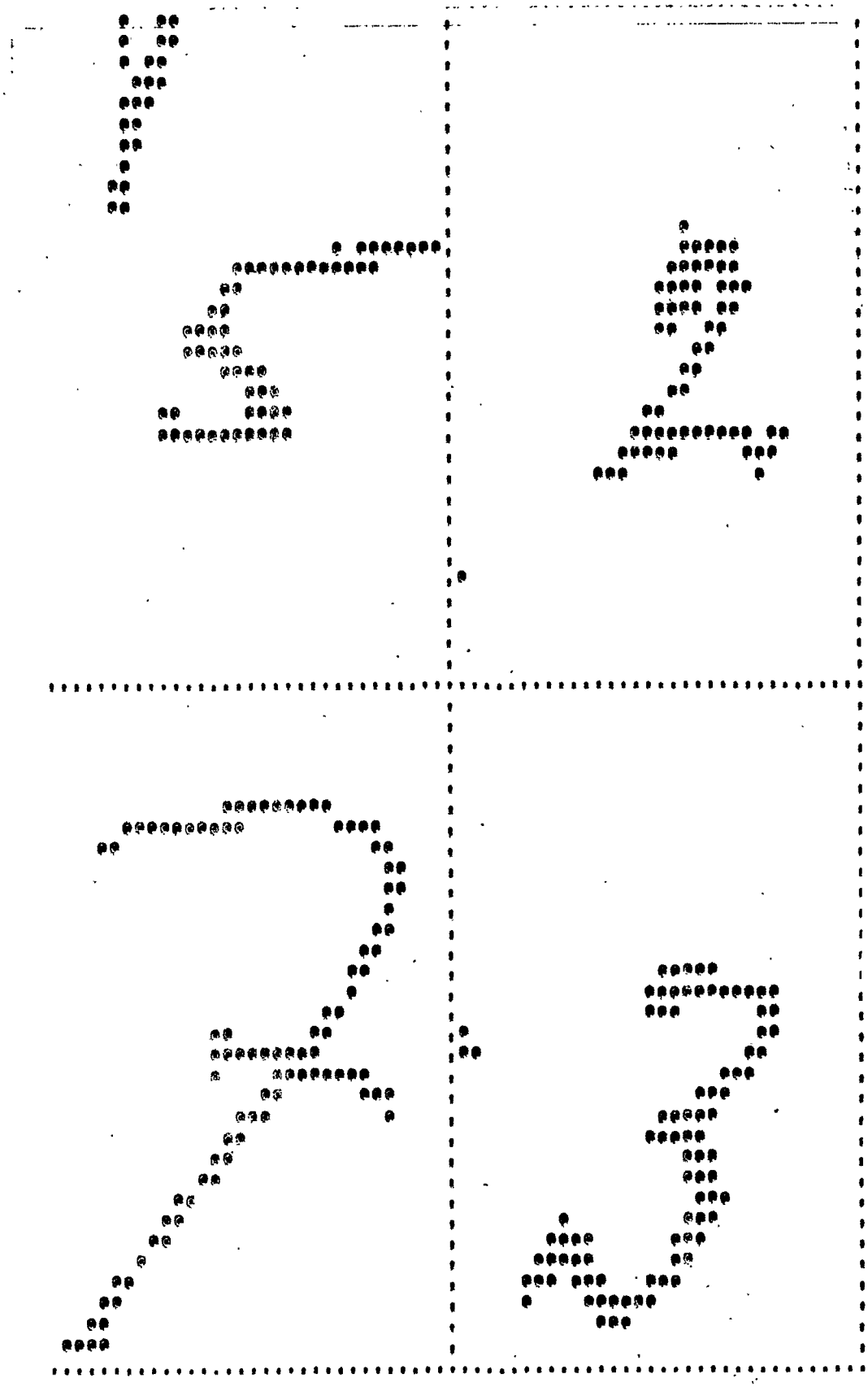


ILLUSTRATION 3

Handprint Character Samples



began linking the algorithms together.

As was mentioned earlier, all three algorithms were developed and simulated as independent research projects. Being "feature" type algorithms, the feature extraction sections were written as main programs with the less complicated pre-processors and recognition, or character decision sections, becoming subroutines. Illustration 4 illustrates this configuration of programs.

Each of the three programs was made to run smoothly by itself -- that is, the run/update/rerun sequence had become more or less routine and each was considered a moderate success as a research tool.

At this point we decided to merge the algorithms -- that is, modify all three systems to process the same input data set and report joint as well as individual character decisions. Illustration 5 is an example of the summary printout that gives both

ILLUSTRATION 4

Original Program Configuration

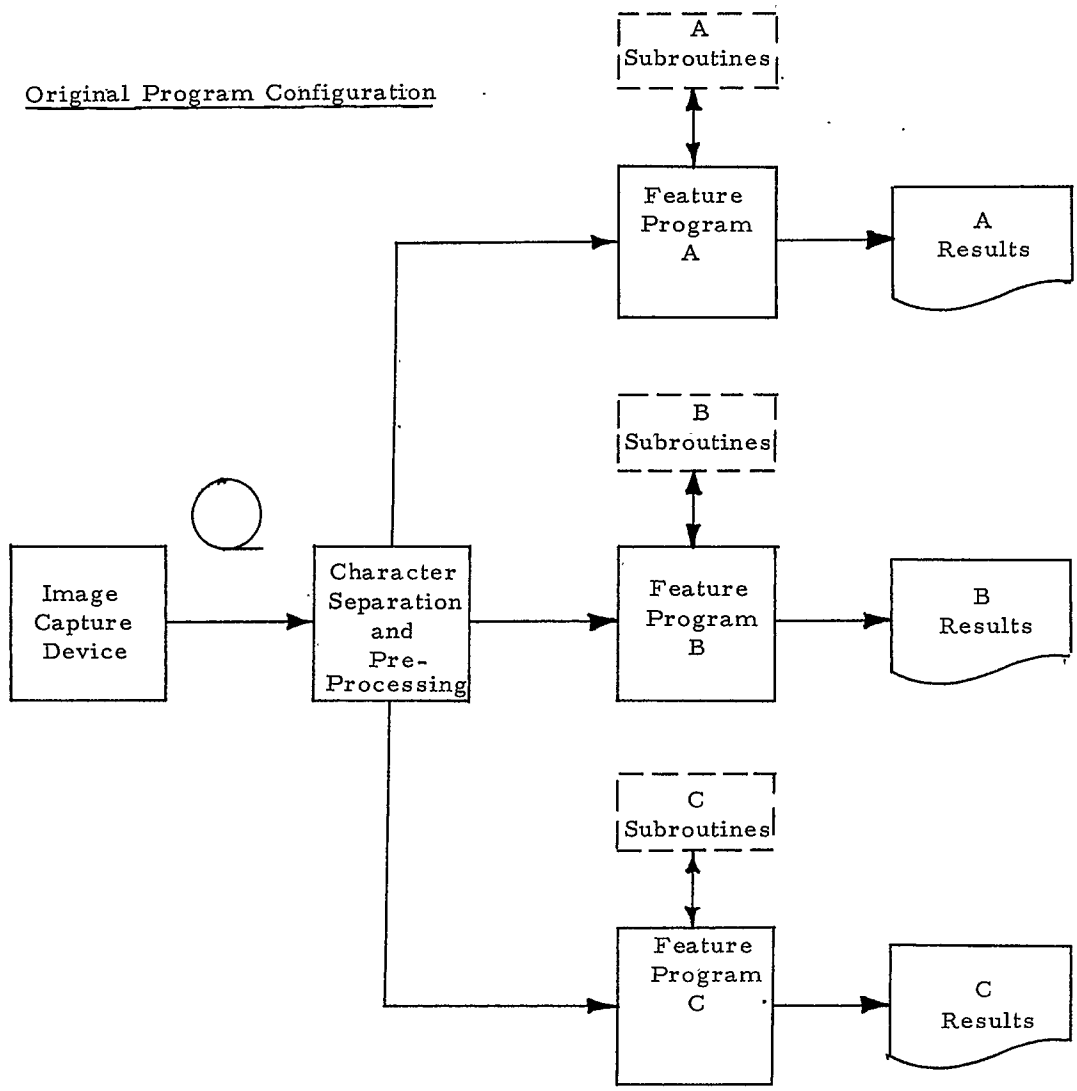


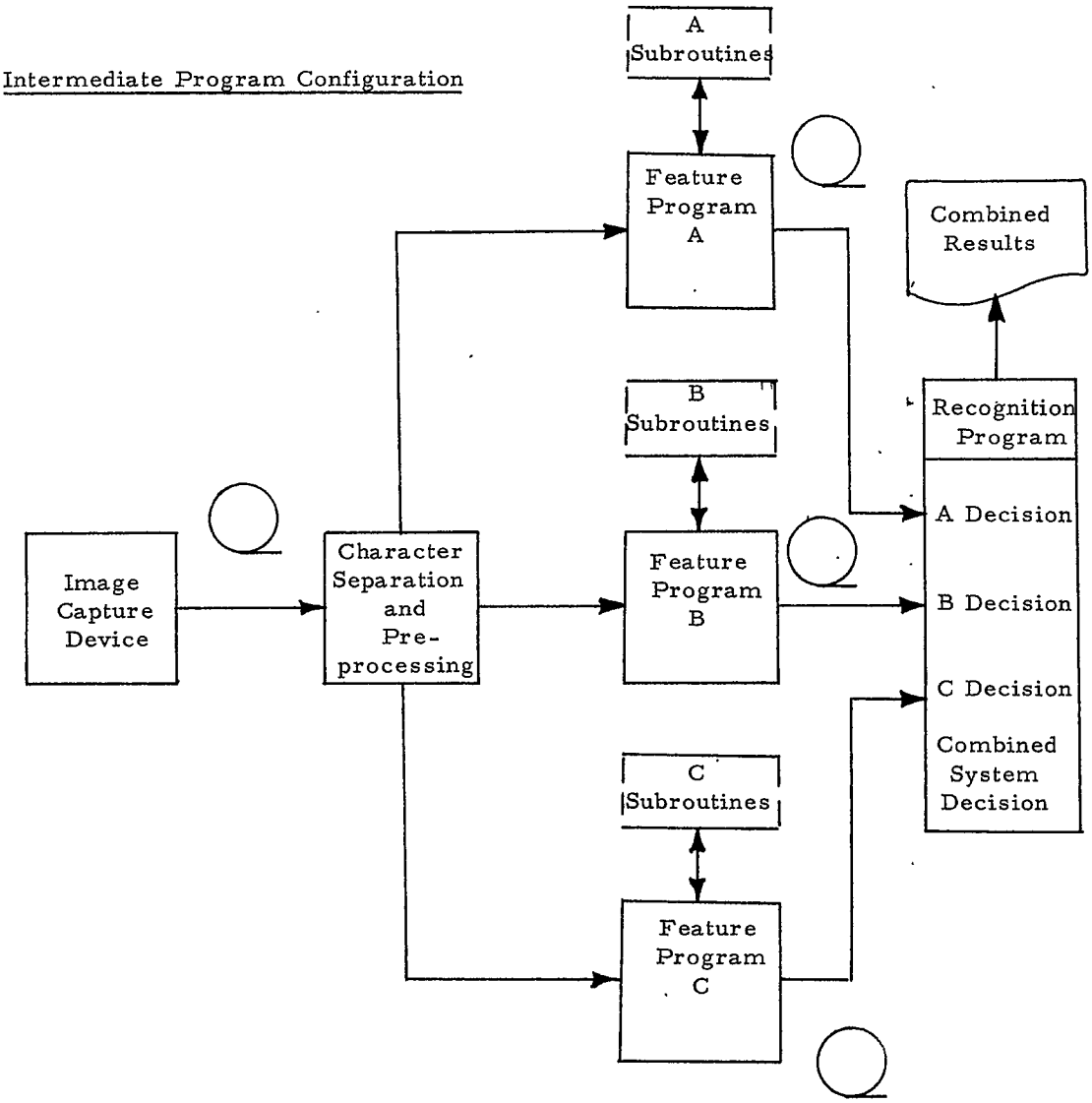
ILLUSTRATION 5

Recognition Summary Printout

| Algorithm Type | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | * | R | REJ | Total Characters | Percent |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | READ | 111 | 138 | 56 | 54 | 87 | 96 | 78 | 79 | 36 | 57 | 0 | 0 | 0 | 0 | 792 | 77,88 |
| | REJ | 2 | 3 | 37 | 30 | 16 | 30 | 1 | 9 | 14 | 21 | 0 | 0 | 4 | 3 | 163 | 16,03 |
| | ERROR | 1 | 2 | 18 | 13 | 7 | 4 | 1 | 3 | 5 | 8 | 0 | 0 | 0 | 0 | 62 | 6,10 |
| | C=ERR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ,00 |
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | * | R | REJ | | |
| B | READ | 91 | 137 | 44 | 62 | 93 | 74 | 37 | 34 | 25 | 12 | 0 | 0 | 0 | 0 | 609 | 59,88 |
| | REJ | 22 | 4 | 61 | 24 | 14 | 43 | 33 | 45 | 21 | 61 | 0 | 0 | 1 | 3 | 328 | 32,25 |
| | ERROR | 1 | 2 | 6 | 11 | 3 | 13 | 10 | 12 | 9 | 13 | 0 | 0 | 3 | 0 | 80 | 7,87 |
| | C=ERR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ,00 |
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | * | R | REJ | | |
| A/B | READ | 111 | 137 | 69 | 66 | 96 | 100 | 57 | 72 | 39 | 64 | 0 | 0 | 0 | 0 | 811 | 79,74 |
| | REJ | 2 | 4 | 42 | 25 | 12 | 28 | 23 | 16 | 14 | 15 | 0 | 0 | 4 | 3 | 181 | 17,80 |
| | ERROR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ,00 |
| | C=ERR | 1 | 2 | 0 | 6 | 2 | 2 | 0 | 3 | 2 | 7 | 0 | 0 | 0 | 0 | 25 | 2,46 |
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | * | R | REJ | | |
| C | READ | 101 | 143 | 89 | 64 | 92 | 106 | 71 | 60 | 39 | 65 | 0 | 0 | 0 | 0 | 830 | 81,61 |
| | REJ | 8 | 0 | 17 | 24 | 12 | 16 | 6 | 25 | 10 | 13 | 0 | 0 | 0 | 3 | 131 | 12,88 |
| | ERROR | 5 | 0 | 5 | 9 | 6 | 8 | 3 | 6 | 6 | 8 | 0 | 0 | 4 | 0 | 56 | 5,51 |
| | C=ERR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ,00 |
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | * | R | REJ | | |
| A/C | READ | 100 | 140 | 74 | 54 | 87 | 100 | 70 | 62 | 35 | 56 | 0 | 0 | 0 | 0 | 778 | 76,50 |
| | REJ | 14 | 3 | 36 | 41 | 22 | 29 | 10 | 28 | 18 | 27 | 0 | 0 | 1 | 3 | 228 | 22,42 |
| | ERROR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ,00 |
| | C=ERR | 0 | 0 | 1 | 2 | 1 | 1 | 0 | 1 | 2 | 3 | 0 | 0 | 3 | 0 | 11 | 1,08 |
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | * | R | REJ | | |
| B/C | READ | 99 | 139 | 77 | 60 | 91 | 91 | 49 | 55 | 29 | 57 | 0 | 0 | 0 | 0 | 747 | 73,45 |
| | REJ | 15 | 4 | 34 | 36 | 17 | 37 | 31 | 33 | 25 | 26 | 0 | 0 | 4 | 3 | 258 | 25,37 |
| | ERROR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ,00 |
| | C=ERR | 0 | 0 | 0 | 1 | 2 | 2 | 0 | 3 | 1 | 3 | 0 | 0 | 0 | 0 | 12 | 1,18 |
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | * | R | REJ | | |
| A/B/C | READ | 113 | 140 | 95 | 79 | 101 | 119 | 80 | 80 | 43 | 69 | 0 | 0 | 0 | 0 | 919 | 90,36 |
| | REJ | 1 | 3 | 15 | 14 | 7 | 10 | 0 | 8 | 9 | 14 | 0 | 0 | 1 | 3 | 81 | 7,96 |
| | ERROR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ,00 |
| | C=ERR | 0 | 0 | 1 | 4 | 2 | 1 | 0 | 3 | 3 | 3 | 0 | 0 | 3 | 0 | 17 | 1,67 |

ILLUSTRATION 6

Intermediate Program Configuration

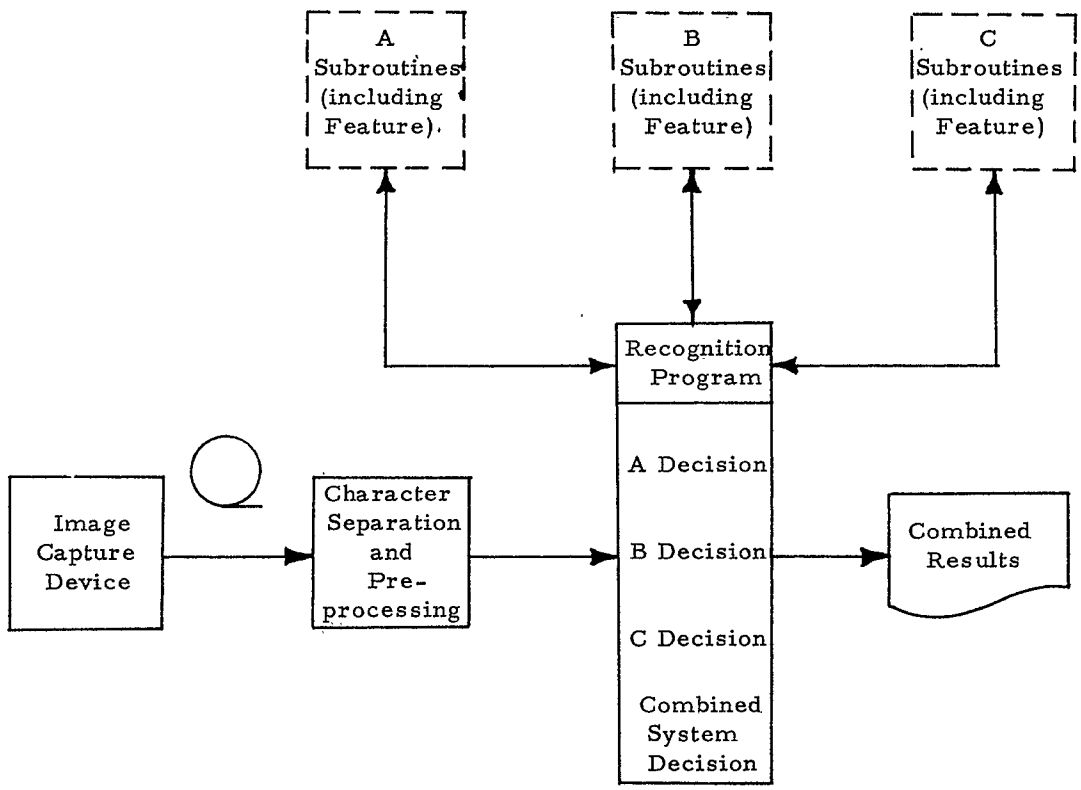



individual and joint performance results. In order to retain system flexibility, we left the feature extraction main programs as they were originally written but added mag tape output to a separate recognition and character decision program. Illustration 6 shows the modified configuration. We could now modify any or all the feature extraction programs and rerun the recognition program with new results, and still retain small core requirements. This new program called for a reorganization of the decision process and the addition of code necessary to combine the individual character decisions into a new "system decision", plus other programming problems associated with taking inputs into the system decision function from any one, any two, or all three algorithms.

The last major change came about when it appeared necessary to build in a "feedback loop" between algorithm "C" and the character decision. That is, based on a reinterrogation of the feature matrix with knowledge of the character decision, it may be desirable to reset some parameters and process the character image again. One way to accomplish this is to set up the communication system to "remember" the status of the program and flag the image to be reprocessed; then the entire series of programs could be rerun and the character so flagged would be processed with the appropriate modified parameter settings. The alternative approach was to make the feature

ILLUSTRATION 7

Final Program Configuration




extractor into a subroutine with the parameters in common. We selected the latter approach for reasons of speed. Since making algorithm "C" into a subroutine would leave us with a large core requirement anyway (over 65,000 words) we decided to make all three algorithms into subroutines. This was accomplished by removing the mag tape functions and rearranging blank and labeled common to maintain communications. A block diagram of the final system is shown in Illustration 7.

## IV. SOME INTERESTING ASPECTS OF THE SIMULATION

The programs are written in U. C. C.'s Fortran V and run on their dual Univac 1108 Dallas installation. There are 2 programs, 3 major subroutines, and 29 other subroutines and functions represented by over 4500 Fortran instructions. The simulation requires approximately 110K words of memory, and processes 1024 character samples in about 20 minutes. Approximately one man-year of programming effort was required to put the system together. Over 100,000 character samples have been processed by the algorithms.

The direct computing charges amount to approximately $.05 per sample, and on this basis, the simulation is a very efficient, economical vehicle for research and development of OCR. Of much heavier cost impact is the programming and debugging labor required to realize the operational programs. This cost probably represents four to five times the productive run costs. Heavy programming costs, however, are unavoidable during a development effort of this sort; more to the point, the programming is an integral part of the development process. Debugging must occur sometime, better in the software than hardware.

The programming techniques, for the most part, are rather straight-forward. Since the simulation played a key role in the development cycle, the central idea was to minimize program "down time" during modifications.

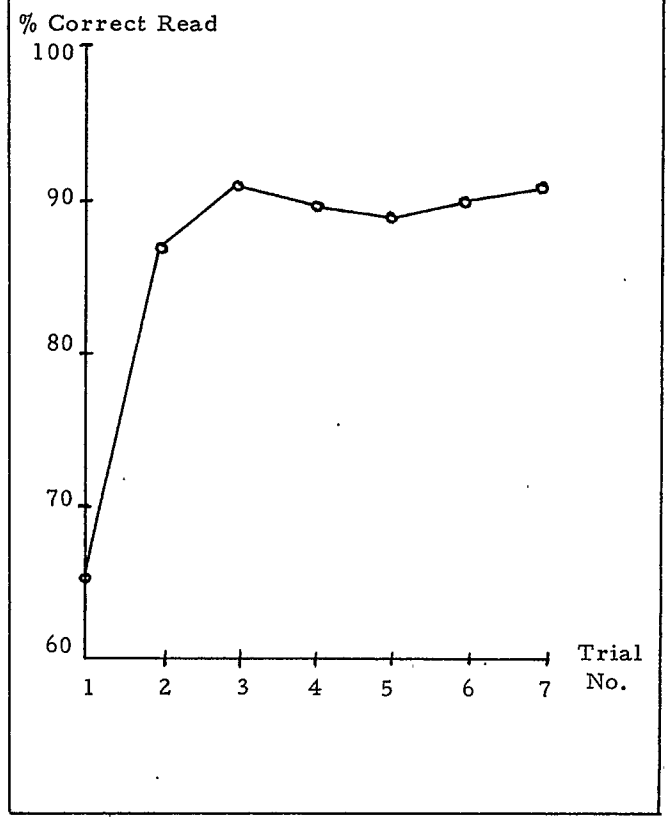
During development of esoteric OCR algorithms, it is necessary to expand, shorten, modify, or delete both large and small areas of programs. One of the main difficulties encountered was due to the fact that no section of code was sacred - nothing could be considered permanent and solidified. This tends to increase the debugging time and costs. The programming flexibility inherent in straightforward coding was responsible for keeping these costs from becoming excessive.

At the time of this writing, somewhere in the program we are destroying the linkage to the video input tape unit and are terminating abnormally during input of the 28th image. The effort required to detect, isolate, and correct bugs of this nature has increased with the complexity of the program. In the initial, simpler versions, the detection and isolation was trivial in that we knew immediately which algorithm didn't run successfully. Furthermore, the nature of the output usually indicated which section of that program was in error. Now, however, we can have one (or more) bug(s) in one (or more) of the subroutines(s), and the output gives us no indication of how many, or where. In this case, we must test the linkage (and reinstate it, if necessary) after the return from each major subroutine just to isolate the problem. Then we must carefully insert diagnostic print statements to determine which section of that routine is in error.

## V. THE OCR RESULTS

The last step in developing and improving an OCR system is completion of the decision process; this is referred to as the "training" phase. In this phase a large number of character samples are processed by simulation, and the decision

ILLUSTRATION 8
Illustrating Increased Learning During Training
(1000 Character Samples per Trial)




algorithm is updated to read as many of the samples as possible. Illustration 8 shows the results of a training phase for handprint data which was completed in June 1973 using the simulation described earlier. The illustration shows the results of reading 1000 samples of handprint on the 1108 computer for each of seven trials; between each trial the decision algorithms were modified. The shape of a typical learning curve is obvious, where rapid progress is made early in the effort, and performance eventually levels off. Approximately 8000 samples were processed in this particular effort, and a great deal was learned about the strengths and weaknesses of the algorithms involved as well as about the characteristics of handprinted numerals. The relatively low performance level of 90 percent read rate is indicative of the extreme variability of this particular set of input data. These results can be contrasted with the performance obtained with another set of handprint samples of somewhat higher quality. Table 9 illustrates this comparison. As can be seen, the second data set (consisting of approximately 20,000 samples) yields a 99.5 percent read rate where the first set gave a 90 percent rate. This sort of performance variation is not unusual for widely differing qualities of video input images.

TABLE 9

| | No. of Algorithms | Read | Reject | Error |
|---|---|---|---|---|
| Set 1 | 1 | 82% | 12% | 6% |
| | 2 | 78% | 21% | 1% |
| | 3 | 90% | 7% | 3% |
| | **No. of Algorithms** | **Read** | **Reject** | **Error** |
| Set 2 | 1 | 98.5% | 1% | 0.5% |
| | 2 | 98.5% | 1.5% | 0.0% |
| | 3 | 99.5% | 0.5% | 0.0% |

Comparative Recognition Results between two separate sets of hand-print data. Set 1 is much poorer quality than Set 2.

Table 9 also illustrates another interesting aspect of this OCR system: the verification effect of multiple recognition algorithms. The programs report the results of making character decisions based on only one algorithm, two algorithms, or all three algorithms. The use of two complementary algorithms (which also contain a certain amount of redundancy) drastically reduces the error rate over that of one algorithm, while affecting the read rate only slightly. Using the information provided by all three algorithms has a lessor effect in reducing errors, but the read rate is improved significantly over that of any single algorithm. This latter effect stems from the complementary nature of the recognition algorithms.

## VI. CONCLUSION

It has been our experience with several simulation efforts of this sort that the performance predicted by software closely matches that accomplished by the eventual hardware. The simulation serves as a tool for development and in addition provides a forecast of expected performance.

Simulation of OCR algorithms during the development stage of a new OCR system is invaluable, and we continually are expanding our use of this technique. The benefits of this simulation tool are limited primarily by the foresight and ingenuity of the engineer/programmer team.