WHITE PAPER

Next-Generation Fingerprint Card Scanning: New Advanced Features and Functionality

AWARE

Next-Generation Fingerprint Card Scanning: New Advanced Features and Functionality

Aware offers a next-generation fingerprint card scanning software solution that utilizes mature, field-proven COTS components, and recently equipped with advanced capabilities that come out of ongoing research and development, as well as customer feedback. The products now include new algorithms that contribute to their performance and functionality, including OCR correctness confidence and line removal. Central to our newly improved products are new form recognition capabilities for enhanced fingerprint image extraction and form alignment. Our ongoing R&D has resulted in these and other improvements and major new versions of the FormScannerMB and FormScannerSE applications as well as improvements to the AccuScan and SequenceCheck SDK products. The following sections describe the new features added to these products. Product descriptions and a summary of improvements are also provided.

Advanced Algorithm Research and Development

Optical Character Recognition (OCR) Confidence Metric

OCR can potentially be extremely useful in the fingerprint card scanning process in order to automate the accurate capture of text-based data from cards. The inherent difficulty of the OCR problem is exacerbated for legacy forms due to the varied quality of form images. In addition, given the mediocre performance of some OCR/ICR engines on poor quality images, Aware has determined that for OCR/ICR to be useful to an operator, an appropriate mechanism had to be developed to aid the operator in quickly identifying possible OCR/ICR errors. In this way, an operator's attention could be directed to OCR strings that might have a high probability of error, thus improving operator efficiency and reducing their tedium in validating the OCR result.

Unfortunately, very few OCR engines offer confidence metrics for their OCR results, and the ones that do only have a confidence metric on a character by character basis. Consequently, a prototype language-independent[1] confidence metric was developed by Aware to enable a confidence measure to be computed for entire words.

Although our approach involved training on the entire English dictionary as a proof of concept, it is more accurate and useful to train on an appropriate subset related to the fields of a record. For example, one could train on a subset of colors, or Arabic names, or acronyms, thus tuning the confidence metric to its local domain. Note that the infrastructure for this training was developed in a general way to enable its application to other languages.

Figure 1 shows a screen shot of how the confidence metric can be used in the context of a data entry solution. In this case, the three letter tuple at the end of the word "Jobn" was flagged as having a medium probability of an OCR error due to the abnormal suffix "obn". The mixed letter/number combination was also flagged using a simple rule that relates to the type of field (numeric).

Future work will involve more extensive testing of the confidence metric, as well as training on appropriate subsets of words related to the field values.

1 "Language independent" refers to languages with an alphabet.

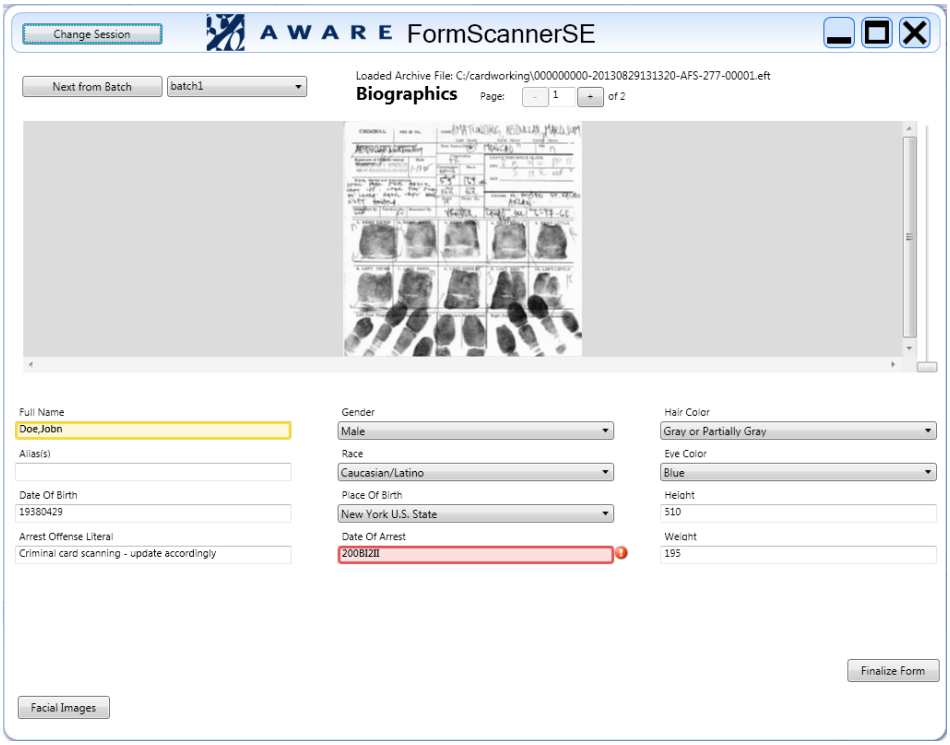

Figure 1 - Screen shot of a three level confidence metric applied to OCR text recognition. Yellow signifies an OCR string that has a medium probability of having an OCR error. Red indicates a high probability of an OCR error.

Line Removal

A new API function, AwSeqRemoveLines () has been added to the SequenceCheck SDK library that can remove lines from non-fingerprint areas of an image. Line removal comprises two equally important components: line detection and an algorithm for "removing" lines. Any line removal algorithm must include line detection as a preprocessing step, since lines cannot be reasonably removed unless they are first detected. At its core, line detection is an image segmentation problem, which requires a very precise definition of what constitutes a "line" in order to facilitate its detection.

First and foremost, we can accept that a line is defined as a set of points that satisfy a linear equation. The points that make up a line in the context of a fingerprint image (the application targeted by this work) differentiate themselves from the background in the same way as the relevant data for the fingerprint differentiates itself from the background. In other words, if we consider dark fingerprints on a light background, lines would also be dark linear structures on a light background and vice versa. Accepting this preliminary definition, the following factors affect how a line in such an image is further defined:

1. **the scale at which points in the image are considered to be points**
2. **the degree of connectivity required between points to consider points to be adjacent**
3. **the number of adjacent points (length) required to constitute a line segment**
4. **the angle of the line segment in the image**

With these factors in mind, an algorithm was implemented that provides parameters that affect the behavior of its line segment detector with respect to these criteria

Once line detection identifies the location of all the lines in the image to be removed, an "inpainting" algorithm is used to "remove" the regions masked by the lines from the input image. "Inpainting" is a term derived from art restoration, traditionally done by skilled artists who would "retouch" images to remove unwanted marks, scratches, etc. Automated inpainting algorithms can be quite sophisticated and time consuming due to the complexity of the processing. In general, the underlying image structure surrounding the area of the image to be removed is used to guide the reconstruction of the removed areas. The inpainting problem how-

ever, is ill posed; i.e., unless assumptions are made, missing information cannot generally be recovered with 100% accuracy from the surrounding image structure. Furthermore, fast numerical implementations are difficult to achieve.

Fortunately, inpainting lines in the context of fingerprint images can rely on a relatively simple image model, provided line removal is not attempted inside fingerprint regions. As in ghost removal (discussed in Section 2.3), an enhanced ridge flow mask can be used to prevent inpainting within fingerprint regions.

The main disadvantage of the algorithm is that blurring tends to occur when the width of the line exceeds 15-20 pixels. However, given the nature of the lines in fingerprint images (usually extracted from forms and on the order of 10-15 pixels in width at 500 ppi), this is not a significant concern, especially taking into account the speed of the algorithm.

Clearly, detection of lines is critical, and significant experimentation with the various parameters was done to determine optimal combinations of parameters for useful operation in the context of fingerprint images. It is also necessary to provide a simple design that would enable a user to tailor the line detection to a particular application using intuitive controls without having to understand all the details of the internal algorithm. To this end, a solution implementation in the form of a standalone library function accessible through Aware's SequenceCheck SDK was developed to facilitate flexible use of the line removal capability.

The function takes several parameters to enable a user to tailor the algorithm's behavior to their particular application. The primary user-set parameters are:

1. **minimum line length in inches –** this sets the lower limit on the length of the line that is removed
2. **angle deviation in degrees –** a value of zero enables removal of all lines regardless of angle; a value greater than zero indicates that only lines that deviate less than the specified number of degrees from being horizontal or vertical are removed
3. **a boolean flag -** set to TRUE if a ridge flow mask is to be used to preserve fingerprint regions, or FALSE if line removal should be done with no restrictions
4. **an aggressiveness factor –** a value that ranges from 0-100 specifying the aggressiveness of the line removal

In the context of fingerprint images, it is unlikely that line removal would ever be done without the use of a ridge flow mask. The reason for the existence of the Boolean is to enable this function's more general use, particularly in the context of OCR images. This function was found to be useful in cleaning up OCR images as a preprocessing step for some of the OCR engines tested in an OCR trade study conducted by Aware.

Two examples of images showing the line removal process are depicted in Figure 2 through Figure 5. Figure 2 and Figure 3 show the effect of the aggressiveness parameter on a roll image extracted from a relatively poor form image. Figure 4 and Figure 5 show the effect of the aggressiveness parameter on dual thumbs extracted from a better quality form. For the sake of the examples, no other constraints were used (i.e. minimum line length was set to zero and all angles of lines were allowed).

There are very minor differences at an aggressiveness factor of zero, with or without the ridge flow mask. Note the high effectiveness of the line removal on the better quality image even at this low level. Differences begin to appear at an aggressiveness factor of 50, but are mitigated quite well by the enhanced ridge flow mask. Recall that the "enhanced" mask is the original ridge flow mask processed to maintain closed boundaries for ridge flow regions. Not surprisingly, there is significant degradation of ridge flow at the highest aggressiveness factor, resulting in severe distortion without the ridge flow mask (as expected). Note the effectiveness of the ridge flow mask in preserving ridge flow areas in Figure 2 allowing the high aggressiveness factor to remove even the pencil marking in the top right corner of the image.

Regarding other parameters, increasing minimum line length can be effective in ensuring that small ridge lines missed by the ridge flow mask are not distorted. In the context of OCR, this filter is extremely useful in preserving characters yet allowing the line removal process to remove longer non-OCR related lines. Similarly, constraints on the angle parameter can also help to minimize fingerprint ridge flow degradation in instances where a ridge flow mask may not be entirely complete.

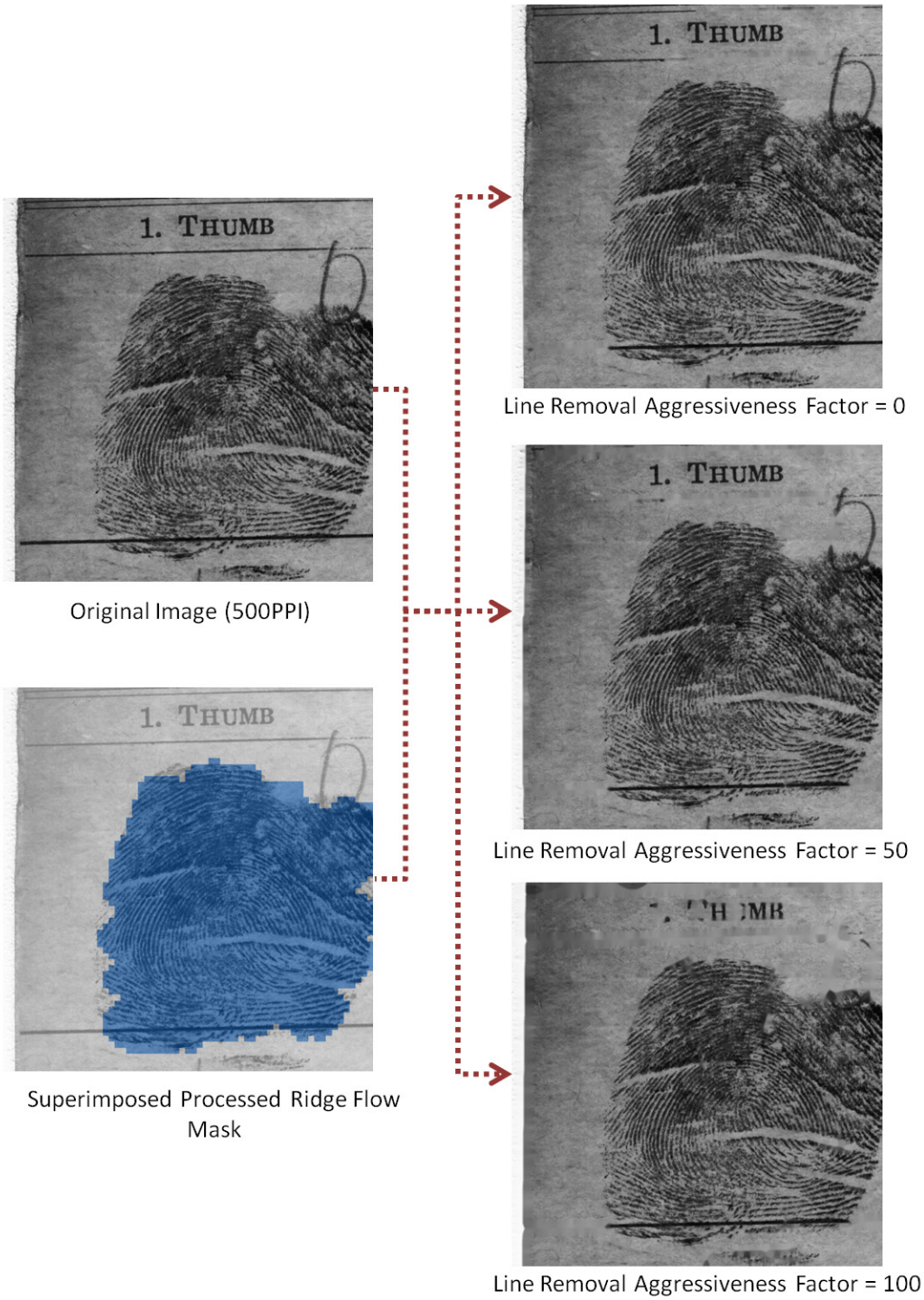

Figure 2 – Line removal mitigated by enhanced ridge flow mask

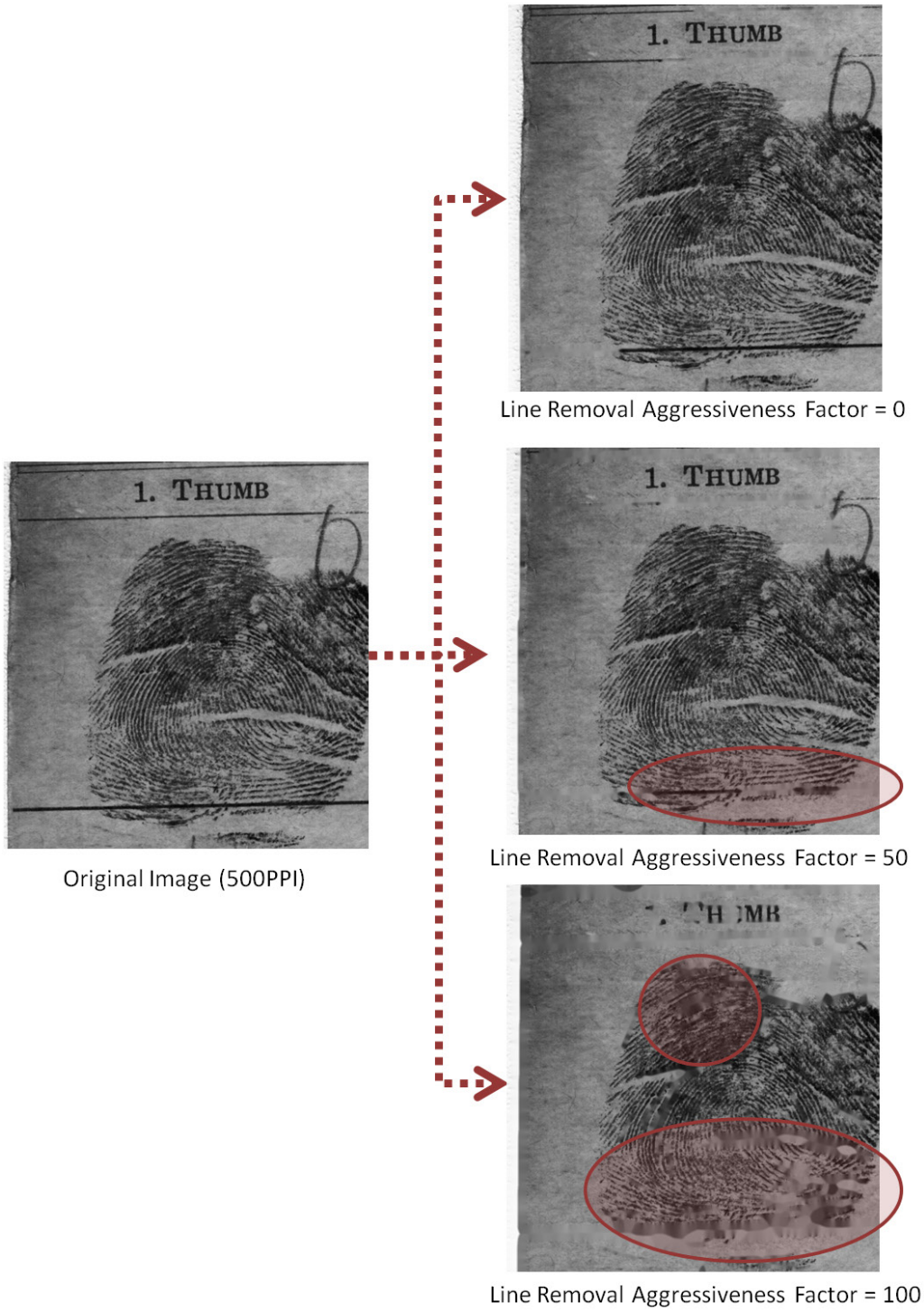

Figure 3 – Unrestricted line removal. Circled areas show ridge destruction due to line removal inpainting algorithm.

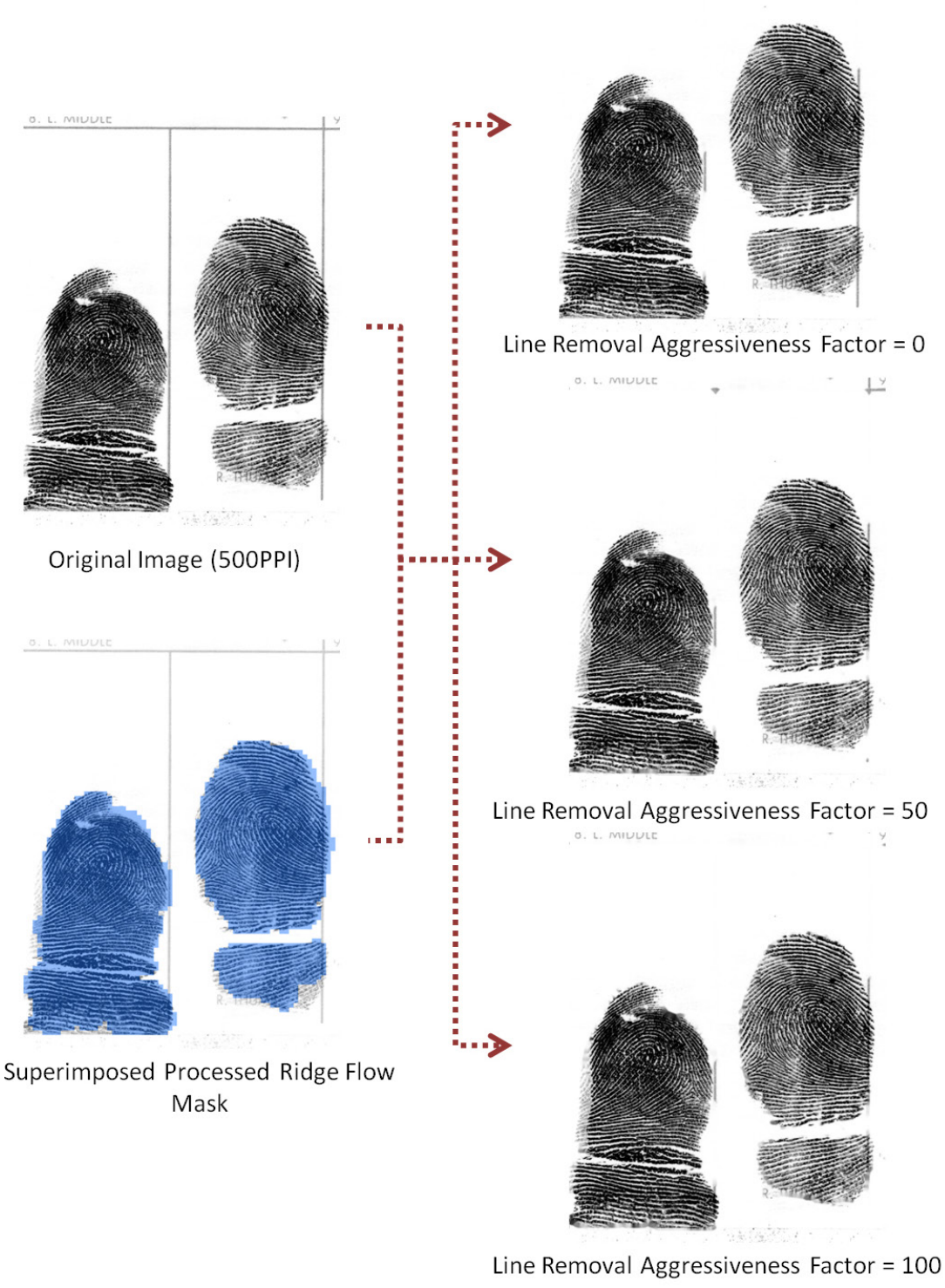

Figure 4 – Line removal mitigated by enhanced ridge flow mask

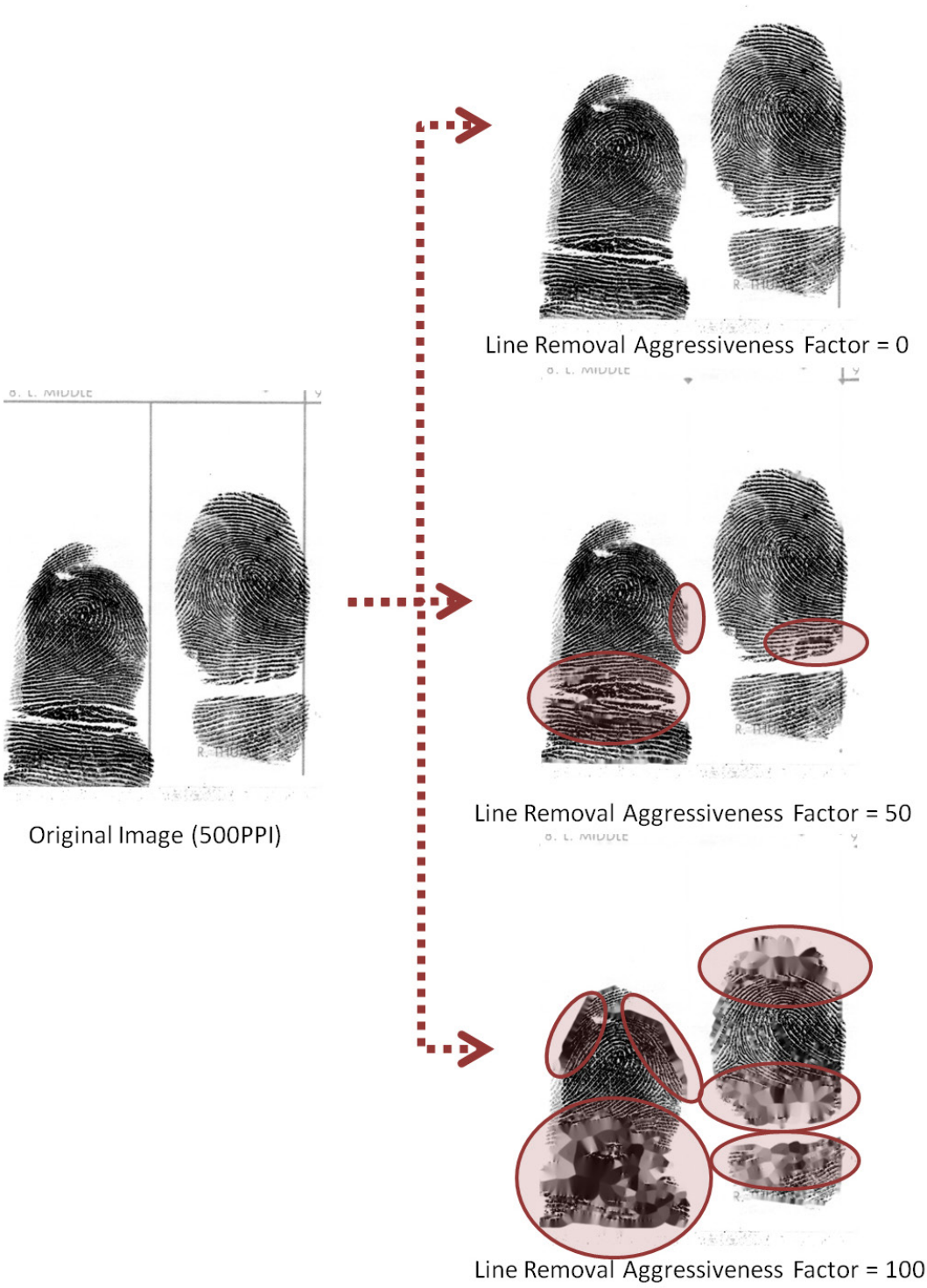

Figure 5 – Unrestricted line removal. Circled areas show ridge destruction due to line removal inpainting algorithm.

Form Recognition

Solution Description

In the context of fingerprint card scanning, it has been observed that a relatively low number of failure modes cause the vast majority of addressable low-quality data. These failure modes include: a) different card types in the same batch, b) mis-oriented cards and c) flipped cards. In addition, there is a need to provide better alignment estimation for fingerprint data extraction and OCR analysis of textual information. An automatic form recognition capability was identified as being valuable to advanced card scanning applications to enable mixed card scanning and at the same time, prevent incorrect scans by improving quality control and improving overall form alignment.

The automatic form recognition capability developed during the course of this work provides a solution that makes significant gains in solving these issues. The ability to automatically recognize fingerprint forms has several very important benefits for advanced fingerprint card scanning:

1. Forms no longer need to be pre-sorted, as form recognition is capable of recognizing forms that have been previously trained
2. Form orientation and form side (front or back) is no longer a concern as the form recognition algorithm automatically determines the orientation and side
3. Crop box extraction is significantly more accurate and consistent for fingerprint data providing cleaner images for extraction, which inevitably results in better segmentation and ultimately better matching.
4. Crop box alignment is accurate enough to enable text box extraction which can more adequately support OCR for semi-automatic form completion.

Form recognition capability for Aware FormScannerSE/MB suite of applications was developed under the following stipulated assumptions:

1. A minimum of three of each form type to be recognized must be available to the user for training.
2. Forms used for training are not necessarily “clean”, i.e. they may (and typically do) contain fingerprint and textual information.
3. With respect to quality, forms used for training must be (somewhat) representative of the population of forms to be recognized.
4. Forms used for training must be guaranteed to be in a known, consistent orientation and view (front or back)

The solution implemented involved developing algorithms for several key components:

1. Form signature extraction and training
2. Form classification
3. Determination of Form Offsets

Note that form orientation detection is achieved as a byproduct of signature matching and classification. The next several pages describe the implemented algorithms and present some performance metrics that evaluate the efficacy of the solution.

Form Signature Extraction and Training

Due to the large diversity of form type and image quality, a robust algorithm was developed to extract the salient information from forms to enable signature matching. Typical solutions to pattern classification select a set of relevant features that can be consistently extracted from each pattern and then generate a signature representation of each class which is capable of differentiating between them. Typical brute force approaches can be powerful, but are rather indiscriminate in their selection of features if no a priori information about the patterns is taken into consideration and if the only goal is statistical separation of all classes.

In the case of mis-oriented (square) forms, form patterns have a very specific relationship to the oriented form: all relevant feature points are rotated some multiple of 90 degrees. For rectangular forms, this relationship is even more constrained, as all relevant points are only rotated some multiple of 180 degrees. For this reason, a very specific set of features was selected that could take advantage of this a priori information, to both simplify the task of form recognition and at the same time, simplify orientation detection and alignment as well.

An example set of fingerprint forms is shown in Figure 6. The general idea for the algorithm is shown schematically in Figure 7. The original form image is scaled from its original size to 50 ppi. This resolution, in conjunction with a specific kernel was determined to be optimal based on significant empirical testing much later in development. The resulting binary image is then dilated in two different ways:

1. using a vertical structuring element to obtain vertical lines in the image, and
2. using a horizontal structuring element to obtain horizontal lines in the image

A radon transform integrates the resulting line data in two different directions to obtain one signature for the horizontal and one signature for the vertical line projections. The result of the transform is shown as a two dimensional image to highlight the bar-code-type signature corresponding to each projection.

During training, this process is repeated for each form in the training set and the results fused to obtain a "prototype" for each projection. The final form signature is a concatenation of the two signatures.

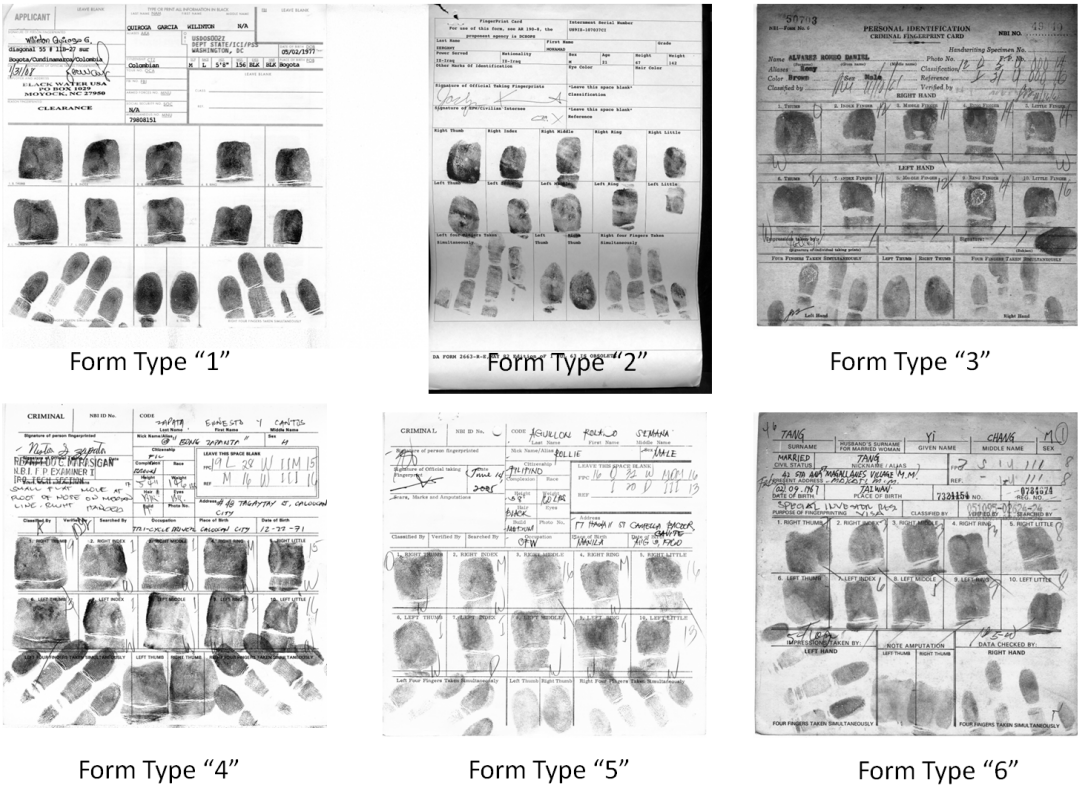

Figure 6 – Examples form images.

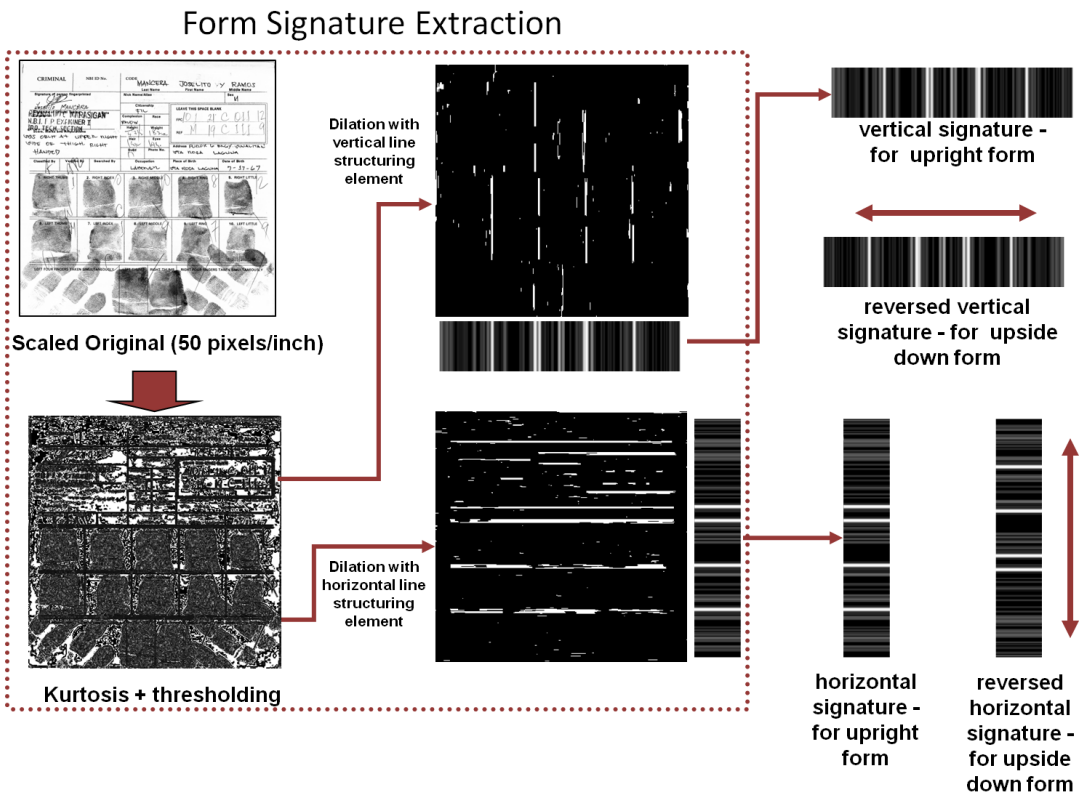

Figure 7 – Schematic showing the algorithm for form signature extraction.

Form Classification

The vector representation of the prototype strongly suggested a correlation classifier as the classifier of choice. The design of the implemented classifier is quite simple; the prototype vector for each class is simply correlated with any unknown signature. The result yielding the maximum correlation is chosen as the "winner", identifying the form type. Several modifications to this algorithm were made to make it more robust and more flexible for general form recognition:

1. Upright prototype signatures are also correlated with reversed prototype signatures to identify forms that are upside down – the result is simultaneous form identification and form orientation detection for upside down forms.

2. In the case of square forms, horizontal/vertical signatures are correlated with vertical/horizontal signatures respectively to identify forms rotated 90 degrees - the result is simultaneous form identification and form orientation detection for 90 degree rotated forms.

3. In the case of square forms, horizontal/vertical signatures are correlated with reversed vertical/horizontal signatures respectively to identify forms rotated 270 degrees - the result is simultaneous form identification and form orientation detection for 270 degree rotated forms.

4. In cases where the back side of forms is also available (as in ADF environments) the same logic is applied to both front and back forms and the correlation results fused to provide enhanced form type, orientation and side detection as well.

5. Once a prototype is trained, a second pass is made correlating the prototype with all of the other forms in the training set to compute a mean and standard deviation for the training set. This is used in production to compute a confidence in the classification result.

The implemented method proved to be quite effective for both form classification and form orientation detection.

Results

Six form types were tested; examples of each form type are shown in Figure 6. Four forms were square, equivalent in dimension, while the last two were each rectangular and different in dimension. A total of ten forms per class were selected randomly from each form type and each trained to obtain the prototype for its class. Fifteen additional forms (not from the training set) were randomly selected to from each form type to form a test set consisting of 6*15 = 90 forms. In addition, additional forms were generated by rotating each form 90, 180 and 270 degrees, resulting in a total of 360 forms.

Counts of all input and predicted form types and orientations are shown in the confusion matrix in figure 8. Only one instance of form Type 4 was mistaken for form Type 5. However, this resulted in four errors because the same image was tested in four different orientations, each time confused as a differently oriented form type 5. Note that in the case of the uniquely dimensioned forms, form types can be easily determined by that feature. Consequently, the only confusion possible for form Types 1 and 2 are whether or not they are upside down. Nevertheless, results for the other four square forms clearly show remarkable robustness with respect to the form recognition algorithm, despite significant form similarities and significant degradation of the original form.

Determination of Form Offset – Coarse Alignment

Another major benefit of using a correlation classifier is the automatic determination of form alignment offset from the correlation computation. This can be explained by noting that if two signature vectors are similar (i.e. the form signature "matches" the prototype signature) the correlation will be maximized when the two vectors are perfectly aligned.

It is obvious that all forms could potentially have different offsets relative to each other and relative to the prototypes, since forms are physically placed on the platen (true whether this is done manually or by machine via an automatic document feeder). There is no way of knowing beforehand what the alignment between an input form and any prototype will be. Consequently, the optimal algorithm will shift the signatures relative

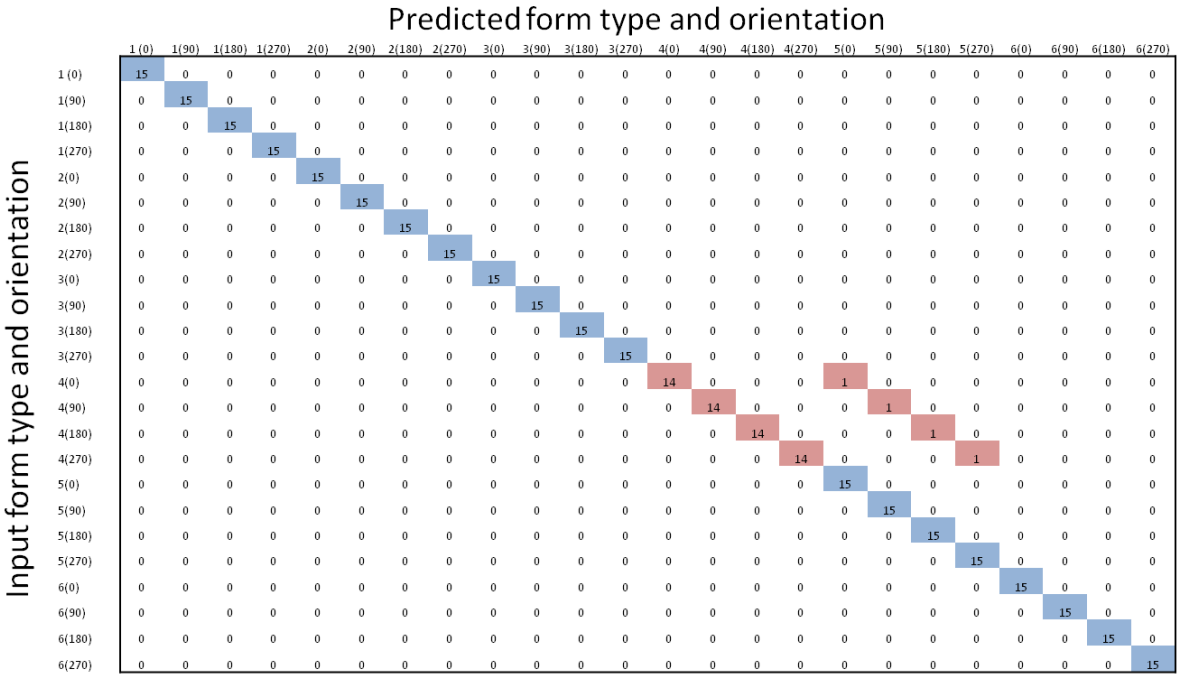

Figure 8 - Confusion matrix showing the performance of form recognition and orientation detection. The syntax is: F(A) where F is the form type and A is the orientation of the form in degrees. A total of 6 form types, 15 images and 4 orientations each for a total of 360 images were tested. Note that only one instance of form type 4 was mistaken for form type 5, resulting in 4 errors because the same image was tested in 4 different orientations, each time confused for form type 5.

to each other and use the maximum correlation as the final correlation result. This indirectly determines the offset of the form relative to the prototype.

An additional complexity is that the final prototype signature for each form type is a fusion of prototype signatures across the training set, whose offset relative to the crop template must be determined. Recall that the crop template in FormScanner+ is the set of crop boxes "drawn" by the user to tell the software where regions of interest exist in the form image. During this process, a form image of the particular type is loaded so that a user has an example image as a reference for crop box creation. The signature for this "reference image" is also now correlated with the stored prototype to determine the offset of the reference image relative to the prototype. When an unknown form is then matched to a prototype, the reference image offset is added to obtain the final offset for the unknown image. This guarantees that crop boxes drawn in the reference image are properly aligned in the unknown form image.

Determination of Form Offset – Fine Alignment

Although quite effective, coarse alignment using the offset obtained from the correlation classifier has a limiting average accuracy of approximately 10-30 pixels for a 500 ppi image. This is a consequence of the 50 ppi analysis resolution used for form recognition and inherent inaccuracies due to image analysis. As a result, an additional fine refinement process was developed to refine the form alignment even further.

During crop box template creation, images within crop boxes are extracted and processed using a line detection algorithm developed for line removal (see Section 2.2). Figure 9 shows an example of the lines detected in a fingerprint crop box region. These lines are then stored along with the prototype signature to be used later during form recognition.

During form recognition, and after coarse alignment of an input form, these same crop box locations are used to extract images from the form. Lines from the prototype are then "redrawn" into a synthetic image, as

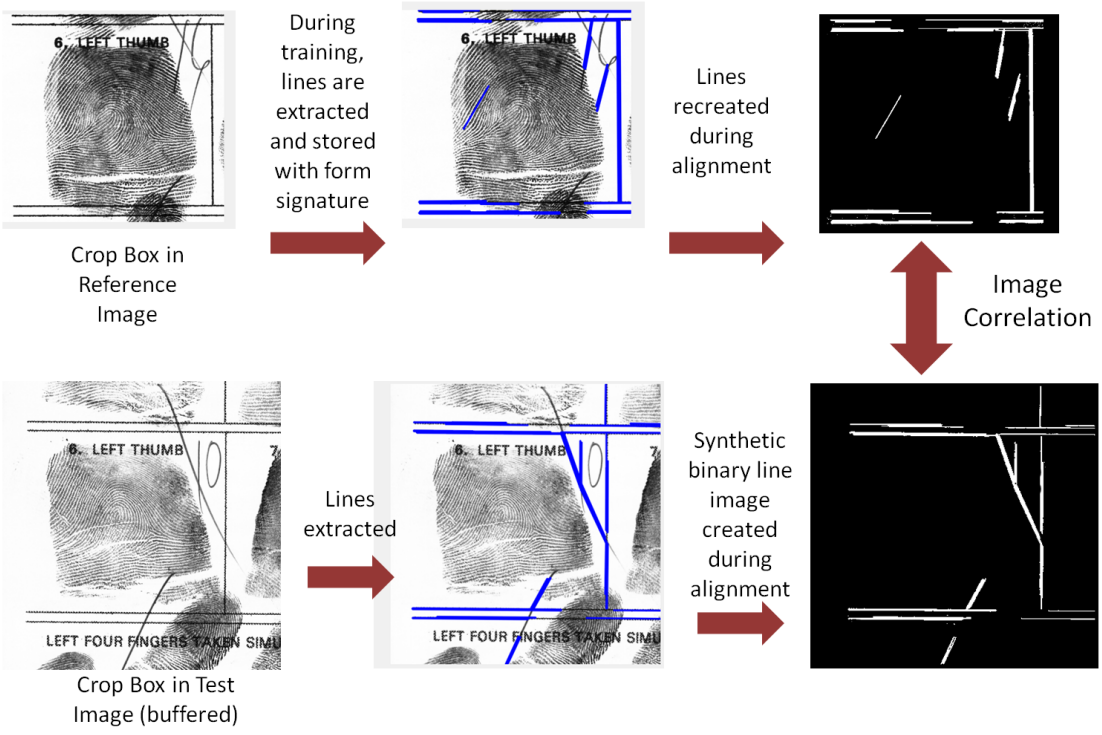

Figure 9 – During alignment, the stored reference crop box lines are used to generate a synthetic binary line image for image correlation with a synthetic binary line image from the test image.

are the lines extracted from the input image (see Figure 9). The two regions are then correlated in the Fourier domain (to save time due to the much larger image correlation) and a fine offset determined from the location for the maximum correlation.

Results from experiments show a significant improvement in form alignment, as shown in figures 10 and 11. Separate plots for horizontal and vertical offsets are presented to simplify the data presentation. The same fifteen forms (for each form type) used in the form recognition and orientation test are "ground truthed" to determine their offsets relative to a reference form. This ground truth is then compared to actual offsets computed during the form recognition process. Mean errors and standard deviations are computed for each of the six forms and plotted using 95% confidence intervals.

The data clearly shows significant differences at greater than 95% confidence for all forms except form Type 4 but only in the case of the vertical offset. The results also show the significant benefit of the fine alignment stage is that in some cases (e.g. forms 2, 3 and 6) dramatically improves alignment over the coarse alignment obtained from form recognition alone.

Future Work

The confidence measure obtained during correlation classification can also be used as a measure to determine outliers, towards a method for automatic form learning. For example, if the computed confidence for a "winning" class is still extremely low, this may indicate that the form is similar to the "winning" class, but still different, perhaps different enough to constitute its own separate class. This could be either used in production to flag potential new forms, or to simplify the training process even further by not requiring an operator to even group similar form types during training, counting on the software to automatically "grow" the class list dynamically.

The single error shown in the confusion matrix could also have been averted if the confidence of the form recognition had been included in the decision (instead of relying on a fusion of the correlation scores). Although the correlation score for the error was greater for the winner, its correlation score was actually an outlier with respect to its own distribution, while the second highest correlation (the true match) matched its class with a reasonably high confidence. Incorporating this information in the future should result in even better performance.

Regarding the very high alignment error for form Type 4, after some considerable investigation, it was determined that the reason for the misalignment was sub-optimal line detection, resulting in extremely poor fine alignment. In the future, a check will be made to insure adequate detection of lines prior to fine form alignment, with dynamic modification of line detection parameters to optimize the line extraction.

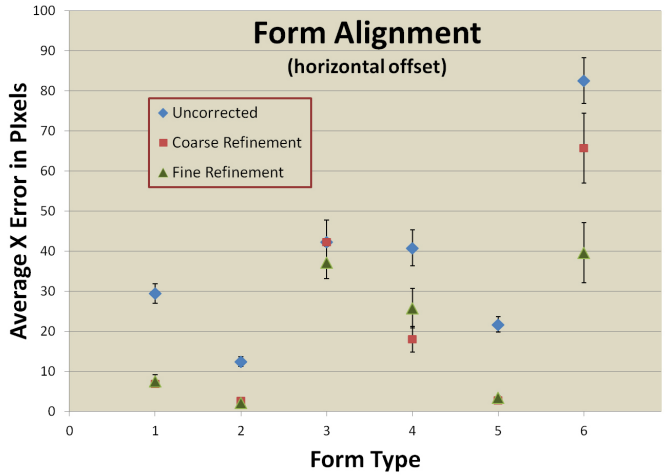

Figure 10 – Mean form X offset errors are shown for 6 form types, for the uncorrected form, coarse refinement determined directly from form recognition, and fine refinement. All confidence intervals are 95% obtained from 15 forms tested for each type.

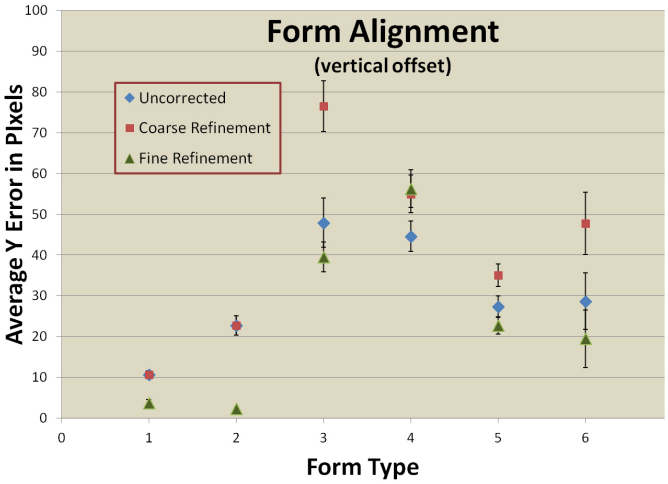

Figure 11 – Mean form Y offset errors are shown for 6 form types, for the uncorrected form, coarse refinement determined directly from form recognition, and fine refinement. All confidence intervals are 95% obtained from 15 forms tested for each type.

Summary of Product Enhancements

Form Recognition

AccuScan - The AccuScan SDK has been enhanced with form card recognition capabilities. Form rotation errors (90, 180,270 degrees) can be automatically detected. Form recognition now allows automatic selection of the appropriate cropping template.

FormScannerMB - Because it is based on AccuScan, this application can now make use of the new form recognition capabilities added to AccuScan. A training tab has been added to the application. Training includes defining Scan Types. A Scan Type defines the area to scan, number of card sides to scan, and the resolution (500ppi/1000ppi) for the scan. Each Scan Type also includes a list of Card Types that will be recognized. Definition of a Card Type involves scanning multiple copies of the card, and results in creation of a template for use in recognizing the form.

A Scan Type needs to be selected whenever scanning a batch of cards. As the cards are scanned, each card is checked against the Card Type templates for that Scan Type to determine the card format being scanned. Cropping of fingerprint images is then done based on the Card Type found.

OCR Software Integration

FormScannerSE - An OCR abstraction library was created to support several OCR libraries, including ABBY and Tesseract. This OCR library has been integrated into the FormScannerSE application and can be used during data entry to recognize text from pre-configured areas of the card.

Compliance with ANSI/NIST-ITL 1-2011

FormScannerMB - FormScannerMB includes support for storing whole card images into transactions. A configuration option was added to FormScannerMB to allow selection of either Type 16 or Type 20 records to store the whole card images.

The verification file shipped with FormScannerMB had to be modified to include support for Type 20 records.

Support for Windows 7 64-bit

AccuScanMB - The AccuScan SDK library was modified to support running on Windows 7 64-bit.

A 64-bit version of the Aware ScannerReadiness utility was created and added to the AccuScan SDK. This utility uses a commercial test target to check that a scanner is still in calibration.

FormScannerMB - The application now runs on Windows 7 64 bit.

Q/A and Rework

FormScannerSE

The form training process in FormScanner SE has been modified to allow indication of text areas used for OCR. Information for these text areas is stored along with the crop information in the template for the card format.

FormScannerMB - FormScannerMB makes use of the OCR regions set in the card format templates to determine which text areas to apply OCR. Visual feedback of confidence of the OCR result is conveyed to the user through the use of color codes. The user has the option to override the results of the OCR conversion.

Epson 11000XL Support

AccuScan - Aware has modified the AccuScan SDK to enable the 11000XL with Appendix F certified scanning.

FormScannerMB - Support for the 11000XL scanner is now included in the application.

FormScannerSE

Support for the 11000XL scanner is now included in the application.

Manual Batch Scanning Support

FormScannerMB

Because the Epson 11000XL scanner does not support an ADF unit, a new manual batch scanning mode was required in the FormScannerMB application. This mode is intended to aid an operator in rapidly scanning a batch of cards by manually placing each card within the batch onto the scanner. Operation of FormScannerMB is otherwise the same as with a scanner that supports an ADF unit.

Aware Fingerprint Card Scanning Software Product Overview

FormScannerMB

This application is designed for batch scanning of fingerprint card sets. It makes use of Aware AccuScan to control the ADF units of certain scanners. It collects several quality metrics when scanning a card set including presence of fingers, NFIQ and AFIQ quality and sequencing errors. It creates ANSI/NIST based transactions (including support for DOD EBTS transactions. These transactions include cropped out fingerprint images, whole card images, and very limited biographic data support (which store default field values based on configuration). FormScannerMB utilizes the AccuScan SDK, and is therefore FBI IQS certified.

FormScannerSE

This application is designed to interactively support creation and editing of individual ANSI/NIST based transactions. This includes support for DOD EBTS transactions. When used independently of FormScannerSE, it supports all steps needed to create a transaction from a fingerprint card including scanning. When used with FormScannerMB, its primary task is for manual entry of biographic data into transactions scanned with FormScannerMB. FormScannerSE utilizes the AccuScan SDK and is therefore FBI IQS certified.

AccuScanMB SDK

This SDK provides batch FBI Appendix F scanning for a limited number of flatbed scanners with ADF units. Appendix F scanning can be done at both 500ppi and 1000ppi. It can support multiple scanners attached to the same computer.

AccuScan SDK

This SDK provides FBI Appendix F scanning using many different models of flatbed scanner. Appendix F scanning can be done at both 500ppi and 1000ppi depending on the capabilities of the scanner. Unlike AccuScan, it does not support batch scanning using an ADF unit.

SequenceCheck SDK

This SDK provides functionality for working with fingerprint images including rolls, single plain fingers, four finger slap images and palm images. Functionality includes segmentation (finding fingers within an image), sequencing (checking for duplicate fingers within a set) and quality metrics.

A new API function, AwSeqRemoveLines () has been added to the SequenceCheck SDK library that can remove lines from non-fingerprint areas of an image. Lines within the fingerprint part of the image are left as is. This function includes an aggressiveness parameter, minimum line length parameter, and an angle deviation parameter to enable its more general use in both OCR extraction and fingerprint line removal. This function is described in more detail in the section below on Advanced Algorithm Development.

A second API function, AwSeqRemoveGhostImages () has been added to the SequenceCheck SDK library for ghost image suppression. This function lightens areas surrounding fingerprints that contain residual fingerprint and/or darkening artifacts generated during scanning. This function includes an aggressiveness parameter and an image class code for artifact removal. This function is described in more detail in the section below on Advanced Algorithm Development.

About Aware, Inc.

Aware is a leading provider of biometrics software products and development services to government departments, system integrators, and solution suppliers globally. Our products include SDKs, software components, workstation applications, and a modular, centralized, service-oriented platform. They fulfill a broad range of functions critical to biometric authentication and search using fingerprints, face, and iris, including sample autocapture, image quality assurance, abstraction of capture hardware peripherals, centralized data processing and workflow, subsystem connectivity, and biometric matching algorithms. The products are used to enable identity-centric security solutions with biometrics for applications including law enforcement, border management, credentialing and access control, and defense and intelligence. Aware is a publicly held company (NASDAQ: AWRE) based in Bedford, Massachusetts.

AWARE

Please contact Aware or visit our website for additional information:
sales@aware.com | www.aware.com