Automated Traffic Accident Data Entry

Project No. 1068

Sharon B. Johnson and Kimberly Hofer
Computer Science and Computer Engineering
313 Engineering Hall
University of Arkansas
Fayetteville, AR 72701

12 October 1999

Prepared in Cooperation with the U.S. Department of Transportation,
Federal Highway Administration.

CONTENTS

Automated Traffic Accident Data Entry

Sharon B. Johnson and Kimberly Hofer
Computer Science and Computer Engineering
313 Engineering Hall
University of Arkansas
Fayetteville, AR 72701

Abstract The purpose of this project was to develop a method to automatically enter traffic accident records into an existing database and to provide a computerized method of archiving the forms. This involved redesigning the existing forms, developing a method for scanning them, performing optical character recognition on all the fields, and designing a system for storage of the forms. This project involved mainly finding appropriate existing software to accomplish all of the above.

I. Preliminary Testing

Initial testing was done with the existing traffic accident report forms designed by the Arkansas State Police (ASP), using software commonly supplied with scanners (Paper Port and OmniPage). It was found that these forms were not spaced correctly for typewritten input, either vertically or horizontally. There were also a number of "black boxes" on the forms which caused problems with the optical character recognition (OCR) process. A major problem with the OCR of the forms occurred when characters touched or extended across lines.

It was also found that the storage requirements for the forms was excessive. It was estimated that a year's worth of accidents would take about 30 gigabytes of storage.

The conclusion as a result of initial testing was that the forms would have to be redesigned to increase the accuracy of the OCR process and decrease storage requirements.

II. Forms Development

OmniForm, from the Caere Corporation, was chosen for forms development. This software has settings designed for typewriter spacing, and was found to be very accurate compared to other products which we tried.

OmniForm allows for the naming of each of the input fields in a form which can be used for interfacing the entries with a database. Another Caere product, the OmniForm OLE Automation Programmer's Reference, will provide the name, location, and type of each of the input fields. This proved to be very useful in that the forms can be redesigned without requiring any changes in the OCR processing as the names and locations of each of the fields can be updated in a file to be read by the OCR programs.

In order to alleviate the problem of typewritten characters crossing lines or labels on the forms it was determined that the forms should be done in some color other than black. We obtained examples of pantone colors and tested them on scanners and copiers to find what colors could be optically filtered out during scanning but still be maintained during copying. A number of colors were found that could be used. After consultation with the

ASP and AHTD, a fairly light blue (cyan) was chosen.

Filtering out the form during scanning had another benefit in that the storage requirements for the forms was also considerably reduced. A form which would have taken about 125-150 megabytes of storage now required only 20-30 megabytes. This brought up the problem of reproducing the form, but that was left to be dealt with later.

While most of the redesigned forms were to be printed in cyan, we found it necessary to add some features to be printed in black. One was the forms identifier. There are 5 separate forms which may be part of an accident record. We added a form number, in black, in exactly the same location on each form. This was to be used by the OCR programs to identify which file to use for identifying the input fields.

Another feature we added were fiduciary marks to the upper left and lower left of each form. These are used to aid in deskewing the form after scanning, ensuring that the forms are "lined up" (i.e. each field is in the correct position), and for determining if the form was scanned upside-down.

III. Scanning

Most of the testing and development was done using Hewlett-Packard 4C scanners, with a sheet feeder. These scanners are sufficiently accurate to use for this project, however they are much too slow to be used in an actual implementation of the project. A high-speed black and white scanner was purchased to test for use in the implementation of the project. This was a Panasonic KV-SS50 scanner. This scanner proved to be very inaccurate. The size errors from the top to the bottom of a page could be as much as 1/5 inch, which is larger than the height of a typewritten line. There was also a considerable amount of "waviness" on the horizontal. The scanner was replaced 4 times, with the same results with each replacement scanner.

We concluded that the problem was with the design of the scanner. The paper to be scanned follows a horizontal, then vertical, path across the lamp, rather than the lamp moving across the paper as in a flatbed scanner. A fast flatbed scanner would be more appropriate for use in this project. The Panasonic scanner was delivered to AHTD for use in other applications.

Any scanning software that is capable of doing unattended scanning, adjusting the brightness of the image, and storing the image in Tagged Image File Format (TIFF) CCITT Group 4 compression format, would be appropriate for use. We used PaperPort, OmniPage, and PixView for testing purposes.

IV. Preprocessing

After the scanning, the image files are processed to ensure that they are not upside down. They are also deskewed if necessary, and the images are "moved" to ensure that the fiduciary marks are in the correct location. While all of these operations were possible to program in a combination of Visual BASIC and Borland C, we were unable to find any algorithms or inexpensive software capable of writing the resultant images in the Group 4 compression format.

The solution was to purchase a set of Active-X modules (ScanFix) from TMS Sequoia. These modules, which interface with Visual BASIC, accomplish all the required operations and both read and write the desired file format. This solution, however, adds about $200, in licensing fees, to each installation of the system.

The preprocessing software also does some preliminary OCR on the images to determine whether or not the accident report number and form number are readable. If they are, the file is saved with a filename containing the report and form number. The header of the image file is then modified to ensure all the files have the same format header. A field is added to the header to indicate the form number. If the report number is readable, but not the form number, the image is still accepted but “flagged” so as to not be OCR’d.

V. Optical Character Recognition

The OCR portion of the project was the most time-consuming endeavor. We initially experimented with the OCR package that is included with most scanners - OmniPage. While OmniPage is a very accurate OCR package, we found that we were unable to find out where on the form any of the data was located.

We then purchased OmniPage Pro. This package allows “zoning” of the image, so that the source of the data is known. However, only about 15 zones are allowed. One of the forms has 265 separate fields. This could have been done with multiple passes on the same form, but would have been very time consuming. Another problem with this package was that the zoning appeared to be destructive. If the zones were defined from top to bottom or left to right, subsequent zones appeared to be empty, i.e. zone 1 appeared to destroy the contents of zone 2. A third problem with using this software was that it was not automated, and required constant operator intervention.

The solution to this was to purchase the Professional OCR Developer’s Kit from Caere Corporation. This package provides modules to interface with a number of languages, including Visual BASIC. This package features almost unlimited zoning capability, allowing us to define zones for each entry on the traffic accident record forms. The output may be read from memory and entered directly into the database. We found none of the “self-destructive” tendency as was apparent in OmniPage Pro.

The primary problem we found with this package was that the zoning procedure was very time-consuming. We solved this by indicating the form number as the filename extension and processing all identical form numbers consecutively. The zoning for each form then had to be done only once.

Use of this package adds about $800 in licensing fees to each installation, however the project would not have been feasible without it.

VI. Forms Storage and Retrieval

As stated previously, TIFF with CCITT Group 4 compression was chosen to use for storage of the forms. This is a widely used format, supported by virtually all scanning software. The Group 4 compression provides the smallest, lossless, images of the formats supported by the OCR software. An umcompressed image requires 1 MB of storage. A compressed version, including both the data and the form, requires approximately 150 KB of storage.

Optical removal of the form image on scanning also considerably decreased the size of the image files which were stored. The average size of the compressed image files was reduced to the 20 to 30 KB range.

Removal of the form image, however, created an interesting problem. Since the images are being saved to provide an easily accessible archive of the original accident reports, the original appearance of the forms had to be reproduced. This could be accomplished by maintaining one scanned copy of the original forms (without any data entered) and merging that image with the image of the data alone.

After a considerable amount of searching we found a package which would allow us to do that. This package was ImagnX from Pegasus Imaging Corporation. This software has Active-X controls which can be used from Visual BASIC to display the images. The controls allow for scrolling, zooming, and all the other functions normally associated with image viewing software.

In addition to these features, it will also provide the memory location where uncompressed versions of the images are being stored. This allowed us to load the image of the form, the image of the data, and merge the two for display on the screen. An image print function is also included so that hard copies of the records can be made. There are no licensing fees for this software as long as there are less than 1000 installations.

VII. Implementation

The implementation of this project involves four steps - scanning, "fixing", OCRing, and retrieving the traffic accident report forms.

Scanning: This can be accomplished using any scanning software that can do batch scanning, has adjustable brightness controls so that the form image can be removed, and can store the resultant image in a 300 dpi TIFF file using CCITT Group 4 compression. The image files may be stored in any directory.

Fixing: After scanning, the program *FixForms.exe* must be run. The operator may choose the directory from which the raw forms are to be retrieved. This program "moves" the image as necessary to ensure that the upper left fiduciary mark is in the correct location. The form number and accident number are then OCR'd and used as the filename for the image. The TIFF image header is modified so that all headers are in identical format (not all scanning programs create the same format header), and the form number is placed in the header. This is used later to determine which form image should be used to display this report.

The operator may view the images, and correct the form number or accident number if necessary. The operator also may correct any images scanned upside down. All accepted images are written to a predetermined directory - {ReadyForOCR}.

This program is written in Visual BASIC 5 and requires the following packages :

OCR Engine, Caere Corp.	$800 license
ScanFix, TMS Sequoia	$200 license
ImagnX, Pegasus Imaging	no license

OCRing: After *FixForms*, *DoOCR.exe* must be run. This program uses the images in subdirectory {ReadyForOCR}. The images are OCR'd and the resultant data is written into the database. The zones to be OCR'd are in .zon files, which contain the field locations and the name to be used for the field in the database. The processed images are then moved to subdirectory {DB-Images}.

This program is written in Visual BASIC 5 and requires the following package :

OCR Engine, Caere Corp.	$800 license

Retrieving: The images in the {DB-Images} subdirectory may be viewed using the program *DBQuery.exe*. The images are retrieved by accident number, but may be selected based on county, road, etc. All of the images associated with a particular accident will have the same filename, but different extensions. All forms for one accident are retrieved, and the user may “scroll” through the individual forms. The forms may also be printed at this time.

This program is written in Visual BASIC 5 and requires the following package :
ImagnX, Pegasus Imaging   no license

VIII. Conclusions

The results of this project convinced me that this approach is feasible, but the chances that it will be implemented are very slim. In order for this to be successful, the report forms must be typewritten, and the entries on the forms must be totally contained within the allocated space for each item. The revised forms also have to be accepted by the Arkansas State Police, and used by all the reporting agencies within the state.

I think the one important idea which arose from this project was that of being able to optically remove the form image when scanning, store only the data image, and merge the two images when the form must be reproduced. This requires that the form be in a color which can be “removed”, but it would also be possible to “mask out” the form after scanning using an XOR operation. TMS Sequoia (from whom we purchased the ScanFix software) does have FormFix software which is designed to do this. As long as OCR does not need to be performed on the image, this would be acceptable.