

Chapter Three

Advanced Technologies for Display and Database Automation

3.1 Summary

The objective of this activity was to investigate advanced technology tools for transferring data into the Flight Standards Service (AFS) databases. A detailed study of the AFS database systems and the manner in which the Aviation Safety Inspectors (ASI) interacted with these database systems was completed in Chapter One. Based upon the results of this study, several advanced technology tools were identified and investigated. These technology tools included optical character recognition, personal digital assistants, and speech recognition. Each will be explained in detail in this report.

3.2 Optical Character Recognition

The data an ASI collects follows a circuitous path before it is entered into the appropriate AFS national database. Most ASIs today collect data during an inspection or investigation using brief notes written down on a notepad. When they return to the office they fill out the appropriate form(s) referring to the written notes from their note pads and their memory. The data from these forms is then entered into a local database at the office, often by a data entry clerk. This is a time consuming process that is prone to data entry errors of omission and commission. Time and money would be saved and data integrity would be improved if the number of times the data were handled was minimized. At some time in the future it is planned that the handling of the data would occur once using a single point entry job aid like the Inspectors Field Kit (IFK). Unfortunately this will not occur in the near future for all AFS offices. In the mean time, an optical character recognition application would provide the means to reduce the number of times the data were handled. With the help of it, data can be input into the computer by scanning the raw data directly from the initial form. This study was undertaken to find out how accurate this technology can be and what configuration(s) of software and hardware should be used to obtain the best results. The process and the results of the study are discussed below.

Various Optical Character Recognition software products were investigated for this activity. We identified 36 different OCR packages that were currently available. Features such as handwriting recognition, form removal, image enhancement options, and de-skew capability were identified as minimal OCR capabilities. In order to be compatible with the IFK we also looked for the ability to support industry standard programming languages (Visual Basic and C) within the Microsoft Windows operating environment. This would allow an OCR application to be tightly integrated into existing, as well as future, AFS systems. These requirements reduced the number of potential products to 12. Many of these had features that were not applicable to the proposed AFS office environment such as form creation utilities, stand alone receive and send FAX capability, batch scanning and classification algorithms, and graphic editing capabilities. The cost of these additional features was not deemed necessary just to get the basic required OCR capabilities. After reviewing product information and trade journal reviews, OMNItools by Nestor Products was purchased and evaluated.

As with all technology assessment activities, this review of products is current as of March, 1995. The introduction of new technologies is continuous and a better product may be available in the near future. We will continue to remain cognizant of these new systems as they are introduced.

OmniTools

OmniTools, developed by Nestor Incorporated, accurately recognizes hand-written and machine-printed characters from scanned or faxed images. In addition to character recognition, the system identifies documents, resizes and straightens incoming faxes, removes preprinted forms, automatically detects hand-printed and machine-printed words, reads optical mark (check box) entries, and provides several output options including alternate choice characters. It provides an Application Program Interface (API) for industry standard programming languages, such as Visual Basic, and also allows handwriting recognition solutions to be developed from Access, Excel, Foxpro, Lotus 123, and Paradox.

3.2.1 Recognition Process

The process of handwriting recognition using this product includes five steps: (1) preparing an image file, (2) defining a zone definition form, (3) performing the actual recognition, (4) displaying the results and (5) verifying the results. Except for the last, all of the steps can be done with the OMNItools software.

3.2.1.1 Preparation of Image Files

Since OMNItools cannot prepare image files, a basic scanning product was used to create the image file. This file contains the image of the hand-writing and form that will be recognized. An image can be captured by a camera, a scanner, or some other devices. All the image files used in this study were created using the Hewlett-Packard ScanJet IIc scanner and the Deskscan II software¹. Using this hardware and software combination, three attributes of the resulting image could be modified: image file format, brightness, and resolution. These attributes are commonly available to most scanners, and are discussed in detail below.

Image file format

OMNItools accepts either Paintbrush or Bitmap format files. Preliminary testing indicated that there was no difference in recognition accuracy between these two file types. Since the size of a Paintbrush file is usually four to five times smaller than the size of a Bitmap file, the image files were stored in a PaintBrush file format. The number of colors used in a PaintBrush file can also be set. Two color (black and white) PaintBrush files were used for all the tests in this study. The reason for this decision is that OMNItools has difficulty interpreting PaintBrush files that contain more than two colors.

Brightness

The Brightness attribute controls the contrast of an image. It is similar to the brightness control used in a photocopier. The darker you set the brightness attribute, the more details you can see. The range of the brightness scale for a ScanJet IIc scanner is from 0 (darkest) to 255 (lightest). By trial and error, two values, 80 and 150, were chosen to be used in this study. When the brightness is set to a value of 80, the scanned image is clear and sharp for both dark and light color lines. When the brightness value is set to 150, dark color lines appear clearly, while light color lines are not visible.

Resolution

The resolution is determined by the number of dots per inch (dpi) used to represent an image. In DeskScan II, the dpi setting can be set as low as 12 dpi or as high as 1200 dpi. A setting of 100 dpi resolution was used in the beginning. It was quickly determined that this value was too low because the image quality was very poor. A setting of 300 dpi was found to be the lowest setting that produced a high quality image for character recognition in the image file.

The density of data on the original form is also very important. The data density cannot contain too much data when the image is processed. How much is too much was determined empirically. When the form's data density was too high, an error message "Image is too large and complex" would be displayed. This phenomena was discovered during the initial familiarization period. This situation did not occur during the actual evaluation.

It warrants repeating that the brightness and the resolution attributes discussed above are specific to the scanner. They are not part of the OMNItools. Readers must keep this point in mind when these scanner settings are discussed in context of the OMNItools attributes.

3.2.1.2 Form Definition

A zone definition form (ZDF) file contains information of how OMNItools should recognize the image. The specific area or zone on the form that will contain data must be identified. You can also specify more information about each zone and about the whole form so that the software can carry out a more accurate recognition. There are three kinds of zones which can be defined in a ZDF file: registration zone, data zone, and optical mark recognition (OMR) zone.

Registration zone

This zone defines static information that can be deleted from the scanned image using a feature called Form Removal. Using a common tax form as an example, OMNItools would be able to remove those printed boxes, lines, and characters from the tax form as the scanned image was being processed. It would read the tax form as if it only contained the hand-writing of the author. To use Form Removal, registration zones have to be defined and two forms must be supplied: one blank form (called Master Form) and one form filled with data (Data Form). The registration marks are used as anchors between the Master Form and the Data Form so that the static data can be removed accurately.

Also, through trial and error it was found that 1)Form Removal worked best if the registration marks were some simple shapes like small, filled-in squares and 2)Locating the registration marks close together at the top third of the page provided the most accurate combination for the Form Removal feature to function properly. Locating these marks was not an easy task. It was discovered that characters or complex graphics cannot be used as registration marks. This would result in an error saying "Can't find the registration mark".

Define data zones

Data zones are areas which contain the hand writing that needs to be recognized. A single form can contain multiple data zones. Several attributes can be defined for each data zone, each of them intended to specify the number/type of characteristics within a data zone which would result in a higher rate of recognition accuracy. The attribute options available are:

I) Hand print / Machine print.

Specifies whether the zone contains hand-writing, machine-print characters, or both.

II) Characters Per Inch (CPI)

A data string typically has a constant letter spacing. If the number of characters per inch is known in advance, it can be specified so that the software will locate each character more accurately.

III) Punctuation Marks

Some data strings may or may not contain punctuation marks. Setting attribute "Containing Punctuation" to "none" will reduce the possibility of recognizing some characters as punctuation marks if no punctuation exists.

IV) Single Word, Single/Multiple Lines

The data format may be in one word, one line, or multiple lines. This option may help to eliminate recognizing extra spaces and recognizing one character as two.

V) Checking Context

If this is set, OMNItools uses an expert system for grammar in an attempt to guess the next character based on the context of the words if there is any uncertainty about that specific character. For instance, if the software is not sure whether the character is a 'v' or 'u' following a 'q', it would identify the character to be a 'u' because of higher grammatical probability.

VI) Define Dictionary.

A Dictionary can be used (or defined) to restrict the word for a specific zone. For instance, suppose a data zone is restricted to a dictionary that is defined as containing only two words ("MALE" and "FEMALE"). If, for example, a word "MALA" (or anything similar) was recognized, the word will be changed to "MALE" because of the dictionary constraint.

VII) Character constraint

It is usually known in advance what type of characters will appear in a data zone. For example, A-Z, and/or 0-9, or some unique combination are valid ranges.

Optical Mark Recognition (OMR) zones

These zones are areas on the form designated for check boxes. It can be chosen to be either a single radio box or a multiple choice group.

Form level information

There are some attributes which would affect the whole form. These options are:

I) Form Removal - ON/OFF

If this option is set, the static data will be removed from the image.

II) Remove long lines - ON/OFF

If this options is set, all the long lines will be removed.

III) Restore/de-skew image - ON/ OFF

If this option is set, OMNItools will try to restore image by de-skewing some lines that may have been skewed during the process of capturing the image from the source.

IV) Drop-out ink - ON/OF

This option tells OMNItools whether the image to be recognized contains drop-out ink or not.

V) Result file format - ON/OFF

Result file format can be text, text with quotes, or ZRF (file format defined by OMNItools). If the results are going to be verified using the OMNItools, the result file format has to be in Zone Recognition File (ZRF) format.

3.2.1.3 Recognition

The actual recognition can be done by specifying the image file and the ZDF file. It usually takes less than 10 seconds to finish one recognition with about 100 characters using a Pentium-90 machine.

3.2.1.4 Results Display

OMNItools provides a tool to view the ZRF file. It can display the results of all the data and OMR zones in a scrollable window. It can be chosen to show how accurate the recognition was for each character. Threshold confidence levels can be set such that it will only show the recognized character if the confidence level is higher than the specified value. Using this tool, results can also be stored in a plain text file or copied to other Windows' application.

3.2.1.5 Verification

The verification tool allows users to see the recognition and the original data at the same time. This allows the users to compare the results visually and correct any mistakes made by the software. As mentioned before, the results have to be stored in ZRF file in order to use this tool. The modified results can also be stored in the same ZRF file.

3.2.2 Methodology

3.2.2.1 Data Preparation

A modified version of the standard AFS form, Program Tracking and Reporting System (PTRS), was used for all the tests. This form was created using a software application called VISIO. There are a number of modifications made to the basic form in order to be suitable to be tested. Four registration zones were defined. Data fields were enlarged to have a density of 6 CPI. Letters within any check box were removed to avoid any confusion between these letters and the actual hand-writing. The gray scale of boxes that surrounded the data fields were set to be the lightest shade available in the VISIO application. This was done to minimize the interference with the recognition process. The modified PTRS form is shown in the Appendix.

3.2.2.2 Data Collection

Ten Galaxy Scientific Corporation employees participated in this study. A sample form and a blank form were given to each participant. They were instructed to copy the information from the sample form to the blank for in their own handwriting. The sample form contains five class of data that included every recognizable character:

- 1) Uppercase : three sets of upper case alphabets (A-Z)
- 2) Lowercase : three sets of lower case alphabets (a-z)
- 3) Digits : five sets of digits (0-9)

4) Special : three sets of special characters ("\$()+~/")

5) Checkmark : thirty-six check mark zones

There are a number of attributes can be changed in the ZDF file that were discussed in previous section. Some of these attributes remained fixed for the study while others were varied. The following section lists the fixed and variable attributers.

Fixed attributes:

I) Hand print / Machine print.

Handprint was used throughout the entire study.

II) Punctuation

No data will contain any punctuation.

III) Single line

All data was regarded as single line data.

IV) No context checking

The sample characters were listed alphabetically, therefore context was not used.

V) No dictionary defined

For the same reason as above, no dictionary was defined.

VI) No drop-out ink

No drop-out ink was used.

VII) Result file format

Text file format was used.

Variable attributes:

I) Form Removal

The options tested were On or Off. The static data was removed from the image if "On" was selected.

II) Remove long lines

The options tested were On or Off. If " On" was selected then all the long lines were removed.

III) Restore/de-skew image

The options tested were On or Off. This attribute can only be used when Form Removal is "Off", otherwise the error message "Can't find the registration mark" would occur.

VI) Character constraint

This attribute only applies for the data zone. There are two kinds of constraints used in the tests: restricted constraint and relax constraint. For the restricted constraint option, the data zone was restricted to one of three settings:

- alphabetic characters defined to be from A and Z;
- digits defined to be from 0 and 9;
- special characters defined to be anything (A-Z,0-9,\$()+-/).

For relax constraint option, every data zone was defined to recognize all characters.

V) Characters per inch (CPI)

This attribute only applies to the data zone. It can be set to any positive value. The data form was originally designed for a character spacing of 6 CPI. Therefore this option was set to either 6 CPI or 0 CPI for these tests. The 0 CPI option means that the characters per inch attribute is undefined. In other words, the software has to locate the characters by itself.

VI) Brightness of the image

The options tested were 80 or 150.

There are a total of 48 possible combinations for above attributes. [Table 3.1](#) illustrates the options more clearly:

Table 3.1 Options

Attributes	Options	
Characters per inch	0 (Undefined)	6
Form Removal	Yes	No
Remove Long Lines	Yes	No
Restore/de-skew image	Yes	No
Character Constraint	Restricted	Relaxed
	(A-Z or 0-9)	(all characters)
Scanner Brightness	80	150

Ten sets of hand writing were collected. Each set of data was recognized using all 48 configurations mentioned above. The results were stored in text file format and printed out. The printout were counted manually to determine the number of correctly recognized characters per set of data.

3.2.3 Results & Analysis

For each configuration that represented the 48 unique combinations of variables, an average percentage of accuracy was calculated across the ten participants. The data for each variable attribute was divided into two groups (since each attribute has two options). A line graph was plotted using these two groups. The difference between these two lines indicates how much the specific attribute affects the recognition accuracy.

Since data zones and OMR zones have different attributes, their results were analyzed separately. The results were illustrated with the graphs. It needs to be emphasized for the reader that each graph represents a different set of configurations of variables to illustrate the effects of each attribute. For example, Configuration 1 in [Figure 3.1](#) may not be the same as the Configuration 1 in Figure 3.2. Detailed information for each graph indicating configurations and actual values can be found in the Appendix.

3.2.3.1 Data Zones Average results

Figure 3.1 shows the average results for each configuration. The table shows the corresponding statistical values. Descriptions for each configuration are in the Appendix. There are several rises and falls on the graph. They are caused by different variable attributes. The effects of each attribute will be discussed individually.

Average	Standard Deviation	Minimum	Maximum
74.28%	5.85%	61.88%	84.51%

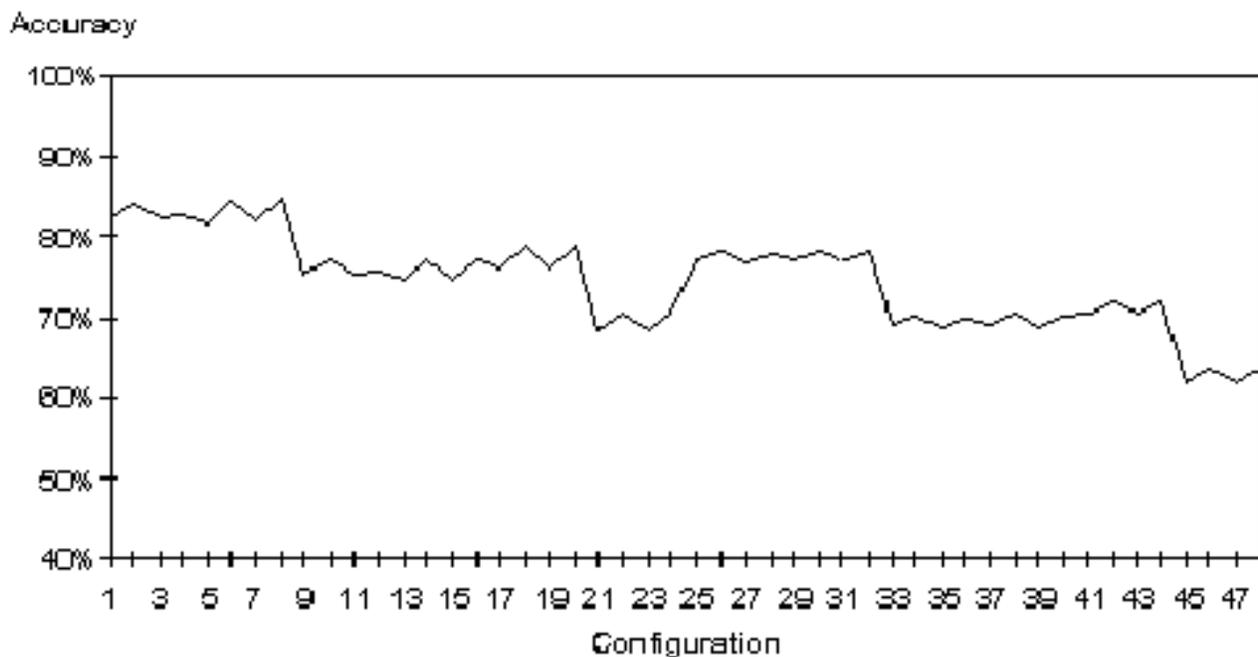


Figure 3.1 Average results for each configuration

Character constraints

Figure 3.2 shows how the attribute Character Constraint affects the recognition accuracy. The first eight configurations have a restricted constraint while the last eight configurations have a relax constraint. There is a drop in accuracy (7.06%) when the constraint is changed from restricted to relax. This is quite reasonable because there are a number of letters which are easily recognized as digits and vice versa (e. g., 5 & S, 1 & l, 9 and q, etc.). It is also found that the trend appears to be the same for the brightness variable and the form removal attribute.

Constraint	Restricted	Relax	Difference
Average	82.85%	75.79%	7.06%
Std Dev.	1.12%	1.16%	-

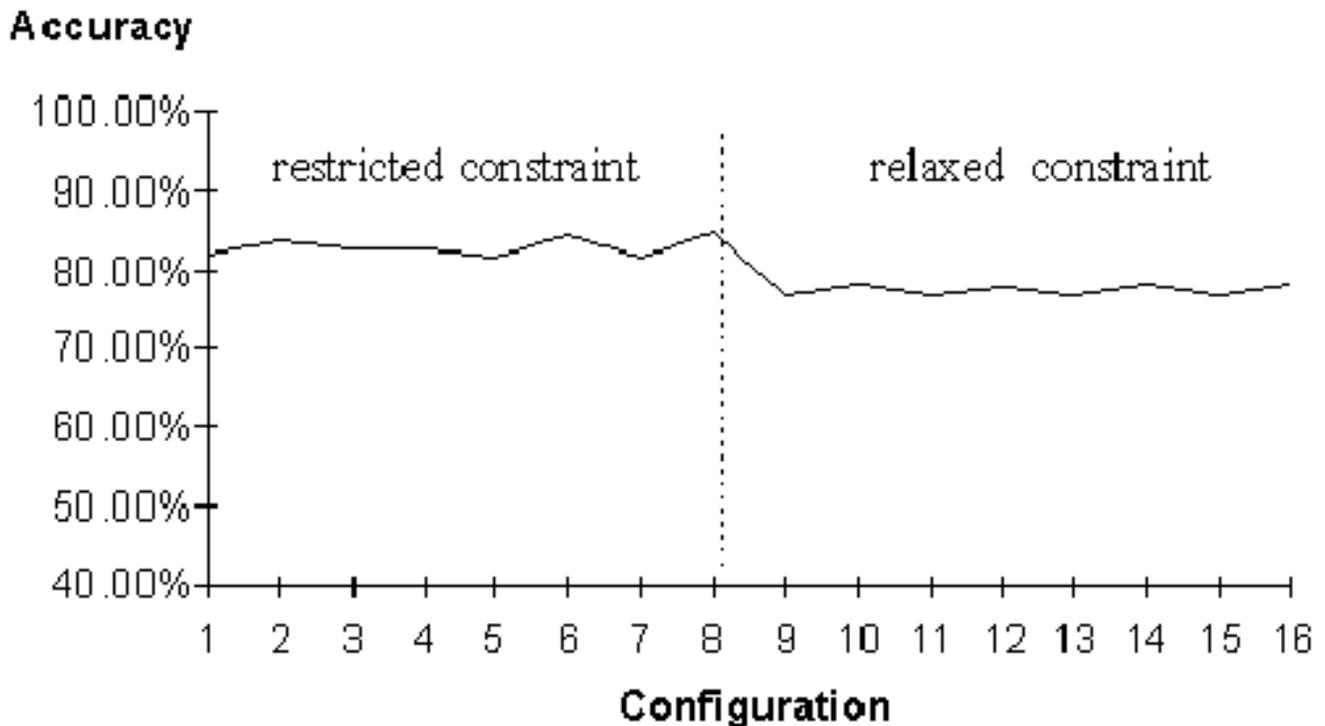


Figure 3.2 Configurations with brightness = 150 and no form removal applied

Brightness

Figure 3.3 shows the results of restricted constraint without form removal. It again shows that there is a decrease (5.54%) in accuracy in the middle of the graph. That is the result of a change of the brightness attribute from 150 to 80. Similar trends were found using different combinations of character constraint and form removal attributes.

Brightness	150	80	Difference
Average	82.85%	77.31%	5.54%
Std. Dev.	1.21%	0.65%	-

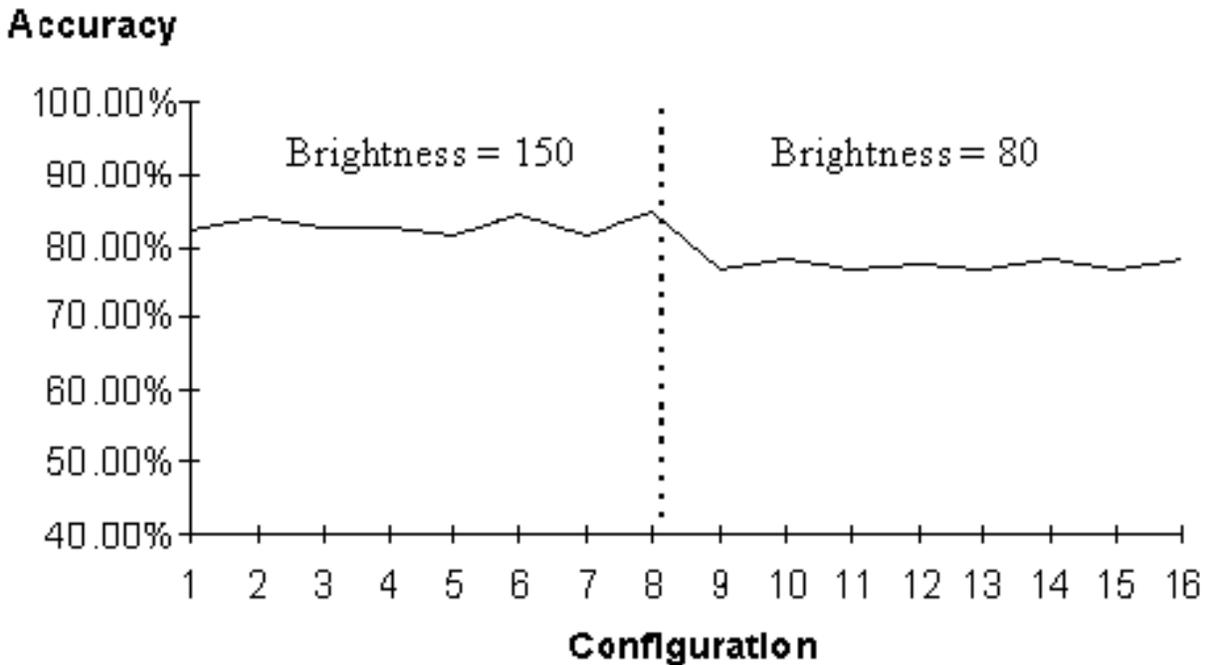


Figure 3.3 Configurations with restricted constraint and no form removal applied

Form removal

Figure 3.4 shows the results of configurations with restricted constraint. Two series of data are shown on the graph. One uses Brightness = 150 and the other uses Brightness = 80. Both are showing the same distribution. The respective decreases (5.33% and 6.31%) in the middle of the graph are caused by the form removal feature. It was believed that the removal of the form would improve the recognition accuracy. The results were opposite from what was expected and there is no obvious explanations for this.

Brightness = 150			
Form Removal	No	Yes	Difference
Average	82.91%	77.38%	5.33%
Std. Dev.	1.39%	1.39%	-

Brightness = 80			
Form Removal	No	Yes	Difference
Average	77.38%	71.07%	6.31%
Std. Dev.	0.71%	0.77%	-

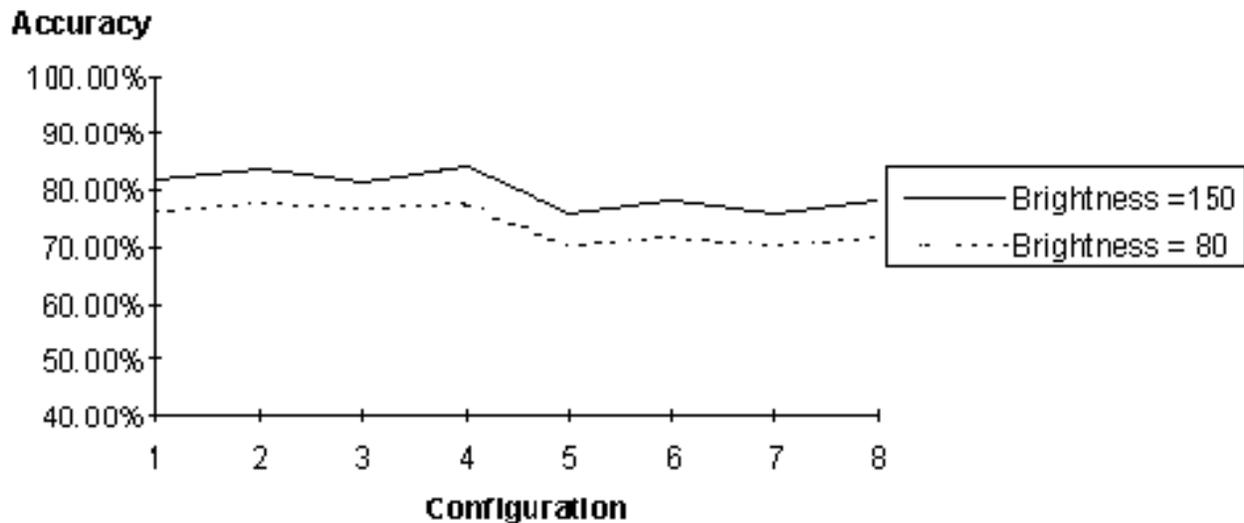


Figure 3.4 Configurations with restricted constraint

Character constraints, brightness and form removal were found to be the only attributes that affect the recognition accuracy. It is also found that defining the characters per inch attribute would help improving the recognition accuracy a little (as shown in [Table 3.2](#) below). The de-skew and long line removal attributes do not affect the recognition accuracy at all.

Table 3.2 Average and Standard Deviation

CPI	6	0	Difference
Average	75.11%	73.45%	1.66%
Std. Dev.	5.92%	5.79%	-

Digits with different character constraint

Figure 3.5 shows the results for digits class only. It uses brightness = 150 without form removal. The accuracy drops dramatically (20.95%) when the constraint is changed from restricted to relax. The change is so big because there are only 10 possibilities (0-9) for restricted constraint and there are 36 possibilities (0-9,A-Z) for relax constraint. The other attributes do not affect the accuracy of digits class at all.

Constraint	Restricted	Relax	Difference
Average	92.15%	71.20%	20.95%
Std. Dev.	1.77%	1.19%	-

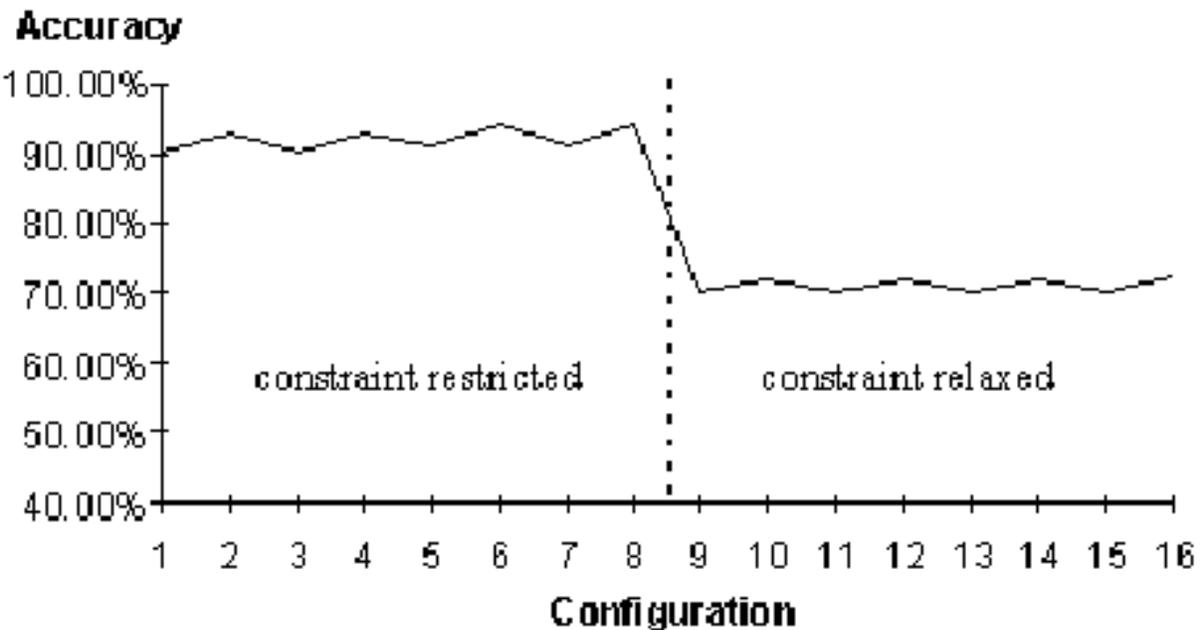


Figure 3.5 Digits class with brightness = 150 and no form removal applied

Special characters with different brightness

The only attribute that affected the accuracy for special characters class was Brightness. Figure 3.6 shows the results of special character recognition without form removal and de-skew. The recognition accuracy falls (25%) when the brightness is changed from 150 to 80. This behavior is difficult to explain.

Brightness	150	80	Difference
Average	76.94%	51.94%	25%
Std. Dev.	2.25%	2.59%	-

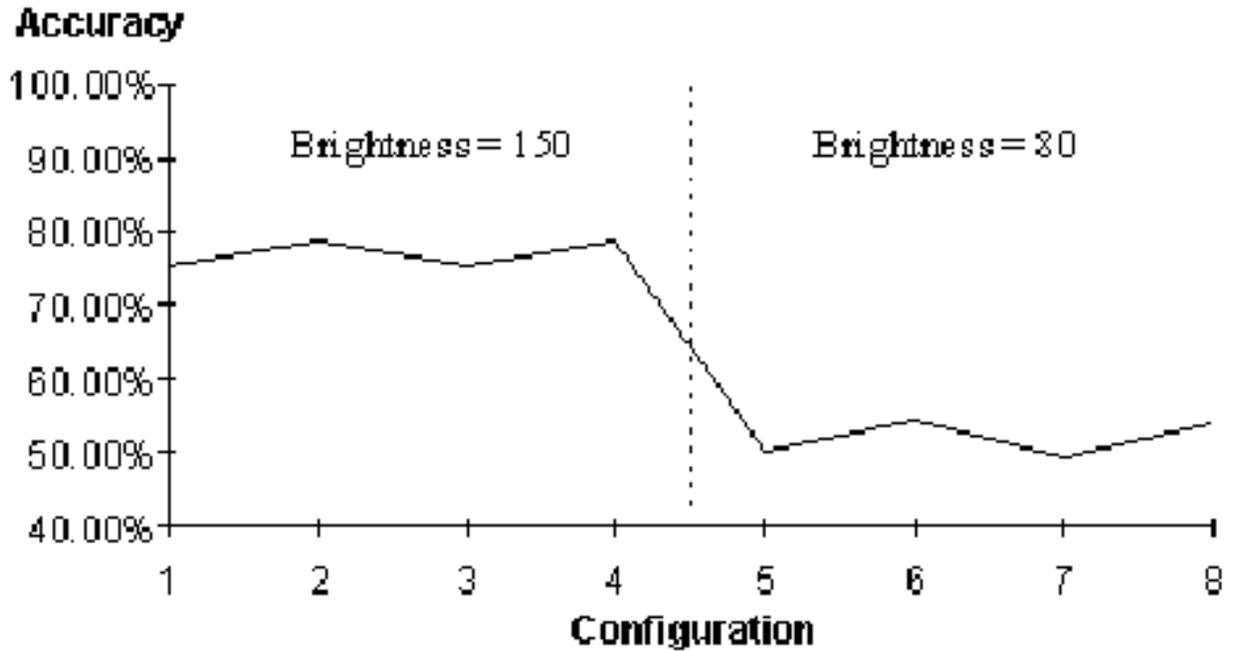


Figure 3.6 Special character class without form removal and deskew

Upper and lower cases characters

Figure 3.7 shows the recognition accuracy results of upper cases and lower cases. It can be observed that the accuracy of upper case characters was much better than that of lower cases. One of the reasons may be that some of the lower case hand writing characters are very similar in shape to each other. For example, e & a, g & q, r & v, n & h, etc.

Class	Upper cases	Lower cases	Difference	Overall excluding lower cases
Average	83.73 %	62.86%	20.87%	10.95%
Std. Dev.	3.99%	7.09%	-	-

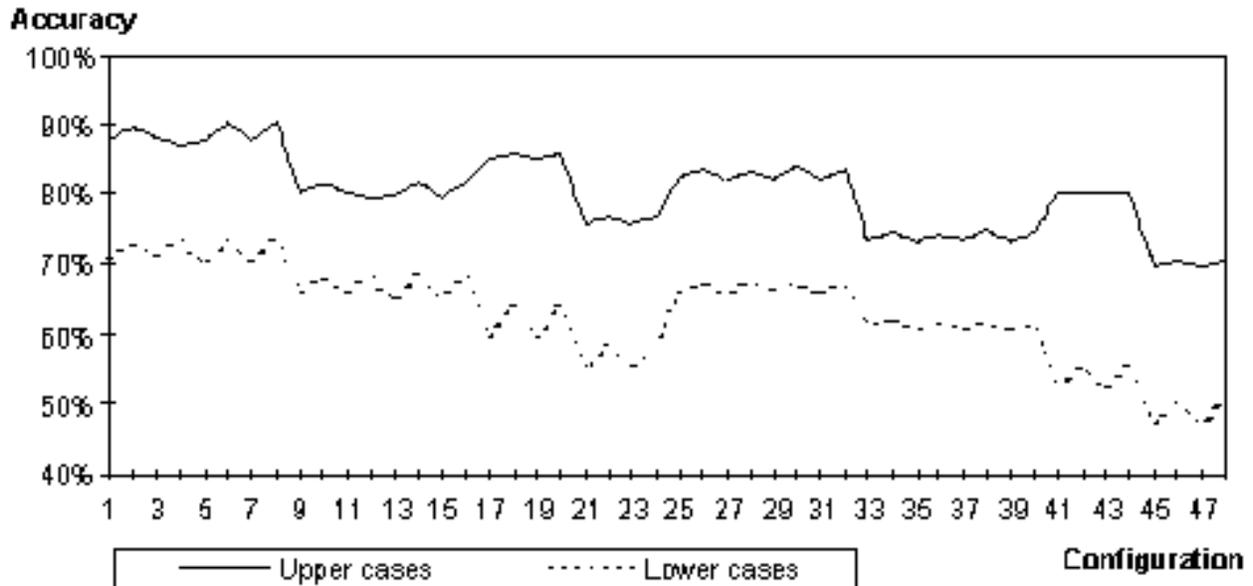


Figure 3.7 Comparison of upper cases and lower cases performance

OMR zones

Figure 3.8 shows the results for OMR zones. The results are quite bad. The reason is that OMNItools was not able to discern if an OMR zone is completely empty. Considering this observation, it assumed that all the uncertain results are actually unmarked. With this assumption, the results improved significantly. The modified results are shown on the same figure. The drop in the middle is caused by enabling the form removal attribute. Again this decrease in recognition accuracy when Form Removal is activated was not expected.

	Original Results	Modified Results	Difference
Average	31.65 %	87.26%	55.61%

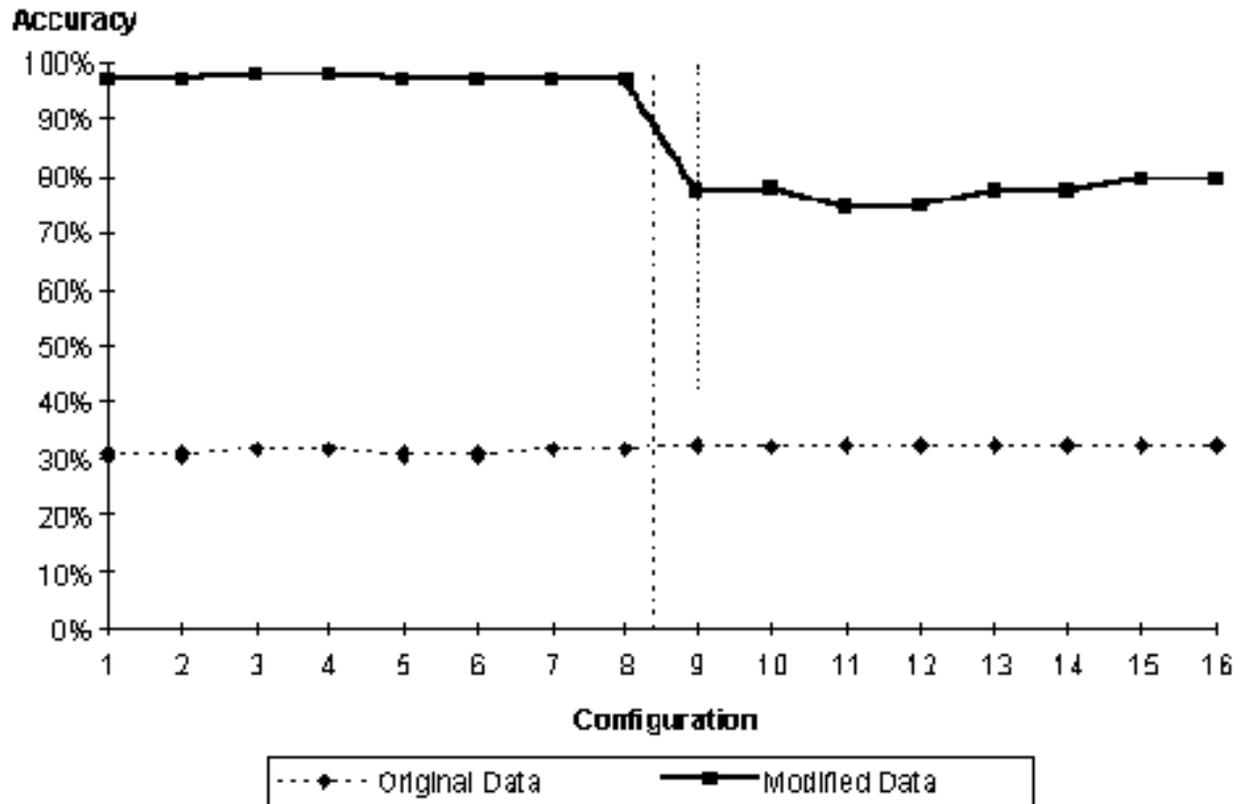


Figure 3.8 Results for OMR Zones

Optical Character Recognition Suggestions

OMNItools is an acceptable hand writing recognition program that achieved approximately 85% recognition accuracy during our evaluations. However, careful design of source data and proper choice of attributes are very important to obtain the best possible results. The following design recommendations and usage for (1) source form design, (2) source form filling, (3) scanner settings, and (4) OMNItools settings are suggestions to maximize the recognition accuracy based upon the results of this study.

Design Recommendation:

1. Source form design
2. Source form tilling
3. Scanner setting
4. Omni tools setting

Source Form Design

1. The source form has to be designed carefully. Borders should be used to surround each character of the input text field. This will control the size and the density of characters (CPI) when data is entered on the form. Either rectangle boxes or comb fields would be a good choice. These boxes or comb fields should also be printed in some lightest shade possible. This will allow the boxes or comb fields to be dropped out easily during the image process.

Source Form Tilling

2. The form should be filled out using black ink pen. It is also suggested to use all capital letters. The form should be designed to carefully organize the density of data otherwise OMNItools may not be able to carry out the recognition.

Scanner Setting

3. The image of the source form should be in black and white only. The resolution of the scanner should be at least 300 dpi. The brightness should be chosen such that the boxes or comb fields (drawn in the lightest shade possible) are not recognized during the image process. Different scanners will require some experimenting in order to find the best value of these (or other) available attributes.

Omni Tools Setting

4. When defining the zone definition form, form removal is not suggested to use. Effort should be put on arranging the zones such that each area contains only either alphabets, or digits, or a small range of characters. It is also helpful to define the characters per inch for each zone. The results of the OMR zone should be modified by replacing the uncertain results as unchecked.

3.3 Personal Digital Assistants

The Personal Digital Assistant (PDA) industry is in a state of transition. The first generation of these devices were greeted with great expectations. They were to provide personal electronic-based tools designed to improve the productivity of its owner in a pocket-sized package. Unfortunately these expectations greatly outpaced reality. The root of the problem was this product was developed in a vacuum then its manufacturers went in search of a consumer market. These first generation devices attempted to be all things to all users. As it turned out, very few consumers purchased these products and those that did were disappointed with their performance.

This lack luster performance by PDA sales has caused concern among the manufacturers and the software providers for these products. New processors and operating systems that were planned for release this year have been delayed indefinitely. Some in the industry have gone so far as to declare the PDA concept as dead. Based upon our research we feel this is too extreme. There is a market for these devices but they must take a user-centered approach when developing the new generation of these systems.

The two key features that would make a PDA a useful tool for an AFS ASI are a PDA operating system that is compatible with their desktop computers (e.g., Windows compatible) and communications connectivity. Unfortunately both of these capabilities are either not available on today's PDAs or are very limited. Because of these reasons we did not pursue further investigation of PDA for ASI use at this time. [Table 3.3](#) contains the specification of several PDAs that we investigated.

These units are roughly seven inches by five inches, less than an inch thick, and weigh about one pound. They utilize a low power processor, up to 4 MB RAM, one Type II PCMCIA slot and a three inch by four inch monochrome LCD display. The PDAs use either a pen-based DOS or a proprietary pen-based operating system. Handwriting recognition is only one element of a pen-based operating system. The specific user interface metaphors are different but overall, they all utilize the pointing ability of the pen for navigating through the operating system or interacting with an application. They also capture graphic images by storing "electronic ink"

As this research effort came to a close in March 1995, a second generation of PDAs have started to emerge. The early press releases describing these new PDAs claim to have addressed the two features that would make these units useful for an ASI. We will be continuing to keep abreast of these products through Task Order 03.

Table 3.3 PDA Specifications

	PDA #1	PDA #2	PDA #3	PDA #4	PDA #5
Processor	Casio Custom (x86 compatible)	ARM 610/20 MHz 68349/16 MHz	ARM 610/20 MHz 68349/16 Mhz	Motorola Dragon MHz	Motorola Dragon NEC VG230/16
HDD	PCMCIA	PCMCIA	PCMCIA	PCMCIA	PCMCIA
Storage	memory card	memory card	memory card	memory card	memory card
RAM	1MB (4 MB ROM)	1MB (4 MB ROM)	1MB (4 MB ROM)	512 KB (4 MB ROM)	3 MB 6 MB Mask
Desktop	PC (option)	Mac & Windows	PC (option)	Mac & Windows	Serial LapLink
Connect	Connectivity (opt)		Connectivity pack		
Display	3.1" x 4" 320 x 256 pixels	3" x 4" 336 x 240 pixels	3.0" x 4.5" 480 x 320 pixels	3.0" x 4.5" 480 x 320 pixels	7.4" diag 640x400 pix

Battery Three AA bat., NiCad, NiCad, Six AAA bat., Six AA
AC/DC Four AA bat. AC/DC AC/DC bat.,
AC/DC AC/DC

Operating Env. GEOS, PenRight! Newton Intelligence Magic Cap Magic Cap GEOS

Size, 6.8"x4.2"x1.0" 8.0"x4.0"x1.25" 7.5"x5.75"x1.2" 7.5"x5.2"x1.0" 9.2"x6.4"x
Weight 0.95 lbs 1.28 lbs 1.7 lbs 1.2 lbs 1.4" 2.4 lbs

Options Message card, Message card, Message card, Pager LapLink,
fax modem, fax modem, printer connection, card, Li-Ion Bat.
printer printer flash storage. keyboard,
connection, connection, head set,
flash storage. flash storage. memory card.

Voice recognition is being investigated as a viable means of allowing ASIs to interact with the AFS database systems. Interaction could be in the form of allowing ASIs to use voice not only to enter data into the system but to also control other applications on their desktop computers. Speech recognition systems can be either speaker-dependent or speaker-independent and either continuous-speech or discrete-speech. Most voice recognition systems today use the speaker-dependent, discrete-speech recognition technology. The effort by most manufactures of voice recognition systems is directed to developing a speaker-independent, continuous speech product. In theory, speaker-independent systems would recognize speech equally well for all users and allow for users to speak in natural, continuous speech. In reality, recognition accuracy of current systems is less than perfect and typically deemed unacceptable for most uses. This less than optimal recognition accuracy is due primarily to the difficulty in accounting for the many variations of the American English dialect. On the other hand, speaker-dependent, discrete-speech systems can achieve optimal recognition accuracy for one user at a time. This is due to the fact that discrete-speech systems requires that users speak with a pause between words and perform a series of exercises to develop a voice template for that specific user. The current effort has concentrated on the latest products that have a quasi-speaker-independent capability which provides the option to increase the recognition accuracy with a brief enrollment exercise performed by the user.

Two important factors were being used as the criteria for evaluating these software packages. First, the appropriate package had to be Personal Computing Manufacturing Community International Association (PCMCIA) compatible. This feature will allow Voice Recognition Software to run on a Notebook Computer, thus enabling an ASI to use the software while he/she is out in the field. Second, the appropriate package must support industry standard programming languages (Visual Basic and C), which will allow voice recognition to be tightly integrated into existing as well as future AFS systems.

The following is a brief summary of several voice recognition software packages that were investigated.

3.4.1 DragonDictate from Dragon Systems

DragonDictate uses the speaker-independent, discrete speech recognition technology and features a large active vocabulary which users can totally customize with their own words. A unique 110,000-word pronunciation dictionary with acoustic models makes adding words easy and ensures immediate recognition. DragonDictate's special vocabulary optimizer automatically personalizes your vocabulary while its dynamic adaptation capability adjusts to your voice and work environment. In addition, it allows you to control various applications by voice.

The major drawback to the DragonDictate voice recognition software is that it is not PCMCIA compatible. However, it does have a C language Application Programming Interface (API).

3.4.2 Kurzweil Voice for Windows from Kurzweil Applied Intelligence, Inc.

Kurzweil Voice enables users to create, test, and enter data simply by speaking. It also allows navigation, which drives the Windows Operating System and Windows-based applications on a command and control basis. Kurzweil Voice incorporates a new version of Kurzweil Artificial Intelligence's (AI) large vocabulary, speaker-independent, discrete speech recognition technology. It also has on-line knowledge, including acoustic recognition models and spellings, for a total of 200,000 words.

Although Kurzweil Voice is impressive, it does not have a PCMCIA interface. It neither has a C language nor a Visual Basic API.

3.4.3 Custom Voice/ICSS from A&G Graphics Interface

Custom Voice/ICCS uses a speaker-independent, continuous speech system. It provides a simple high-level interface to speech recognition via standard Visual Basic programming tools. Custom voice is geared towards the developer. It allows a developer to create customized applications quickly, using industry standard development tools.

Although the system has a Visual Basic API, it only has a 1,000 word vocabulary and it is not PCMCIA compatible.

3.4.4 IBM VoiceType Dictation for Windows from IBM:

This is a discrete, speaker-dependent voice recognition system. It allows users to create text quickly and efficiently simply by talking to a desktop personal computer, notebook or subnotebook PC. The software gives users a basic 30,000 word vocabulary, as well as specialized language models for specific professions (including radiology, surgery, and law) to speed the accurate conversion of words to text. VoiceType Dictation creates a model based on each user's voice, which allows the system to accurately convert speech to text as quickly as 70 to 100 words a minute.

The IBM VoiceType Dictation system was purchased for evaluation because it is PCMCIA compatible and it has a Visual Basic and C language API. The system was evaluated and the results of the evaluation are as follows.

The first effort with this system was to perform the enrollment procedures. This generates a voice template for each individual user that the application uses to aid in recognizing speech from a particular individual. This also provides an opportunity for the user to become familiar with the capabilities of this system. Once this was completed, testing began on the recognition accuracy and speed. Finally, the Speech Software Development Kit (SDK) was investigated to figure out how to incorporate this technology into existing application in the most unobtrusive fashion.

3.4.4.1 Enrollment Process

The enrollment process for the dictation system is long and thorough. Enrollment is a two part process: your dictation and the computer's processing of your voice information. During enrollment, VoiceType gives you a set of sentences to dictate. The system already knows these sentences, so the pronunciation of each word assists the system to learn how these words sound in the user's voice. It does not appear to be too burdensome to request that a user go through this process in order to achieve the highest level of voice recognition. In general, recognition is reasonably quick and highly accurate. In Paced/Command mode, the recognition engine is usually in the 97% - 99% accuracy range and is very quick. This is due mainly to the use of a smaller vocabulary and the simple sound matching processing that is done to recognize words. In Dictation mode, the recognition engine is considerably slower. This engine uses a language model to narrow down the possible word options and better interpret what is being said; not only on the actual word sound, but on the context of the sentence. This helps reach higher levels of recognition accuracy than can be achieved through sound matching alone.

However, the recognition is not flawless. The recognition engine had the most trouble with words containing multiple syllables. Occasionally, the recognition engine would recognize a single, multiple syllable word as multiple words. This would cause the recognition can get way off-track using Dictation mode. Since the engine attempts to use the context of what is being said, a stray word or severely mis-recognized word can totally change the meaning of a sentence. Thus, the context is incorrect and recognition engine will briefly cascade off into the wrong direction. Fortunately, this was not a frequent problem.

Background noise was not a serious problem. Extraneous noises will affect recognition, but there are solutions to minimize its effect. One option is for the user to create several training sessions. One training session could be in the quiet environment of an office and another for a noisy outdoor setting. It was noted that one item that would really help in speaking to the recognition engine is an on/off switch. A hardware push button could be used to activate the microphone only when this button is depressed. This will help in eliminating extraneous noise and vocabulary that might be accidentally spoken by the user.

When there is an error in recognition while dictating, it can be somewhat tedious to correct the engine. Fortunately, there are programming aids that can be created to assist the user in the correction process. The effort put forth in enrollment and error correction does pay off in the long run in recognition accuracy.

Due to complicated modeling that is being done internally, the speech recognition engine uses a large amount of memory (16MB of RAM) and disk space. The larger the recognition vocabulary, the larger the storage requirements. In addition, recognition is fairly hard disk drive intensive. This may come into play in application designed for mobile computers where battery life is a factor.

One final point should be noted. From discussions with IBM technical support, it was discovered that the system does have a speaker independent model that can be used for simple commands and minor dictation. This type of speaker independent model is used in the sample programs that came with the system. Sample games provided called States and Weather were used to evaluate this feature. The independent models used in these programs were crude but functional. They worked, but their accuracy percentage is very low; around 60-70 percent. It does not appear that there is any way to use a speaker independent model, even on a small scale, and bypass the user having to go through the long enrollment process. Anyway, it appears that IBM does not want developers to use this independent model. They will not give any information on how to access the independent model through the SDK.

3.4.4.2 Speech Software Development Kit (SDK)

The SDK is very thorough and very complex. We were only able to scratch the surface of all the functions and capabilities that are accessible through it. There are several good demonstration programs included in the package. We relied heavily on these to get an initial demonstration program operating. One feature that was not known earlier was that the SDK will only work in a 32-bit environment. That means Windows NT or the WIN32S add-on to Windows 3.1 is required. We had trouble creating a dynamic linked library (DLL) with the speech API that Visual Basic could handle. It was discovered that VB cannot handle the 32-bit interface required for DLLs using the speech API. Instead an executable file called TALK was created that was run once the target application was started. This program recognized all words and passed messages concerning recognition to the target application.

When defining command vocabularies for the user to specify fields to input data, the engine does not allow for multiple words in a command. In general, any word in the 'Text' vocabulary can be recognized as a command. The developer must then manually keep track of words said by the user and interpret a given series of words as needed. There are procedures to add words to the vocabulary along with proper pronunciation. However, this process is fairly complicated thus was not deeply pursued.

The engine has the capability to replay, any recognized word, in an audio format. In a sense, the user can speak a particular command and the engine can replay what it recognized. This can be used as a verification mechanism for the user to know that his words were processed properly.

3.4.4.3 PENS Integration

The task of integrating the engine into the PENS software was a complex one. Due the nature on the engine, it will not be as simple as dropping a special control into the project in order to integrate speech. There will some intruding code that will be necessary to process and interpret commands and actions received from the speech engine. Because of this, there will be an additional level of complexity added to the software. This may cause some difficulties with programs that are already very complex.

In order to accommodate the existing PENS software, a command structure must be developed that will allow the user to easily and intuitively set the software's focus on a specific field for input. In addition, the structure must allow the user to easily and quickly input numbers, letters, or dictated text.

The overlying software structure consisted of a series of message levels such that recognized words are passed through different levels of functionality within the PENS software. The first level is the upper level tab navigation. Each form is made up of several tabbed folders. All command words are first passed to the main tab folder for the current data entry form in use. If a command word for tab control is recognized, then that causes a change in the currently active tab folder. Otherwise, if the command word is recognized as a valid data entry section for the currently active tab folder, then that data section is then given the focus. Code for this data section would then check the command words to see if it was attempting to recognize a particular field within this section. Once a data entry field is recognized through a command word, it is then given the focus. At this point, all commands or dictated text are passed to this control for data input.

All messages are thus passed down this chain until it is recognized as a command or field name, or until it is used as input into the active field. This cascading message loop was implemented in a sample PENS application that was developed during this evaluation of the speech software. On this small scale, it appears that this approach will work and is fairly foolproof. It appears that speech recognition would be fully integrated into PENS. However, it will require additional work considering the complexity and sheer volume of forms and controls in the PENS software.

3.4.4.4 Conclusions on IBM VoiceType Dictation System

The IBM VoiceType Dictation system is at the cutting edge of recognition based upon the initial requirements for this type of system. It can provide fast, accurate speech recognition for an data entry and system control application. However, much work still remains to be done. This system will add a considerable amount of computational overhead to a given application. It also requires a large amount of memory and disk space. On the other hand, it can really speed user input; particularly for users that must input large amounts of information or have difficulties with current 'standard' input devices or with typing. Recognition of dictated text by the engine is probably faster and more accurate than what 90% of people can type. If this software is to be used in an application, the primary reason should be to take advantage of its dictation capabilities. One needs to seriously weigh the benefits of this package with the need for this product and the effort that is willing to be given to incorporate it into an application. The demands that this product would put on some mobile computing application might be too great at this time. We do think though that this product is currently capable of adding a substantial level of usability to several applications presently being developed by the Performance Enhancement System program.

3.5 Conclusions and Recommendation

The three technology areas that were investigated all demonstrated that they each have a strong potential to enhance the Performance Enhancement Program. In a follow-on program we are planning to take each of these technologies and integrate them into the Performance Enhancement System program software and perform a series of preliminary evaluations.