

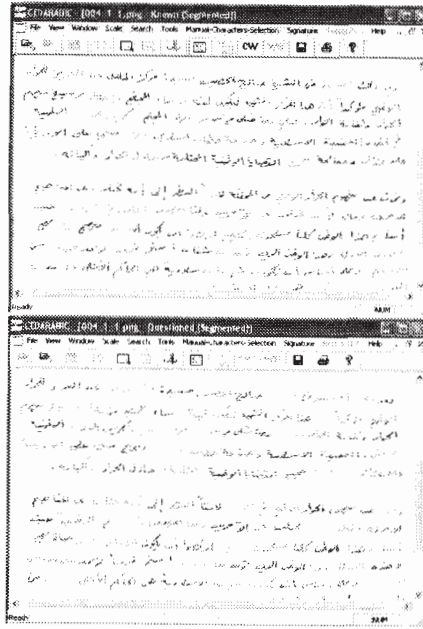


Figure 10: Example of a poorly segmented Arabic document by automatic English segmenter. Truth is on the right and the result of automatic segmentation is on the left. In this case 45% of the segmented words are correct.

5 Word Shape Matching

The word segmentation is an indexing step before using the word shape matching for word retrieval. A two-step approach is employed in performing the search: (1) prototype selection: the query (English text) is used to obtain a set of handwritten samples of that word from a known set of writers (these are the prototypes), and (2) word matching: the prototypes are used to spot each occurrence of those words in the indexed document database. A ranking is performed on the entire set of test word images—where the ranking criterion is the mean similarity score between prototype words and the candidate word based on global word shape features.

5.1 Prototype Selection

Prototypes which are handwritten samples of a word are obtained from an indexed (segmented) set of documents. These indexed documents contain the truth (English equivalent) for every word image. Synonymous words if present in the truth are also used to obtain the prototypes. Hence queries such as “country” will result in selecting prototypes that have been truthed as “country” or “nation” etc... A dynamic programming Edit Distance algorithm is used to match the query text with the indexed word image’s truth. Those with distance as zero are automatically selected as prototypes. Others can be selected manually.

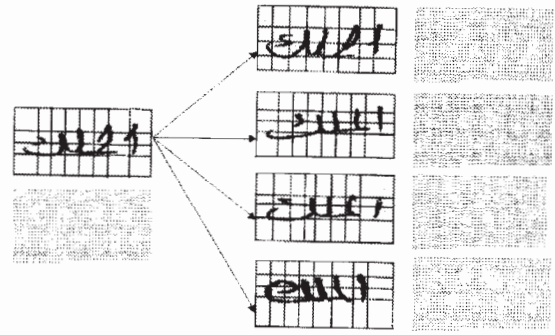


Figure 11: One query word image (left) matched against 4 selected prototype images. The 1024 binary GSC features are shown next to the images.

5.2 Word Matching

The word matching algorithm uses a set of 1024 binary features for the word images. These binary features are compared using the correlation similarity measure 5.2.1 to obtain a similarity value between 0 and 1. This similarity score represents the extent of match between two word images. The smaller the score, the better is the match. For word spotting, every word image in the test set of documents are compared with every selected prototype and a distribution of similarity values is obtained. The distribution of similarity values is replaced by its arithmetic mean. Now every word is sorted in rank in accordance with this final mean score. Figure 5.2 shows an example query word image compared with the a set of 4 selected prototypes.

5.2.1 Similarity measure

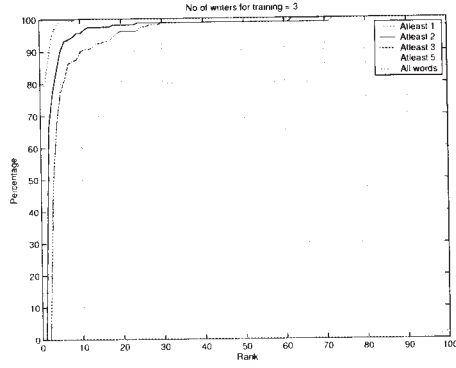
The method of measuring the similarity or distance between two binary vectors is essential. The correlation distance performed best for GSC binary features [10] which is defined for two binary vectors X and Y , as in equation 5.2.1

$$d(X, Y) = \frac{1}{2} \left(1 - \frac{s_{11}s_{00} - s_{10}s_{01}}{[(s_{10} + s_{11})(s_{01} + s_{00})(s_{11} + s_{01})(s_{00} + s_{10})]^{\frac{1}{2}}} \right)$$

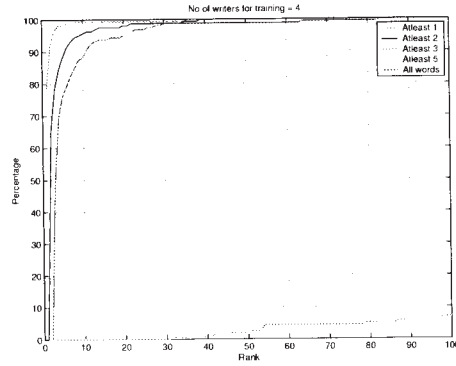
where s_{ij} represent the number of corresponding bits of X and Y that have values i and j .

5.3 Word-Matching Performance

The performance of the word spotter was evaluated using manually segmented Arabic documents. All experiments and results were averaged over 150 queries. For each query, a certain set of documents are used as training to provide for the template word images and rest of the documents were used for testing. Figure 12 shows the percentage of times when correct word images were retrieved within the top ranks scaled on the x-axis. More than one correct word image does exist in the test set. For example, if the word “king” occurs in document 1 twice, and we use writers 1 through 6 for training, then



(a) Recall with 3 writers for training.

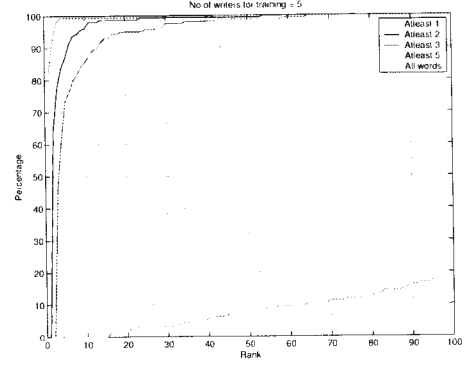


(b) Recall with 4 writers for training.

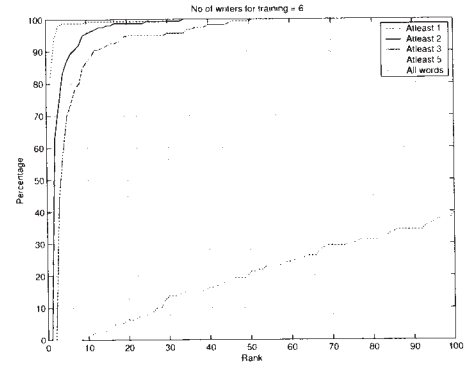
Figure 12: Retrieval performance of the system. All results were averaged over 150 queries. The styles of 3 and 5 writers were used for training (by using their documents) to provide template word images. The rest were used for testing. The different curves report on the percentage of times the word images were retrieved within the ranks scaled on the x-axis.

we have $6 * 2 = 12$ template word images to perform word spotting. The remaining test set contains $(10 - 6) * 2 = 8$ correct matches in the test set. The top curve of the figure shows that at least one correct word is retrieved 90% of the time within the top 12 ranks. The other curves show the same when atleast 2 and 3 correct words are retrieved. Similar plots can be obtained when different numbers of writers are used for training. These plots are shown in Figs. 12 13 and 14 where three, five and seven writers were used for training.

Performance of word spotting can be specified in terms of precision and recall. If n_1 is the number of relevant words in the database, n_2 is the number of words retrieved, and n_3 is number of relevant words among those that are retrieved then $Precision = n_3/n_2$ and $Recall = n_3/n_1$. Precision-Recall curves corresponding to using three, five and

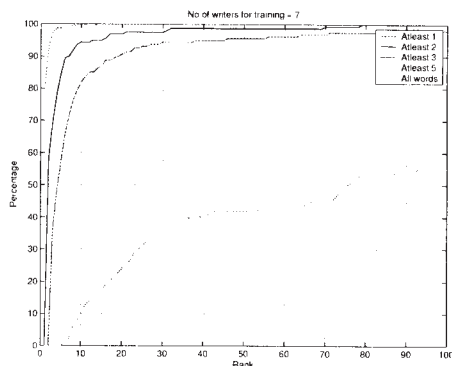


(a) Recall with 5 writers for training.

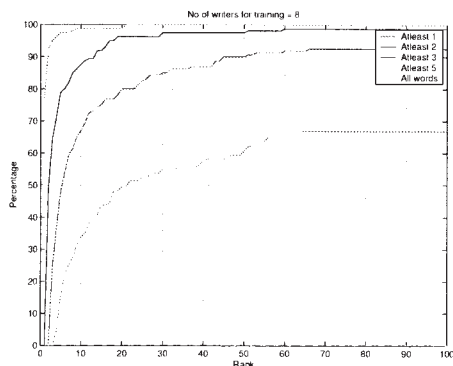


(b) Recall with 6 writers for training.

Figure 13: Retrieval performance of the system. All results were averaged over 150 queries. The styles of 5 and 6 writers were used for training (by using their documents) to provide template word images. The rest were used for testing. The different curves report on the percentage of times the word images were retrieved within the ranks scaled on the x-axis.

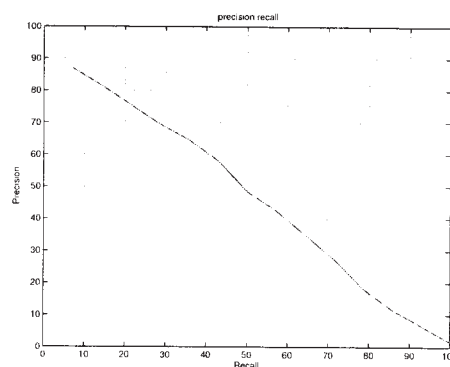


(a) Recall with 7 writers for training.

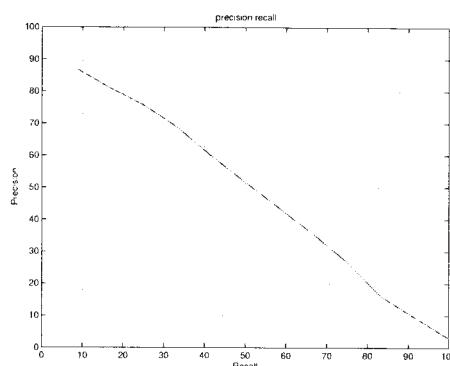


(b) Recall with 8 writers for training.

Figure 14: Retrieval performance of the system. All results were averaged over 150 queries. The styles of 7 and 8 writers were used for training (by using their documents) to provide template word images. The rest were used for testing. The different curves report on the percentage of times the word images were retrieved within the ranks scaled on the x-axis. Results show improved performance over the case when five styles were used as shown in Figure 12 and 13.



(a) Precision Recall with 3 writers for training.



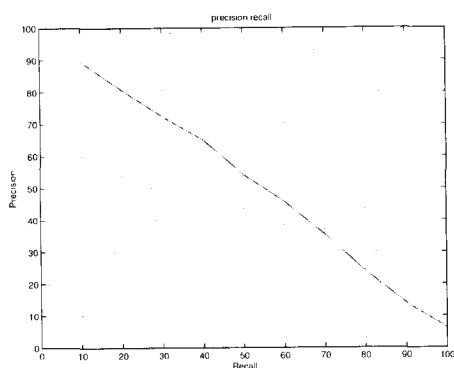
(b) Precision Recall with 4 writers for training.

Figure 15: Precision-Recall curves where the styles of 3 and 4 writers were used for training (by using their documents) to provide template word images.

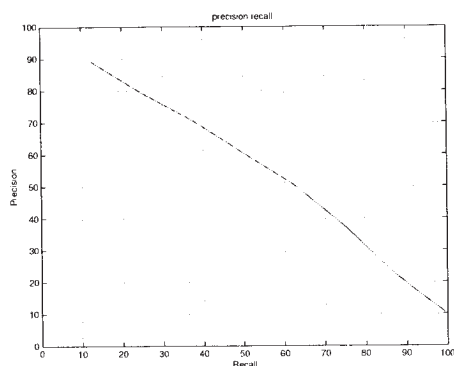
seven writer styles are shown in Figures 15 , 16 and 17 respectively. Finally figure 18 compares the precision at a particular recall with varying number of writers used for training. As the number of writers is increased, the precision at a given recall increases.

6 Conclusion

A technique for Arabic word image spotting in previously segmented images and that of automatic Arabic word segmentation was discussed. A two-step approach involving (i) Prototype selection and (ii) Word matching were described. Performance of the system is very promising with increased recall and better precision as the number of writers in the training set is increased. A precision of 70% is achieved at a recall of 50% when 8 writers were used for training. The CEDARABIC system described as the user interface provides tools for (i) manual word segmentation, (ii) information look-up on truth of every word, and (iii) interface to perform word spotting. A set

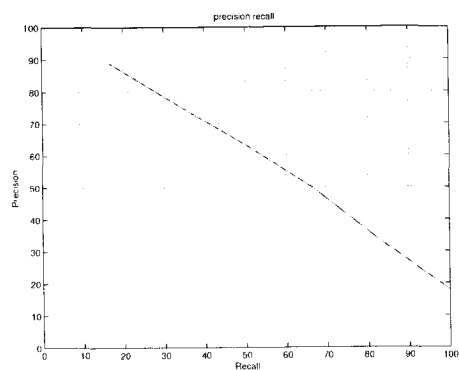


(a) Precision Recall with 5 writers for training.

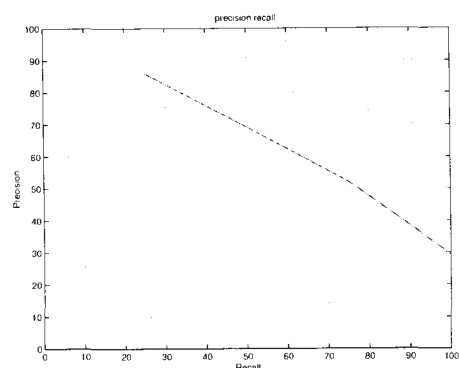


(b) Precision Recall with 6 writers for training.

Figure 16: Precision-Recall curves where the styles of 5 and 6 writers were used for training (by using their documents) to provide template word images.



(a) Precision Recall with 7 writers for training.



(b) Precision Recall with 8 writers for training.

Figure 17: Precision-Recall curves where the styles of 7 and 8 writers were used for training (by using their documents) to provide template word images. Results show an improvement over the case when five styles were used as shown in Figure 15 and 15.

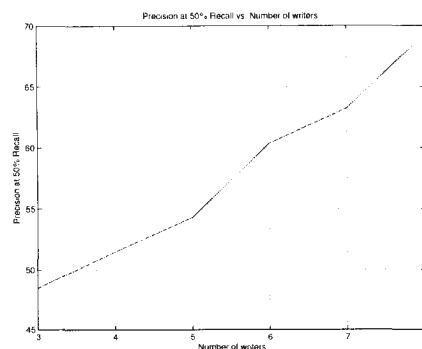


Figure 18: Precision at 50% recall is plotted against different numbers of writers used for training. As more number of writers are used for training, the precision at a given recall increases.

of more complicated features tuned for Arabic will result in better automatic word segmentation performance and it will be a part of future research.

7 Acknowledgment

The authors wish to thank Chen Huang, Vivek Shah, Manivannan Arivazhagan and Thanh Nguyen for their assistance.

References

- [1] S. Kuo and O. Agazzi, "Keyword spotting in poorly printed documents using 2-d hidden markov models," in *IEEE Trans. Pattern Analysis and Machine Intelligence*, **16**, pp. 842–848, 1994.
- [2] M. Burl and P. Perona, "Using hierarchical shape models to spot keywords in cursive handwriting," in *IEEE-CS Conference on Computer Vision and Pattern Recognition*, **June 23–28**, pp. 535–540, 1998.
- [3] A. Kolz, J. Alspector, M. Augusteijn, R. Carlson, and G. V. Popescu, "A line-oriented approach to word spotting in handwritten documents," in *Pattern Analysis and Applications*, **2(3)**, pp. 153–168, 2000.
- [4] R. Manmatha and T. M. Rath, "Indexing of handwritten historical documents-recent progress," in *Symposium on Document Image Understanding Technology (SDIUT)*, pp. 77–85, 2003.
- [5] B. Zhang, S. N. Srihari, and C. Huang, "Word image retrieval using binary features," in *Document Recognition and Retrieval XI*, **SPIE Vol. 5296**, pp. 45–53, 2004.
- [6] S. N. Srihari, B. Zhang, C. Tomai, S. Lee, and Y.-C. Shin, "A system for handwriting matching and recognition," in *Symposium on Document Image Understanding Technology Greenbelt, MD*, April, 2003.
- [7] S. N. Srihari, S.-H. Cha, H. Arora, and S. Lee, "Individuality of handwriting," in *Journal of Forensic Sciences*, **47(4)**, pp. 856–872, 2002.
- [8] S. N. Srihari, B. Zhang, C. Tomai, S. Lee, Z. Shi, and Y. C. Shin, "A system for handwriting matching and recognition," in *Proceedings of the Symposium on Document Image Understanding Technology (SDIUT 03), Greenbelt, MD*, pp. 67–75, 2003.
- [9] G. Kim, V. Govindaraju, and S. N. Srihari, "A segmentation and recognition strategy for handwritten phrases," in *International Conference on Pattern Recognition, ICPR-13*, pp. 510–514, 1996.
- [10] B. Zhang and S. N. Srihari, "Binary vector dissimilarity measures for handwriting identification," *Proceedings of the SPIE, Document Recognition and Retrieval*, pp. 155–166, 2003.

Word spotting in Arabic script documents

A. Lawrence Spitz and Jim Yaghi

DocRec Ltd

34 Strathaven Place, Atawhai, Nelson 7001, New Zealand

spitz@docrec.com, jim@docrec.com

Abstract

Several applications of processing of document text based on character shape codes have been demonstrated for documents in English and other languages set in Roman script, but ability to deal with Arabic-script documents has not yet reached the same level of sophistication.

Arabic script does not naturally divide into characters in the image domain complicating the design of shape code sets. Many Arabic fonts are quite ornate adding complexity to the shape classification process and the Arabic language is agglutinative and highly inflected and thus requires a huge lexicon of surface forms.

These attributes combine to make reconstruction of Arabic considerably more complex than has been shown for Roman-script documents.

We are developing a technique for spotting the occurrences of particular words of interest in Arabic script document images.

1 Introduction

A character shape representation has been shown to support many applications such as text reconstruction[1], postal address recognition[2], duplicate detection[3], language identification[4] and topic identification[5] amongst others. Most of the work has been applied to Roman-script documents, though there have been reports of successful application in both Thai [6] and Japanese [7] documents.

In Section 2 we describe some of the attributes of Arabic text which must be taken into consideration in shape coding.

Section 3 describes the image processing necessary to perform Arabic Shape Coding. Section 4 describes the shape representation for Arabic script. Section 5 will deal with issues of representation ambiguity.

Section 6 will describe our word-spotting experiments and Section 7 discusses current status and future plans.

2 Arabic Document Image Attributes

2.1 Typography

Let us define a *grapheme* as being a connected sequence of characters within or comprising a word. The Arabic alphabet has 28 letters, 22 of which have four forms depending on whether the letter is in isolation, in the initial, medial or terminal position of a grapheme. The remaining 6 letters have only two forms, isolated and terminal. The sequence of characters within a word is connected until and unless one of these six letters is encountered, thereby ending the grapheme. Spaces between graphemes are smaller than spaces between words.

Arabic characters often contain ancillary marks such as hamza **أ** and nuqta **ق**. Some Arabic characters share skeletal forms and therefore are distinguished only by the number and position of these nuqta **ث ت ب**.

Justification in Arabic script is accomplished using *keshide*, or horizontal stretchiness of character images, obviating the need for hyphenation. Thus, words are never fragmented across text line boundaries.

2.2 Fonts

There is a wide variety of Arabic fonts in use ranging from simple to very ornate. Fortunately most body text is set in one of a few simple fonts.

2.3 Language and Lexicography

In agglutinating languages, words may be comprised of several morphemes, but are easily divided into the components that make them up. These components are usually one root and several affixes. Each affix typically has one meaning and represents

only one grammatical category. On the other hand, words in derivational languages do consist of several morphemes but their affixes represent several grammatical categories at the same time and the affixes are not easily detachable because they interdigitate with the root. Some languages have a high amount of agglutination or fusion, allowing for a large number of morpheme combinations. Latin languages are primarily fusional. Arabic and English are both fusional and agglutinating, although Arabic is more fusional.

Arabic words can take several forms depending on their position in the sentence. Verbs change form depending on (1) the subject's number (singular, dual, plural), gender (masculine, feminine), and person (first, second, third); (2) the verb's tense (past, present, imperative); (3) voice (passive, active); and (4) mode (indicative, subjunctive, jussive). The English equivalent of Arabic's indicative mode of the verb study is either study or studies as in 'they study', 'she studies', and the subjunctive mode equivalent is English's non-finite verb study as in 'I recommend that she study hard', etc. Whilst the English representation of the verb *study* only uses two forms, Arabic uses a large number of word forms, as can be seen below. The Arabic equivalent of the verb *study* takes a large array of orthographic representations depending on its syntactic context. This example shows the active voice, present tense verb inflected for different subjects. Other word forms can be generated from the remaining tense and voice combinations. A second pass of word form modification may take place when adding object pronominal suffixes, conjunction particles, and verb intensifiers. The latter modifications may be primarily agglutinating but they do add a second level of complexity to word forms.

Table 1: Tense: Present, Voice: Active for Arabic

Pronoun	Indicative	Subjunctive	Jussive
He	yadorusu	yadorusa	yadoruso
She	tadorusu	tadorusa	tadoruso
You (masc.)	tadorusu	tadorusa	tadoruso
You (fem.)	tadorusyna	tadorusy	tadorusy
I	>adorusu	>adorusa	>adoruso
They (dual, masc.)	yadorusAni	yadorusaA	yadorusaA
They (dual, fem.)	tadorusAni	tadorusaA	tadorusaA
You (dual)	tadorusAni	tadorusaA	tadorusaA
We	nadorusu	nadorusa	nadoruso
They(plural,masc.)	yadoruswna	yadoruswA	yadoruswA
They (plural, fem.)	yadorusona	yadorusona	yadorusona
You (plural, masc.)	tadoruswna	tadoruswA	tadoruswA
You (plural, fem.)	tadorusona	tadorusona	tadorusona

This variation in verb forms causes a single verb to have hundreds of orthographic representations. The same type of complexity applies to nouns, where each noun is marked for gender, number, definiteness, and grammatical case.

In addition, the fact that the language typically indicates the consonants, semi-vowels, and vowels of words and makes the marking of short vowels optional adds yet another level of difficulty for graphic recognition. Arabic text may have some, all, or none of the short vowel diacritics marked. In most cases, diacritics are omitted entirely and added only when the context may cause confusion in meaning. Fortunately, many of these diacritics are small marks that appear above or under a base grapheme; hence, they can be ignored completely. It might be wise to eliminate short vowel diacritics altogether in order to reduce the size of lexicon. Even then, the size of the lexicon is extraordinary.

It is these derivational properties of the morphology and the inflectional nature of the syntax which make it difficult to create an encompassing lexicon of the language. Using a small corpus is unlikely to facilitate finding examples of the majority of words in the majority of contexts where they occur.

3 Image processing

Most of the image processing is the same as for Roman text. See [9]. Text lines are processed from left to right but at the end of the line the shape code sequences are reversed.

3.1 Text-line fiducials

In shape coding Roman script, characters are classified by the spaces that they occupy: the ascender zone above the x-line, the zone between the baseline and the x-line, and the descender zone below the baseline.

Shape coding Arabic script requires different fiducials largely because the concept of x-height does not apply well to Arabic script. Nevertheless for want of better terms we have continued to use x-line to define the vertical position that delimits the ascender zone from the zone immediately adjacent to the baseline. Likewise x-height is the distance from the baseline to the x-line and x-zone is the zone between those two fiducials.

3.2 Quanta

In Arabic, joined character pairs¹ are the rule rather than the exception since in most cases non-terminal characters within a word are connected to the following character.

Thus we were presented with three options for segmenting Arabic text. In each case we generate a sequence of Arabic Shape Codes (ASCs) for each quantum encoded.

3.2.1 Grapheme segmentation

In this instance we would generate a shape code for each connected sequence of letters. There are an average of 3.6 and a maximum of 12 letters per grapheme in the lexicon. This requires a fairly large shape code set to represent the complexity contained in the grapheme, but is the easiest from the standpoint of image processing since all that needs to be distinguished before encoding is the difference between intra-word (grapheme) spacing and inter-word spacing.

3.2.2 Character segmentation

To encode actual characters one-to-one, accurate segmentation must be performed. All joining character pairs do so with a horizontal segment at the baseline, but unfortunately some characters have baseline-aligned horizontal segments within them. Thus character segmentation becomes tricky.

The benefit of true character coding is large however since the transformation from the image domain to the lexical (character) domain is trivial.

3.2.3 Over-segmentation

In this instance, drawing from the experience of those engaged in (Roman) handwriting recognition[8], we divide the image not only at grapheme spaces but at all instances of horizontal segments at the baseline. This will clearly isolate all characters but will also break some characters into multiple segments.

In order to find segments, we search a line from one end to the other delimiting graphemes when the distance between the highest point of the current pixel column and the lowest pixel column exceeds the average stroke-height by a margin. The margin is decided by how rapidly the pixel column height increases. Following this approach allows us to split graphemes at even the smallest teeth.

We over-segment graphemes and simply search for *teeth* or bumps above the connector between letters. Instead of considering the letter **ك** in grapheme-medial position to be made up of one x-height character, we count it as two x-height segments. So the word **كتب** will be considered to be made up of one

ascender segment **ك**, followed by two x-height segments **ت** and **ب** and the word **بصر** is made up of three x-height segments followed by a descender. The letter **ص** here was counted as two segments, the curved part was one while its tooth was the second.

If 100% accurate segmentation of any of the three types cited, could be performed there would be no need for any indication in the shape coded regime of grapheme breaks within a word. However we cannot count on perfect performance of any of these algorithms and so in particular to support over-segmentation code streams we must insert *end-of-grapheme* indicators to allow re-alignment in the case of errors.

3.3 Word boundaries

In Roman script, a single line of text contains a relatively large number of small inter-character spaces, and a smaller number of wider inter-word spaces. The distribution of space width shows two peaks indicating the modal widths of inter- and intra-word gaps. Because of the connected nature of Arabic, there are many fewer spaces per line. Although grapheme spaces are still more numerous than word spaces, grapheme spaces are often not markedly different from word spaces.

Arabic text has a large number of auxiliary marks above, below, and sometimes between letters. These marks may be diacritics, hamzas, or niqat (dots). Because diacritics in Arabic are not consistently placed on all letters, we do not currently handle them. A majority of business text and news uses few diacritic markings if any at all. However, the hamzas and niqat are extremely important for distinguishing many letters from one another as some letters share the same basic skeletal form and only differ in number or position of niqat. We assume all hamzas and niqat are placed somewhere within the width of the region spanned by the character's skeletal. These auxiliary markers appear either above or below the base-line, so we characterize each segment with a property indicating the position of its auxiliary marks if there are any. Letters such as **ش** have three niqat placed very close to one another in a triangular form above the letter's teeth. These niqat may or may not touch, and they may be offset so that they are in the region of either of the two segments that compose its teeth, but their shape is guaranteed to be triangular. We simply associate the markers with one of the segments of the letter and at the word recognition stage we use algorithms that allow these different possibilities to match the same letter.

1. Not to be confused with ligature characters (e.g. lam-alef) that represent character pairs with an entirely distinct form.

4 Shape classification

Having settled on a shape code representation based on over-segmentation, we define a four bit shape code for each segment thus encoded. Two bits define whether the skeletal of the character occupies the ascender, x-height and descender zones and one bit each to define the location of auxiliary marks above or below the skeletal.

Word Shape Tokens (WSTs) are shape code sequences delimited by word separators.



Figure 1: A few words of Arabic text showing the shape codes for each segment

5 Representation ambiguity

If we had a perfect OCR there would be no ambiguity. That is for each word shape code there would be one, and only one, corresponding entry in the surface form lexicon. Shape coding is a computationally efficient but lossy process and therefore, even if done perfectly, there is an ambiguity of representation since one shape code may represent more than one grapheme, character or segment.

We measure ambiguity in several ways, two of which yield a concise metric for the expected utility: average number of surface form words per WST and the proportion of WSTs that represent exactly one surface form word.

The actual ambiguity is a function of lexicon size and diversity (generality) [9] and the characteristics of the shape code set.

Our initial experiments based on coding entire graphemes with a single shape code yielded unacceptable results with an average of 67 words encoded by each WST

Our next experiment was to attempt character segmentation which resulted in a dramatic reduction in representation ambiguity but was too highly sensitive to incorrect character segmentation.

We have finally settled on an over-segmentation of the graphemes. We believe the results, shown in Table 2 below, will be sufficiently good to support a number of shape code based applications. This shape representation leads to a lexicon structure as shown in Table 3

Table 2: Ambiguity of various Arabic shape code representations with English, French and German for comparison.

	Lexicon Size	Shape code set size	Ambiguity	Singleton ratio
Arabic				
Grapheme	24320070	16	67.22	0.19
Character	24320070	12	3.78	0.47
Segment	24320070	12	2.46	0.59
Roman				
English	246906	5	1.40	0.84
French	514637	5	1.73	0.76
German	316035	6	1.51	0.75

Table 3: Snippet of lexicon indexed by word shape token

Word Shape Token	Word 1	Word 2	Word 3
0000 0001 0100 0000 0000 0000 0010	مناسب suitable	متكسب gainfully employed	
0000 0100 0000 0010	يكعب make cubic	يلعب he jests	يلعب he plays
0000 0000 0000 0000 0001 1000 0100 0100 0000 0101	مستهلك customer of you		
0000 0000 0000 0000 0001 1000 1000 0000 0000	مستورد importer		
0000 0000 0000 0001 0000 0000 0000 0001 0000 0001 0000 0000 1001	ستستمنحين you will be given a grant	ستستمتعين you will enjoy yourself	

6 Word spotting

We have previously demonstrated the efficacy of character shape coding of Roman script for a word-spotting application [10]. We now adapt this technique to Arabic script.

We used as a database 100 document images of synthesized news articles. We selected 20 nouns from current news feeds to be used as search terms in this experiment. Table 4 shows the various representations of the search terms along with their frequency of detection in . A document was considered to be found if the search term is found in the document.

The first stage was to check whether we achieved 100% precision and recall based on the Unicode search term applied to Unicode news articles.

The second stage was designed to demonstrate the potential for Arabic Shape Coding and for the particular algorithmic transliteration that we have developed. Transliterations of the Unicode search

Table 4: Different representations of search terms and their frequency of occurrence in the database, trasliterated truth and image-based Arabic Shape Codes.

Arabic	Buckwalter	ASC	Approximate translation	Unicode	Shape-coded transliterated Unicode	Shape-coded document image
حقوق	Hqwq	0189	rights	45	45	44
استخدام	AstxdAm	4000110048	usage	52	52	49
لمجلس	lmjls	4024000	for the council	51	52	52
تحقيقات	AEtyAl	4112448	arrest	60	63	63
اتفاقية	AtfAqyp	4114125	contract/ agreement	38	38	37
لندن	lndn	41009	London	48	48	48
غرينتش	grynt\$	18211010	Greenwich	38	38	35
الجميع	AljmyE	442028	everyone	39	42	38
النشر	Aln\$R	4410108	publishing	38	38	32
المعرفة	AlmErfp	4400815	familiarity/ knowledge	39	39	33
باكستان	bAkstAn	244000149	Pakistan	39	39	33
الصحافة	AlSHAfp	44000415	journal	36	39	26
جولة	jwlp	2845	tour	35	35	30
قادة	qAdp	14005	leadership	44	60	58
واشنطن	wAshnTn	84010149	Washington	36	36	34
الغربيين	Algrbyyn	44182229	Westerners	34	34	31
السلطة	AlslTp	44000445	ruling power	24	29	26
تحقيقات	tHqyqAt	10121410	interrogations/ investigations	13	13	13
الطاقة	AlTAqp	444415	energy	20	25	24
محكمة	mHkmp	00405	court	10	10	10

terms into WSTs were probed for in the transliterations of the Unicode news articles. In an error-free environment, the expectation is that recall would be

100%. Precision would be somewhat reduced due to the ambiguity introduced by the shape coding process.

The third stage of the experiment is to evaluate performance based on transliterated search terms and synthesized document images.

Precision and recall measures for each stage are shown in Table 5.

Table 5: Results

	Unicode	Shape-code transliterated Unicode	Shape coded document image
Precision	100	95.4	93.8
Recall	100	100	97.0

From these data we can see that the shape coding process introduces a very small amount of ambiguity and the process of deriving shape codes from images introduces noise that slightly degrades both precision and recall.

A fourth step, not yet realized, is to degrade the document images in various ways and to re-run stage 3.

7 Status and plans

We believe we have developed a useful tool for spotting the occurrences of specific search terms in document images.

What we described here is a core technology around which we believe, based on prior experience with Roman character shape coding and a demonstrated favorable level of shape coding ambiguity for Arabic, will support a wide range of applications.

We have shown reasonable results in terms of precision and recall on Arabic script documents at a low computational burden and with robustness to image quality degradation characteristic to shape coding as opposed to OCR.

Other applications, as alluded to in the Introduction, will follow.

Acknowledgments

We would like to acknowledge the willing assistance we have received during the process of this development from Chuck Bigelow, Kareem Darwish, David Doermann, Ali Ferghaly, Kamal Mansour, Suzanne Nesbit, Douglas Oard, John Trenkle and Sane Yagi.

We would also like to acknowledge partial financial support for this project from the New Zealand Foundation for Research Science and Technology.

References

- [1] A. Lawrence Spitz, "Progress in document reconstruction", *International Conference on Pattern Recognition*, Quebec City, pp 464-467, 2002.
- [2] A. Lawrence Spitz, "Applications of a feeble text image classifier", *Image and Vision computing, New Zealand*, Auckland, pp 331-334, 2002.
- [3] Daniel P. Lopresti, A. Lawrence Spitz, "Comparing the utility of optical character recognition and character shape coding in duplicate document detection", *Document Analysis Systems*, pp 439-450, 2000.
- [4] A. Lawrence Spitz, "Determination of the script and language content of document images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 3, pp 235-245, 1997.
- [5] A. Lawrence Spitz, Arman Maghbouleh, "Text categorization using character shape codes", *SPIE Symposium on Electronic Imaging Science and Technology*, pp 174-181, 2000.
- [6] Doug Cooper, "How to read less and know more: Approximate OCR for Thai", *SIGIR*, Philadelphia, pp 216-225, 1997.
- [7] Masaharu Ozaki, Katsuhiko Itonori, "A fast Japanese word extraction with classification to similarly-shaped character categories and morphological analysis", *Document Analysis Systems*, Nagano, pp 316-325, 1998.
- [8] Horst Bunke, "Recognition of cursive handwriting - past, present and future", *International Conference on Document Analysis and Recognition*, Edinburgh, pp 448-459, 2003.
- [9] A. Lawrence Spitz, "Moby Dick meets GEOCR: Lexical considerations in word recognition", *International Conference on Document Analysis and Recognition*, Ulm, Germany, pp 221-226, 1997.
- [10] A. Lawrence Spitz, "Using Character Shape Codes for Word Spotting in Document Images", *IAPR Workshop on Syntactic and Structural Pattern Recognition*, Narariya, Israel, pp 1-8, 1994.

Challenges in Adapting Machine-Print Arabic OCR for Handwriting

Steven G. Schlosser Robert C. Vogt

NovoDynamics, Inc.
123 N. Ashley St. STE 210
Ann Arbor, MI 48104
steve@novodynamics.com

Abstract

The creation of a handwritten Arabic recognition system capable of processing degraded pages presents many technical challenges. Some of those challenges are particular to the Arabic language and some to handwriting recognition in general. Other challenges are common to both machine-print and handwritten recognition because handwritten Arabic often resembles machine-print Arabic. Therefore, an effective approach to the development of a handwritten Arabic OCR may be to start with a machine-print Arabic OCR.

NovoDynamics has undertaken the task of adapting its high-performance machine-print Arabic OCR for Arabic handwriting. This paper describes the motivation for pursuing this particular approach as well as the initial results obtained. Performance metrics for early versions of an Arabic handwritten system applied to degraded pages are reviewed, and prospects for future progress described. It is shown that NovoDynamics' machine-print Arabic OCR is a solid foundation for Arabic handwriting recognition and that it has characteristics that make it an attractive candidate for adaptation to other languages as well.

1 Introduction

NovoDynamics' machine-print Arabic Character Recognition System (ACRS) and supporting tools have been developed to tackle the inherent difficulties of performing character recognition on low-quality document images. Because machine-printed Arabic is script-based, the underlying algorithms in ACRS are readily adapted to handwritten documents. The system also incorporates Arabic-specific features in the recognition algorithms which can be leveraged for handwritten Arabic recognition. As a consequence of these early design choices, it is plausible that ACRS can provide a solid foundation for the development of a handwritten recognition system. If true, the development time of such a system can be minimized – a substantial benefit in terms of availability and cost. However, at the outset, it is not certain that ACRS can be leveraged in this way, so our development work begins with a feasibility study

whose principal goal is to validate this approach and to provide insight into the development tasks required to produce a high-performance handwriting recognition system, using ACRS technology as a starting point.

This paper thus describes results obtained as part of a handwritten Arabic OCR feasibility assessment. The study was designed to gauge the performance of ACRS when it is used for Arabic handwriting recognition. In the following, the acronym HACRS refers to ACRS when it is used for handwriting recognition. HACRS 1.0 is actually identical to ACRS version 9.1, however, because **no** modifications were made to the machine-print system before obtaining the baseline measurements for handwritten input.

The technical assessment of the study consists of two components: (1) character and word accuracy of non-degraded ("clean") document page images, and (2) character and word accuracy of degraded images. For each component, the origin of specific OCR failures have been identified and used to form a realistic development plan. This report primarily addresses the second component concerned with degraded pages since its results are more indicative for real-world applications.

Section 1.1 highlights some issues concerning Arabic handwriting recognition while Section 1.2 outlines the architecture of ACRS and its individual modules. Section 2 summarizes the feasibility study, and Section 3 highlights recommended tasks for the development of ACRS as a handwriting recognition system. Section 4 briefly describes the initial performance gains obtained primarily from training the recognition system on handwritten Arabic.

1.1 Handwritten Arabic Recognition

Handwritten Arabic documents present special challenges for character recognition. Like handwriting in other languages, many Arabic handwritten texts are unreadable or barely readable by native readers of the language – often context is needed for correct interpretation. The example shown in Figure 1 illustrates some issues that arise in handwritten Arabic.

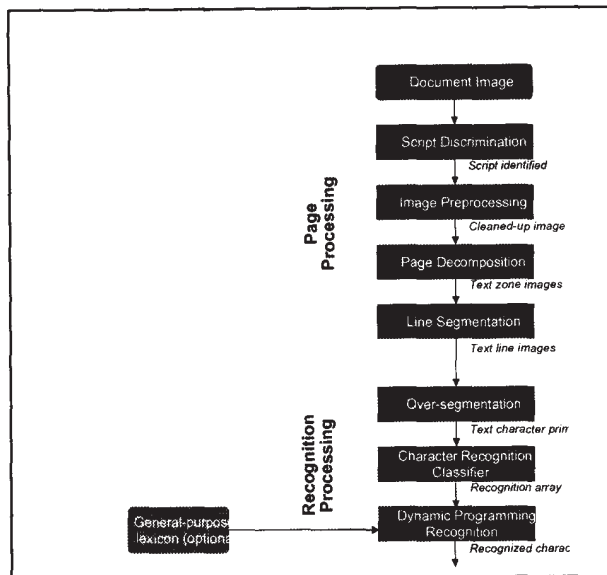


Figure 2. ACRS approach to recognition of individual lines.

header is detected, it is located and separated from the image with the header text contents possibly added to a metadata file. The noise content of the image is measured and a decision model is used to determine whether noise filtering is necessary.

If required, a *noise removal* module is applied; this module is aggressive in removing even heavy salt-and-pepper noise, without removing content that may be dots associated with Arabic characters. Once the image has been cleaned, overall page characteristics such as gross orientation and fax type are assessed. Orientation and the aspect ratio of low-resolution faxes are corrected, if necessary. Finally, small angle image skew (rotation) is measured and corrected.

Page decomposition is then performed on the de-skewed image to segment the page into zones with similar characteristics such as size and stroke width. Graphics, images and other non-text items are removed from further recognition processing. Each text block is then partitioned into individual lines of text by the *line segmentation* module. The text line images are normalized and passed to Recognition Processing.

1.2.3 Over-Segmentation

An *over-segmentation* recognition strategy has been implemented to overcome the limitations of traditional character segmentation when presented with low-quality imagery or connected script. The goal of over-segmentation is to split, or segment, an image of text into *primitives*, pieces containing an individual character or a portion of a character. The task of correctly assembling primitives into characters is performed later in conjunction with character recognition. A primitive that contains multiple characters represents a segmentation error because correct individual characters cannot be reconstructed from it. Therefore, the over-

segmentation algorithm is tuned to produce character fragments, rather than multiple characters, by incorporating language-specific characteristics such as the strong baseline in Arabic text.

The primitives are reassembled into various combinations, called *unions*. Unions are formed from con-

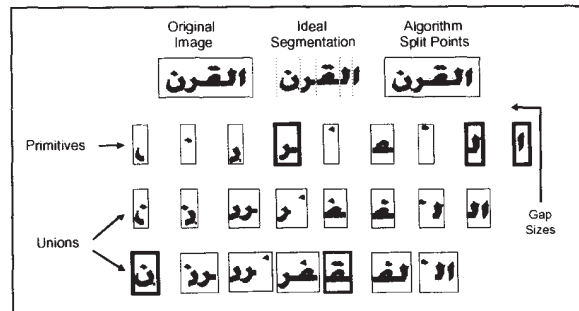


Figure 3. Over-segmentation and the formation of multiple segmentation hypotheses. Heavy box borders indicate correct characters.

tiguous primitives, preserving their spatial relationships. Among these unions are the ideally-segmented characters, as illustrated in Figure 3. The union images are passed to the character recognition module, which simultaneously determines the final segmentation and recognition outputs.

1.2.4 Character Recognition

During character recognition, each union is processed by a *feature extractor* to obtain a vector of descriptive attributes. The feature extractor is a module designed to systematically extract particular characteristics (features) that discriminate one character from another in a given script, for example, character curvature or the presence of dots. These features are then passed to a classifier trained on features from characters of the appropriate language and script. Obviously, a key question with respect to handwriting recognition is whether the features used for machine-print Arabic are fully effective or whether a more suitable feature set must be found.

The output of the classifier is the set of probabilities linking each union to each of the valid characters. The use of probabilities can dramatically reduce the breadth of search in the subsequent dynamic programming step, because the majority of entries are zero and can be ignored. The classifier probabilities for each of the unions are stored in a 3-dimensional array, termed the *recognition array*.

1.2.5 Dynamic Programming

The recognition array is passed to a *dynamic programming* algorithm, which transforms the information in the recognition array into a sequence of characters,

performing both final segmentation and recognition. The dynamic programming algorithm determines the sequence of unions that yields the highest cumulative character confidences across a line. One benefit of the dynamic programming framework is that the character confidences supplied by the classifier may be augmented with bonuses or penalties based on supplementary information such as a general-purpose lexicon, image-domain information such as character spacing, or linguistic constraints such as the rules for assigning language-specific character forms.

2 Feasibility Study

As noted earlier, this paper focuses upon results obtained as part of a handwritten Arabic feasibility study. The following sections summarize what was observed when processing degraded pages of handwritten Arabic.

2.1 Experiment Design

In the degraded page study, the effects of degradation on recognition performance, and the differences between performance results for individual writers were examined, both on the clean "Accepted" set images, and on the mixed "Degraded" set.

For the Degradation Tests, character and word recognition performance were measured for the 496 images in this data set, on the 5 different treatment categories: Clean(none), Skewed, Streaked, Speckled, and Darkened—and then characteristics of the degraded images were analyzed to determine what conditions, in interacting with the ACRS recognition software, might explain the performance differences found.

For the Writer-Specific Analyses, results for character and word recognition rates per page, for each writer, were examined for both the Accepted and the Degraded image sets. The correlations of the character and word rates in terms of the rankings for each writer were also considered. Following this, the writing of those writers representing extreme cases in the distributions were reviewed to look for characteristics that might explain why these were outliers.

2.2 Degradation Tests

Results of the performance analysis for the five different degradation treatments, both character and word recognition rates, are shown in the attached statistical tables and histogram plots found in Appendix A ("Performance Results Charts"). These are placed in order of our expectations for performance (best to worst), namely: Clean, Skewed, Streaked, Speckled, and Darkened. The plots and tables both show the numbers or percentages of pages having particular word or character recognition rates. Cumulative and inverse cumulative plots are also included to make it easier to find the medians, and the numbers of pages having scores more or less than a given percentage.

Overall, these results on degradation are quite good, even somewhat better than expected. For all of the degradation modes except for the worst (Darkened), the decrement in performance is fairly minor, and for the Darkened case, the character recognition rate drops by only a fourth (mean) or fifth (median), while the word recognition rate drops by about half. This is significant because many systems would lose much or nearly all of the text in the Darkened case. Moreover, for the "Speckled" case, performance actually *improved* slightly, over the Clean case. These two results, in particular, are a direct result of the noise analysis and removal capabilities present within ACRS. And, it is worth emphasizing that these results are for handwritten data, using a system that has been designed and trained to operate on *machine-printed* text.

2.3 Experiment Interpretation

A more detailed analysis of these degradation test results reveals some interesting aspects. On the character recognition side, compared to the Clean case (Median 32.9, Mean 35.8, Max 66.9), the Skewed case is very close (Median 31.5, Mean 33.3, Max 61.43). In fact, our expectation is that normally the Skewed images would show *no* performance differences at all, *except* for the fact that some of the text is cut off by clipping, and is thus completely lost. The figures we see here may therefore represent close to an upper bound on performance for the Skewed set, in terms of what text is actually left on the page. In other words, these small differences may be almost entirely due to outright loss of text, rather than difficulties in handling skewed pages.

For the next, Streaked case (Median 29.4, Mean 31.3, Max 63.9), there is a drop of a few percentage points—again fairly minor. The additional errors fall into two groups. For the horizontal lines, even though algorithms to ignore underlines are implemented in ACRS, if the line is connected to a descender, there may be cases where it would not be correctly eliminated, and this would interfere with the recognition of that word and its letters. For the vertical lines, there is a detection and removal algorithm for purely vertical lines such as those that might be generated by a fax machine. This should do a good job of removing most of vertical lines, however, in some cases, these lines are not perfectly vertical and thus might not be adequately detected and removed by the current algorithm.

The Speckled case (Median 34.7, Mean 35.5, Max 70.3) provided a real surprise. While performance was expected to be slightly worse than the above two, in actuality it was slightly *better* than the Clean result, by a couple percent, at both the low and high end. A careful examination of these cases revealed two things. First, the degree of noise is generally light, and the noise is generally less dark than the text—making it fairly easy to remove with our binarization algorithms.

In addition, something not noticed earlier is that the method used to create the speckle also apparently thickens the strokes of the characters somewhat, making them slightly easier to recognize, as well as joining the characters of words together more tightly. The latter effects apparently led to the observed improvements.

Finally, for the Darkened case (Median 25.6, Mean 26.8, Max 66.9), there is a larger drop in performance, as one would expect given the large amount of dark noise that is added to the images—about 7 percentage points in the Median (20% lower), and 9 percentage points in the Mean (25% lower). Still, these are not very significant drops considering the extent to which the quality of the image has deteriorated. Again, this ability to recover text in the face of fairly heavy noise is due to the noise assessment and filtering algorithms present in the ACRS system.

On the word recognition side, the picture is somewhat the same. For the Clean case, the performance on the 496 image set used for this test is reasonably good for handwriting, using a machine print-based system (Median 9.1, Mean 11.9, Max 42.4). The Skewed case is barely different (Median 9.1, Mean 11.5, Max 36.2)—the main differences are felt on the high end (Max 36.2 compared to 42.4), as one might expect due to the clipping effects cited earlier. The Streaked case shows more degradation in performance (Median 6.9, Mean 9.5, Max 40.8), but only a couple of percentage points, again due to the errors in detecting and removing the spurious lines that were mentioned above.

As with character recognition, word recognition rates on the Speckled case (Median 10.7, Mean 13.3, Max 56.1), were actually a bit better than even the Clean case, especially for the best pages. Again, it appears this was due to the thicker characters, leading to better intra-word joining, that resulted from the speckling treatment of the page. Finally, the Darkened case showed a more significant decline, as expected (Median 3.9, Mean 6.8, Max 46.2) for the averages—however, these represent a drop of only 40-60 percent in performance levels—not bad considering the severity of the degradation, and, on the best pages, some *improvement* over the Clean case was found, again likely due to the thickening of the character strokes, similar to the Speckled case.

2.4 Writer-Specific Analyses

The second set of analyses had to do with looking at performance differences between individual writers. These results are summarized in charts which also appear in Appendix A. The first pair of these shows the writer IDs sorted by their Accepted set character recognition performance along the x-axis. We then have the character recognition rates for the Accepted and Degraded image sets in one graph, and the corresponding word recognition rates for the two image sets in the other graph.

These charts show that the deterioration in character and word recognition performance for different writers, actually varies to some extent. Some (such as DR, AA, MC, and AH) are hit harder than others by degradation, while a few (MS, JB, and JC) actually seem to get better. (Though the fact that these are at the low end to start with suggests that the improvement seen may be due to the fact that their worst pages were thrown out in going from the 568 Accepted image set, to the 496 Degraded set). As it turns out, AA and MC degrade more rapidly because their writing is small. DR has taller lines, but this comes mainly from large ascenders—the interior characters are also small and thus easily damaged by degradation or noise. AH suffers under degradation for a different reason—the text lines in this writing are sloppy, not straight, and not well-separated, so that when noise is added, it becomes very difficult to segment them from each other.

It can also be seen from the word recognition chart that character and word recognition performances do not go strictly hand-in-hand.

In the second pair of charts, the correlations of the character and word recognition *rankings* of the 19 different writers are shown, in one case for the Accepted image set, and in the other for the Degraded image set. What the charts show, in particular, is where the two recognition rates are distinctly *different* for certain writers, based on specific characteristics of their writing. The fact that Accepted results differ from the Degraded results, indicates that different writing styles have varying abilities to survive the effects of degradation. Thus, in attempt to summarize the above discussion about performance rates for different writers, it appears there are a small number of basic principles which explain most of the differences seen over all four charts, some of which are more obvious than others. They are:

- *Well-formed, distinct characters will be more easily recognized than sloppy ones, and better able to survive degradation*
- *Likewise, well-formed, distinct words will be more easily recognized than sloppy ones, and more robust with respect to noise*
- *Small letters, if written in a very regular fashion, look more like machine-print text and are likely to be better-recognized by our machine-print text recognizers*
- *Small letters, because of their size however, are less able to survive degradation, especially severe degradation*
- *Tall letters, if written well with uniform scaling, can also be fairly easily recognized, but can also survive degradation processes intact enough to still be recognized*
- *Some writers have very tall ascenders and descenders, but with other characters being*

fairly small. This can reduce character recognition rates on the smaller characters, because after normalizing a line or word to the larger character size, the smaller ones are too small to be easily recognized.

- *On the other hand, some writers who have this style of writing (which is often also very script-like, sometimes with "elided" characters that are barely present—almost like cartoon strokes)—some of these writers also naturally emphasize "key" letters that are needed to recognize the word. If there is enough information in these emphasized letters to clearly identify a word in the language, then the word may still be recognizable, even if some of the characters are not.*
- *Some writers make the separation between words very distinct, and/or tie together the letters of the same word, either in terms of proximity or actual connection. This improves word recognition.*
- *Other writers write the characters of a word in a more spread-out or broken fashion, and often have breaks between the words that are not obviously larger—this makes it difficult for the character segmenter to assign word breaks accurately, especially after degradation.*
- *Some writers make straighter lines of text, while others make curved or skewed lines; the latter are more difficult because the line segmenter is currently built to handle straight, uniformly oriented, machine-print lines.*
- *Some writers put more separation between lines than others. When there is less separation, especially in cases of strong degradation in the form of noise, it is more likely that line boundaries become confused, and the lines possibly merged, which basically loses all of the text in both.*

In looking at the rank correlation charts, it can be seen that DD, JP, and JPH on the 568 Accepted set chart have better word recognition rates than their character rates would suggest, while PP and JM are worse. DD and JP are examples of very script-like writing, with elided characters, but such that key characters are emphasized, allowing word recognition to occur, despite poor character rates. JPH appears to do better, because the word separations are particularly distinct in this writing, while the characters are only moderately good in terms of recognition. For PP and JM, on the other hand, while the characters are reasonably well-formed, the words are somewhat sloppy, and not well-separated.

For the 496 Mixed Degraded image set, on the other

hand, we see in the second chart of this pair that JPH and AH perform better than expected in terms of word recognition, while BW and JM perform worse. Again, JPH, as mentioned earlier, has very good word separation, which augers well in the face of noise and other forms of degradation. AH was lower in both character and word performance under degradation, apparently due to line segmentation issues but, in relative terms, the word rate here survived better because the letters are large and well-written, and the words are reasonably well-separated, even though the lines are somewhat sloppy and not well-separated. BW does more poorly because the writing is small, and there isn't good separation between the words—which gets worse when noise is added to the image. Similarly, JM, as noted earlier, has good characters but the words are sloppy, somewhat broken, and not well-separated, and the lines are not well-separated—all of which lead to segmentation problems when noise is added.

3 Recommended System Development

The ACRS feasibility study results described in Section 2 show that it is very likely to be feasible to construct an effective Arabic handwritten recognition system, using ACRS as the foundation. In fact, the study provides good reason to believe that well-understood engineering modifications to ACRS will result in a high-performance system that is capable of processing a wide range of real-world handwritten material. The results of the feasibility study also highlight the processes within ACRS that require adaptation, extension or replacement before handwriting can be recognized reliably. Most important among these processes are line segmentation, character segmentation and recognition, and word recognition.

Based on the encouraging results of the feasibility study, the envisioned handwriting recognition system will be developed on the foundation of the existing Arabic Character Recognition System. The page and recognition processing capabilities of that system will be extended to accommodate the unique characteristics of *handwritten* Arabic.

More specifically, the adaptations of the ACRS system needed to account for handwritten material are summarized in Table 1.

Module	Requires Modification	Comments
Script Discrimination	Yes	Tune existing handwritten vs. machine-print discriminator
Image Preprocessing	Yes	Tailor page 'characterization' to handwritten Arabic
Page Decomposition	No	Use existing ACRS capability
Line Segmentation	Yes	Modify for handwritten lines
Over-segmentation	Yes	Modify for handwritten segments
Character Recognition Classifier	Yes	Retrain character classifier Reevaluate ligature handling
Dynamic Programming Recognition	Yes	Tune for modified classifier outputs

Table 1. Modifications to the ACRS for Handwritten Arabic Recognition.

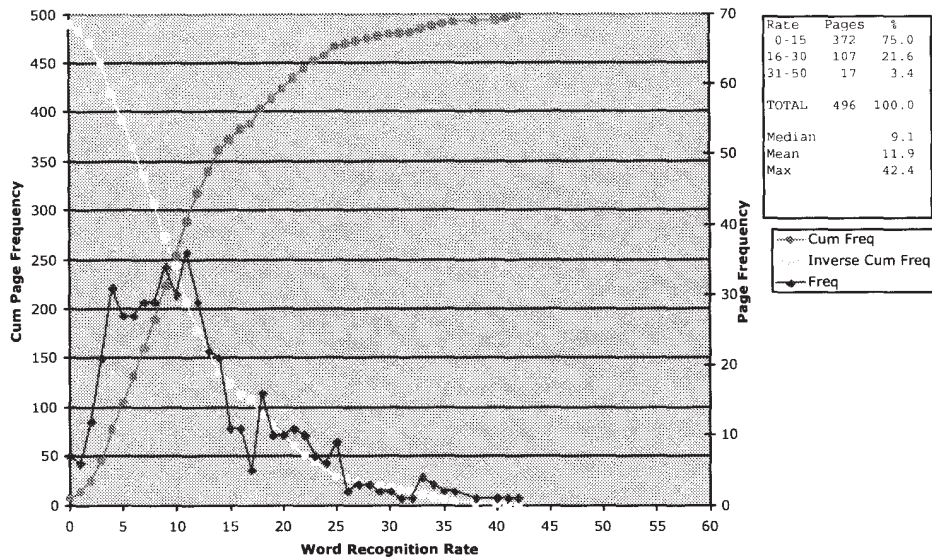
4 Current Status of Handwritten Arabic

This paper has focused on the results of a handwritten Arabic feasibility study carried out by NovoDynamics earlier this year. Since the completion of the study, the development of a handwritten Arabic system based on ACRS has begun, focusing on the modifications noted in the previous section. To date, the results have been excellent. Specifically, the current character recognition rate is approximately 55% (i.e., a 66% increase) even though the newly developed handwritten training and test materials are much more challenging with respect to writing style and neatness. These developments will be the subject of a future report.

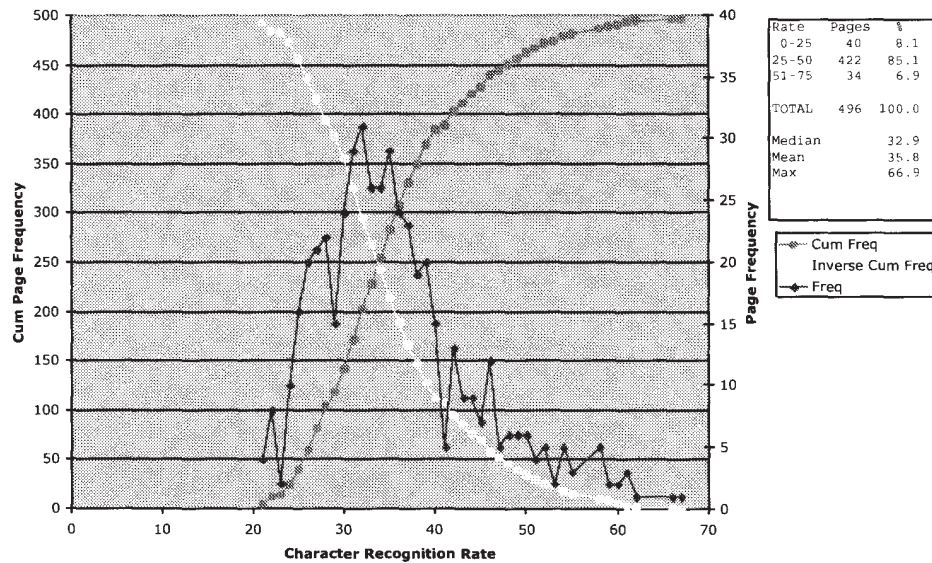
Appendix A: Quantitative Results

A.1 Degradation Test—Clean Set

Word Rate on Clean Images

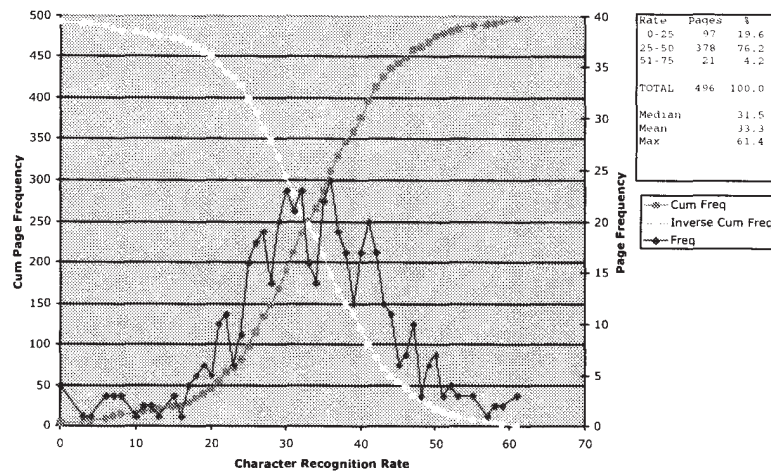


Character Rate on Clean Images

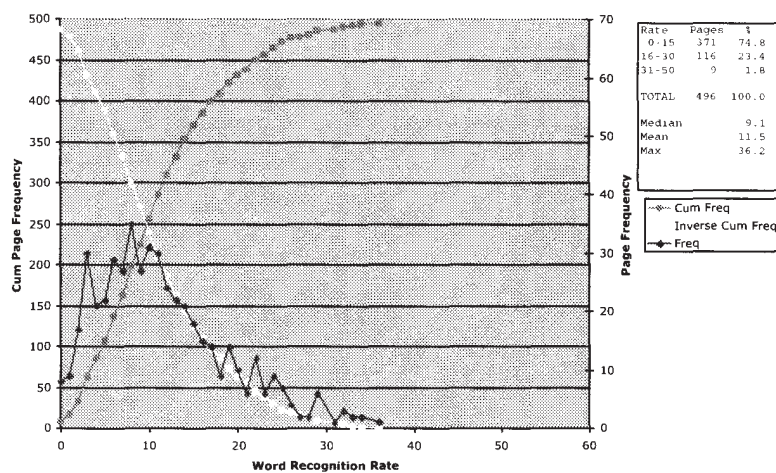


A.2 Degradation Test—Skewed Set

Character Rate on Skewed Images

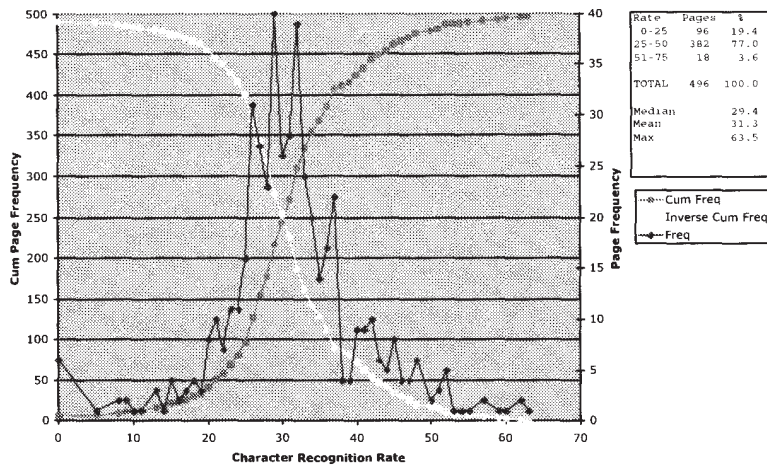


Word Rate on Skewed Images

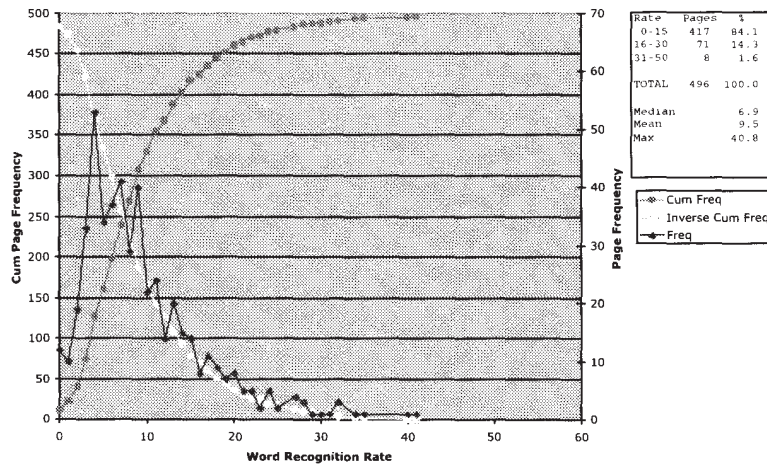


A.3 Degradation Test—Streaked Set

Character Rate on Streaked Images

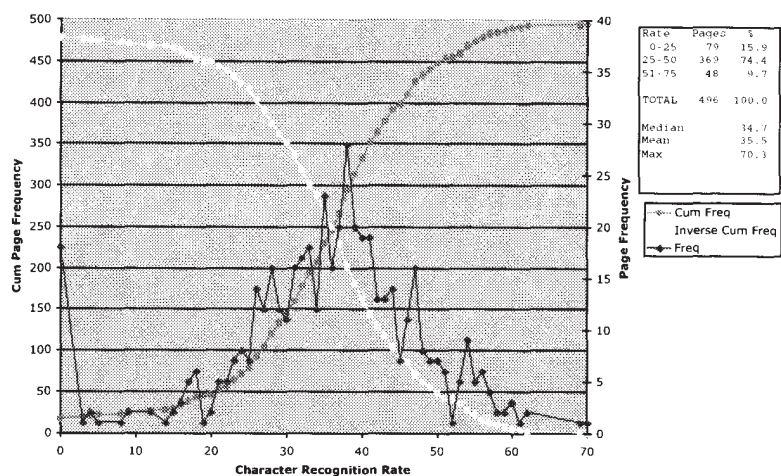


Word Rate on Streaked Images

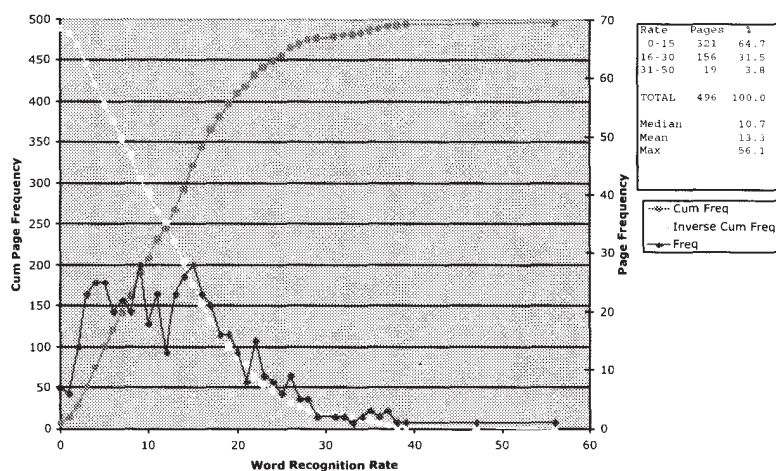


A.4 Degradation Test—Speckled Set

Character Rate on Speckled Images

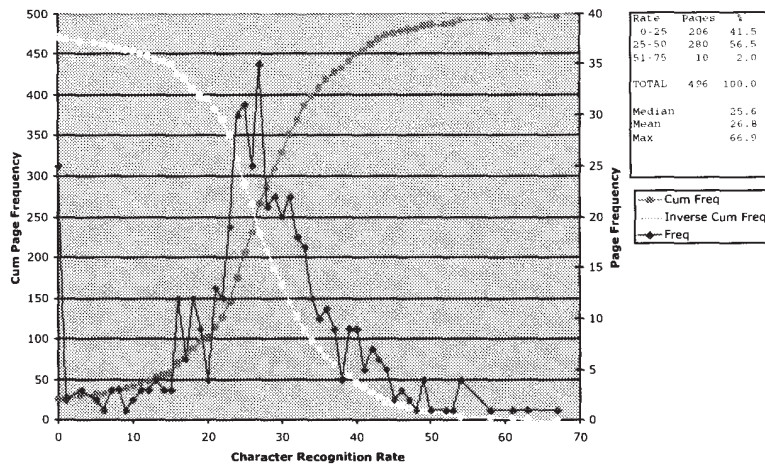


Word Rate on Speckled Images

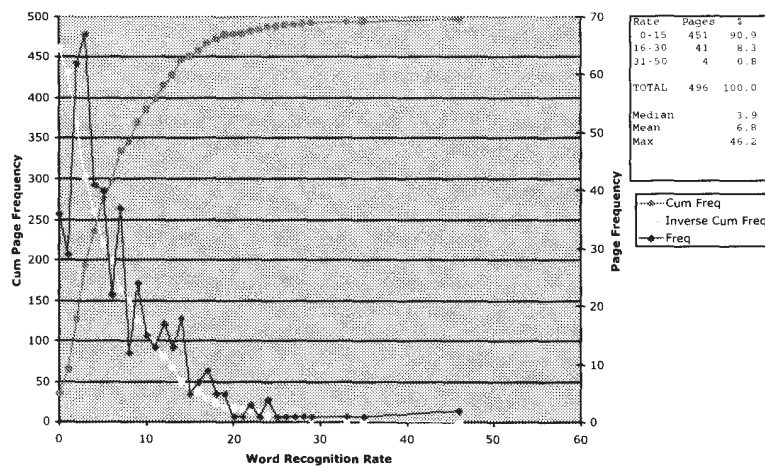


A.5 Degradation Test—Darkened Set

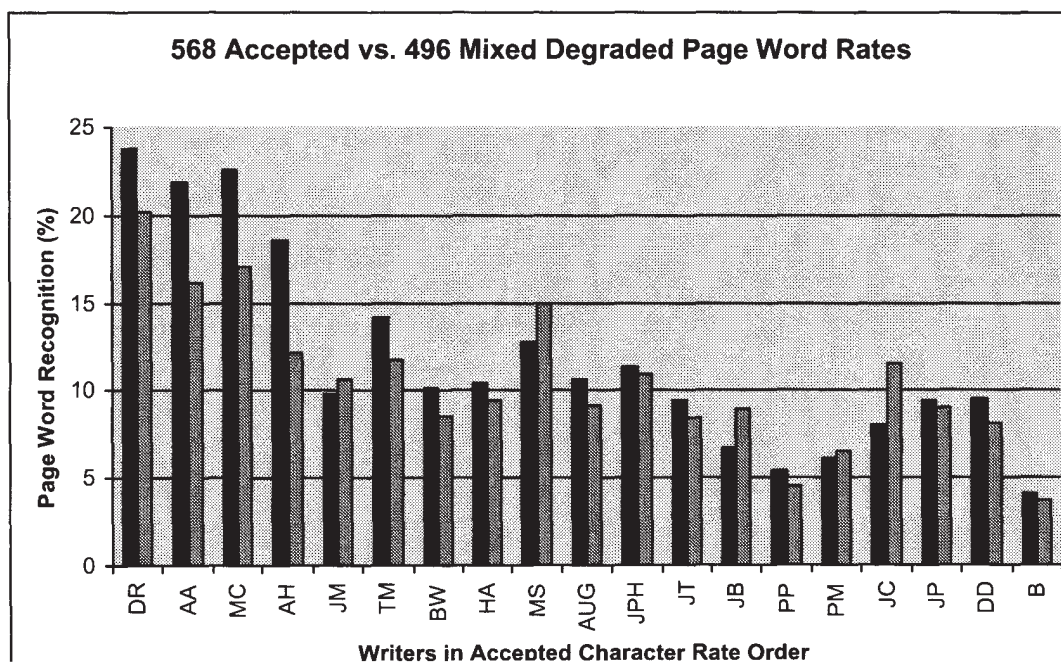
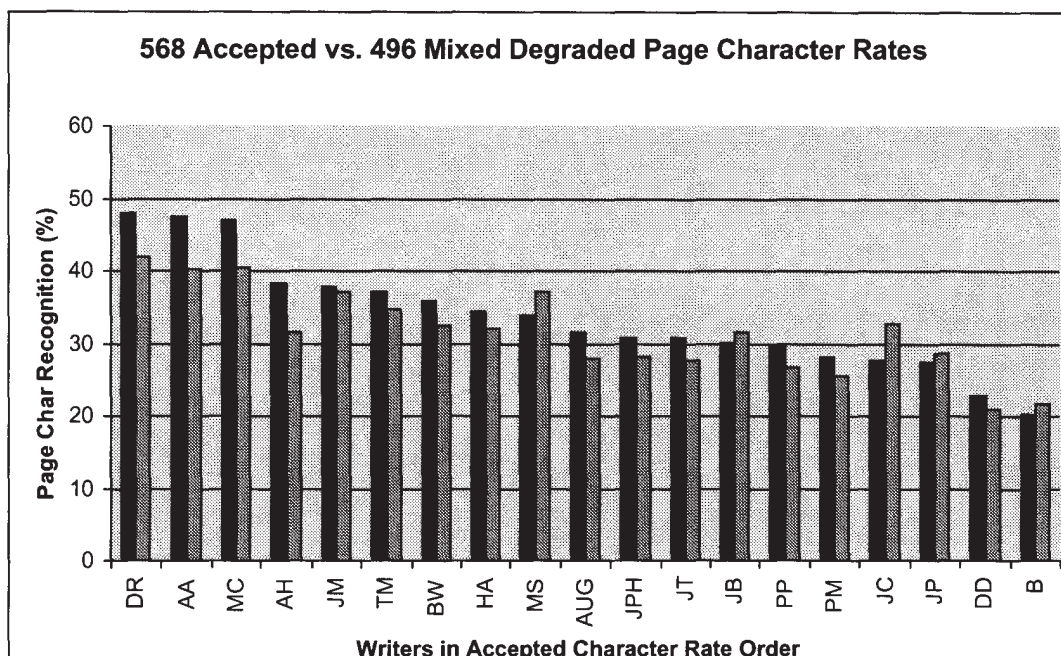
Character Rate on Darkened Images



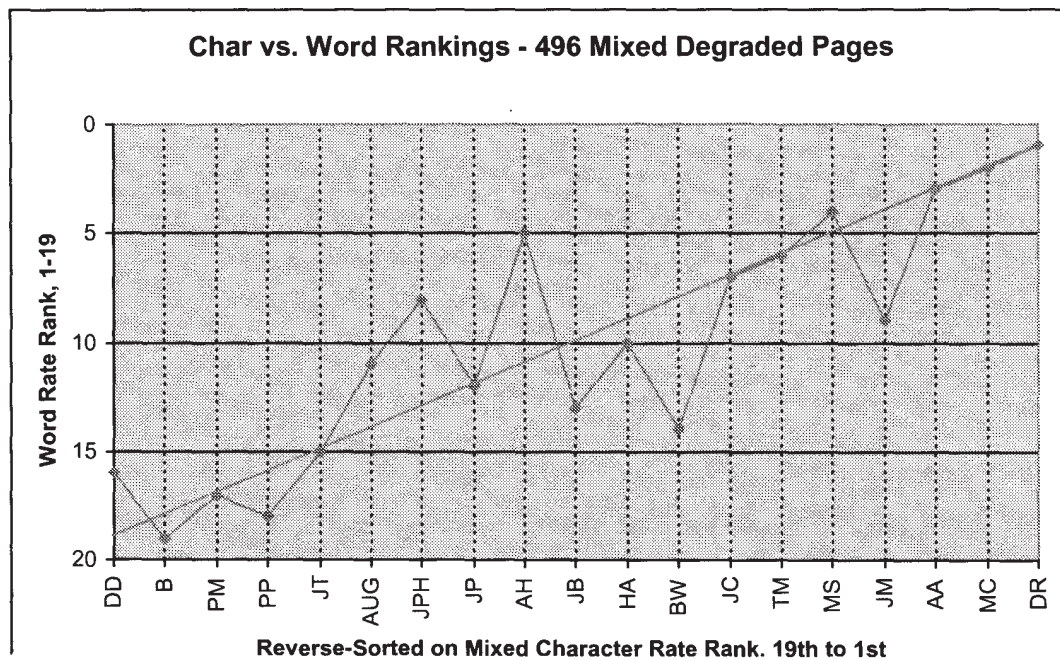
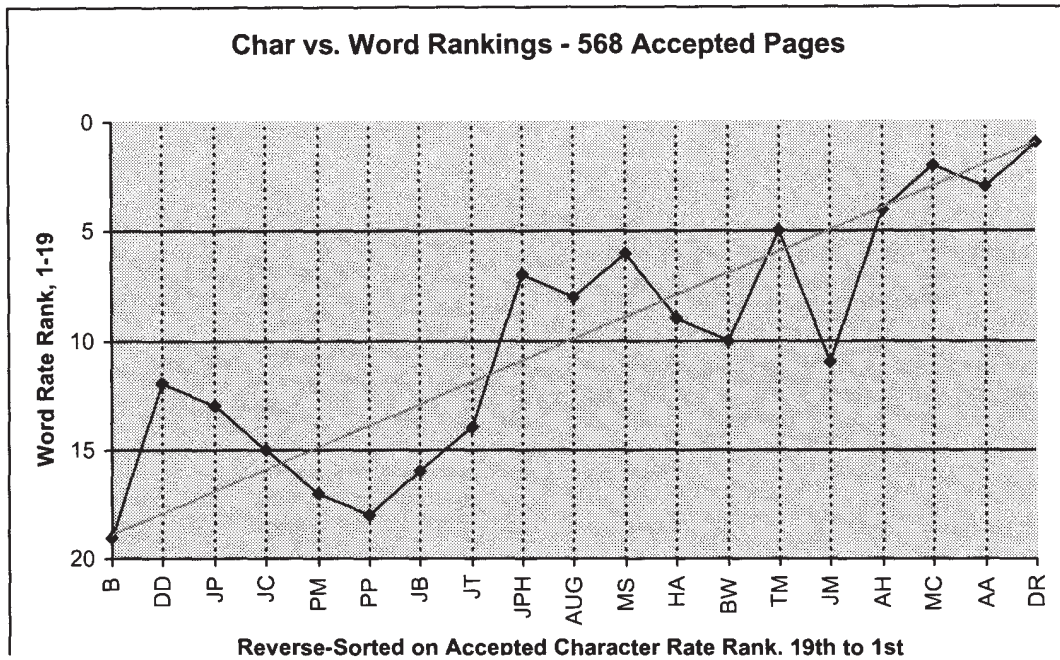
Word Rate on Darkened Images



A.6 Writer Assessment—Performance Rates



A.7 Writer Assessment—Rank Correlations



Cross Domain Applications

Robot Navigation Techniques for Engineering Drawing Analysis

Thomas C. Henderson Chimiao Xu

School of Computing
University of Utah
Salt Lake City, UT 84112 USA
tch@cs.utah.edu

Abstract

Engineering drawings provide a significant challenge to image analysis. The goal is to take a scanned engineering drawing image and produce an interpretation of the contents in terms of characters, digits, arrows, line segments, dimensions, etc. Our goal is to incorporate these results into a legacy system re-engineering process (e.g., image analysis must provide parameter values for manufacturing features like counterbore holes, etc.). We propose to treat the problem in two steps: (1) determine a good set of feasible points which constitute the markings on the page, and (2) subsequently treat these as a 2D floor plan that is explored by a tiny mobile agent that navigates through the drawing and produces a map. The trajectories followed by the agents allow a segmentation of the image into basic geometric units: straight line segments, end points, branch points, etc. In this paper we study static Pseudo-Range Maps (PRMs) to identify point features in the image.

We believe that this approach can be applied to any kind of line drawing material where the objects of interest can be segmented. We conjecture that it will also be possible to characterize and identify processes that create such line drawings (e.g., people, printers, etc.).

1 Introduction

Almost all legacy engineering designs involve 2D CAD drawings. The most common tasks for the re-engineering of legacy systems include the modification of existing designs to make a new design, or the synthesis of 3D models from multiple 2D views (usually available as engineering drawings). Automatic engineering drawing analysis offers a useful tool in the interpretation of engineering drawings when the original electronic CAD is not available. Engineers engaged in reverse engineering rely heavily on the scanned engineering drawings from blueprints as well as 3D data from the available

physical parts. Our goal is to aid the design process by achieving the most accurate image segmentation possible in order to permit the best interpretation of annotations in digitized images of CAD drawings.

The automatic analysis of engineering drawings has posed interesting and challenging research topics in document analysis, pattern recognition, image processing and computer vision for a few decades [19, 22–25]; however, the complete analysis of engineering drawings is still an unsolved problem. Most extant systems exhibit brittle performance when applied to real world image sets. (See Tombre [1] for an overview of the field and Kanungo et al. [2] for insightful commentary.) Most previous work deals with the interpretation of particular symbols and structures in CAD drawings (e.g., straight line vectorization [4, 12, 16], arcs [20], and dimensioning analysis [3, 5]). Our survey shows that previous research focuses on methodologies and algorithms for individual aspects of the problem and only achieves good overall performance in special circumstances. Most of the known algorithms and procedures require noise-free conditions which is usually an unrealistic assumption. Research is still underway in this area due to real world application demand, and no existing system works well universally.

We have studied this problem for a number of years (e.g., see [8–11, 22]). That work explored the use of structural constraints in the underlying document elements, and feedback mechanisms to reinforce paths to successful interpretations. However, one constant issue is the poor quality of the image segmentation of the engineering drawings.

We propose to improve the image segmentation process by taking advantage of the fact that engineering drawings are human artifacts and thus have a regular structure. Moreover, this structure is quite similar in nature to the 2D floor plan of a building layout; e.g., lines are like hallways. Figure 1 shows a typical layout to be explored and mapped by a mobile robot. Similarly, the scanned digital image of an

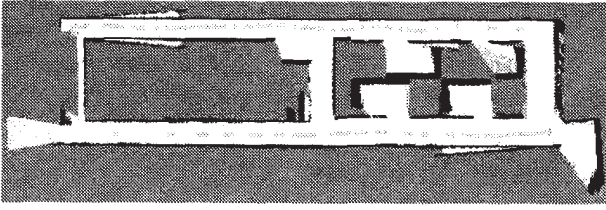


Figure 1: Building Layout for Mobile Robot Mapping

engineering drawing has errors and noise, and therefore poses difficulties for standard image processing techniques. We propose a paradigm shift: view the drawn lines and symbols as hallways, rooms, etc., and use robot navigation and mapping methods to analyze the drawing structure. Mobile agents are placed “inside” the drawn lines and located with subpixel precision. The gray levels serve as a *density* function, and a pseudo-range scan can be performed at any pixel (the background of the line drawing image serves as the solid surfaces). The engineering drawing analysis is now defined as a robot mapping problem. We propose to demonstrate that features important to the drawing analysis can be extracted using this approach. Once these features can be robustly extracted, we believe that it is possible to obtain the line segments and structures necessary for high-quality drawing interpretation. The method also makes it possible to learn interesting structure within the image automatically. Finally, localization methods can be used to match models of known structures (e.g., letters, digits, arrowheads, etc.).

In this paper we explore the static use of pseudo-range maps at individual pixel locations to classify 0-dimensional features (i.e., point features), including: (1) *end points*, (2) *interior corridor points*, (3) *corner points*, and (4) *branch points*. Performance of this approach is demonstrated on engineering drawing images for which ground truth is available.

2 Method

We propose a two-step process:

1. Threshold the image into foreground and background components, and
2. Use mobile robot mapping techniques to interpret point features in the foreground data.

Thus, step 1 produces a binary image and step 2 extracts point features of interest.

2.1 Foreground Segmentation

The images being analyzed are mostly comprised of lines; that is, elongated, thin linear structures. The foreground segmentation algorithm takes as input an image, I , and a length, k and is given as follows:

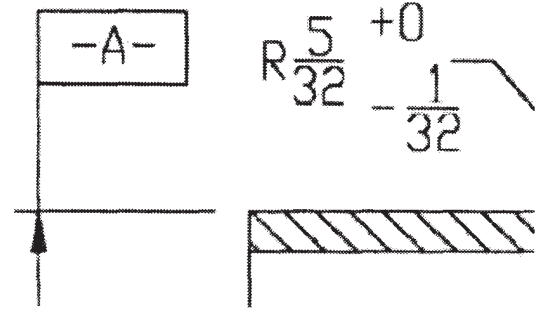


Figure 2: Original Gray Level Drawing Image.

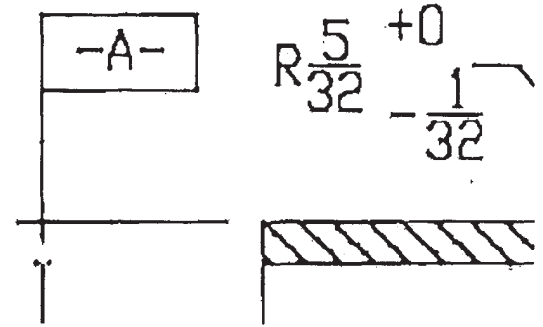


Figure 3: Segmented Image.

- for every pixel, p , in I
 - analyze the linear vector of k pixels centered at p and in the 4 major orientations (0, 45, 90, 135 degrees)
 - pick the orientation, if any, with the strongest contrast between the middle pixels and the outer pixels (outer pixels higher valued than the middle pixels since the drawing is dark on light background).

Figure 2 shows an original gray level image and Figure 3 its segmented counterpart. (Figure 4 shows a segmented image using a standard within-group variance thresholding technique [18].)

Note that the arrow head is missing in Figure 3 as it is not a linear structure. Also note how the fraction numbers and line are messy and not well-separated in Figure 4.

2.2 Mobile Robot Mapping Techniques

Figure 5 shows part of an engineering drawing with the position and orientation of a mobile agent overlaid on it. The agent can produce a pseudo-range map (PRM) at any location; Figure 6 shows the

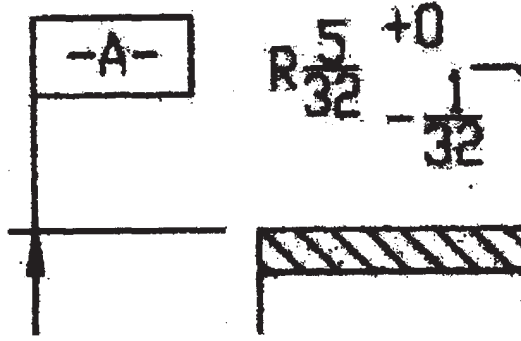


Figure 4: Image Segmented using Standard Method.

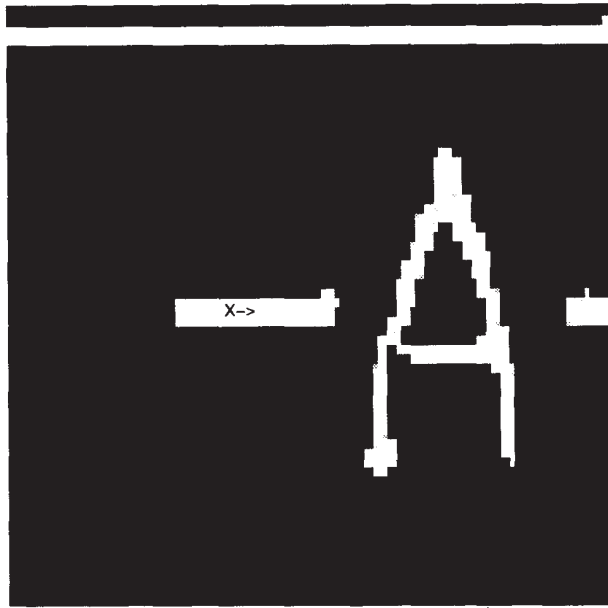


Figure 5: An Engineering Drawing with Agent Overlaid.

PRM for the given location (note that the range is taken at sub-pixel accuracy). The basic idea is that the agent will use the PRM in order to explore the line drawing. This will be done using standard mapping techniques. While exploring the line drawing, features, segments and symbols will be extracted. Performance will be analyzed using a dataset for which we have determined the ground truth.

Mobile robot navigation, localization and mapping is still an area of active research in the robotics community [2, 17]. However, many techniques are already developed which apply to our problem domain[7].

Mobile robot mapping techniques exploit three types of map concepts: *topological*, *geometrical*, and *grids*. From one or more of these standpoints, a map of the domain will be built. Specific approaches in-

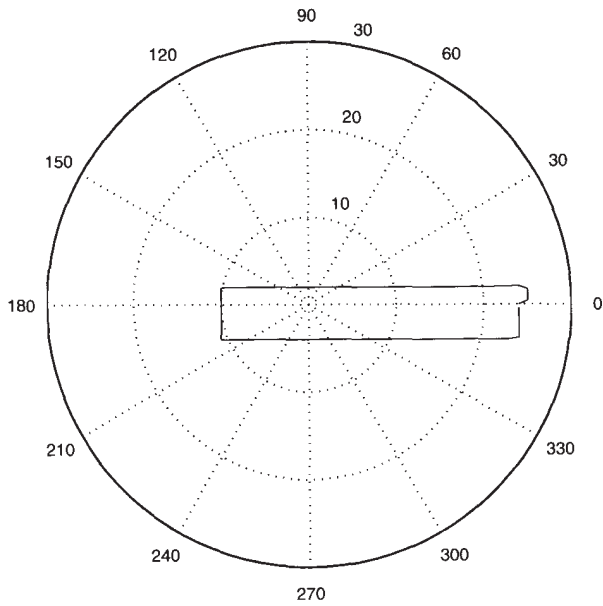


Figure 6: A Pseudo-Range Map (PRM) for the Agent Location.

clude the use of the the visibility graph[14], the generalized Voronoi diagram (GVD)[1], trapezoidal decomposition, or probabilistic roadmaps[13]. We have developed some simple mapping procedures (similar to [7], p. 176) that construct an approximation to the GVD.

2.3 Feature Analysis

We describe here our work on 0-D features in the image, but ultimately, we hope to exploit navigation techniques to discover:

- 0-dimensional features: point or locale-centered features:
 - end points, small blob segments
 - corners
 - multi-branch points
- 1-dimensional linear features
 - straight line segments
 - curved line segments
- 2-dimensional features
 - arrowheads
 - certain symbols

An initial foray into 0-D feature extraction demonstrates the feasibility of this approach. The PRM can be used at each pixel to identify:

1. *endpoints*: the terminal part of a line segment (a dead end in terms of robot exploration)

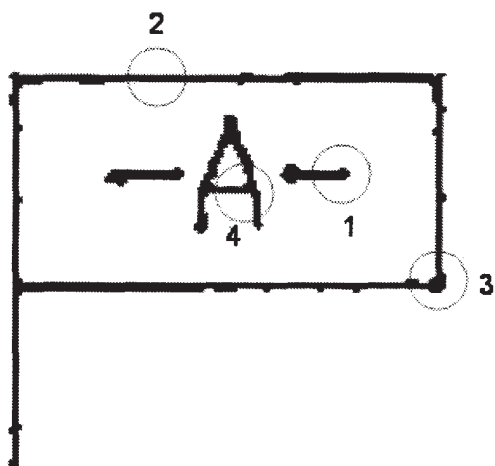


Figure 7: The four 0-D Feature Types (1: endpoint; 2: interior corridor; 3: corner; 4: multi-branch point).

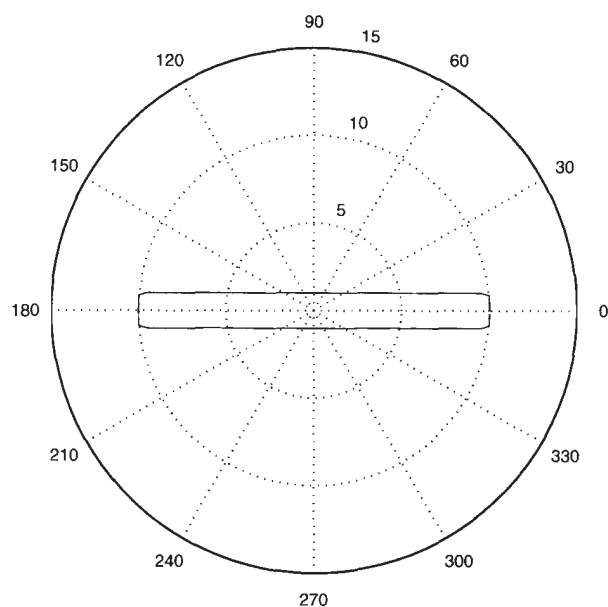


Figure 9: The Pseudo-Range Map for the Interior Corridor Feature (number 2 in Figure 7).

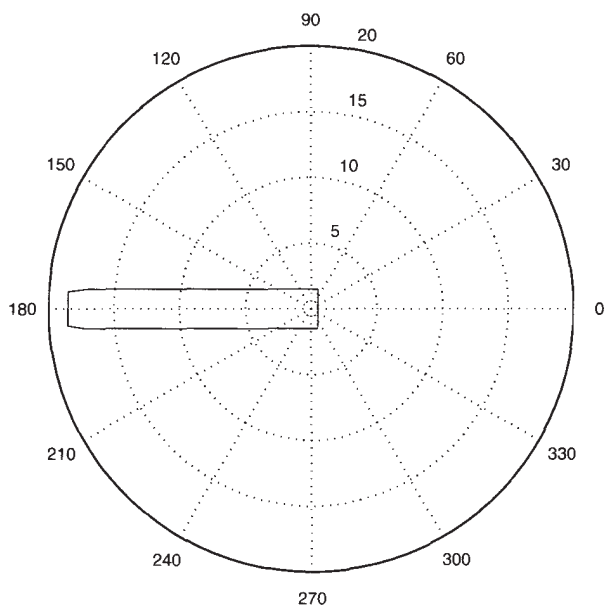


Figure 8: The Pseudo-Range Map for the Endpoint Feature (number 1 in Figure 7).

2. *interior corridor points*: two directions of travel possible, but not a corner (robot can move basically in two directions 180 degrees apart)
3. *corner points*: two directions of travel possible, but at significant angle off 180 degrees
4. *multi-branch points*: more than two directions possible for robot to explore.

Figure 7 shows examples of these four types, and Figures 8 to 11 show the PRM for each location circled in the image.

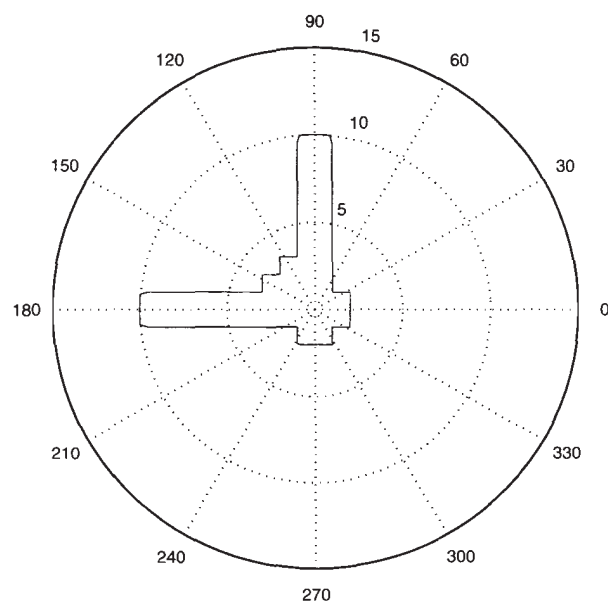


Figure 10: The Pseudo-Range Map for the Corner Feature (number 3 in Figure 7).

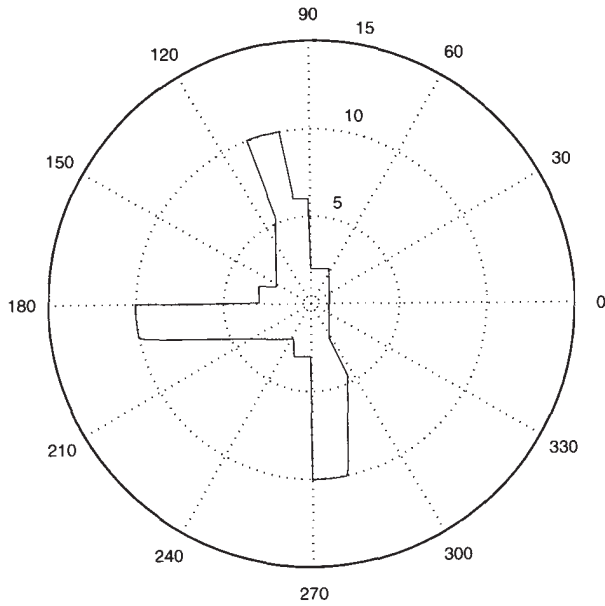


Figure 11: The Pseudo-Range Map for the Multi-Branch Feature (number 4 in Figure 7).

Given a thresholded image, this algorithm labels the 0-D features in the image; Figures 12 through 15 show the feature points extracted from the image. This is done using the polar range plots; the number of branches in the polar function tells the type of structure.

The feature classifiers work by taking into account properties of the PRM like number of peaks, maximum range, and range in the direction opposite to the maximum range (e.g., endpoint pixels have short range opposite the maximum, while corridor pixels have long range in both directions).

The performance of the proposed method, even without serious pre- and post processing, is better than that of the standard line detection algorithms.

In an attempt to determine how well our algorithm works, we explored the use of decision trees as a classification method. In particular, the information-theoretic approach described in [21] was applied, using the following attributes of the PRM:

- *attributes 1-7*: Hu invariant moments[15] of the PRM image region.
- *attribute 8*: area of the PRM image region.
- *attribute 9*: perimeter of the PRM image.
- *attribute 10*: sum of ranges in the PRM.
- *attribute 11*: total distance of the points in PRM perimeter to the point which is used to compute the PRM.

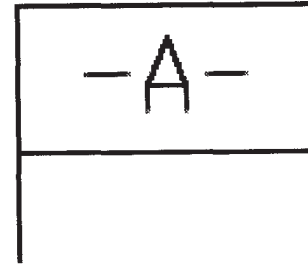


Figure 12: Endpoints Found in Image im00.

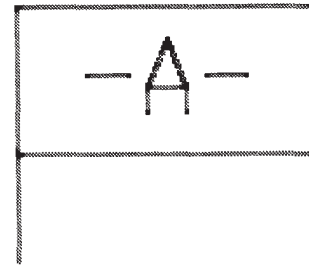


Figure 13: Corridor Points Found in Image im00.

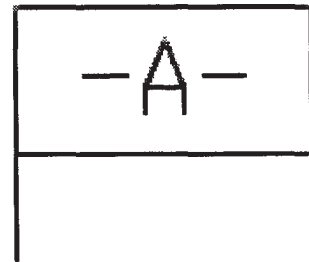


Figure 14: Corner Points found in Image im00.

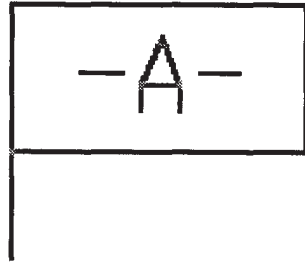


Figure 15: Branch Points Found in Image im00.

- *attribute 12*: sum of the absolute distance of the rows and cols of the PRM region point's to the row and col of the point which is used to compute the PRM, divided by the total number of points in the PRM region.
- *attribute 13*: number of branches in the PRM. (The range of the branch length must be longer than 90% of the maximum range of the PRM.)

A set of training samples were selected, and four decision trees were built: one for each 0-dimensional feature. It is interesting to note the most discriminating attribute (i.e., the first branch is determined by this attribute) for each feature: (1) endpoints: attribute 11, (2) corridors: attribute 2, (3) corners: attribute 12, and (4) branches: attribute 11. These classifiers perform well (see Section 3 on performance), but not as well as the hand written classifier.

3 Performance Evaluation

The overall system performance is evaluated in terms of the quality and computational complexity demonstrated over various image datasets. Noise is introduced during the digitization process; thus there are extraneous as well as missing objects in the resulting image. Both noise and missing data can have a large influence on the image interpretation process. In addition, blueprints might be stained, damaged during storage and usage, and scanners might have different blur, lighting, and scale factors [8]. To objectively measure how well the proposed system analyzes digitized engineering drawings, we compare results over a dataset for which we have ground truth knowledge acquired from engineering drawing datasets.

We have done the following steps for system performance evaluation:

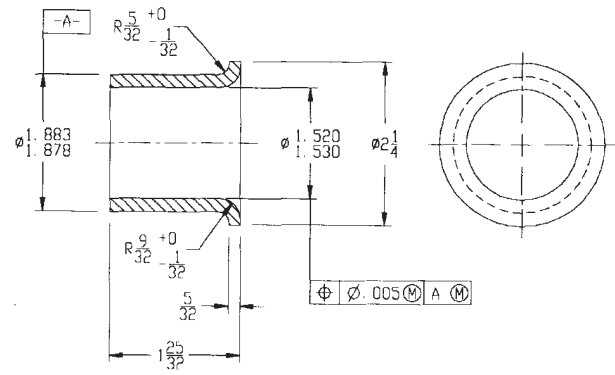


Figure 16: Test Image im00.

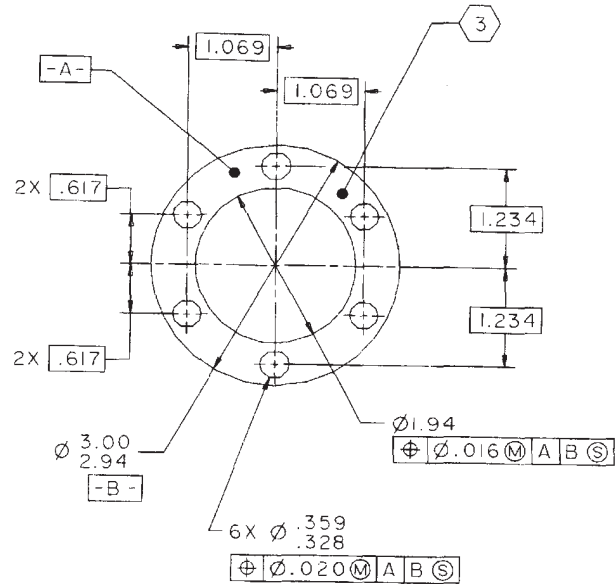


Figure 17: Test Image im17.

- Established a testbed benchmark set of five images and ground truth. These five images are shown in Figures 16 through 20.
- Run our previous feature classifier (non-PRM) on the test images. (See Table 1.)
- Run the PRM classifier on the test images. (See Table 2.)
- Developed a decision tree classifier using training data (numerical features of the PRMs). (See Table 3.)

Figures 16 through 20 show the engineering drawing test images.

To measure the performance of the algorithm, we use *recall* to mean the ratio of correct features found to total number of features, and *precision* to indicate the ratio of the number of correct feature responses to the total number of feature responses. Tables 1

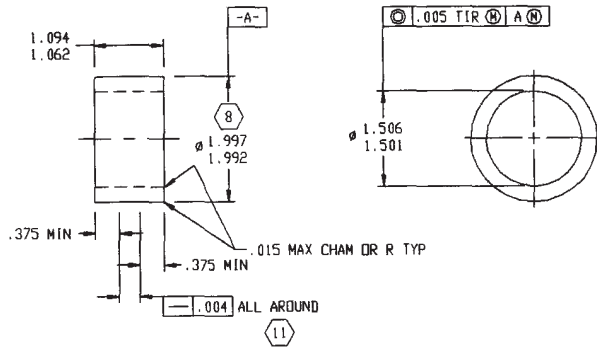


Figure 18: Test Image im18.

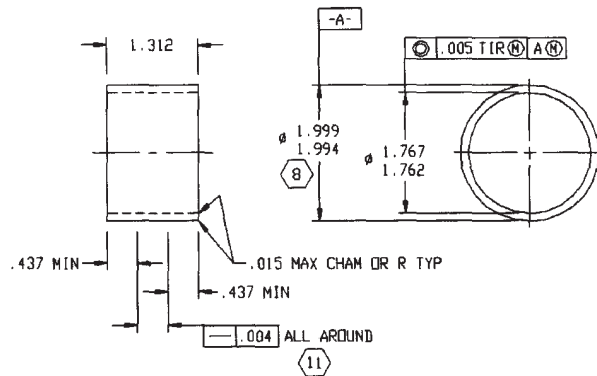


Figure 19: Test Image im19.

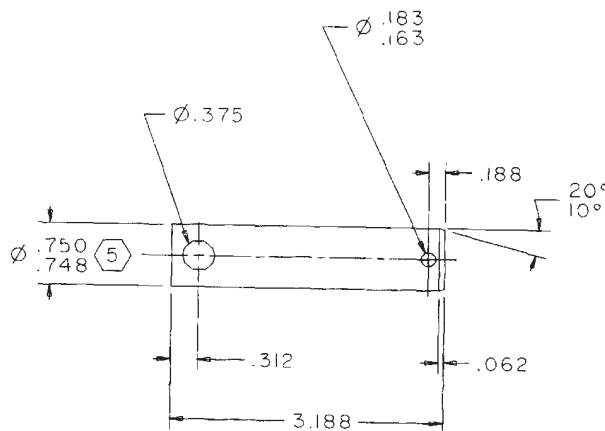


Figure 20: Test Image im22.

through 3 show the *recall* results from these experiments, while Tables 4 through 6 give the *precision* results.

Image	F_end	F_corr	F_corn	F_bra
im00	90.45	99.55	72.60	81.22
im17	92.41	99.25	65.11	89.25
im18	96.38	98.64	73.27	71.90
im19	95.59	98.95	73.88	86.36
im22	84.85	99.92	36.72	86.75

Table 1. Recall Percentage Correct for Non-PRM Method.

Image	F_end	F_corr	F_corn	F_bra
im00	100.0	98.70	100.0	100.0
im17	96.97	98.84	100.0	100.0
im18	100.0	99.07	99.63	100.0
im19	98.64	98.95	100.0	95.87
im22	97.32	98.09	100.0	98.92

Table 2. Recall Percentage Correct for PRM Method.

Image	F_endpt	F_corr	F_corn	F_bra
im00	100.0	87.61	85.48	99.53
im17	97.32	82.61	68.85	100.0
im18	99.55	84.91	84.49	80.99
im19	100.0	84.91	81.34	90.91
im22	97.73	91.85	46.88	97.59

Table 3. Recall Percentage Correct for Decision Tree Method.

Image	F_end	F_corr	F_corn	F_bra
im00	43.36	100.0	42.87	43.25
im17	35.10	100.0	28.64	41.52
im18	60.96	100.0	51.66	35.01
im19	53.94	100.0	63.23	34.92
im22	34.73	100.0	29.20	45.93

Table 4. Precision Percentage Correct for Non-PRM Method.

Image	F_end	F_corr	F_corn	F_bra
im00	35.82	100.0	42.59	50.11
im17	55.39	100.0	38.31	39.45
im18	69.89	100.0	47.83	57.56
im19	68.55	100.0	48.57	58.27
im22	31.06	100.0	28.20	44.50

Table 5. Precision Percentage Correct for PRM Method.

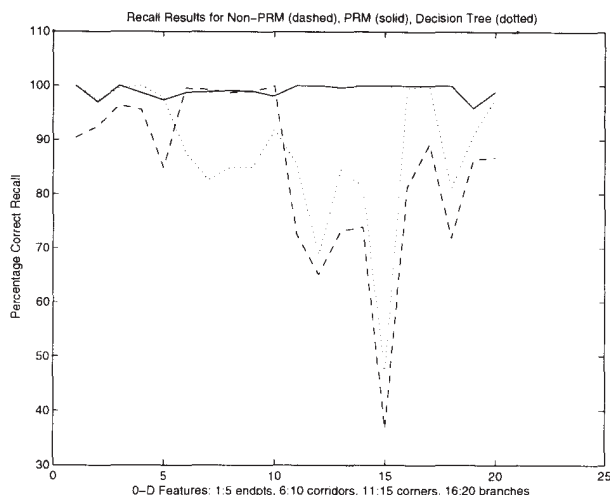


Figure 21: Percentage Correct Recall.

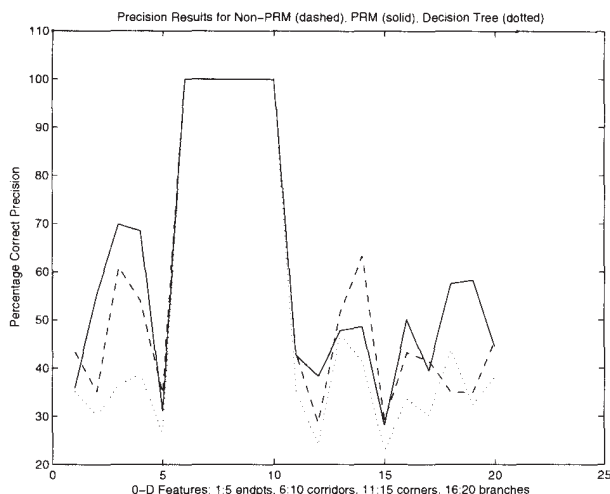


Figure 22: Percentage Correct Precision.

Image	F_endpt	F_corr	F_corn	F_bra
im00	35.17	100.0	35.35	33.71
im17	30.17	100.0	24.15	30.14
im18	36.42	100.0	47.14	43.79
im19	38.79	100.0	41.21	32.43
im22	26.35	100.0	22.96	38.18

Table 6. Precision Percentage Correct for Decision Tree Method.

As can be seen from the performance data in Figures 21 and 22, the static PRM feature classification method works very well with regard to recall, but requires a post-processing step to ensure good precision.

4 Conclusions and Future Work

We have demonstrated that the use of static Pseudo Range Maps yields high percentage detection of point features in engineering drawing images. We

are currently investigating:

- *Agent motion during feature recognition.* First, 0-dimensional features will be analyzed, then 1-dimensional, and finally, 2-dimensional.
- *Learning characteristics of the line drawing data.* Let the trajectory of an agent as it explores a drawing be considered an element. Then one approach to finding related parts of the drawing, or parts that are similar to a known model, is to use the affinity graph method[6] to cluster the elements into similar classes.

We are currently pursuing these two lines of development. In addition, we intend to expand the domain of application to other datasets, including: map features, handwritten and other documents.

Acknowledgments

This work was supported in part by ARO grant number DAAD19-01-1-0013.

References

- [1] F. Aurenhammer. Voronoi Diagrams – A Survey of a Fundamental Geometric Structure. *ACM Computing Surveys*, 23:345–405, 1991.
- [2] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localisation and Map Building (SLAM) Problem. *IEEE-T Robotics and Automation*, 17(3):229–241, June 2001.
- [3] Dov Dori. Dimensioning Analysis toward automatic understanding of engineering drawings. *Communications of the ACM*, 35(10):92–103, October 1992.
- [4] D.S.Guru, B.H.Shekar and P.Nagabhushan. A simple and robust line detection algorithm based on eigenvalue analysis. *Pattern Recognition Letters*, 25(1):1–13, 2004.
- [5] Feng Su, Jiqiang Song, Chiew-Lan Tai and Shijie Cai. Dimension Recognition and Geometry Reconstruction in Vectorization of Engineering Drawings. In *Computer Vision and Pattern Recognition, 2001 (CVPR 2001)*, volume 1, pages 682–688, 2001.
- [6] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2003.
- [7] H. Choset and K. Lynch and S. Hutchinson and G. Kantor and W. Burgard and L. Kavraki and S. Thrun. *Principles of Robot Motion*. MIT Press, Cambridge, MA, 2005.

- [8] Thomas C. Henderson. Explicit and Persistent Knowledge in Engineering Drawing Analysis. Research Report UUCS-03-018, School of Computing, University of Utah, Salt Lake City, Utah, 2003.
- [9] Thomas C. Henderson and Lavanya Swaminathan. Form Analysis with the Nondeterministic Agent System (NDAS). In *Proceedings of 2003 Symposium on Document Image Understanding Technology*, pages 253–258, Greenbelt, VA, April 2003.
- [10] Thomas C. Henderson and Lavanya Swaminathan. NDAS: The Nondeterministic Agent System for Engineering Drawing Analysis. In *Proceedings of International Conference on Integration of Knowledge Intensive Multi-agent System*, pages 512–516, October 2003.
- [11] Thomas C. Henderson and Lavanya Swaminathan. Symbolic Pruning in a Structural Approach to Engineering Drawing Analysis. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 180–184, August 2003.
- [12] Jiqiang Song, Min Cai, Michael R. Lyu and Shijie Cai. A New Approach for Line Recognition in Large-size Images Using Hough Transform. In *International conference on pattern recognition (ICPR'02)*, volume 1, page 10033, 2002.
- [13] L.E. Kavvaki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE-T Robotics and Automation*, 12(4):566–580, June 1996.
- [14] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
- [15] Martin Levine. *Vision in Man and Machine*. McGraw-Hill Book Company, New York, 1985.
- [16] M.Mattavelli, V.Noel, and E.Amaldi. A New Approach for Fast Line Detection Based on Combinatorial Optimization. In *10th International Conference on Image Analysis and Processing*, volume 1, page 168, 1999.
- [17] M. Montemerlo and S. Thrun. Real Time Data Association for FastSLAM. In *Proceedings of International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.
- [18] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*. Addison-Wesley, Reading, MA, 1992.
- [19] S. H. Joseph, T. P. Pridmore, and M. E. Dunn. *Toward the automatic interpretation of mechanical engineering drawings*. Computer Vision and Image Processing(A. Bartlett, Ed.), New York: Kogan Page, 1989.
- [20] Jiqiang Song, Michael R. Lyu, and Shijie Cai. Effective Multiresolution Arc Segmentation: Algorithms and Performance Evaluation. *IEEE-T Pattern Analysis and Machine Intelligence*, 26(11):1491–1506, November 2004.
- [21] Stuart Russell and Peter Norvig. *Artificial Intelligence*. Prentice Hall, Upper Saddle River, New Jersey, 2003.
- [22] Lavanya Swaminathan. Agent-Based Engineering Drawing Analysis. Master's thesis, University of Utah, Salt Lake City, Utah, May 2003.
- [23] K. Tombre and D. Antoine. Analysis of Technical Documents using A Priori Knowledge. In *IAPR Workshop Syntactic and Structural Pattern Recognition, Pont--Mousson, France*, pages 178–189, 1988.
- [24] K. Tombre and Dov Dori. Interpretation of engineering drawings. In *Handbook of Character Recognition and Document Image Analysis*, pages 457–484, 1997.
- [25] Y.Yu, A.Samal, and S.C.Seth. A System for Recognizing a Large Class of Engineering Drawings. *IEEE Transactions on PAMI*, 19(8):868–890, 1997.

Perceptual Organization in Semantic Role Labeling

Prateek Sarkar

Eric Saund

Perceptual Document Analysis
Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304
{psarkar,saund}@parc.com

October 24, 2005

Abstract

Documents are produced for the purpose of human interpretation. Human perceptual factors have played an important role in the design of documents — from the development of glyphs and scripts to the layout of visual components. OCR technology allows recovery of textual content from images of text but does not recover visual information encoded in layout. We explore the role of perceptual organization in the interpretation of documents and related computational challenges. A useful application area is semantic role labeling: the problem of assigning semantic roles to visual or textual structures in documents. We present a generic A^* search formulation that has been applied to reduce search times by orders of magnitude in a semantic role labeling problem.

1 What is Semantic Role Labeling

Documents, as records of human communication, are built up of visual components that play distinct roles in expressing the meaning of a document. While these roles are various, and visual artists are still discovering ways of communicating through images, major categories of semantic roles can be enumerated in the context of document processing. For example, business invoices contain visual components that communicate identity (address block, company logo), object relations (item-price relations, signator-signature pairing), special attention (highlighting, circling). Parsing a document image into its semantically significant components is the problem of Semantic Role Labeling. We develop automatic and

semi-automatic approaches to Semantic Role Labeling especially for large volume image understanding and indexing projects.

2 What is Perceptual Organization

The science of Perceptual Organization, pioneered by Wertheimer, Koffka, and Köhler, studies patterns and principles by which humans organize visual stimuli into perceptual forms in ways that are quite universal across both peoples and situations. This science advocates the Gestalt principles of perceptual grouping, and theories of figure-ground separation. The common Gestalt principles of grouping by proximity, similarity, continuity, closure, symmetry, surroundedness apply to both design and analysis of visual documents. In all scripts, proximity informs the grouping of glyphs into words. Curvilinear alignment, and feature similarity are vital in perceiving more complex textual blocks as visual objects rather than just a collection of glyphs. Enclosures and separators are frequently used to guide perceptual and semantic grouping.

3 Perceptual Organization in document image interpretation

The interpretation of complex document layout structure has remained a research subject. Unlike the parsing of one-dimensional text, image parsing even in very structured domains is algorithmically difficult, because of the lack of a natural ordering in

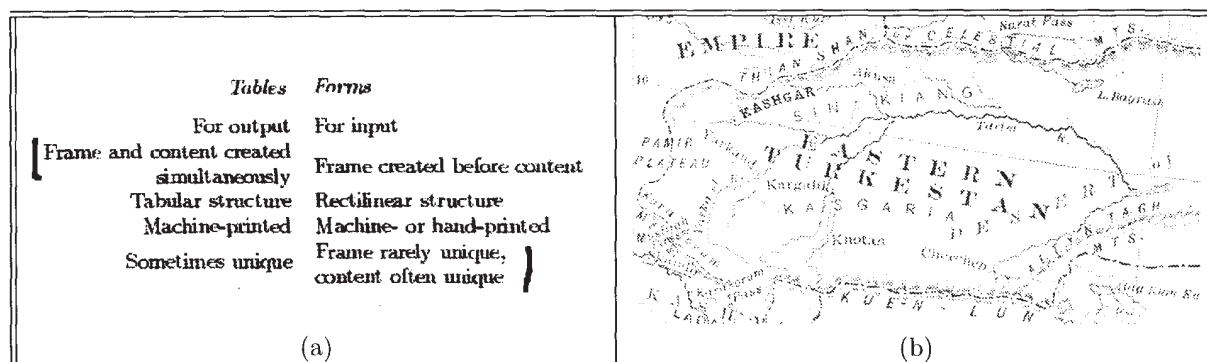


Figure 2: Examples of perceptual grouping in images. (a) The vertical alignment of text line groups is vital evidence that the marked text lines should be grouped into a single table cell. (b) The vertical alignment of text line groups is vital evidence that the marked text lines should be grouped into a single table cell. Parsing under such grouping cues is of exponential complexity.

without further subdivision. The problem is to assign labels $(y_1, y_2, \dots, y_n)$ to a collection of observation atoms $(x_1, x_2, \dots, x_n)$. In a typical pattern recognition formulation, the assignment is chosen by maximizing some objective function $f(x_1 \dots x_n, y_1 \dots y_n)$ over all possible labelings. If each atom can take on one of C label-values, the label assignment problem is a search through a the space of C^n label assignments. Restricting the nature of the objective function avoids the exponential explosion of the search space. For example, if the objective function is factorizable as the product of functions of the form $f_i(x_i, y_i)$, our problem reduces to labeling n observation independently, and the search complexity is nC . But often factorization of the objective function, if present, is not as simple. In addition to intrinsic features of x_i that influence the labeling y_i , we may have:

- *Factors coupling two or more labels* of the form $h(y_i, y_j, \dots)$. Linguistic constraints in OCR are in this category. A Gestalt preference for repeated structure can be modeled with such factors. In parsing a book page, at most one of all the text-word atoms may take on the label "page-number".
- *Factors coupling two or more observations* of the form $g(x_i, x_j, \dots)$. These arise when the atomic observations have constraints on relative size, orientation, or color change (lighting variations). Style consistency [SN05, VN05] models require such factors to model shared fonts, color, or other characteristics of text.

- *More complex factors* such as direct dependence between labels and features of different atoms. For example, to label a dot in an image as a "bullet" induces preference for indentation on adjoining atoms, and alignment with other adjoining "bullet" atoms. Both indentation and alignment are features of groups of atoms.

Most perceptual organization principles induce such complex dependencies among atomic constituents of images. Various techniques have been developed in the graphical models literature for solving the joint labeling problem (e.g., loopy belief propagation, sampling, variational methods, and hybrid techniques.) However these techniques do not offer guarantees of joint optimality. We have developed an A^* search framework, where we find easily factorizable upperbounds for complex objective functions. In particular, when the upper-bound is expressed as a product of n functions $\bar{f}(x_i, y_i)$ the bound can be maximized in linear time. It is then used to guide the search in a best first way, thereby guaranteeing that the optimal solution will always be found. While the worst case complexity of this method is still exponential, the low average complexity can enable the solution of problems that are otherwise intractable.

6 Application: Indexing engineering drawings

A typical repository-indexing or metadata-extraction application requires scanned images of drawings to be indexed by a number of fields such as drawing number, drawing title, date of creation, author, company

PALATINE, ILLINOIS RELOCATION OF STORE ROOMS AT NO. 14, BULLID		
DRAWING TITLE MICROLITE / SIEMENS CONTROL DIAGRAMS		
DATE 7/25/03	STORE NO. OTT54	DRAWING NO. E2.01B
DRAWN BY: RG	SCALE: AS NOTED	
CADD PLOT: PERMIT SET	RELEASED TO CONSTRUCTION	
REVIEWED BY: GR		50 OF 63 DWGS.

The diagram shows a simplified version of the title block above, with labels pointing to specific fields: 'Drawing Title Indicator' points to the top-left corner, 'Drawing Title' points to the title text, 'Drawing Number Indicator' points to the bottom-left corner, and 'Drawing Number' points to the drawing number text.

Figure 3: An example of a title-block image fragment showing the four relevant fields to be extracted, and the competing irrelevant text groups in the neighborhood.

logo, site name, and scale. In our pilot technology development project we focused on extracting drawing numbers and drawing titles. Figure 3 shows an example of an image fragment that contains the drawing number and drawing title. While most drawings include drawing numbers and titles, they are printed at various locations, and in various formats and styles (Figure 4.)

Our objective is to locate these fields in the images without restricting input to a small number of layouts and styles. To this end, an input image is pre-processed to generate a number of candidates for the drawing number and title fields. The first step is perceptual grouping of foreground pixels into potential text fields. Connected components are first grouped to obtain word-like elements, which are further grouped into text-lines and multi-line text objects. Proximity, size similarity, alignment, and surroundedness (a text group does not straddle line enclosures) are used in the grouping process. In the typical engineering drawings in our dataset, this results in 30-200 text groups (see Figure 3.) Each resulting

SHEET TITLE: THERMOPLASTIC ROOFING DETAILS (BASE BID OPTION "A")		PALATINE, ILLINOIS RELOCATION OF STORE ROOMS AT NO. 14, BULLID DRAWING TITLE MICROLITE / SIEMENS CONTROL DIAGRAMS	
A3a		E2.01B	

SHEET TITLE: ROOF FRAMING PLAN NOTES		KEY PLAN SHEET TITLE FLOOR PLAN - PLUMBING AREA 1	
A-6		PP111	

SHEET TITLE: PARTIAL UPPER LEVEL FLOOR PLAN		SHEET TITLE: CAR WASH EXTERIOR POWER PLAN	
MD2.2		10.7.2B	

Figure 4: Examples of index fields in snippets of engineering drawings: drawing number and drawing title.

group in the image is sent to an external OCR engine, and any deciphered text is recorded.

We assume that each relevant field is grouped into a single unit, and not merged with other fields. In other words, each group found by our grouping algorithm, along with its OCR text, is treated as an atom for labeling. Our problem then is to label at most one of these groups as *drawing title* (*DT*), and at most one as *drawing number* (*DN*). To aid in the process, we also attempt to find a *drawing title indicator* (*DTI*) (e.g., "Sheet Title",) and a *drawing number indicator* (*DNI*) (e.g., "Sheet Number".) All remaining fields are to be labeled *irrelevant* (*IRR*). We thus have $C = 5$ classes, $n \approx 50$ (on average) objects to label, and a search space of $\approx 50^5$ competing label-hypotheses.

A label-hypothesis is represented as $(l_{dn}, l_{dni}, l_{dt}, l_{dti})$ meaning that the group l_{dn} is labeled *DN*, group l_{dni} is labeled *DNI*, and so on. $l_{(.)}$ can take on values $1 \dots n$ or 0, the latter indicating that the field is missing. Our problem is to find a label hypothesis that maximizes some score. The score (likelihood) of a label-hypothesis is assumed to

be of the following form:

$$L(l_{dn}, l_{dni}, l_{dt}, l_{dti}) \propto f_{dn}(l_{dn}) \cdot f_{dni}(l_{dni}) \cdot f_{dt}(l_{dt}) \cdot f_{dti}(l_{dti}) \cdot u(l_{dn}, l_{dni}) \cdot v(l_{dt}, l_{dti}) \cdot w(l_{dn}, l_{dt}) \quad (1)$$

The formula contains two kinds of functions:

- *Unary functions* f_{dn} , f_{dni} , f_{dt} , f_{dti} , which compute a goodness score (likelihood) that a group is a *DN*, *DNI*, *DT*, *DTI* respectively based on measurements that are properties of the group alone. Examples of such properties are *size*, *aspect ratio*, *position on page*, *regularity of size of group components*, *average stroke width*, *closeness of OCR text to expected textual content*, etc. In our pilot experiment we use only the last mentioned feature.
- *Pairwise functions* u , v , w , which compute scores on pairs of label candidates. These functions may thus depend on *relative location* and *relative alignment* of two groups, whether the two groups are *neighbors in a Delaunay triangulation*, whether they *share an enclosure* demarkated by line or white-space separators, etc. Pairwise functions can also be used to model mutual exclusivity constraints such as $(l_{dn} \neq l_{dt})$, *i.e.*, the same group cannot be both a drawing number and a drawing title. In our pilot experiments only the relative location of two labeled groups and mutual exclusivity were used in the pairwise functions.

In general, functions involving three or more groups can also be included. The overall score is proportional to the product of all such functions, and the winning label-hypothesis is the one that maximizes this product. If the objective function includes only unary functions, each component unary function can be maximized independently to arrive at the result. The pairwise (and higher order) terms, introduce complexity because such independent term-by-term maximization is no longer useful. Nevertheless, the pairwise functions are important for distinguishing, for example, the drawing number from the many other alpha numeric fields that appear on a page. So we follow the A^* solution suggested in the previous section, by replacing the pairwise terms by fixed upper bound to obtain an overall upperbound score of the form:

$$\hat{L}(l_{dn}, l_{dni}, l_{dt}, l_{dti}) \propto f_{dn}(l_{dn}) \cdot f_{dni}(l_{dni}) \cdot f_{dt}(l_{dt}) \cdot f_{dti}(l_{dti}) \cdot \hat{u}\hat{v}\hat{w} \quad (2)$$

This upper bound is now factorizable, and can be used in our A^* search. We can sort all n candidate text groups in an image in decreasing order of $f_{dn}()$, $f_{dni}()$, $f_{dt}()$, $f_{dti}()$, respectively. The resulting four lists contain candidates groups, ordered from best to worst according to our upper bound, for the *DN*, *DNI*, *DT*, *DTI* labels respectively. The top candidates from each list are combined to form our most-promising label-hypothesis.

For the search algorithm we construct a priority queue, designed such that the head of the queue is always the most-promising label-hypothesis yet to be evaluated. A label-hypothesis is evaluated by computing its true score according to formula 1, comparing it to the running best hypothesis (according to true score). The search is over when the true score of the running best hypothesis exceeds the upper bound score of the next most promising candidate waiting at the head of the queue.

Of course, the queue does not have to be populated with all n^4 label candidates at start. It can be grown dynamically as the search proceeds. Whenever we pop the head of the queue, we also insert four successors of the popped label-hypothesis. A successor of a label-hypothesis can be obtained by picking any element of the hypothesis, say the index of the *DN* label $- l_{dn}$, and replacing it with the next most promising *DN* index, easily obtained by moving down the $f_{dn}()$ sorted list by a notch.

In practice, the queue will seldom process all of n^4 possible label hypothesis, because with a good upper bound, the top label-hypothesis will be found among a few most promising candidates. In our experiments we noted that only of the order of a hundred label-hypothesis candidates were being explored among a potential n^4 candidates, where n was typically between 50 and 200. If the queue for an image grows much larger, it indicates extremely weak discriminating evidence from the unary functions. In such cases our algorithm has a high probability of being in error, even if we let it run till the end, and it may be useful to simply *reject* this input as undecipherable.

This is work in progress. In initial experiments on approximately 1000 test images, the drawing number field was correctly identified about 80% of the time. Observed errors derive from the following major causes: failures of the grouping algorithm to generate correct atomic units of text for labeling; OCR errors; errors in the unary functions to identify the textual properties of target labels.

7 Concluding remarks

Metadata tagging of document content is complementary to brute-force search over unstructured text. With the rise of XML standards for representing metadata in electronic databases, this function is gaining increased societal as well as commercial interest. Discovery of atomic content items, and the tagging of their semantic roles, is a key aspect of this problem.

We have outlined a formal approach to Semantic Role Labeling which decomposes the problem into construction of atomic units using perceptual organization techniques, and subsequent labeling of these units using unary and multiway evidence. We have demonstrated an A* approach to managing inherently exponential search, in cases where the atomic units are fixed. A greater research challenge arises when the atomic units cannot be computed with confidence, but instead multiple candidate parsings must be entertained.

This Semantic Role Labeling approach to document content tagging is amenable to other document recognition data sets and we are interested in expanding the set of domains to which it may be applied.

References

- [MV98] E. Miller and P. Viola. Ambiguity and constraint in mathematical expression recognition. In *Proceedings of AAAI-98*, pages 784–791, 1998.
- [NKS93] G. Nagy, M. Krishnamoorthy, S. Seth, and M. Viswanathan. Syntactic segmentation and labeling of digitized pages from technical journals. *IEEE Trans. PAMI*, 15(7):737–747, 1993.
- [Sar02] P. Sarkar. An iterative algorithm for optimal style conscious field classification. In *Proceedings of the 16th ICPR*, 2002.
- [SM04] E. Saund and J. Mahoney. Perceptual support of diagram creation and editing. In *International Conference on the Theory and Applications of Diagrams*, 2004.
- [SN05] P. Sarkar and G. Nagy. Style consistent classification of isogenous patterns. *IEEE Trans. PAMI*, 27(1):88–98, January 2005.
- [VN05] S. Veeramachaneni and G. Nagy. Style context with second order statistics. *IEEE Trans. PAMI*, 27(1):14–22, January 2005.
- [Wer38] Max Wertheimer. *Laws of Organization in Perceptual Forms (Translation from 1923 German article by W. Ellis)*, pages 71–88. Routledge & Kegan Paul, London, 1938.

Pictographic Recognition Technology Applied to Distinctive Characteristics of Handwritten Arabic Text

Mark A Walch

The Gannon Technologies Group
1000 North Payne Street
Alexandria, Virginia 22314
(703) 373-1962
mwaltch@gannontech.com

Donald T Gantz, PhD

George Mason University
4400 University Drive
Fairfax Virginia 22030
(703) 993-1511
dgantz@gmu.edu

Abstract

Pictographic Recognition technology is a Graph-Theory-based method for locating specific words or groups of words within handwritten and machine printed document collections. This technique converts written and printed forms into mathematical graphs and draws upon key properties of graphs such as topology and geometric features to locate graphs of interest based upon specified search terms. In its initial implementation, Pictographic Recognition technology has functioned as a search tool by identifying individual characters in strings of handwritten script—both English and Arabic. Arabic, however, through its internal segmentation resulting from intra-word gaps, presents the opportunity to treat groups of characters as distinctive objects that can be recognized directly without the need to isolate individual characters. This paper focuses on current research toward using Pictographic Recognition techniques to identify Arabic word segments.

1 Introduction

Handwritten Arabic poses unique challenges for recognition technologies. First, Arabic is written in a cursive form so there is no clear separation between characters within words. Second, Arabic characters change their form depending on their word position: initial, middle, final or standalone. Third, Arabic words incorporate external characteristics such as diacritical markings. Fourth, Arabic writers take broad “latitude” with handwritten forms by stacking characters as well as omitting certain characters. This latter characteristic creates considerable dissimilarities between handwritten and machine printed forms of Arabic.

Of the items cited above, it is Arabic’s unconstrained cursive form that arguably creates the most difficult problem for recognition technologies. In conventional recognition systems such as those used for hand or machine printed Latin characters, it is possible to

“segment” words into individual characters before actual recognition begins. While it is a major impediment to machine recognition, segmentation comes quite naturally to human readers because humans appear to perform segmentation in conjunction with recognition rather than as separate sequential tasks. That is, given that a human reader has a prior knowledge of accepted character forms coupled with an understanding of context, readers identify these forms directly within a cursive word, rather than isolating them first and recognizing them later.

Aside from Arabic’s cursive form, the stacking and omission of characters pose the next greatest challenge to recognition technologies. Stacked characters work contrary to the expectation of sequential horizontal placement of letters and omitted letters negate the certainty that characters will always be found in expected locations. What human readers can readily overcome constitute significant obstacles to algorithmic solutions. These obstacles are particularly effective in barring technologies developed to recognize machine print from making a smooth transition to handwriting—in Arabic as well as other languages.

Pictographic Recognition technology offers a viable approach toward recognizing cursive forms of Arabic and English as well as “pictoform” languages such as Chinese. Pictographic Recognition shares common roots with Optical Character Recognition, but it is a fundamentally different approach. Pictographic Recognition looks for the “Pictographic Signature” of words in their native form. In this context, Pictographic Recognition is somewhat similar to Optical Word Recognition (“OWR”) but differs significantly from OWR in its ability to evaluate the actual characters and character groupings that comprise the unknown words.

To date, Pictographic Recognition has been implemented as a Prototype application that performs word searches based on pictographic signatures. These

signatures consist of graph-based topologies embedded in written or printed words. These topologies usually represent characters, but they may also reflect small character groups or parts of characters such as loops, cusps and line crossings. Current languages where Pictographic Recognition has been implemented include machine printed, hand printed and cursive English as well as machine printed and handwritten Arabic. A more detailed discussion of the basic concepts can be found in the white paper entitled: "Pictographic Matching: A Graph-based Approach Towards a Language Independent Document Exploitation Platform" which will be made available by the Authors upon request.

Aside from the focus on graphs as its foundation, another aspect of Pictographic Recognition that is distinctive from OCR is the format of output data. OCR typically converts images of documents into text. Pictographic Recognition converts images into a specially structured format that retains much of the information extracted from the original image including, candidate character possibilities as well as data related to figure topology and geometry. These data are retained so they can be applied during the actual searching process, so every item processed is evaluated directly against a search term of interest.

The interim product from Pictographic Recognition takes the form of a "Results Matrix" rather than the text results characteristic of OCR. The Results Matrix maintains considerable data about each word and character in a document collection including word and character alternatives, attendant confidence scores and related geometric information. The manner in which this information is stored can be defined as *Planned Indeterminacy*. That is, unlike OCR which "forces" its results into text output, Pictographic Recognition, refrains from word determination until an actual search is performed. At the time of the search, the full body of information stored in the Results Matrix is brought to bear to match the search term with words in the document collection being searched. In this way, the search process is "empowered" by using all the relevant data available at exactly the time it is needed. The value of these data is realized by a special search engine that incorporates Dynamic Programming techniques to find the best fit between a search term of interest and detailed word data rendered from document collections.

Pictographic Recognition starts with images of documents. The images are parsed into individual objects representing words and characters within these words. Each of the parsed pieces is converted into a graph containing a wealth of physical measurements and mathematical descriptors. These measurements become the basis for a classification scheme that decodes the graph into its actual identity as a single or a

group of letters. Because of the abundance of measures available, classification can be sharpened by focusing on the specific set of measures that separates each particular letter's graph from others. As items are classified, they are loaded into a Results Matrix that retains information concerning alternative identities as well as geometric information. As searching is performed, the search term is compared against the contents of the matrix to find the best mapping of the search terms of interest with the matrix elements built from document collections.

The previous paragraphs have presented a "primer" on Pictographic Recognition. The remainder of this paper discusses the Authors' most recent research efforts related to this topic. These efforts focus on a specific aspect of Pictographic Recognition that "bundles" both segmentation and recognition activities into a single action that can be very efficient and effective in recognizing Arabic script. In particular, the discussion that follows will focus on the ability of Pictographic Recognition to treat connected multiple character strings within Arabic words as objects that can be recognized individually. At the heart of this recognition process is the ability to encode both the features and topology of a character into a form that can be used for rapid identification. The technique herein described is strongly rooted in an algorithmic approach, based on Graph Theory, which treats handwriting and printed text as mathematical graphs. The principles that have been successfully applied to individual characters will now be applied to connected character strings.

2 Conceptual Framework

Graph Theory is a branch of Mathematics that focuses on representing relationships as line diagrams containing nodal points and the linkages among these points. In graph terminology, the nodes are referenced as "vertices" and the links as "edges". Graphs are a handy and effective way to represent written language since graphs can be created directly from the pen strokes used to compose letters and words. Words, characters and numerals take their form as graphs written on paper assuming distinctive shapes representing the letters of the alphabet as well as numerals and punctuation. The edges and vertices of these written graphs connect and cross and are straight or curved. Graphs accurately capture the essence of written language since they contain both the topological structure and geometric information sufficient to replicate complete written forms. Graphs are the foundation of written language and a search methodology that focuses on the graphs offers the potential to permit searching that encompasses several languages.

Graphs contain all the information extracted from writing condensed into a concise mathematical format

that is highly computable. Within graphs, this information takes two forms. The first form is the graph's *topology* that can be seen in the connectivity among the major graph components. The way pen strokes are crossed and connected is the framework for graph topology. Topology is the structure of the graph. The second form of information contained in graphs is the *geometry* of the graph. Geometry is expressed in terms of distances, angles and characteristics of graph components. The depth or shallowness of a curve, the distance between line crossings or the sharpness of angles are all examples of graph geometry. Geometry characterizes the *shape* of the graph. Collectively, topology and geometry account for the structure and shape of graphs. The topology and geometry of graphs are also quite computable. That is, they can be expressed as data to support computer-based processes that can be performed on graphs such as indexing and searching.

The foundation of Pictographic Recognition is an algorithm that describes graph topology as a numeric code. Any two graphs with the same structure will generate the same code. Any two graphs generating the same code are said to be *isomorphic*. Linked to the graph topology and its attendant numeric code is graph geometry. Graph geometry can also be expressed as a feature vector used to compare graphs. A feature vector is a multi-dimensional expression of the multitude of measurements that can be extracted from graphs. When the topology of graphs is coupled with feature vectors as a combined data structure, the computability of this data structure offers a very effective way to use graphs as indices for characters and words contained in the images of documents.

The computational engine for Pictographic Recognition is an automated method for identifying and matching isomorphic graphs and storing these graphs in a database. Graphs are isomorphic when they are structurally identical—with the same number of edges and vertices connected in the same way—although they may appear to be different. Two graphs are considered isomorphic when there exists a one-to-one correspondence between their internal structures. That is, they have the same number of edges and vertices connected to form exactly the same topology.

Figure 1 illustrates this point. Although the graphs appear to be quite different, they are structurally identical: isomorphic. They appear different because their geometry is different. The topology is the same, but the angles and distances are different.

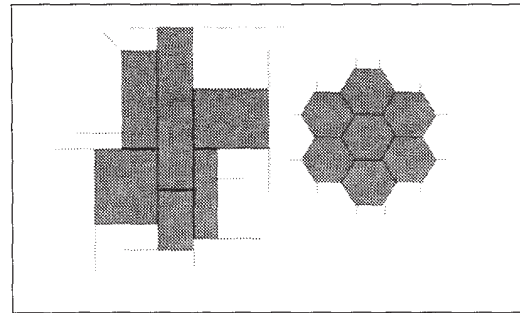


Figure 1: Illustration of two Isomorphic Graphs with different features.

Figure 2 illustrates the concept of Isomorphic Graphs as it applies to written characters. In this figure, each row shows three instances of the letter “a” written as the same isomorphic graph class. The numbers on the left hand side of the figure are the code names derived by the Pictographic Recognition process for each class of graph. Graph classes reflect graph topology. The class labeled “2;192” consists of graphs built from a loop and one edge. Class “4;112” contains graphs with 2 edges only. These typically characterize the letter “u” and the letter “a” written to be open at the top. Class “4;98.0.64” encompasses characters with a leading edge, an enclosed area—in graph terminology this is called a “face” --and a trailing edge.

2;192	
4;112.0	
4;98.0.64	

Figure 2: Sample letters “a” for three different graph isomorphic classes.

Figure 3 shows how graph isomorphic classes transcend individual letters. This figure presents letter “a” and “e” for isomorphic graphs classes coded “4;112.0” and “4;98.0.64”.

	a	e
4;112.0		
4;98.0.64		

Figure 3: Example of letters “a” and “e” sharing same isomorphic graph.

Graph isomorphism is a critical concept underlying Pictographic Recognition. In fact, Pictographic Recognition employs a database that maps a collection of known graphs—such as the letters of the alphabet—against all possible unknown graphs such as those that could be extracted from imaged words. Building such a database is actually accomplished by generating all possible graphs up to a certain size (order), then testing each of these graphs to determine whether they contain the topologies of any known graphs. As these topologies are detected, a mapping is established to indicate the presence of a known character topology embedded in an unknown graph form. To construct such a database it is necessary to consider only planar graphs. Planar graphs are graphs that can only exist in a plane, such as lines on a page of paper.

In its current implementation, Pictographic Recognition focuses on locating and identifying individual characters within handwritten words. The present discussion looks at Pictographic Recognition applied in a different manner—as a means of identifying character groupings rather than individual characters. The nature of written Arabic offers the perfect opportunity to apply Pictographic Recognition to multi-character clusters.

Written Arabic is considered to be a “cursive” language. The structure of written Arabic is similar to cursive English with some significant exceptions. These exceptions include:

1. Certain characters, such as short vowels, are omitted from the handwritten form but included in the machine printed form.
2. Arabic characters change their form depending on the location within a word.
3. Rules govern which characters connect and which characters do not connect.
4. Arabic characters may be stacked vertically as well as placed in a horizontal sequence.

Figure 4 illustrates these properties by contrasting formal machine printed Arabic (above) with handwritten versions of the same words (below).

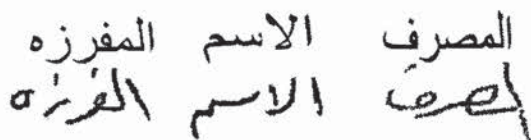


Figure 4: Comparison of machine printed and handwritten Arabic words.

Of the four distinctive Arabic properties herein enumerated, the present discussion focuses on the intrinsic segmentation of Arabic words into character

groups. These groups occur because certain Arabic characters always connect while others never connect. There is really no parallel of this concept in cursive English, except certain lead capitals that may not connect with following characters in a script word.

The “intra-word gaps” within Arabic words divide the word into smaller parts that can be treated by Pictographic Recognition in a way quite similar to the way individual characters are treated. For purposes of this paper, these segmented character strings are referenced as “word segments”.

These word segments consist of freestanding characters as well as small character groupings. The following table shows the character count distribution for word segments compiled from the Arabic Gigaword Corpus produced by the Linguistic Data Consortium. This corpus is a massive compilation of Arabic words obtained from numerous electronic media sources.

Table 1: Distribution of word segments by size from Arabic Gigaword Corpus.

Number of Characters in Segment	Number of Segments	Percent	Cumulative Percent
1	7,396,009	0.45.41	0.4541
2	4,608,726	0.28.30	0.7370
3	2,724,877	0.16.73	0.9043
4	1,077,822	0.0662	0.9705
5	334,438	0.0205	0.9910
6	104,649	0.0064	0.9975
7	17,520	0.0011	0.9985
8	23,679	0.0015	1.0000

In this table, the first column indicates the number of characters in the observed word segments. The second column shows the number of segments detected of a particular character length as shown in the first column. The third and forth columns show each row as a percentage and as a cumulative percentage of the total. This table shows that 99 percent of Arabic word segments contain 5 characters or less. And 90 percent of the Arabic word segments contain 3 characters or less. This observation may be in part an artifact of Arabic’s “triconsonantal” roots in which vowels are inserted into consonant-based templates.

The following table provides a similar analysis from a 1,000 page handwritten Arabic document collection obtained by the Authors. Note the striking similarities between the distributions of counts of characters within character sequences between this sample and the above selection from the Arabic Gigaword Corpus.

Table 2: Distribution of word segments by size from handwritten Arabic document collection

Number of Characters in Segment	Number of Segments	Percent	Cumulative Percent
1	29,746	0.5247	0.5247
2	14,622	0.2579	0.7826
3	7,756	0.1368	0.9194
4	3,004	0.0530	0.9724
5	1,187	0.0209	0.9934
6	357	0.0063	0.9997
7	14	0.0002	0.9999
8	5	0.0001	1.0000

The fact that Arabic word segments are built from so few characters helps to constrain their complexity and makes them ideal candidates to be treated and classified by Pictographic Recognition in a manner similar to individual letters. Most Arabic words are built from 2 to 4 word segments. Focusing on these segments and treating them as individual objects to be recognized converts the problem of recognizing multiple connected characters within a Arabic word into a matter of recognizing a few individually segmented forms. However, the difficulty related to recognition based on word segments is that segments—even for the same string of characters—will not take exactly the same form and, thus, will not necessarily generate isomorphic graphs even if they represent the same string of characters.

Making use of word segments requires a method that can focus on what is common among the segments while de-emphasizing the differences. In other words, the task becomes one of finding the “essence” of word segments. Pictographic Recognition offers a means of capturing this essence. Specifically, the essence can be found in forms embedded within word segments that are significantly large to capture the principal features of any individual segment and yet simple enough to be common to several word segments representing the same set of characters. To Pictographic Recognition the essence of word segments is found in embedded isomorphic graphs.

3 Embedded isomorphic forms

Written representations of character sequences are usually quite similar graphically and distinguished only by a few differences such as extra or omitted strokes. Because of these differences, the graphs these characters produce will be different in terms of the strict definition of graph isomorphism. That is, to be isomorphic, graphs must be identical. However, there will often exist an embedded graph that transcends multiple writing samples and is isomorphic in each

sample. That is, this graph becomes a common embedded form. Figure 5 shows two versions of the English word “is”. In their native form, these words have several differences principally related to the additional strokes along the bottom of the right hand sample and the extra lines in the “dot” above “s” on the left side. However, they do have a significant common embedding that is shown in the lower portion of the figure.

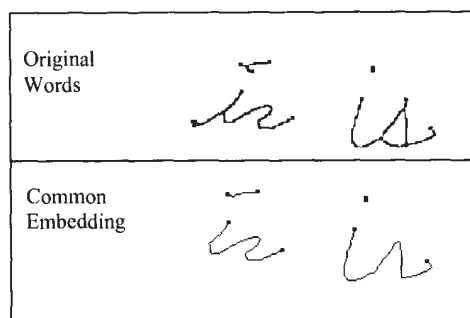


Figure 5: Comparison of original form and common embedded forms for English word: “is”.

Embedded forms offer an effective tool for capturing the essence of written character strings. Because of the great fluidity and variability of handwriting, the likelihood of two handwritten strings or words matching is small. However, these different strings could well have much in common in terms of embedded forms. Arabic word segments behave in a manner similar to cursive English word strings. As such, Arabic word segments contain embedded graphs that are common among the segments. Figures 6 and 7 provide some examples to show these embeddings.

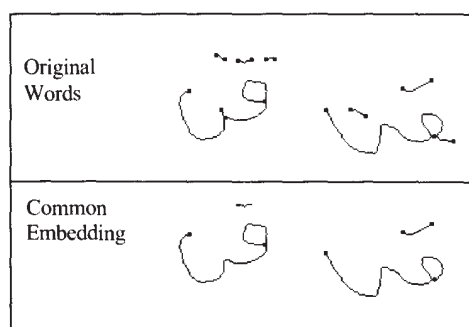


Figure 6: Comparison of original form and common embedded forms for Arabic word قن

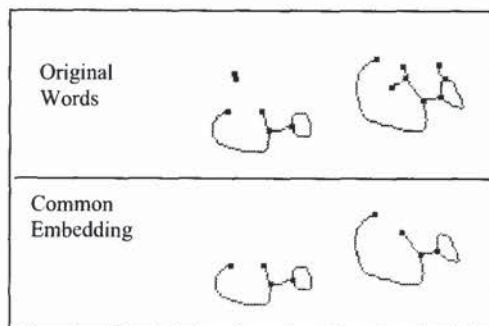


Figure 7: Comparison of original form and common embedded forms for Arabic word من

4 Conceptual Foundation for detecting Embedded Isomorphic Graphs

Treating written objects as graphs presents computational opportunities to detect common embedded forms. Graphs existing in one plane, such as graphs produced from written forms on paper are considered to be “planar graphs”. As such, planar graphs are a subset of all possible graphs. Furthermore, within the collection of planar graphs, a database can be built considering all the non-isomorphic forms of such graphs. The order of a graph indicates the number of vertices (or nodes) in the graph. The following table shows the number of non-isomorphic planar graphs up to order eleven.

Table 3: Distribution of isomorphic graphs up to order 11.

2	1
3	2
4	6
5	20
6	99
7	646
8	5,974
9	71,885
10	1,052,805
11	17,449,299

This table indicates there are 17,449,299 possible unique graphs up to and including graphs of Order 11. All graphs of order lower than 11 are also included in this amount. Thus, any possible graph up to this size can be pre-calculated and all instances of known character graphs can be mapped to these pre-calculated models. While 17 million is a large number, it is a very manageable and very computable number by present computing and storage standards. Review of graphs existing in language indicates Order 11 is a reasonable model both for English and Arabic.

An evaluation of character graphs contained in the reference data provided by the National Institute of Science and Technology shows that for written English

over 99.5 percent of characters as actually written create graphs of Order 11 or less. Similarly, analysis of a database of cursive English shows over 99.5 percent of all written characters produce graphs of less than or equal to Order 11. Experience indicates a higher percentage for Arabic because of innately simpler character forms. The fact that there are 17,449,299 theoretically possible graph forms that will contain virtually one hundred percent of all known graph forms provides the practical foundation for building the mapping database.

Creating this database presents another challenge, however. And creation is possible through an indexing technique that assigns a unique code to each unique graph isomorphism. That is, each graph with edges and vertices connected in a unique way can be assigned a unique numeric code to express its singularity of structure. Detailed discussion of how this code is derived is beyond the scope of this paper, but it involves arranging a graph’s adjacency matrix into a particular state. Adjacency matrices are tabular representations of the connections of vertices by edges within a graph. If graphs are isomorphic, it is possible to arrange their adjacency matrices to match exactly. So, all the adjacency matrices of all isomorphic graphs can be arranged into exactly the same order and this order can be “hashed” into a numeric value. Thus, a key can be built that will reflect each unique graph topology and this key can be used as access into a database of all possible graphs up to a certain size. The calculations to build this database are considerable, but they need only be performed once and stored as data. The best discussion of building isomorphic graph keys can be found in the body of the U.S. Patent Application entitled: “Systems and Methods for Source Language Pattern Matching” (Walch).

The concept of the isomorphic graph database was invented to store information regarding individual characters. However, the concept readily extends to character groups such as n-grams and word segments. In the case of Arabic word segments, each segment is treated as a graph and stored in the database. Traditional segmentation in the sense of horizontal “cutting” of the word segment is not necessary. Rather, the task becomes one of finding the “essence” of the word segment—the embedded form that characterizes an individual word segment and many other word segments resulting from the same character combinations.

Although graph isomorphism is a very important ingredient for capturing embedded graph forms, it must be stressed that isomorphism alone is insufficient for recognizing these written forms since similar topologies may result from completely different characters. The graph’s feature vector, as measured by angles, distances

and other descriptors, must also be considered in concert with the graph's topology. Within a set of graphs of the same topology (isomorphic graphs), these graphs can be classified based on their physical characteristics. In the case where these graphs represent the letters of the alphabet, classification separates the letters "a" from "e" even though the underlying graphs are isomorphic. Again, refer to Figure 3 which shows different letters constructed from isomorphic graphs.

Features to support graph classification cannot be mathematically divined. Rather, features for classification must be obtained by modeling from exemplars obtained from actual writing specimens. Thus, classifying Arabic word segments is a two-step process. The first step involves a modeling stage where a classification algorithm is trained to distinguish isomorphic graphs that represent different character combinations. The second step is to apply this classifier to unknown word segments to be recognized.

5 Modeling Arabic Word Segments

Building a model to classify Arabic word segments requires that a suitable training set of graphs be established. This set of graphs should contain the various graph forms generated for different character combinations as well as common embedded forms. The modeling set can be accomplished through the following steps.

1. Obtaining a set of Arabic writing samples
2. Isolating and labeling the individual word segments within these writing samples.
3. Converting these word segments into graphs.
4. Extracting the common embedded isomorphic graphs from the word segment graphs for each character combination.
5. Training the classifier using the word segment graphs and principal embedded graphs.

Steps 1 and 2 are pretty much straightforward. They involve collecting writing specimens and labeling the character sequences within these specimens. The most important aspect of these two steps is to obtain a collection of writings sufficiently broad to encompass character sequences likely to be of interest in future searching.

Two strategies will achieve this objective. The first strategy is to train on a listing of likely key words from an authority list. That is, given *a priori* knowledge of likely terms of interest, collect handwriting specimens containing these words. The second strategy is more complex. This strategy involves building a language model containing words that provide the broadest coverage of the language. One way the magnitude of this task can be approximated is to select a sampling of words from a large collection of words in common

usage, such as the Arabic Gigaword Corpus, and extract and compare word segments with a specific document collection.

A sampling of words with a frequency greater than 500 drawn from the Gigaword Corpus will yield a list of 3,172 words. These words will produce 1,777 unique word segments. Compared against a collection of handwritten Arabic language documents containing 32,507 words and 56,961 word segments, the segments extracted from the Gigaword Corpus match 87 percent of the words from the document collection. This suggests that simply building a training set from an attainable collection of commonly used words offers a viable approach for collecting a comprehensive set of Arabic word segment exemplars to support word-segment-based recognition.

Once the samples have been collected, the next step entails converting the word images into graphs. This step is followed by another step that involves extracting common embedded sub-graphs that consistently characterize the word segments. These embedded isomorphic graphs can be extracted directly using the isomorphic database concept. As a word part is encountered, it is converted into a graph and its isomorphic key is generated. This key serves as the basis for looking up the graph in the isomorphic database. Once the key lookup is complete, the isomorphic database returns a list of all embedded sub-graphs.

The isomorphic database will identify all embedded sub-graphs based solely on topology. As such, these graphs may not actually represent the corresponding components of word segments.

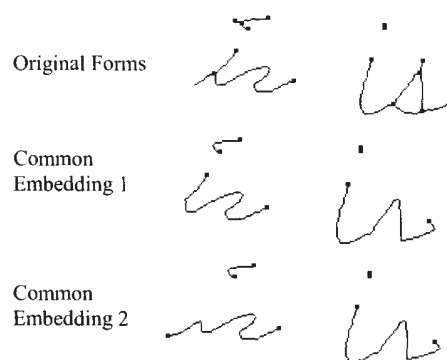


Figure 8: Comparison of Original and two embedded forms for handwritten word "is".

Figure 8 shows two possible combinations of embedded sub-graphs for handwritten samples of the English word "is". Whereas the original graphs are not isomorphic, the isomorphic database extraction method found two

pairs of common embedded graphs that were isomorphic to each other. However, the first set of embedded graphs provides a better match to the actual corresponding parts of the original words.

Finding the embedded isomorphic graphs that best match the original words can be accomplished by filtering techniques rooted in geometric features. That is, as embedded graphs are extracted, their geometric features can be compared to find those offering the closest match. These geometric features include lengths of graph segments, directions among graph components, location of "corners" and other shape-related features and the like. The result of this process is to establish "clusters" of similar embedded graphs for Arabic word segments. It should be noted that a segment may produce several different clusters based on the variations of writing used to generate the word segments.

6 Classifying Graphs

A graph feature vector consists of the wealth of measurements that can be obtained from graphs. These measurements include directions, distances and various descriptors. Directions are quantified as angles between graph components such as the angle from one vertex to another or the angle from the centroid of an edge to a vertex. Similarly, distances quantify the amount of space between graph components such as the number of pixels between two vertices. Graph descriptors include factors that articulate the shape of graphs. Two such measures are Bezier values, which provide a means for capturing the essence of curves in as few as four numbers and Bending Energy that is an indicator of curvature. Figure 9 shows three classes of features that are part of a character graph's feature vector.

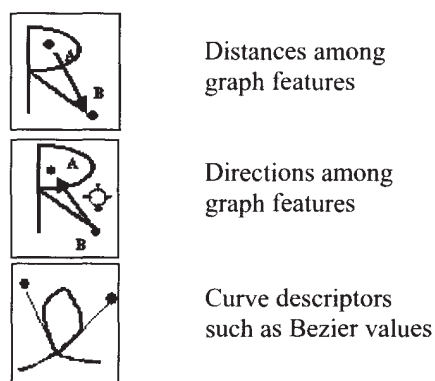


Figure 9: Three principal classes of features used for Pictographic classifying of graph forms.

When isomorphic graphs are aligned, a direct one-to-one correspondence between each geometric measurement comprising the feature vector can be

established. That is, when two graphs are identified through the isomorphic graph matching process, their feature vectors are also aligned and corresponding features can be compared in detail.

Figure 10 shows graph alignment between two separate embedded graphs from Arabic word segments for the Arabic word من. These segments were extracted from different words from different writers.

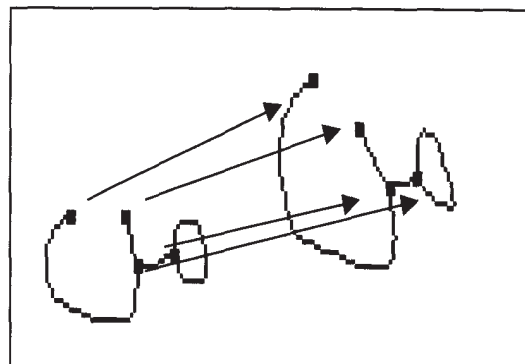


Figure 10: Alignment for two embedded forms for Arabic word من taken from different writing samples.

The isomorphic graph matching process ensures that the two versions of the Arabic word "من" are matched and aligned. Alignment means that all vertices are matched in corresponding pairs as indicated by the arrows in the figure. Given the point-to-point alignment achieved by isomorphic graph matching, graph feature vectors can be aligned to reflect the corresponding relationships between two graphs. The process of matching graphs and aligning the geometric features in the feature vector is a very robust technique for recognizing varying forms of characters and symbols when other techniques—principally derived for recognition of printed fonts—are much less forgiving of these variations in form. Once topologies are matched and feature vectors aligned, the graphs are ready to be modeled as individual objects that represent both single and multiple character strings.

Considerable data are available from graph-based feature vectors. Even a relatively simple graph can generate a vector with over a hundred measurements. This wealth of data is well suited for a modeling method utilizing Stepwise Discriminant Analysis. Discriminant Analysis is a powerful multivariate statistical technique that can be used to identify the subset of variables that best distinguishes one graph from another. Using Stepwise Discriminant Analysis, head-to-head comparisons can be made between aligned feature vectors to identify the specific set of measures that distinguishes one object from the other.

Modeling for ultimate classification through Discriminant Analysis involves considering each set of isomorphic graphs separately. Each set of graphs may represent different character strings but have the same underlying topology.. During this modeling process, a specific set of variables is isolated for each form of each character set within the isomorphism. This feature set constitutes the “Alphametric Kernel” for the word segment. The Alphametric Kernel is defined as the specific set of measures that will reliably distinguish that a particular word segment from all other segments represented by the same isomorphic graph.

Similarly, a Regression Tree Classifier can be used to isolate those features that distinguish one character string representation from another. The Classifier distills the abundance of measurements to a succinct set of critical measurements that distinguish one form from another. In either case, the result is a set of powerful variables labeled by the Authors as the Alphabetic Kernel.

Through the modeling process, Alphametric Kernels are generated for all forms of all word segments extracted from the training data. These kernels taken in concert with the specific isomorphism provide the basic data to decode unknown word segments into their appropriate text character strings.

7 Recognition of Arabic Word Segments

Recognition of Arabic word segments can be accomplished through a process that parallels the method used for modeling. This process is characterized by the following steps.

1. Extract objects likely to be word segments from handwritten documents.
2. Convert these segments into graphs.
3. Generate an isomorphic database key from the word segment graphs.
4. Use the key to find the graph in the isomorphic database.
5. Extract the significant embedded graphs.
6. Classify these graphs based on the training data.

The application of Pictographic Recognition to word segments differs slightly from its application to individual characters. The principal difference is that individual character identification requires a preliminary form of segmentation that breaks the word into small individual parts that can be assembled in different ways to capture the actual characters. In the case of the word segments, advantage is taken of the intra word gaps as the means of segmentation. In both cases, the output can be handled the same way.

When applied at the individual character level,

Pictographic Recognition distinguishes itself from OCR by retaining its results in a matrix that retains alternative candidate results. Since word segments will resolve to character sequences, results derived from Arabic word segments can be handled and stored in the same way.

8 Preliminary Recognition Results

A preliminary test was structured to measure the effectiveness of Pictographic Recognition in classifying individual segments within Arabic words. This test involved a training set consisting of 120 Arabic single paragraph writing samples each containing 109 words. Collectively, these words contained 247 word segments. Another set of 120 handwritten Arabic documents containing the same paragraph but from different authors was used as the testing set. In both cases, the documents were manually labeled at the word and character level to provide ground truth both for training and testing purposes.

The training documents were automatically segmented into words and word segments using the existing functionality of the current Pictographic Recognition software platform. The word segments were further processed automatically to identify common embedded isomorphic graphs. A threshold was set to extract only those embedded graphs that occupied more than 75 percent of the graph structure of the original word segment from which it was extracted. As embedded graphs were extracted, they were labeled by their appropriate isomorphic graphs. A Regression Tree Classifier (CART® “*Classification and Regression Tree*” by Salford Systems) was used to build classification trees for the different embedded isomorphic graph groups extracted. For each isomorphic graph group, the classification tree contained the salient variables and decision rules to distinguish the various character combinations associated with the graph. The number of unique character combinations within each isomorphic graph group ranged from as few as ten to over one hundred. This process is identical to the manner by which individual characters are also handled for classification by Pictographic recognition. Tables 5 - 7 show examples of Arabic word segment classification for three isomorphic graph groups. The graph groups shown in these tables encompass the sets shown in Table 4.

Table 4: Profile of selected graph groups.

Graph Group Code	Training /Testing Sample Size
6;96.128.16	1,930
4;64,128	3,155
6;112.0.16	2,453

Table 5: Classification results for top nine word segments within graph group 6;96.128.16.

True Class	Count	Percent Correct	CLASSIFIED AS									
			None	تما	ثر	سر	سما	كل	لمز	لو	مع	يد
None	1510	67.61	1021	37	35	18	10	45	89	33	63	58
تما	29	89.66	2	26	0	0	0	0	0	0	1	0
ثر	48	97.92	1	0	47	0	0	0	0	0	0	0
سر	34	82.35	4	0	0	28	0	0	0	0	0	0
سما	45	77.78	7	0	0	0	35	1	2	0	0	0
كل	54	92.59	4	0	0	0	0	50	0	0	0	0
لمز	60	93.33	3	0	0	0	0	0	56	0	1	0
لو	39	79.49	3	0	0	0	0	0	0	31	0	0
مع	63	90.48	2	0	0	0	0	0	4	0	57	0
يد	48	91.67	2	0	0	0	0	0	0	1	1	44

Table 6: Classification results for top nine word segments for graph group 4;64,128.

True Class	Count	Percent Correct	CLASSIFIED AS									
			None	أ	س	سا	شا	في	لسا	ها	خو	يد
None	1958	67.67	1325	12	70	38	118	110	17	48	127	93
أ	264	96.59	4	255	0	2	0	0	0	1	1	1
س	115	93.04	4	0	107	1	0	1	0	0	2	0
سا	112	89.29	1	0	6	100	0	0	0	5	0	0
شا	87	85.06	6	1	1	1	74	2	0	1	1	0
في	79	82.28	9	0	2	0	2	65	0	0	1	0
لسا	123	99.19	1	0	0	0	0	0	122	0	0	0
ها	83	86.75	4	2	1	0	2	1	0	72	1	0
خو	232	89.22	15	0	0	0	5	4	0	0	207	1
يد	102	84.31	10	0	0	0	1	0	0	0	5	86

Table 7: Classification results for top nine word segments for graph group 6;112.0.16.

True Class	Count	Percent Correct	CLASSIFIED AS									
			None	أ	شا	لسا	نب	هر	خو	يد	ير	ين
None	1728	73.73	1274	56	74	32	46	78	21	17	27	103
أ	90	72.22	20	65	1	0	0	0	1	3	0	0
شا	73	83.56	9	0	61	0	1	0	0	0	0	2
لسا	81	96.30	1	0	2	78	0	0	0	0	0	0
نب	52	78.85	5	0	1	0	41	0	0	0	0	5
هر	64	84.38	8	0	0	0	0	54	1	0	1	0
خو	63	92.06	3	0	0	0	0	1	58	0	0	1
يد	145	84.83	5	3	0	0	0	1	0	123	6	7
ير	53	84.91	4	0	0	0	0	1	0	1	45	2
ين	84	72.62	15	0	2	0	3	0	0	2	1	61

The size of the testing sample was intentionally set to the same quantity as the training sample. These three isomorphic graph groups were selected from the hundreds of possibilities generated because they are among the most frequently occurring graphs for Arabic word segments.

Figure 11 shows sample Arabic characters that produce the three graphs used in the analysis. It should be noted that individual isomorphic graph groups transcend both individual characters as well as word segments.

Graph Class	Example	Example 2
6;96.128.16		
4;64.128		
6;112.0.16		

Figure 11: Examples of graphs applied to Arabic characters.

For each of the three sets of graphs, training was performed using only the top nine character strings based on frequency. This limit was imposed to test classification of items not within the training set. Testing was performed with many more character strings to determine how items not in the training set would be handled during classification. The total number of unique word strings in each isomorphic group is shown in the following table.

Table 8: Number of items processed and accuracy for three selected graph group.

Group	Number of Word Segments	Number Classified
6;96.128.16	122	9
4;64.128	111	9
6;112.0.16	119	9

In Tables 5-7, the leftmost column in each shows the top nine character strings for each isomorphic graph group. The tenth category, labeled "None" is used for those cases where a test item did not match any of the nine selected character strings. The "Count" column shows the number of items for each word segment. The "Percent Correct" column shows the percentage of test items that correctly classified. The remaining

columns show the distribution of classification results. The total numbers of items processed and percentage of correct answers are shown in Table 9.

Table 9: Number of items processed and accuracy for three selected graph groups.

Group	Count	Percent Correct
6;96.128.16	1,930	89.04
4;64.128	3,155	90.89
6;112.0.16	2,433	83.12

In addition, the groups scored 67.67 (Group 6;96.128.16) , 67.61 (Group 4;64.128) and 73.73 (Group 6;112.0.16) in terms of correctly identifying items that were not among the nine possible character sets that could be classified. These results show considerable promise for applying Pictographic Recognition to Arabic word segments.

In practice, Pictographic Recognition employs a scoring methodology that creates a composite score from the individual classification results of all embedded sub-graphs extracted from an Arabic word segment. The classification results shown in Tables 5 - 7 provide a view of how three of these embedded graphs would perform.

Figures 12 - 14 show some sample Arabic word components and the composite scores they produce. The scores are tabulated as the sum of the classification scores of individual embedded graphs. Below each image, the top 5 scores are shown with an asterisk ("*") placed next to the score of the correct answer.



	Word Segment	Score
1	طا	788*
2	لا	391
3	حا	388
4	لها	226
5	ها	181

Figure 12: Sample character graph form and top five classification scores for Arabic word segment طا . This graph had ten embeddings.



	Word Segment	Score
1	نب	1086*
2	في	333
3	قن	314
4	في	291
5	ب	285

Figure 13: Sample character graph form and top five classification scores for Arabic word segment نب. This graph had 20 embeddings.



	Word Segment	Score
1	با	277
2	جا	252*
3	حا	159
4	ها	137
5	يا	136

Figure 14: Sample character graph form and top five classification scores for Arabic word segment با. This graph had 2 embeddings.



	Word Segment	Score
1	تو	890*
2	بط	573
3	به	239
4	ب	230
5	لو	202

Figure 15: Sample character graph form and top five classification scores for Arabic word segment تو. This graph had 12 embeddings.

In these examples, three of the figures had the correct answer in the first position with a broad spread between first and second position. In one case, (Figure 13) the correct answer came in second position closely following the first position score.

9 Conclusions

Word segments intrinsically formed in Arabic words offer an extraordinary opportunity toward both word recognition and word detection. Pictographic Recognition provides a tool capable of recognizing either the actual word segment or common embedded graphs within the word segments. The common embedded graphs are a critical ingredient toward successful word segment recognition because the high degree of variability in handwriting reduces the likelihood that written character strings will regularly appear exactly alike. However, the embedded graphs offer a means of capturing the "essence" of these character strings in a physical and mathematical form that can reliably be captured and classified. Effectively, the embedded graphs offer a means of standardizing strings of characters into reliable consistent forms.

Initial findings indicate Pictographic Recognition shows promise for recognizing word segments both in terms of accuracy of recognition and reduction of false positives. The embedded graphs appear to capture the essence of the written character strings from which they are extracted. However, implementing a general-purpose search and recognition system requires compilation of a database with sufficient exemplars to cover as many word segments as possible. Complete coverage is not necessary, because Pictographic Recognition can always rely on its character-based processes to analyze Arabic word segments never encountered previously.

Immediate next steps include greatly expanding the collection of word segment exemplars used for training to achieve a broader overall Arabic language coverage. In conjunction with the expanded training sets, the Authors will continue to refine the techniques herein discussed toward the goal of building an accurate and reliable Arabic word search and recognition tool based on Pictographic Recognition.

Document Analysis Applications

A Document Triage System for Army Application

Francis Fisher, Kelvin Marcus, Richard Chang, Joi Turner

U. S. Army Research Laboratory
2800 Powder Mill Road, Adelphi, Maryland

Abstract

Military campaigns routinely put Soldiers in contact with foreign documents that they are generally unable to read. The PDA-based Basic Language Translation System (PDA BLTS), an ARL designed handheld document triage system, permits the Soldier to triage, or sort by importance, foreign language documents. The system assists the Soldier in the collection and triage of captured documents by identifying those documents relevant to the Soldier's mission or the overall military campaign so that the Soldier can expedite the processing of those documents. Through document triage the military linguist or translator is provided those documents that may be most relevant to military operations. BLTS was designed in response to the need for expedient document triage and delivery through a portable system to be used by Soldiers. This paper will describe the engineering methodology used to develop this tool.

1 Introduction

PDA BLTS is a handheld document triage system consisting of a laptop, five PDAs with PDA-mounted cameras, and two scanners, as shown in Figure 1. The BLTS PDA gives the Soldier a handheld device to capture and translate document images in order to determine the importance of captured foreign language documents. The translated document and keyword search results are returned and displayed on the BLTS PDA. The BLTS Server receives document images from multiple BLTS PDAs, processes those document images, and returns the triage results to the appropriate BLTS PDA. The BLTS Scanners (flat-bed and sheet-fed scanners) attach directly to the BLTS Server and provide an additional method of document input for translation and review. Documents processed by BLTS can be reviewed on the BLTS Server using the BLTSDocumentReview software.

The design goal for PDA BLTS was to develop a portable, handheld device to be used by Soldiers for document translation and triage. Before beginning the design process several problems were outlined and discussed in order to determine practical yet optimal solutions for hardware and software. Those issues included system architecture, PDA selection, scanner selection, camera selection, and software design.

This paper will describe the technical work performed, engineering methodology used, and the choices made in developing the BLTS system. It will cover what we did to test, evaluate, and solve problems, based on what problems were already known; what new problems appeared and how they were handled; and how the system was integrated.

2 System Engineering

The first question to answer was how to meet the user's need for a small hand-held document triage device while working within the limitations of available technology. While there are some small X-86 compatible tablet computers available now, we were unable to find suitable products for the BLTS application when this effort was started. Even the current X-86 based tablets are much larger than typical PDAs. If we limit the Soldier-carried device to a PDA then we seriously limit the processing

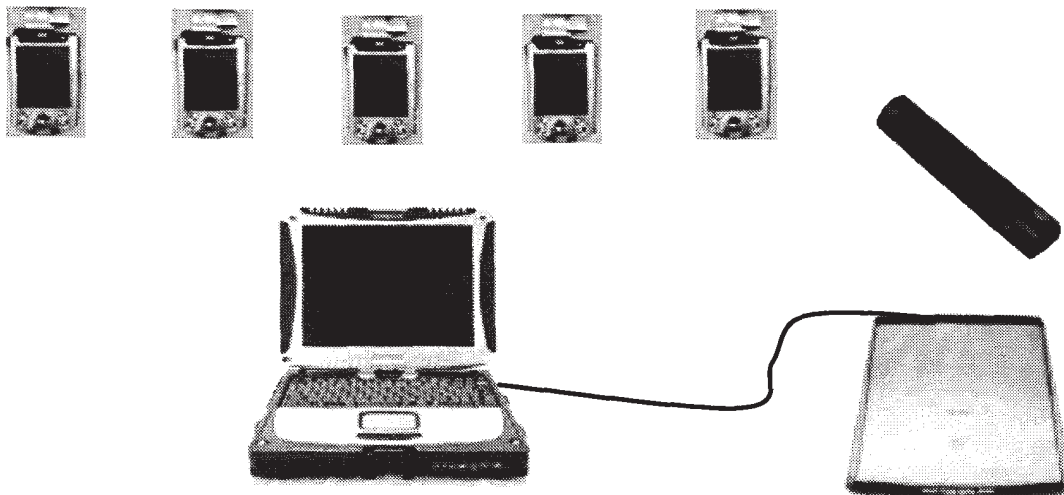


Figure 1: PDA BLTS Components

BLTS Process Flow

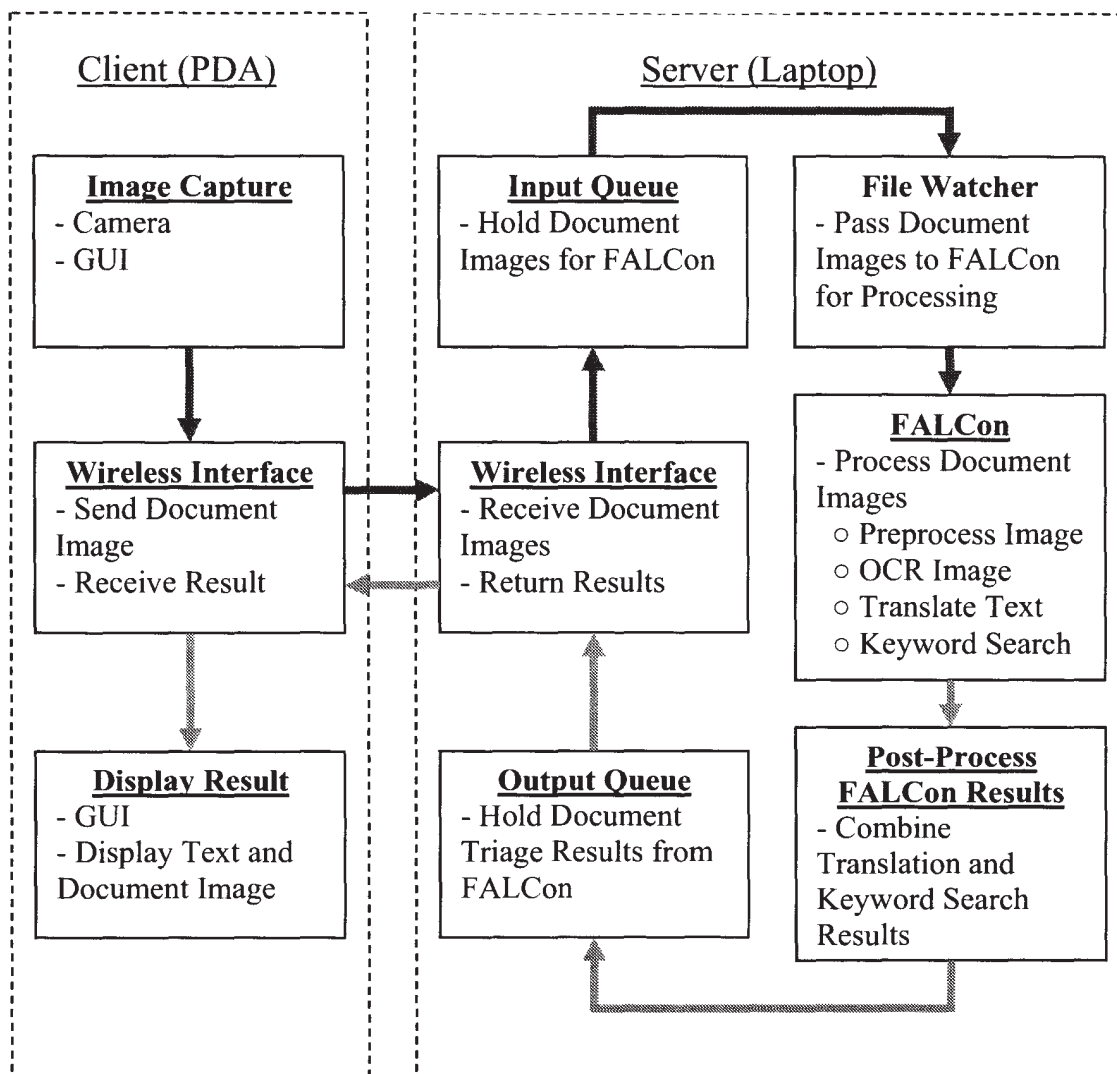


Figure 2: Process flow for PDA BLTS

capability that is available. We decided to use a client/server architecture with a laptop for the server and PDAs as the clients. This permitted us to use the existing FALCon document triage software to provide the core functionality. The PDA is then used as a document capture and display device. A wireless network is used to transfer document images from the PDA to the laptop, and to transfer the translation result from the laptop back to the PDA for review. The process flow for PDA BLTS is shown in Figure 2.

The next issue then was to select an appropriate method for document input for the PDA. Using any type of scanner would dramatically increase the size of the handheld system and would not be feasible. We decided instead to use a digital camera module that would plug into the PDA client device. Based on initial work using digital cameras to input document images for translation [1] and on test data collected in the lab, we knew that VGA resolution cameras were simply not sufficient for document image processing unless the area to be translated was severely limited. The release of 1.3 mega-pixel cameras was the final system component that ensured the success of the PDA BLTS system design.

Another issue in the design of PDA BLTS was the limitation of the wireless data link. If the Soldier operates the BLTS PDA beyond the range of the data link then document images cannot be sent to the BLTS Server for processing. Due to the processing limitations of the PDA, we found it difficult to implement an automated queuing system for delivery of document images to the server for processing. This could also have confused the user if document images sent for processing were not returned in a reasonable time because they were being held on a queue. Instead, we put the burden of operation on the user. We provide a message to the user when the wireless link is not available and they are given the option of saving the image for retransmission at a later time.

With the system engineering solutions in hand, we then turned to selecting hardware components for PDA BLTS as described in the following sections.

2.1 Hardware Selection

In the design of PDA BLTS we were required to select suitable hardware components for all of the BLTS functions. All hardware selections were performed in conjunction with Engineering Systems Solutions of Frederick, Maryland, the integration contractor selected for this project. All of the PDA BLTS components needed to be operational in extreme field environments. Since this is a prototype for pilot field-testing, there was not a requirement that the components be MIL qualified.

In designing BLTS, we have worked toward a solution that gives the Soldier a handheld device for document triage. Due to the PDA's limitations of low processing speed, small memory capacity, and the incompatibility with COTS software developed for the Microsoft Windows XP/2000 operating systems, we were forced to utilize a client server architecture for PDA BLTS for the current system designs. Given the time constraint, there was no way to port the COTS (MT, OCR, etc.) that form FALCon to the PDA, financially or technologically. FALCon was an in-house product that could easily be integrated into the current system design, and would be completely accessible by PDAs through a client/server architecture.

2.1.1 PDA Selection

As part of system design we were required to decide on commercial versus rugged hardware platforms. There are many rugged PDA platforms available on the market today. The problem is, we will not know what will be available tomorrow. The problem is even worse in the commercial PDA market where product lifetime is typically less than two years. An additional selection criterion was the cost of the equipment. Rugged PDAs are typically 3 to 5 times more expensive than the commercial equivalents. Users were questioned on their preference for quantity versus ruggedized and their preference was for a larger quantity of commercial systems for initial pilot testing. The PDA selection criteria were: COTS product; fastest processor available; SDIO memory slot for camera; operate in extreme environmental conditions.

After purchasing and testing sample PDAs, we selected the HP h5555 for version 1 of PDA BLTS. When we started production of version 2 we found that the h5555 had been discontinued. We then re-evaluated available products and selected the hx2750 PDA by HP. The hx2750 provided a 330% increase in battery life and doubled the range of operation for the wireless network.

2.1.2 Laptop Selection

The requirements in the selection of the laptop for the BLTS server were to run the Forward Area Language Converter (FALCon) software (the core translation software for BLTS), to provide wireless network connectivity to the BLTS PDAs, and support operation at temperatures from 30° to 130° Fahrenheit. The laptop selected is the Panasonic Toughbook CF-18. Built with Centrino technology, the CF-18 runs software for the BLTS Server, scanner, and FALCon and supports wireless network connectivity for five PDA's. The server queues documents from the PDA and scanner, passes the documents to FALCon for translation and keyword search, and returns the translated results to the appropriate PDA as shown in Figure 2. If the document processed is from the scanner, then the FALCon results are stored on the server for later review.

2.1.3 Scanner Selection

The selection criteria for the BLTS scanner included small size, extended temperature operation, and simplicity of operation. From previous prototype fielding of FALCon, we found that users had difficulty using sheet fed scanners to process bound or stapled documents in the field. Based on those issues we started the initial research on the application of digital cameras for document input as reported in [1]. Flat bed scanners have been greatly reduced in size in the 7+ years since our prototype fielding of FALCon. For BLTS, we selected the smallest COTS flat bed scanner that we could find. We then tested three samples of that scanner at 130 degrees F during the day and 30 degrees F at night to simulate harsh environments encountered in some military operations. Although the temperature range tested far exceeded the manufacturers specifications, the scanners tested performed well. Based on this testing we selected the Canon LIDE80 flat bed scanner for the BLTS application.

During user training for PDA BLTS, users requested sheet-fed scanners to replace the flat bed scanners. The sheet-fed scanners are much smaller than the flat bed scanners making them more portable. The trade off is that the sheet fed scanners are not able to scan bound and oversized documents. Version 1 of PDA BLTS was delivered with flat bed and sheet fed scanners, version 2 of PDA BLTS was delivered with sheet fed scanners only.

2.1.4 Digital Camera Selection

The BLTS PDA Camera selection criteria was, for the most part, geared toward finding a camera with the highest possible resolution that would provide high quality images at close range with minimal optical distortion. The market for high-resolution compact flash or Secure Digital (SD) PDA cameras is very limited. Of the small number of devices available, we reviewed data sheets and selected four different models for evaluation. One of the cameras selected for evaluation included a photo flash capability, which we thought might be helpful. Unfortunately, the addition of the flash unit to the camera resulted in a package that was so large and so heavy that it was not at all suitable for this application. The poor quality of the optics further reduced the usability of this camera. The Veo Phototraveler and the HP Photosmart turned out to be the same cameras with a minor difference in the layout of the camera body. Both of these cameras provided good imaging capability at close range with minimal optical distortion. We selected the HP Photosmart mobile camera, a 1.3M pixel SD camera, because it provided good quality document images and was compatible with the h5555 PDA. We used the software development kit from Veo with the HP Photosmart camera to provide a full solution for the integration of the digital camera into the BLTS PDA and software.

Through previous work with 3.3M pixel digital cameras done in [1], we developed a preprocessing algorithm for document images to improve FALCon OCR accuracy. During an additional evaluation of a 5.1M pixel camera we discovered an improved method in our algorithm to increase OCR accuracy for both low and high-resolution cameras. We currently use this preprocessing algorithm on document images captured by the current 1.3M pixel SD camera before passing them to FALCon for translation.

2.1.4.1 Document Image Size Limitations

In order to evaluate the PDA camera, tests were also performed to determine the optimal range of font sizes and page sizes that would yield the highest OCR accuracy when capturing and processing document images with PDA BLTS. These tests confirmed that document images with font sizes greater than 10pt and 250 DPI yield the optimal OCR results, with accuracies from 92% - 95%. OCR accuracy fell quickly when document images with font sizes less than 10pt were processed as shown in figure 3. The same was true for document images with 320 DPI as shown in figure 4, but the drop in overall OCR accuracy was much smaller for font sizes of 10pt and less as would be anticipated given the increased number of pixels on each character.

Based on these results, the camera selected for PDA BLTS should be sufficient for capturing and processing half-page document images and should produce optimal OCR results for documents with font sizes of 10pt or larger.

3 System Software Components

The following sections describe the software components of PDA BLTS.

BLTS Server

The BLTS Server software acts as a link between the BLTS PDA and the Forward Area Language Converter (FALCon), the back-end software that produces all machine translations and keyword results. The BLTS Server keeps track of all connected BLTS PDA's. It is also responsible for receiving all

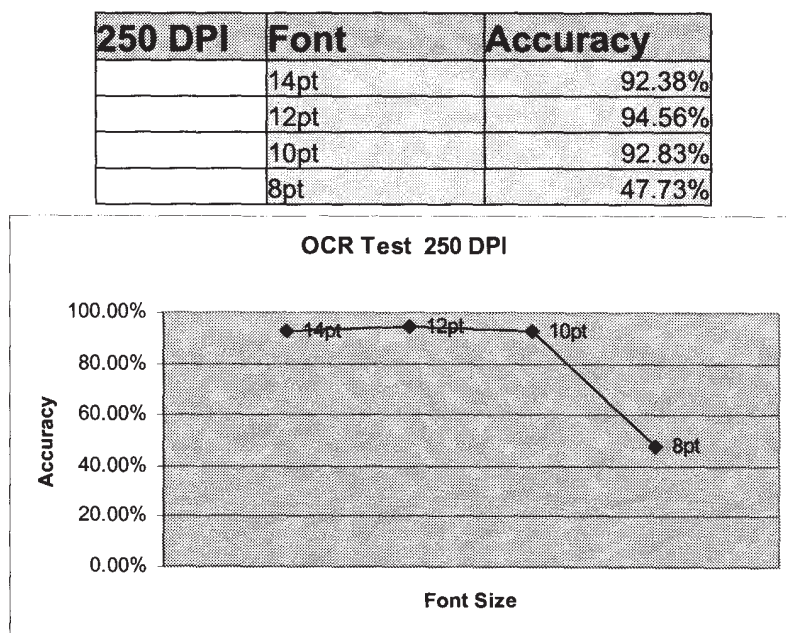


Figure 3: OCR Font Test result for 250 DPI documents

320 DPI	Font	Accuracy
	14pt	95.59%
	12pt	93.41%
	10pt	94.37%
	8pt	85.93%

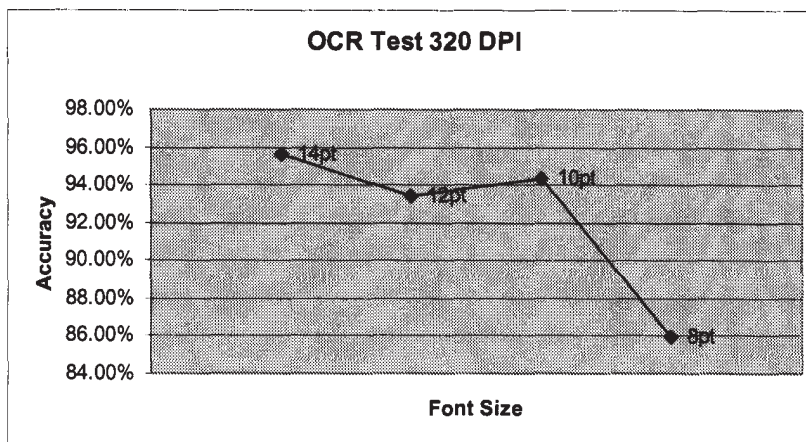


Figure 4: OCR Font Test result for 320 DPI documents

document pictures and sending all translation results to and from the PDA's. If the Soldier is near the BLTS Server with documents in hand, he or she can scan the documents, which are then automatically routed to the BLTS Server through the BLTS Scanner software. The images are then stored and translated for review.

3.1 BLTS PDA

The BLTS PDA software uses wireless communication to exchange data with the BLTS Server. The software provides a user interface to capture document images using the camera and displays the translation results for review.

3.2 FALCon

FALCon, the Forward Area Language Converter, is a tool that utilizes optical character recognition (OCR) and machine translation (MT) to help non-linguists identify the importance of foreign language documents captured in the field. Scanned document images or digital camera images can be loaded into the FALCon software which then converts the images into text characters and translates the text into English. Keyword lists can be created and specified by the user, which would be used to search the resulting English text, or if the appropriate foreign language lists exist, search the foreign language text.

During the development of BLTS, the original build of the FALCon software had undergone several key changes. The Machine Translation software, Transphere's Apptek, used by FALCon for Arabic text translation was updated from version 2.6.17 to 2.7.0.35 to address key issues where during transliteration some English word letter combinations would consistently err, word definition selection would translate to the least probable selection, and proper names would not completely translate.

3.3 BLTS Document Review- Server

All images and resulting documents can be reviewed using the BLTSDocumentReview on the laptop. This software provides a display that conveniently groups the document pictures with their respective Optical Character Recognition (OCR), Machine Translation (MT), and Keyword search results. A scaled down version of this software is also used to review MT and Keyword search result files on the BLTS PDA.

4 Software Design

The software design issues were numerous. The first issue was to figure out how to allow the system to support multiple users, and process documents concurrently. Each BLTS Server supports up to 5 BLTS PDAs; so, 5 Soldiers can send documents for processing at the same time. Since FALCon processes one document at a time, FALCon became the bottleneck in the PDA BLTS process. The solution was to adapt a multi-threaded server architecture with a document queuing process. This allowed users of the five BLTS PDAs to send documents simultaneously and to continue capturing and sending document images for processing. Controlling document return flow was the next issue to be resolved. An output queue was implemented on the BLTS Server to hold translation results. When translation results are available for a BLTS PDA on the server, the server checks to see if the PDA is available. If the PDA is not available the results are stored in the queue. If the PDA is available, then the results are returned to the PDA for review by the Soldier.

4.1 Software Improvements

The initial design for BLTS used FALCon as the backend translation software, but the processing time for this first implementation was very slow (two and a half minutes to process a half page document). The process time includes the image transmission from the PDA to the server to the moment the user is notified on the PDA that the result is ready for review. By testing the individual software components, we discovered that FALCon's initial sequence of language library loads for machine translation was contributing heavily to the drag on process time. Each time the FALCon process was called, the MT libraries would be reloaded on the server, causing unnecessary delay in the overall performance of BLTS. However, when FALCon was run in batch mode, the initialization process was only run once, at the beginning of the batch. The first document processing time was still over two minutes, but the remaining documents were processed at a rate of less than forty-five seconds. We had the FALCon team modify FALCon so that the batch mode would remain in memory and monitor a folder for new documents to translate. As a result each half-page document takes approximately 45 seconds to process.

In response to user feedback, the Soldier is given an option on the BLTS PDA that forces the server to store all translation results for that PDA on the server. The Soldier can then process larger quantities of documents without interruption and retrieve all of the results for review at a later time. Also from user feedback, documents and results are stored in non-volatile memory on the PDA in case of battery failure or system malfunction so that results will not be lost.

The first version of PDA BLTS did not take into consideration the cumbersome task of system configuration. The user had to set the network SSID, IP address and WEP encryption for the server and all five PDAs. During the initial training of the system, we noted that this section took the longest to train the Soldiers. The requirement to configure the network and security components detracted from regular system use. Because of the frequency in the need to configure the PDA, it was necessary to automate this process. The current configuration only needs a system number, which is located on the device, as user input. The network and security settings are automatically configured and re-configured, on the PDA, if that information is lost due to battery failure or a hard reboot.

4.2 User Interface Design

Another major design area was the user interface. Through many field tests and exercises we have found Army Soldiers to be both intelligent and resourceful. If you design a good product that they can use, they will use it. In all aspects of the design of the user interface we attempted to make system operation simple and flexible within the limitations of the hardware, development environment, and operating systems used. Most of the BLTS PDA software features are controlled by menu bar buttons across the bottom of the PDA display. These buttons change as the operating mode changes. All of these “soft” buttons are linked to the corresponding hardware button located under the menu item. In this way the Soldier can operate the system without the need to locate, remove, and use the stylus. The users fingertip can generally manipulate those windows that do require touch screen use for proper operation. Along with design considerations that encompassed physical attributes of the system, we had to keep in mind that the main purpose for the BLTS PDA is data management. Document information obtained from the BLTS process had to be formatted in a display so that users could make decisions upon review.

4.2.1 BLTS Document Review

The Soldier's main objective when using PDA BLTS is to aid human translators by making key decisions to ensure critical documents are identified and passed to the appropriate authorities as quickly as possible. As such, the Soldier must understand the importance of the system's output in relation to their mission. FALCon creates four document outputs during the triage process; the document image, OCR, MT, and keyword results. Each document has its importance to the end user and PDA BLTS is designed so that access to the results is simple, convenient, and understandable.

4.2.1.1 FALCon Output Format

The document image is the original view of the document either scanned or photographed. The image by itself is a digital link to the original document. By viewing the image, the Soldier can associate it to the physical document without knowledge of the language through simple pattern matching. The OCR - Optical Character Recognition output is not as helpful to the Soldier because it is not easy to see differences between the actual document and the OCR'ed document. The OCR'ed document may be more useful to the human translator who can correct OCR errors and obtain better translations. The appearance of the MT results could be best described as “word soup”, or a mixture of non-cohesive words, phrases, and foreign character strings. This document can aid in the Soldier's ability to easily identify nouns, proper nouns, and action verbs and provide additional information not included in the keyword list. The keyword result document displays the frequency of mission related keywords found in either the machine translated document or the OCR document. FALCon allows the Soldier to edit and add customized keyword lists based on the current mission requirements.

4.2.2 BLTS Document Review-Server

BLTSDocumentReview is a server side application that aids users stationed at the laptop to view all four FALCon output files, as shown in figure 5. All documents scanned or transmitted from the PDA's are stored, organized on the server and accessible from BLTSDocumentReview. This application allows users to review data from a specific PDA source or the scanner. The user can determine the time and date that each data input file was captured and can also traverse through all of the input files in order. Most important, each document image is linked to its processed results. When all documents have been analyzed, the folder containing the mission's data is then closed and tagged with pertinent information regarding the mission. This data can be sent to higher commands for further investigation.

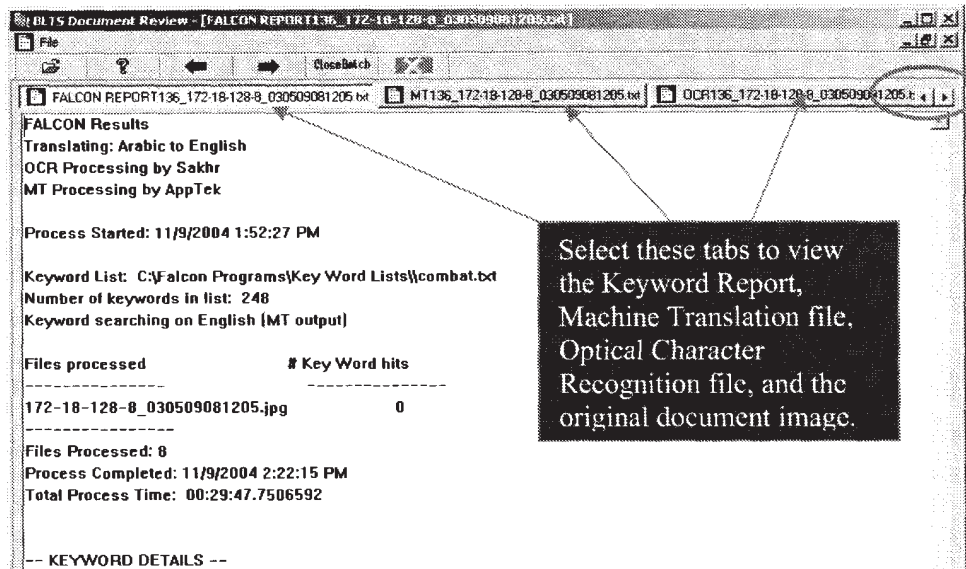


Figure 5: BLTS Server Document Review

4.2.3 BLTS Document Review-PDA

Document review on the BLTS PDA was an interesting problem. The display on the PDA is quite small and has low resolution so it is difficult to display a great deal of visual information. Since the latency between sending a document for translation and receiving the result could be more than a minute, we felt that it was very important to provide the user with the ability to see both the resulting text translation and the original document image. Our initial implementation required the user to open the text translation result from Windows File Explorer for review, then close that view and open the document image to see what the original document was. The user then had to close all views and close File Explorer to continue. This was very cumbersome. After considering different alternatives we implemented a dual view display, shown in figure 6, that shows the original document as a thumbnail view at the top of the screen with the text translation result below. If the user taps on the document image it is enlarged to fill the screen and a limited amount of text is displayed. If the user taps the document image again the display reverts the original thumbnail view with large text area. In this way the Soldier can see both the original document image and text result on one display.

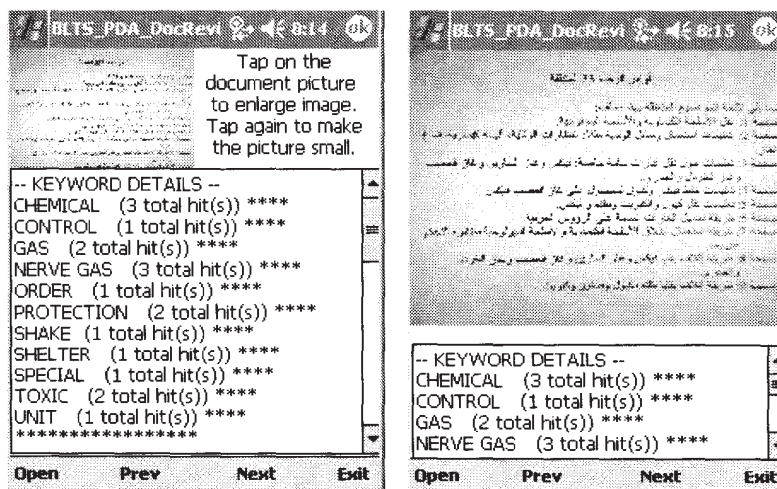


Figure 6: BLTS PDA Document Review. Machine Translation can be viewed by scrolling down in the text view.

5 Conclusion and Future Work

The various COTS software components required to implement a document triage system on a PDA were not available when this effort was started. Handheld PCs were also not available. Hence, to develop a handheld document triage system required the implementation of a client/server architecture system that utilizes a PC-based server and PDA-based handheld devices. Handheld PCs may soon be small enough to permit a handheld document triage system to be build that is based on FALCon or similar COTS based products. As an alternative to “waiting for the COTS market”, the Army is using a Small Business Innovative Research (SBIR) contract to fund the development of custom software to perform document triage on handheld PDA platforms.

Systems developed for field use must take the needs of the Soldier into account. User interfaces must be simple and easy to use, not because the Soldiers are not able to work with complex interfaces, but because the Soldiers have a larger mission to perform. We cannot anticipate that our system will receive the users full attention and we must design accordingly.

The version of PDA BLTS described in this paper will be replaced in the future by a document triage system that requires only the PDA and PDA mounted camera. This future version is already under development by contractors to the U.S. Army and some of that work will be presented at SDIUT'05.

6 Acknowledgements

We would like to acknowledge the assistance of the U.S. Army Program Manager – CHIMS for providing funding and program management support for this effort. We would like to thank ARL's Multilingual Team for providing modifications to the FALCon software as needed to support the proper operation of BLTS. We would also like to thank Dr. Michelle Vanni, Mr. Gideon Bass, and others working with Dr. Vanni, for their efforts to improve the Arabic machine translation software used in the PDA BLTS application.

7 References

[1]F. Fisher. Digital Camera for Document Acquisition. *In Proceedings for Document Image Understanding Technology*, pages 79-81, 2001

Multi-Language Handwriting Derived Biometric Identification

Donald T. Gantz, PhD John J. Miller, PhD
George Mason University
Fairfax, Virginia

Mark A. Walch
The Gannon Technologies Group
Alexandria, Virginia

Abstract

This paper provides a discussion of key technologies underlying the development and implementation of a Handwriting Derived Biometric Identification system. Three applications for author identification through handwriting are in development based on this system:

1. Individual Identification
2. Document Clustering
3. Forensic Document Examination

Individual Identification involves relating a document of unknown authorship to a data base of reference samples of handwriting from known authors.

Document Clustering encompasses grouping documents based on handwriting characteristics without knowledge of specific author identity.

Forensic Document Examination involves building a statistical foundation that will support court room testimony by expert witnesses that will withstand Daubert Challenges raised against handwriting analysis as evidence.

This paper focuses on the technical foundations of Individual Identification.

Handwriting Derived Biometric Identification exploits the rich set of measurements available through Isomorphic Graph Matching which is a technique based on Graph-Theory that is used to identify the same written forms in different writing samples. By statistically comparing measurements on similar objects across different writing we are able to identify those writing characteristics that best distinguish or characterize individual authors. An author's biometric identity is defined through the measurements that are determined to characterize that author's writing in the sense that those measurements have the power to distinguish the author's writing from that of other authors. Handwriting Derived Biometric Identification is a computationally intense process that utilizes statistical discrimination algorithms.

1 Overview

An automated process parses a handwriting sample to identify individual characters. This process uniquely associates each parsed character with a graph. For purposes of this paper, the software tool that makes this association is referenced as the "Graph Builder". Each graph is a collection of nodes connected by loops and curves. Nodes are located at the ends of curves or where curves cross. The following example demonstrates that there is a very large set of physical measurements defined for even the simplest graphs.

Character:	a
Graph Model:	4;112
Number of Edges:	3
Number of Vertices:	4



Measurements for Graph 4:112	Number of Measurements
Absolute Distance	
Vertex to Vertex	6
Vertex to Edge Centroid	12
Vertex to Edge Contours	12
Edge Centroid to Edge Centroid	3
Edge Contours to Edge Contours	3
	36
Centroid Distance	
Vertex to Graph Centroid	4
Edge Centroid to Graph Centroid	3
Edge Contours to Graph Centroid	3
	10
Graph Direction	
Vertex to Vertex	18
Vertex to Edge Centroid	24
Vertex to Edge Contours	24
Edge Centroid to Edge Centroid	6
Edge Contours to Edge Contours	6
	78
Centroid Direction	
Vertex to Graph Centroid	12
Edge Centroid to Graph Centroid	9
Edge Contours to Graph Centroid	9
	30

assigned to the column graph. Entries in the table are constrained in that percentages are only calculated for those letter/graph pairs which were observed at least ten times in the author's modeling London letters.

									-							6		
									8							-	-	
								4								9	8	-
								-								6	-	5
								-		-	1					-	1	-
			-					1		7	1			-		1	1	1
			4					0	-	-	2			6		3	2	1
			-			6		0	6	1	-			-	6	4	-	2
C			9			-		-	-	2	2			1	-	-	2	-
h			8		-	1		1	1	0	4	4		1	1	-	4	1
a		4	-		4	1		2	1	-	-	-		2	1	5	-	2
r	-	-	0		-	-		2	8	2	2	6	-	-	3	-	6	8
a	2	1	-	-	3	1	-	-	-	4	4	4	1	1	-	1	4	-
c	-	1	6	2	-	2	0	6	9	-	-	-	-	1	6	2	-	6
t	1	2	4	-	2	0	-	4	6	0	0	1	1	2	6	0	-	4
e	9	-	-	6	2	-	1	-	-	-	-	2	2	-	-	-	3	-
r	2	0	0	4	4	0	6	0	0	0	0	8	8	0	0	0	2	0
a	19.2	--	48.1	--	--	--	--	--	--	--	--	--	--	32.7	--	--	--	--
c	--	100.0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
d	--	--	--	--	--	--	--	--	--	--	--	--	--	100.0	--	--	--	--
e	--	42.0	--	7.0	--	28.7	--	--	9.8	--	--	--	--	--	--	12.6	--	--
h	--	100.0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
i	--	33.8	--	--	--	--	66.2	--	--	--	--	--	--	--	--	--	--	--
l	--	100.0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
n	--	75.8	--	--	--	--	24.2	--	--	--	--	--	--	--	--	--	--	--
o	42.9	--	57.1	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
r	--	29.7	--	56.0	--	--	--	--	--	--	14.3	--	--	--	--	--	--	--
s	--	56.7	43.3	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
t	--	--	--	--	--	--	--	--	25.4	--	55.2	--	--	--	--	--	19.4	--
u	--	--	--	--	--	--	--	--	100.0	--	--	--	--	--	--	--	--	--
w	--	--	--	--	--	--	--	--	--	--	100.0	--	--	--	--	--	--	--

Author=1

Character					-					-								9	8	8
					4					5								6	-	-
					-					-								-	-	-
					1				-	1								1	7	1
					0					4	1				-			3	-	2
					0					-	2				6		6	4	1	-
					-			-	9	-		5			-	1	-	-	2	2
					1	-	-	4	8	1	-	-		1	1	1	1	0	0	4
					2	4	4	-	-	2	5	2	1	2	1	1	-	-	-	-
					8	-	-	6	0	8	-	2	2	-	2	3	6	2	0	2
-		-	-																	
1	-	2	3	-	1	1	4	-	-	1	5	-	1	-	-	4	4	-	4	
-	2	-	-	6	1	2	-	6	6	2	-	0	1	9	6	-	-	0	-	
1	-	1	2	4	2	0	1	4	4	0	1	-	2	6	6	0	0	-	0	
2	6	9	2	-	-	-	2	-	-	-	2	1	-	-	-	-	-	3	-	
8	4	2	4	0	0	0	8	0	0	0	8	6	0	0	0	0	0	2	0	
a	0	0	28	0	0	13	0	0	10	0	0	0	0	0	0	0	0	0	0	
c	0	14	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	
d	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	
e	0	0	23	58	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
l	0	0	0	13	0	0	14	0	0	0	0	0	50	0	0	0	0	0	0	
n	0	32	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	
o	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
r	0	50	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	
s	0	11	13	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	25	0	
u	0	0	0	0	0	8	0	0	0	0	0	0	0	0	16	0	0	0	0	
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	

Table 2 presents the same style of breakdown for another author. Table 3 presents the number of times that letter/graph pairs were observed in the first author's two held out London letters. Again, frequencies in Table 3 are constrained in that frequencies are only tabulated for those letter/graph pairs observed to occur at least ten times in the author's two held out London letters. We can use the tabulated information in these style tables to investigate how strongly the patterns of letter/graph associations identify unknown authors.

Hypothetically, consider Author A's two held out London letters to be written by an Unknown Author. We can get expected frequencies if Author A were the Unknown Author by multiplying the number of occurrences of each letter of the alphabet by the decimal values represented by the percentages, as seen in Tables 1 and 2, in the row for that letter. We

can then compare the resulting expected letter/graph pair frequencies to the actual observed letter/graph pair frequencies for the Unknown Author. An obvious metric to use for measuring the differences between the expected and observed frequencies is the sum over all table entries of the squared differences between observed and expected values. We did these calculations for a set of 100 authors from our database; that is, we compared the pattern of observed letter/graph pair frequencies for Author A's two held out London letters to the modeling London letter data for each of the 100 authors. The resulting 100 sums of squared differences (after standardization) are presented in the histogram of Figure 1. In Figure 1, Author A is represented by the symbol '1' and each of the other 99 authors is represented by the symbol '0'. Author A, the true identity of the Unknown Author, is very clearly discriminated from the other 99 authors.

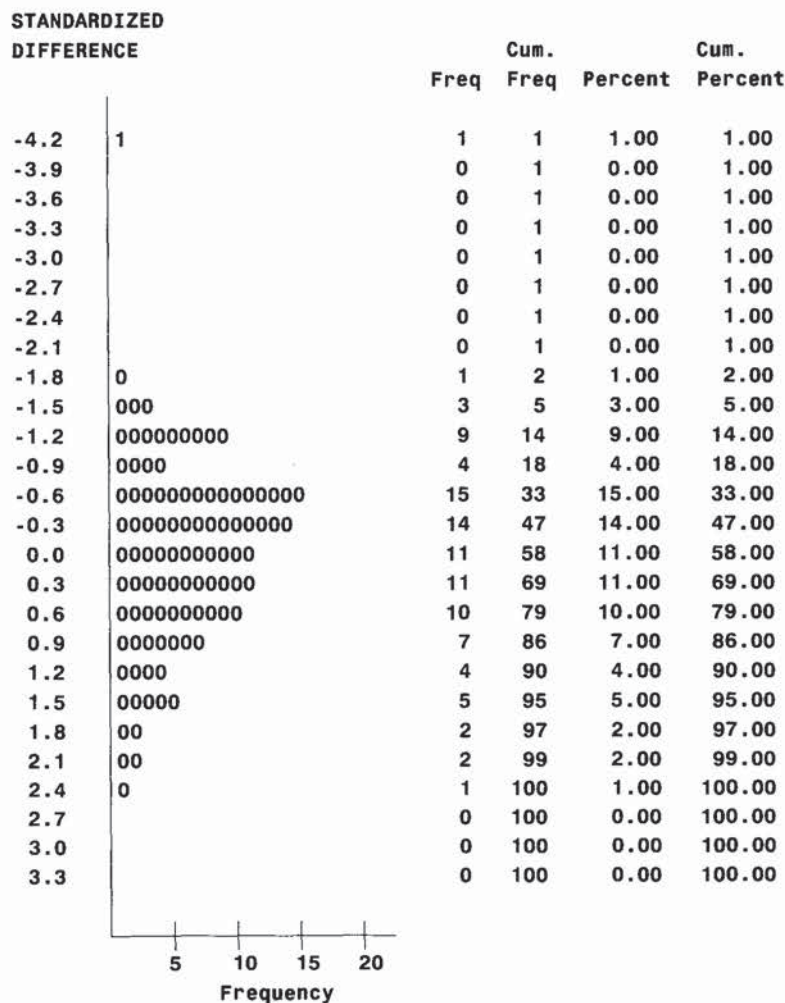


Figure 1: Standardized Sums of Squared Differences

This strong discrimination was typical regardless of the author playing the role of the Unknown Author. Each of the 100 authors had the lowest sum of squared differences among all authors when that author played the role of the Unknown Author.

The above computations show that:

1. Authors are very consistent relative to the associations between letters of the alphabet and the graphs assigned to them.
2. When the writing samples from authors in a database are as rich as they are in our London letter database, the patterns of letter/graph associations can be a powerful identifier of authorship.

3 Biometric Identification

The paper will now turn to the problem of author identification for authors who are observed to be very similar in that the Graph Builder assigns the same graph to their writings of a particular letter. The graph in question, as noted above, will have hundreds of measurements defined for it. The quantity of writings available from the author will determine the number of occurrences of the letter/graph pair. Typically, there will be many more graph measurements than there will be observed occurrences of the letter/graph pair. This makes the data analysis subject to *the curse of dimensionality*.¹

3.1 Discriminant Analysis

One of the statistical methods we use to distinguish the data for similar authors is stepwise discriminant analysis.² We apply this technique by isolating corresponding letter/graph pairs from different handwriting specimens. That is, we inspect different writing samples to locate instances of the same letter written as the same graph, assigned by the Graph Builder. In practice, this task is accomplished by automation that uses recognition to identify the character and uses the Graph Builder to assign the appropriate graph. Once the pairing is done, discriminant analysis will find a small subset of the graph measurements that do a good job of

¹ In this article, the curse of dimensionality refers to the fact that the amount of data available (in terms of occurrences of a letter/graph pair) is insufficient to support an analysis with such high dimensional data (i.e., such a high number of measurements). Some technique must be employed to reduce the measurements to a small set (the Biometric Kernel).

² SAS/STAT Users Guide, Version 9.1.

discriminating the data on this letter/graph pair for two authors. Stepwise discriminant analysis will

1. Select a small number of measurements for discriminating the two authors' data; and
2. Provide a weight (canonical coefficient) for each selected measurement.

We combine the values of the selected measurements using the weights; the resulting new measurement (canonical variable) gives the best available discrimination of the data from the two authors. Figure 2 presents a histogram for such a canonical variable and shows the resulting separation of the data from two authors.

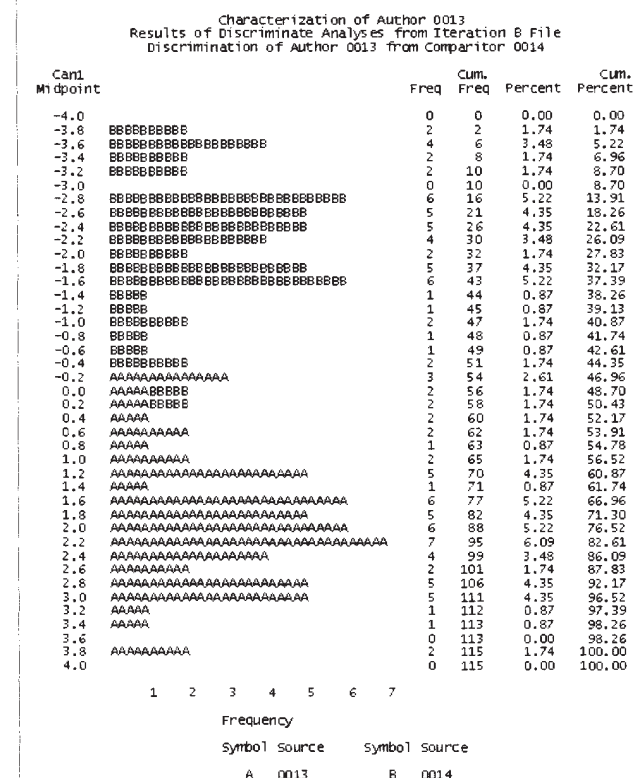


Figure 2: Discriminant Analysis for Two Authors

3.2 The Biometric Kernel

Say that we have 100 authors in our data base who have been observed to be very similar in that the Graph Builder assigns the same graph to some of their writings of a particular letter. We can use stepwise discriminant analysis to compare Author 1 one-by-one to each of the other 99 authors. We combine the results of the 99 stepwise discriminant analyses to reduce the original list of hundreds of measurements defined for the letter/graph pair to a still large but much smaller list of measurements with demonstrated power in discriminating Author 1 from

each of the other 99 authors.³ By repeating the entire process several times, we are able to isolate about a dozen of the measurements that are powerful for discriminating Author 1 from each of the other 99 authors. We refer to these final dozen (or so) measurements as Author 1's *Biometric Kernel* for the particular letter/graph pair.

Following the steps in the preceding paragraph for each author in the pool of similar⁴ authors, we define a Biometric Kernel for each author. We form a database for identification by storing the following information for each cohort of similar⁵ authors:

1. The names of the Biometric Kernel measurements for each author.
2. For each pair of authors (say, Author A and Author B), the weights (canonical coefficients), using Author A's Biometric Kernel, that form a canonical variable that discriminates Author A's data for the specified letter/graph pair from Author B's data for the same pair; and the corresponding weights associated with discrimination of Author A's data from Author B's data using Author B's Biometric Kernel.
3. The means and standard deviations of Author A's data and Author B's data for each canonical variable computed in #2.

3.3 The Competitive Matrix

We conceptualize the stored information for a cohort of similar authors as defining a *Competitive Matrix*; see Figure 3.

	Comparator Author				
	B1	B2	B3	B4	B5
Modeling Author	A1				
	A2				
	A3				
	A4				
	A5				

Figure 3: The Competitive matrix

³ The results are a subset of the graph measurements with weights; each comparison to a new author provides a new subset of measurements and associated weights.

⁴ Authors are similar in that in their three London Letters used for modeling our Graph Builder assigned the specified graph to some of their writings of the specified letter.

⁵ See Footnote 4.

The structure of a Competitive Matrix is determined by associating the matrix's rows with the author whose Biometric Kernel is the basis of the discrimination; we say that rows are determined by the *Modeling Author*. Each column is then associated with a single *Comparison Author*, that is, the author whose Biometric Kernel is not used as the basis of the discrimination. Otherwise stated, the matrix uses each author twice, once as a model author and once as a comparator author. In this way, it is possible to distill those measurements that characterize each author-to-author comparison. Since the same authors exist both as rows and columns, the diagonal axis of the matrix would contain cells where authors are compared against themselves. Since discriminating an author's own data from itself is meaningless in the current context, cells along the diagonal of the matrix are excluded from consideration.

The physical database of three modeling London letters for each author is modeled by a collection of Competitive Matrices. For the examples of this paper, a database of 100 authors is used. Each Competitive Matrix corresponds to one specific letter/graph pair. There is a Competitive matrix for every letter/graph pair observed to occur via use of character recognition and the Graph Builder application.

4 Testing the Model

We now discuss testing our Competitive Matrix modeling of author characteristics by using the Competitive Matrices to assign probable authorship to the characters in the two held out London letters for all authors in the database.

4.1 Competitive Matrix Voting

Given a character of hypothetically *unknown authorship* from the held out London letters,

1. Quantification Scheme: Associate the character with a Letter and Graph pair by character recognition and the Graph Builder.
2. Evaluate potential authorship *among similar authors in the data base*, i.e., among all authors who were observed to use the referenced Letter/Graph pair in their three London letters which were used for Biometric Kernel modeling. The evaluation is accomplished via *Competitive Matrix Voting*.

In Competitive Matrix Voting, each author plays the roles of both row (i.e., *Modeling*) author and column (i.e., *Comparator*) author; hence the matrix structure.

Each *cell* (intersection of a row and column) is assigned a vote, either 1 or 0 for the row author and a

vote, either 1 or 0, for the column author. The voting logic is presented in Figure 4.

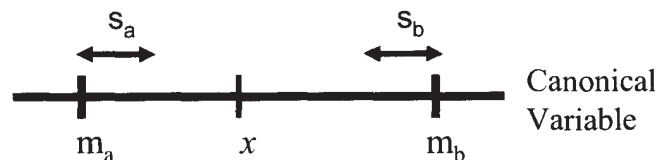
Competitive Matrix Testing a character of un

- H_0 : A is the True Author
- Model: based on assumption H_0 is true
 - So use the Biometric Kernel for Author A
- H_{alt} : B is the True Author
- Statistic is Canonical Variable for (letter/graph) pair based on the Biometric Kernel for row Author A.
- Voting Algorithm: if $|z_a| \leq |z_b|$ and $-2 < z_a < 2$ then Vote for A
if $|z_b| \leq |z_a|$ and $-2 < z_b < 2$ then Vote for B

It is possible that neither A nor B receives a vote. We call such a situation a 'no-vote' case.

$$z_a = (x - m_a) / s_a$$

$$z_b = (x - m_b) / s_b$$



m_a and m_b are the means of the data for Author A and Author B.

s_a and s_b are the standard deviations of the data for Author A and Author B.

x is the canonical variable (statistic) value for the character.

parator

	B1	B2	B3	B4	B5
A1					
A2					
A3					
A4					
A5					

Modeling Author

Figure 4: Competitive Matrix Voting

The next step summarizes each author's row votes and column votes.

Row Voting Using the Competitive Matrix with m Authors

- For Author A, Add the votes for A across all columns (all comparator authors)
- This amounts to $m-1$ tests of H_0 : *A is the true Author*
- One test for each column Comparator Author B
- Each failure to reject the H_0 with a plausible value for x is a vote for A
- An average vote close to 1 means that the unknown data is more consistent with the data used to model Author A than with the data from comparator authors.

Column Voting Using the Competitive Matrix with m Authors

- For Comparator Author B, sum the votes for B across all rows (all modeling authors)
- This amounts to 1 test of H_0 : *A is the true Author* for each *modeling* Author A
- Each rejection of H_0 in favor of H_{alt} : *B is the Author* with a plausible value for x is a vote for B
- An average vote close to 1 means that the unknown data is more consistent with the data for author B than with the data used to model most other Authors

As one's intuition would expect, an author's row and column votes are correlated; that is, both row and column votes are low or both are high. Typically, we have observed that the true author will have a fraction of row votes close to 1 and a fraction of column votes close to 1. However, as seen in Figure 5 some authors other than the true author might also have a fraction of row votes close to 1 and a fraction of column votes close to 1. In Figure 5, each author from the Competitive Matrix is represented by a

symbol (1 or 0) plotted according to that author's fraction of row votes (horizontally) and fraction of column votes (vertically). The plotting symbol for

the true author is '1' and the plotting symbol for all other authors is '0'.

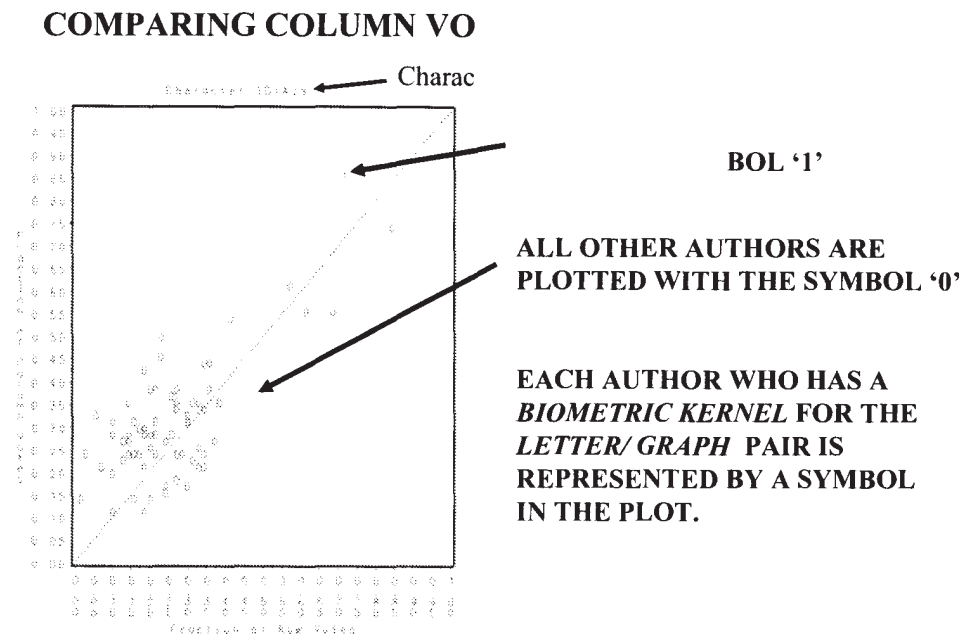


Figure 5: Comparing Row and Column Voting

We seek a single value (statistic) that reflects both how close the fraction of row votes is to 1 and how close the fraction of column votes is to 1. Note that

each symbol (author) in Figure 5 can be associated with a two-by-two frequency table. This table association is defined in Figure 6.

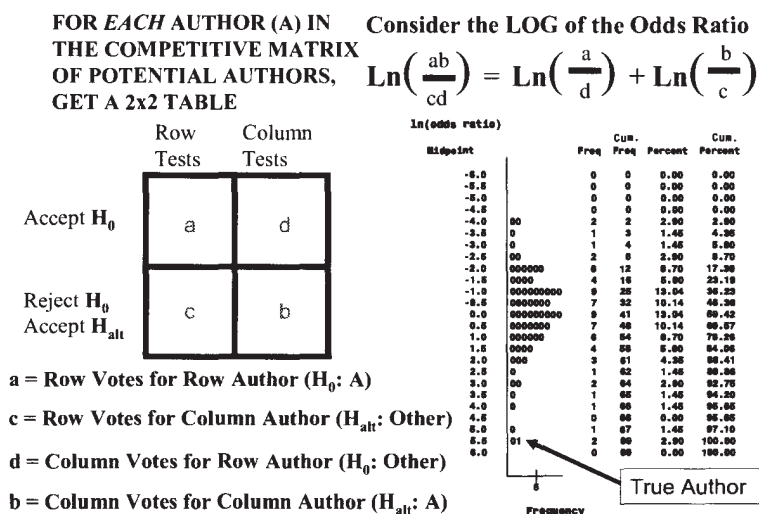


Figure 6: The Log of the Odds Ratio

4.2 The Log of the Odds Ratio

Considering this two-by-two table, a single value (statistic) with the desired properties is the log of the odds ratio, that is, $\log(ab/cd)$. A histogram for $\log(ab/cd)$ for the data displayed in the plot of Figure 5 is presented in Figure 6. Note that the true author (symbol '1') and another author (symbol '0') appear tied for the highest value of $\log(ab/cd)$. These two authors are out in the far tail of the distribution of $\log(ab/cd)$ values for all of the 69 authors who are in the Competitive Matrix.

Figure 6 parses the $\log(ab/cd)$ value into the sum of two log odds values:

$$\log(ab/cd) = \log(a/d) + \log(b/c).$$

This breakdown of the log of the odds ratio helps us to appreciate the power for true author identification in this quantity. Figure 7 plots a versus d for the true authors associated with all characters in the held out two London letters for all authors in our database. Note in Figure 7 the strong pattern of plotted points with $a > d$; such points will give large values of $\log(a/d)$.

Alternatively, Figure 8 plots a versus d for the authors who are not the true authors for the same

characters in the held out two London letters. Note in Figure 8, the symmetric pattern of plotted points around the 45 degree line where $a = d$; for such points the values of $\log(a/d)$ will be symmetric around 0.

Figure 9 plots c versus b for the true authors associated with all characters in the held out two London letters for all authors in our database. Note in Figure 9, the strong pattern of plotted points with $b > c$; such points will give large values of $\log(b/c)$.

Alternatively, Figure 10 plots c versus b for the authors who are not the true authors for the same characters in the held out two London letters. Note in Figure 10, the symmetric pattern of plotted points around the 45 degree line where $b = c$; for such points the values of $\log(b/c)$ will be symmetric around 0.

Figure 11 presents a histogram of $\log(ab/cd)$ for all authors (both true authors and others) from the Competitive Matrices for each character in all held out London letters. Note the smooth, symmetric structure of the histogram. The parts of the histogram associated with values for true authors are darkened in; note that the values for true authors are predominantly positive whereas the values for other authors are randomly positive or negative.

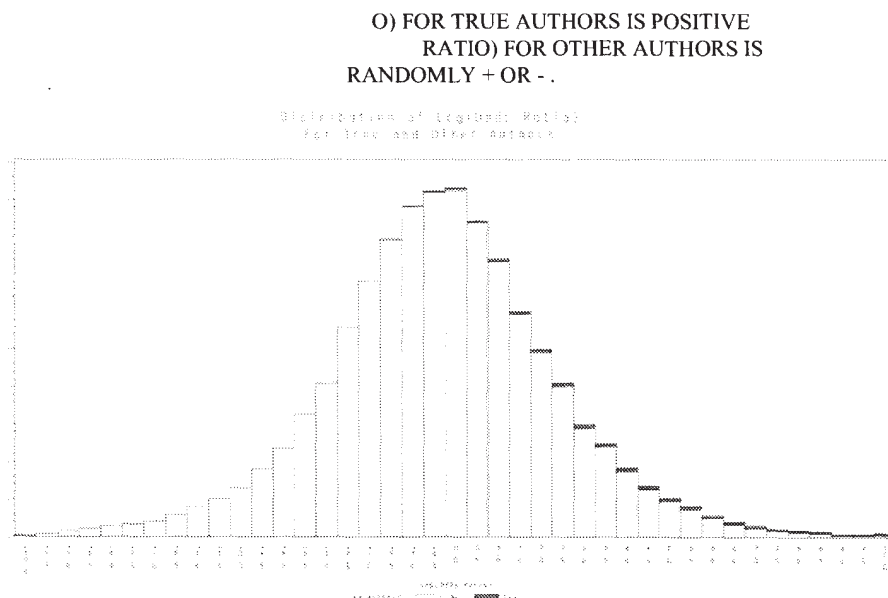
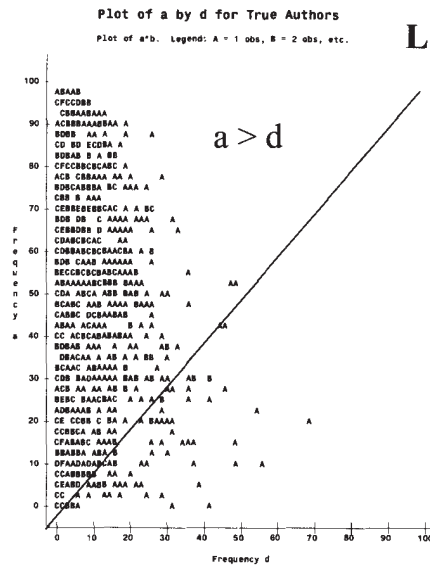


Figure 11: The distribution of the Log of the odds ratio

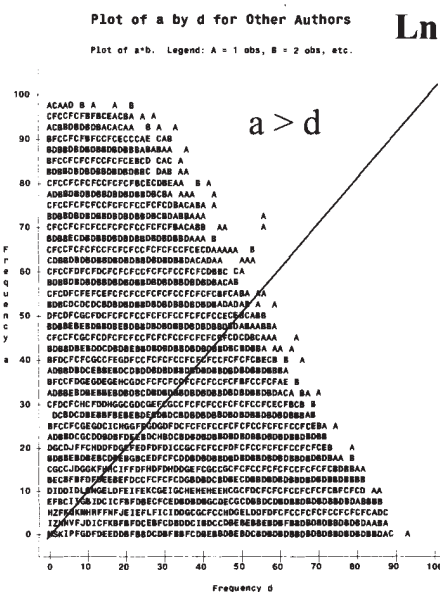


$$\text{Ln}\left(\frac{ab}{cd}\right) = \text{Ln}\left(\frac{a}{d}\right) + \text{Ln}\left(\frac{b}{c}\right)$$

	Row Tests	Column Tests
Accept H_0	a	d
Reject H_0 Accept H_{alt}	c	b

a/d = Odds that an Accepted H_0 is from a Row Test

Figure 7: Plot of a versus d for True Authors



$$\text{Ln}\left(\frac{ab}{cd}\right) = \text{Ln}\left(\frac{a}{d}\right) + \text{Ln}\left(\frac{b}{c}\right)$$

	Row Tests	Column Tests
Accept H_0	a	d
Reject H_0 Accept H_{alt}	c	b

a/d = Odds that an Accepted H_0 is from a Row Test

Figure 8: Plot of a versus d for Other Authors

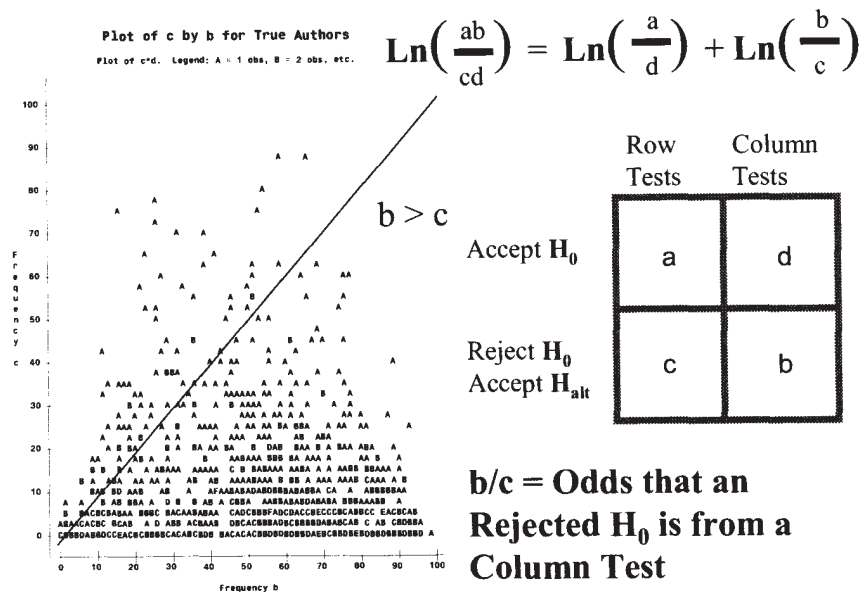


Figure 9: Plot of b versus c for True Authors

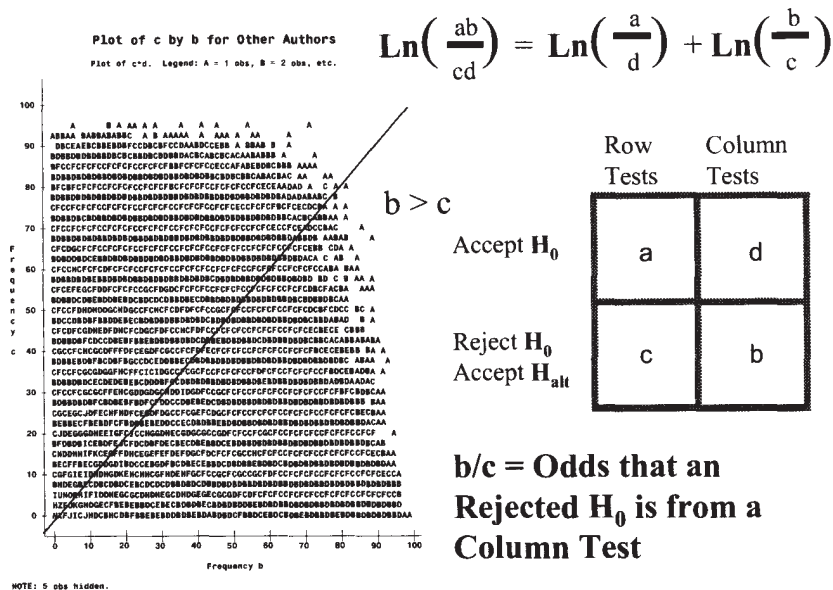


Figure 10: Plot of b versus c for Other Authors

If we take multiple characters from a single author's held out London letters, then the sum of the $\log(ab/cd)$ values for these letters assesses the aggregate likelihood of authorship for the multiple characters. A true author will have a preponderance

FOR A PARTICULAR AUTHOR AMONG A GROUP OF 100 AUTHORS,

- TAKE 15 CHARACTERS AT RANDOM FROM THE *HELD OUT* PARAGRAPHS.
- TEST EACH OF THE 15 CHARACTERS
- SUM THE TEST CHARACTER LN(ODDS RATIO) VALUES FOR EACH OF THE 100 AUTHORS



(number of characters) from 5 to 30. The two graphs in Figure 13 summarize the accuracy results. The first graph presents the average percent of samples for which the true author was in first place, second place, etc., when authors were ranked by decreasing sums of logs of the odds ratio. The second graph presents the number of samples for which 96 out of 100, 97 out of 100, etc., true authors had the largest sum of the logs of the odds ratio thereby selecting them as our candidate for true author.



Figure 13: The accuracy of the Biometric Identification

5 Conclusion

We have demonstrated that the quantification of handwriting by the assignment of a graph to each character provides a very powerful basis for biometric identification. The concepts of Biometric Kernels and Competitive Matrices are used to build a biometric identification database for use in determining the identity of an unknown author of a writing sample.

We are applying the technology described in this article to applications in the areas of Individual Identification, Document Clustering and Forensic Document Examination.

Tradeoff Studies about Storage and Retrieval Efficiency of Boundary Data Representations for LLS, TIGER and DLG Data Structures

David Clutter **Peter Bajcsy**
National Center for Supercomputing
Applications (NCSA), University of Illinois at
Urbana-Champaign (UIUC)
605 East Springfield Avenue, Champaign, IL
61820

Abstract

We present our theoretical comparisons and experimental evaluations of three boundary data representations in terms of storage and information retrieval efficiency. We focus on three boundary data representations, such as, location list data structure (LLS), digital line graphs (DLGs) and topologically integrated geographic encoding and referencing (TIGER) data organizations. These three boundary data representations are used frequently in the GIS domain, and are known as ESRI Shapefiles (LLS), the SSURGO DLG-3 soil files (DLG), and the U.S. Census Bureau 2000 TIGER/Line files (TIGER). Boundary information is viewed as an efficient representation of image documents describing spatial regions. The goal of our work is to study the impacts of choosing boundary information representation on document image management and information retrieval, as well as to improve our understanding of the processing noise introduced during representation conversions.

Our storage and retrieval efficiency tradeoff evaluations are based on load time, computer memory, and hard disk space requirements. The experimental measurements are obtained with test data sets derived from the SSURGO DLG-3 soil files and the U.S. Census Bureau 2000 TIGER/Line files. Based on our experiments, we concluded that LLS files will provide the fastest boundary retrieval (40 times faster than TIGER and 2.5 times faster than DLG) at the price of file size (storage redundancy for LLS files is between 70% and 180% in our experiments). DLG format offers a smaller file size, but is less efficient for boundary retrieval. TIGER format also offers a compact physical representation, at the cost of more processing for boundary retrievals. These findings provide quantitative support for institutional document image management decisions.

1 Introduction

Boundary information is viewed as an efficient representation of image documents describing spatial regions and is important for document image management and information retrieval. Boundary information represents one type of vector information [1, Chapter 15]. Boundaries (or contours or outlines) are mathematically described as convex or non-convex polygons. One boundary can be formed by a set of polygons, for instance, a donut shape boundary. Each polygon consists of an ordered set of points or vertices. In most GIS applications, points are georeferenced so that boundary information can be integrated with raster information. GIS examples of boundary information would be parcels, eco-regions, watersheds, soil regions, counties, Census tracts or U.S. postal zip codes. The goal of our work is to study the impacts of choosing boundary information representation on document image management and informational retrieval, as well as improve our understanding of the processing noise introduced during representation conversions.

In general, boundaries can be spatially related or can be spatially independent. The spatially related boundaries can be either partially overlapping or totally overlapping, such as, one contour being a subset of another set of boundaries. For example, watershed and U.S. postal zip codes boundaries are spatially independent while the U.S. Census Bureau tracts and blocks are spatially dependent in such a way that every tract is formed by a set of blocks.

Given the variety of boundary information, researchers have developed numerous file formats for storing boundary information. These file formats are designated in general for storing any vector data. Vector data contain points, lines, arcs, polygons or any combinations of these elements. Any vector data element can be represented in a reference domain defined by a latitude/longitude, UTM or pixel

coordinate system. The challenge in storing vector data is to organize the data such that the positions and geographic meanings of vector data elements are efficiently stored and easily extracted.

Among all vector data representations in files, the following data structures have been used frequently: location list data structure (LLS), point dictionary structure (PDS), dual independent map encoding structure (DIME), chain file structure (CFS), digital line graphs (DLGs) and topologically integrated geographic encoding and referencing (TIGER) files. For detailed description of each data structure we refer a reader to [1].

The motivation of our work came from the fact that while boundary data types are preferred over raster data types when it comes to storing boundary information, there are multiple memory storage schemes for boundary information, as listed in the previous paragraph. However, choosing the storage scheme that minimizes memory requirements might have a detrimental impact on boundary information retrieval efficiency. Thus, our objective is to evaluate quantitatively the tradeoffs between storage and retrieval efficiency of multiple boundary data representations for LLS, TIGER and DLG data structures. The outcomes of our evaluations are useful for (a) institutional decisions about archiving and retrieving geospatial boundary information, and (b) custom applications that perform processing of large size, geospatial boundary data sets.

In this work, we evaluate three boundary data representations for efficient boundary information storage and retrieval. These three data representations include (1) Census 2000 TIGER/Line files defined by the U.S. Census Bureau and saved in topologically integrated geographic encoding and referencing (TIGER) data structures, (2) shapefiles defined by the Environmental Systems Research Institute (ESRI) and stored in location list data structure (LLS) data structures, and (3) SSURGO DLG-3 soil boundaries prepared by the United States Geological Survey (USGS) and stored in digital line graphs (DLGs) data structures. We overview the three data file formats first. Next, we present our experimental results, and pairwise analysis of experimental results. Finally, we summarize our work and add a few observations about other possible trade-off metrics that might be considered for making institutional decisions.

2 SSURGO DLG-3 Soil Files

The Soil Survey Geographic (SSURGO) Digital Line Graphs (DLG) files provide geographical information on the boundaries of soil types [9], [10], [11]. The SSURGO data sets provide the highest spatial resolution of soil type information among the three soil

geographic data bases, such as, the Soil Survey Geographic (SSURGO) data base, the State Soil Geographic (STATSGO) data base, and the National Soil Geographic (NATSGO) data base.

2.1 File Format Description

DLG File Structure: The DLG file structure is designed to support all categories of spatial data that can be represented on a map. Three distinct types of DLG are defined. Large-scale DLG data is digitized from 1:24,000-scale USGS topographic quadrangles (SSURGO). Intermediate-scale DLG data is digitized from 1:100,000-scale USGS quadrangles (STATSGO). Small-scale DLG data is digitized from 1:2,000,000-scale sectional maps (NATSGO). Furthermore, three levels of DLG data were defined in terms of the number of attributes. It was found that the widest user community would be served by DLG Level 3 (DLG-3) data, which allows for the highest resolution (SSURGO) and highest number of attributes to be encoded (Level 3). The lesser levels of DLG encoding are unused. DLG-3 encodes attributes using two codes: a major code and a minor code. Similar attributes share a major code. The SSURGO DLG-3 soil database uses both the major code and minor code to encode the primary key into a relational database to further describe an area.

We gathered the SSURGO DLG-3 files for a few counties in Illinois from <http://www.ncgc.nrcs.usda.gov/branch/ssb/products/ssurgo/data/index.html>. There are two files for each county, such as, *dlg.zip* (digital line graph or DLG) and *tab.zip* (ASCII attribute data available in Microsoft Access 97 or later template database). The files contain soil boundaries of 18,000 soil series recognized in the United States. For the integration purposes, we have explored the following information from the DLG-3 documentation: (a) file naming convention, (b) spatial resolution, (c) spatial accuracy, (d) geographic coordinate system and (e) storage format. In terms of file naming convention, the *dlg.zip* file would contain files with the following suffixes:

- af - soil polygon DLG-3 file,
- aa - soil polygon attribute file,
- sf - special soil point and line DLG-3 file, and
- sa - special soil point and line attribute file.

Regarding spatial resolution, soil survey is mapped at a scale ranging from 1:12,000 to 1:63,360. The SSURGO soil boundaries meet the accuracy standards for the USGS 7.5-minute topographic quadrangles or the 1:12,000 or 1:24,000 orthophotoquads. Finally, the storage format is Digital Line Graph optional format with the attribute table data archived in ASCII table or INFORMIX table format.

DLG Georeferencing Information: In terms of a geographic coordinate system, coordinates are derived from the North American Datum of 1983 reference

system that is based upon the Geodetic Reference System of 1980. DLG data are recorded in either the Universal Transverse Mercator (UTM) system or are projected using the Albers Equal-Area Conic projection. SSURGO DLG-3 data are normally reported in the UTM system. STATSGO DLG data are reported using the Albers Equal-Area Conic projection.

DLG Data Description: DLG data are reported as nodes, lines, and areas. Lines are composed of a series of nodes, and areas are composed of lists of lines (or optionally nodes). The composition of an area or a line can be encoded either as a list of the nodes that make up the element, or as a list of points. Due to this hierarchical structure, each element must be encoded with a unique identifier.

A node is a coordinate on a map. Each node has an Easting value and a Northing value in the UTM coordinate system. Nodes define the points of each line and are encoded with (1) a unique identifier and (2) the coordinates that the node represents. Nodes can also be encoded with attributes, if desired. Additionally, the DLG format specification allows for a list of all lines that begin and end at a node to be encoded in the record for a node. This is redundant information, however, for it is reflected in the line records as well.

Lines are a series of nodes. Each line is encoded with a unique identifier, as well as its starting node and ending node. The coordinates that a line follows are also listed. In addition, a line can be encoded with attributes.

An area is an enclosed section. Areas can be encoded as either a sequence of lines or a sequence of nodes. When encoded as a sequence of lines, the area will contain a list of the lines that the boundary of the area follows. This list contains the unique identifier for each line; negative values signify that the points in the line should be reversed. Islands within an area are delimited by a '0' in the list of lines. Areas are specified in a clockwise direction around the perimeter of the area, and islands are specified in a counter-clockwise direction. In addition, an area can be encoded with major and minor code pairs. When encoded as a sequence of nodes, the area will contain a list of the nodes make up the boundary of the area.

Software Development for SSURGO DLG-3 Files: First, we implemented a loader for SSURGO DLG-3 files and added it to the list of other GIS files supported by the NCSA I2K software package [5]. Next, we extended our 2D visualization to support visualization SSURGO DLG-3 files. We can visualize multiple georeferenced vector data structures (boundaries and sets of points) simultaneously. Third, we develop a conversion function from SSURGO DLG-3 data structure to ESRI Shapefile (LLS) data structure that was needed for tradeoff comparison purposes.

The details of boundary information retrieval from DLG-3 file format can be described as follows. The

DLG file format defines objects using a hierarchical structure. The lowest objects in the hierarchy must be retrieved prior to higher objects in the hierarchy. Thus, in order to retrieve an area, all lines that make up the area's boundary must be retrieved beforehand. Therefore, the DLG-3 loader in I2K will read all the defined lines first. The lines are kept in a lookup table, and indexed by their unique identifier for later use. The size of this structure is directly proportional to the number of lines.

Next, the areas are retrieved by populating I2K defined data structures for boundary information denoted a ShapeObject. In the ShapeObject, an area has a list of the coordinates that make up its boundary. This list is dynamically constructed when reading an area. Areas that share a boundary will have copies of the common coordinates. Once all areas have been read and processed, the lookup table containing the lines can be safely discarded. Finally, the coordinates for the areas are copied into a ShapeObject.

2.2 Theoretical Evaluation

Memory requirements: The DLG-3 optional format used in SSURGO soil databases provides a compact physical representation of the boundaries of soil types over a geographic area. There is little redundancy in a DLG-3 file. Each area is a list of lines that do not cross. The lines must share the same endpoints in order to fully define an area. Thus, the only redundant information is the endpoints of each line. The points of adjacent polygons will be specified only once; in a line, or series of lines. The boundary between adjacent, non-overlapping polygons is represented as the same series of line identifiers in the file. In addition, representing all data in a fixed-length ASCII form makes for smaller, highly compressible files. Abundant white space exists in DLG-3 files to maintain the fixed length. Typical compression algorithms will compress a series of identical characters efficiently. Thus, when a DLG-3 file is subject to compression, the white space will compress well.

Boundary information retrieval requirements: The boundary information retrieval from DLG-3 file format can require significant processing resources. All boundary coordinates are stored as ASCII characters in a DLG file. In order to use the polygons specified in a file, each coordinate must be converted into a native numeric value. This conversion can be quite costly, and takes approximately 27% of the time to load SSURGO DLG-3 files in I2K.

3 Census 2000 TIGER/Line Files

The Census 2000 TIGER/Line Files provide geographical information on the boundaries of counties, zip codes, voting districts, and a geographic hierarchy of census relevant territories, e.g., census tracts that are composed of block groups, which are in turn composed

of blocks. It also contains information on roads, rivers, landmarks, airports, etc., including both latitude/longitude coordinates and corresponding addresses [2]. A detailed digital map of the United States, including the ability to look up addresses, could therefore be created through processing of the TIGER/Line files.

3.1 File Format Description

Because the density of data in the TIGER/Line files comes at the price of a complex encoding, extracting all available information from TIGER/Line files is a major task. In this work, our focus is primarily on extracting boundary information of regions and hence other available information in TIGER/Line files is not described here.

TIGER/Line files are based on an elaboration of the chain file structure (CFS) [1], where the primary element of information is an edge. Each edge has a unique ID number (TIGER/Line ID or TLID) and is defined by two end points. In addition, each edge then has polygons associated with its left and right sides, which in turn are associated with a county, zip code, census tract, etc. The edge is also associated with a set of shape points, which provide the actual form an edge takes. The use of shape points allows for fewer polygons to be stored.

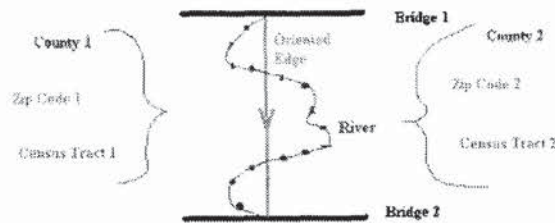


Figure 1: Illustration of the role of shape points.

To illustrate the role of shape points, imagine a winding river that is crossed by two bridges a mile apart, and that the river is a county boundary and therefore of interest to the user (see Figure 1). The erratic path of the river requires many points to define it, but the regions on either side of it do not change from one point to the next, only when the next bridge is reached. In this case, the two bridge/river intersections would be the end points of an edge and the exact path of the river would be represented as shape points. As a result, only one set of polygons (one on either side of the river) is necessary to represent the boundary information of many small, shape defining edges of a boundary.

This kind of vector representation has significant advantages over other methods in terms of storage space. To illustrate this point, consider that many

boundaries will share the same border edges. These boundaries belong to not only neighboring regions of the same type, but also to different kinds of regions in the geographic hierarchy. As a result, storing the data contained in the TIGER/Line files in a basic location list data structure (LLS) such as ESRI Shapefiles, where every boundary stores its own latitude/longitude point, would introduce a significant amount of redundancy to an already restrictively large data set.

In contrast to its apparent storage efficiency, the TIGER vector data representation is very inefficient for boundary information retrieval and requires extensive processing. From a retrieval standpoint, an efficient representation would enable direct recovery of the entire boundary of a region as a list of consecutive points. The conversion between the memory efficient (concise) and retrieval efficient forms of the data is quite laborious in terms of both software development and computation time.

Another advantage of the TIGER/Line file representation is that each type of GIS information is self-contained in a subset of files. As a result users can process only the desired information by loading a selected subset of relevant files. For example, each primary region (county) is fully represented by a maximum of 17 files. Therefore, the landmark information is separate from the county boundary definition information, which is separate from the street address information, etc. Those files that are relevant to the boundary point extraction, and the attributes of those files that are of interest, are the following:

- Record Type 1: Edge ID (TLID), Lat/Long of End Points
 - Record Type 2: TLID, Shape Points
 - Record Type I: TLID, Polygon ID Left, Polygon ID Right
 - Record Type S: Polygon ID, Zip Code, County, Census Tract, Block Group, etc.
 - Record Type P: Polygon ID, Internal Point (Lat/Long).
- We denote this subset of files as "Census boundary records".

3.2 Theoretical Evaluations

This work extends our previous study about the tradeoffs between U.S. Census Bureau TIGER and ESRI Shapefile data representations that are documented in [7].

4 ESRI Shapefiles

A shapefile is a special data file format that stores non-topological geometry and attribute information for the spatial features in a data set. The geometry for a feature

is stored as a shape comprising a set of vector coordinates in a location list data structure (LLS). Shapefiles can support point, line, and area features. Area features are represented as closed loop polygons.

4.1 File Format Description

A shapefile must strictly conform to the ESRI specifications [4]. It consists of a main file, an index file, and a dBASE table. The **main file** is a direct access, variable-record-length file in which each record describes a shape with a list of its vertices. In the **index file**, each record contains the offset of the corresponding main file record from the beginning of the main file. The **dBASE table** contains feature attributes with one record per feature. The one-to-one relationship between geometry and attributes is based on record number. Attribute records in the dBASE file must be in the same order as records in the main file.

All file names adhere to the ESRI Shapefile 8.3 naming convention. The 8.3 naming convention restricts the name of a file to a maximum of 8 characters, followed by a 3 letter file extension. The main file, the index file, and the dBASE file have the same prefix. The suffix for the main file is ".shp". The suffix for the index file is ".shx". The suffix for the dBASE table is ".dbf".

Examples:

1. main file: counties.shp
2. index file: counties.shx
3. DBASE table: counties.dbf

The implementation of shapefile loading, writing and visualization routines was straightforward since the I2K ShapeObject data structure maps directly to the shapefile file organization.

4.2 Theoretical Evaluation

There are numerous reasons for using ESRI Shapefiles. ESRI Shapefiles do not have the processing overhead of a topological data structure such as a TIGER file. They have advantages over other data sources, such as faster drawing speed and edit ability. ESRI Shapefiles handle single features that overlap or are noncontiguous. They also typically require more disk space but are easier to read and write. However, the drawbacks of ESRI Shapefiles are in their storage inefficiency and poor scalability. We will quantify these tradeoffs in the experimental section.

5 Experimental Evaluations

In this section, our goals are (a) to experimentally evaluate the tradeoffs between storage and retrieval efficiency, and (b) to explain the tradeoffs by comparing fundamental format differences. In order to perform experimental tradeoff evaluations, we used two

datasets including (1) the SSURGO soil boundaries for Madison County, IL, stored in DLG-3 file format and (2) the U.S. Census Bureau boundaries of Illinois counties, zip codes, census block and census tracts stored in TIGER/Line file format. The preparation of these two data sets is outlined in Section 5.1. The results of all experiments are provided in Sections 5.2 and include comparisons of DLG & LLS, and DLG & TIGER & LLS. Sections 5.3, 5.4 and 5.5 explain the pair-wise format comparisons based on the experimental results.

5.1 Data Preparation

It is apparent that the experimental evaluations will depend on the size of test data. Ideally, one would like to show results as a function of input file size. However, the practical difficulty arises when one is looking for those test data sets that contain identical boundary information but are represented by LLS, TIGER and DLG files. We were not able to find such files.

We explored the possibility of finding software tools that would convert vector files from one file format to another so that we could create multiple test files with identical boundary information stored in LLS, TIGER and DLG formats. We have concluded that while LLS formats (ESRI Shapefiles) are supported by most GIS software packages, there is a very limited support for DLG and TIGER file formats. This corresponds to our assessment of the implementation complexity to support loading of TIGER, DLG and LLS formats in this order from the most time consuming to the least time consuming. The implementation effort usually doubles when both loading and writing routines have to be supported.

Based on our findings about conversion tools and the availability of GIS software packages at our institution, we created data sets by (1) implementing TIGER to LLS, and DLG to LLS conversions, and (2) using ArcToolBox for LLS to DLG conversion. We created several test data sets that are described next.

In the first experimental tradeoff evaluation, we used a file pair consisting of the original DLG file (SSURGO soil boundaries) and the LLS file converted using I2K. This file pair is denoted as the test data set #1.

In the second experimental tradeoff evaluation, we prepared a triplet of files consisting of (a) the original TIGER files for the state of Illinois, (b) the LLS files obtained by extracting the U.S. Census Bureau boundaries of counties, zip codes, census block and census tracts from the TIGER files and converting them by using our software implementation, and (c) the DLG file converted from the already obtained LLS file using ArcToolBox. This triplet of files provides a test data set for fair performance evaluations in terms of "Total Load Time" and Load RAM Required" parameters. However, this test data set cannot be used for

Table 1: Test data#1: SSURGO Soil Database, Madison County, IL. Loading time includes all SSURGO soil boundaries. Hard disk measurements pertain to all boundaries in the original SSURGO files.

	Total Load Time (s)		Load RAM Required (MB)	Hard Disk (MB)		Number of Nodes
	Zip	Unzip		Zip	Unzip	
LLS (Shapefile)		41.36	290	65	90	2,787,490
DLG	105.72	103.72	380	23	79	2,787,790

Table 2: Test data#2: U.S. Census Bureau 2000 TIGER/Line files for the state of Illinois (102 counties). Loading is constrained to block groups, zcta, census tract, and counties ((Total Load Time and Load RAM Required parameters). Hard Disk and Number of Nodes measurements for LLS and DLG formats contain only block groups, zcta, census tract, and county boundaries, whereas the same measurements for TIGER format include all types of boundary information for the state of Illinois.

	Total Load Time (s)	Load RAM Required (MB)	Hard Disk (MB)		Number of Nodes
	Unzip		Zip	Unzip	
TIGER	1300.2	200	112	940	2,176,719
LLS	12.7	37	27	47	641,955
DLG-3	12.9	52	8	24	457,850

performance evaluations in terms of “Hard Disk” because the TIGER files include all boundary types (including voting districts, and so on), of which four were extracted to LLS and DLG file formats. This file triplet is denoted as the test data set #2.

We expanded the second experimental tradeoff evaluations in Section 5.2 by partitioning the test data set #2. We used sub-sets of the original TIGER files for the state of Illinois in order to vary the number of nodes. In order to explore load time dependency on the number of nodes (boundary points), we selected 1, 2, 3, 4, 10, 15, or 24 counties from the original TIGER files, and formed several triplets of test data sets (TIGER, LLS and DLG). We always chose a subset of counties forming geographically contiguous regions so that neighboring counties would have some overlap of boundary points. This set of file triplets is denoted as the test data set #3.

5.2 TIGER, LLS, and DLG Tradeoff Evaluations

The experimental results of our tradeoff evaluations between storage and retrieval efficiency are presented in Tables 1 and 2. As described in the previous section, the test data sets #1 and #2 {(DLG, LLS) and (TIGER, LLS, DLG)} were formed from the original DLG and TIGER files by converting them into other file formats

using Arc ToolBox and our software. Each file format was then read in separately, and the storage and loading measurements were recorded in Tables 1 and 2.

Before explaining the experimental results by comparing pairs of file formats presented in Sections 2, 3 and 4, we posed the following two questions. First, is there any dependency of storage on the boundary content? In other words, if we had a file with watershed and zip code boundaries, would the results be different from evaluating Census tracts and blocks, and how? Second, can we predict the total load time as a function of the number of polygons/nodes without exhaustive experimentation? Or in other words, what would be the dependency between boundary information retrieval and the number of retrieved nodes?

Storage Dependency on Boundary Content: The answer to the first question is related to the amount of boundary overlap. Ideally, one would experiment with sets of boundaries that span cases from a zero overlap (e.g., non-adjacent county boundaries) to an overlapping hierarchy of polygons (census blocks, block groups and tracts). Our data sets represent the cases of partial overlap (SSURGO) and large overlap (TIGER) of boundaries. Thus, the experimental results will vary as a function of boundary content in the following way: the more overlapping boundaries, the smaller hard disk requirements for TIGER format in

comparison with DLG and LLS (in this order), and the smaller load RAM requirements for LLS format in comparison with DLG and TIGER.

Our conclusion is supported by comparing the number of loaded nodes versus the number of unique nodes using the test data sets #1 and #3, and by inspecting the LLS files. By evaluating the ratio s of these two numbers (loaded nodes versus unique nodes) using the test data #2 (partial boundary overlap), we obtain s equal to 2.02 (5630800/2787490). The same evaluation of the ratio s using the data set #3 (large boundary overlap) led to an average ratio value equal to 2.6416. The measurements using the test data set #3 (ZCTA, Block Group (BG), Census Tract (CT), and County boundaries for 1, 2, 3, 4, 10, 15, and 24 Illinois counties) are shown in Figure 2.

We took additional measurements to compute the ratio s for (a) watershed and county boundaries ($s = 84,601/47,636=1.776$), and (b) watershed and ZCTAs boundaries ($s=344,533/201,767=1.708$). We observed that approximately 70% of the points in both (a) and (b) are shared between multiple boundaries. Thus, the inefficiency of LLS format due to the duplicate points of neighboring boundaries would not decrease below $s=1.7$ for the test data.

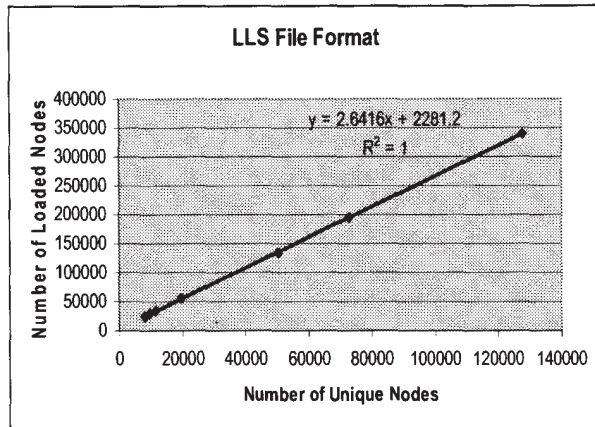


Figure 2: Storage efficiency measurements of LLS files using the test data set #3 (Hierarchical boundary content). The points correspond to evaluations for data sets with boundaries for 1, 2, 3, 4, 10, 15, and 24 Illinois counties.

Boundary Information Retrieval Dependency on Number of Nodes: In order to answer the second question about the relationship between a load time and a number of nodes, we divided the Total Load Time into four components: t_1 , t_2 , t_3 and t_4 (see Equation below and Figure 3). The first component t_1 corresponds to the time to construct polygons from an ordered list of edges. The second component t_2 is for the time to create an ordered list of edges from an unordered set of edges. The third component t_3 represents the time to convert ASCII characters to numeric type values. The last component t_4 is the time

to load any sequence of bytes (ASCII characters or binary values) from a file. We introduce these time components based on our understanding of the three vector file formats.

$$\text{Total Load Time} = t_1 + t_2 + t_3 + t_4 \quad (1)$$

The zero and non-zero time components are summarized for each file format in Table 3. The total load time as a function of the number of nodes can be predicted by knowing that the time components t_1 , t_2 , t_3 and t_4 are linear with the increasing number of nodes. The quadratic dependency of the time component t_2 (creation of ordered list of edges) as a function of the increasing number of nodes is avoided by the fact that the unordered edges are grouped by counties rather than by states. Based on our empirical observations, $t_1 < t_2 < t_3$ for a fixed number of nodes, which leads to superior total load time for LLS format in comparison with DLG and TIGER formats (in this order). Our theoretical predicted Total Load Time as a function of the number of nodes is shown in Figure 3 and is independent of test data sets (addressed as the question number 1 above).

Table 3: Total Load Time decomposition.

Total Load Time=Sum(t_i)	t_1	t_2	t_3	t_4
LLS	X	0	0	X
DLG	X	0	X	X
TIGER	X	X	X	X

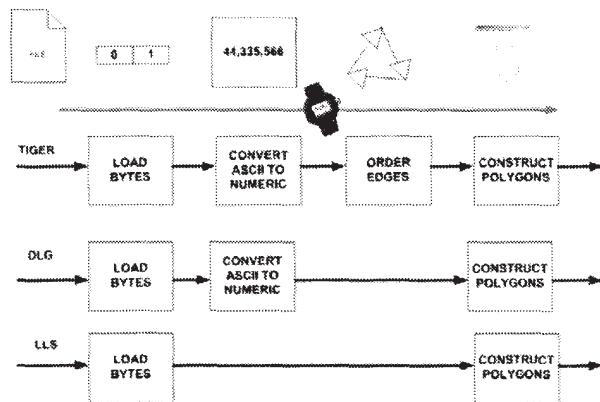


Figure 3: Total Load Time decomposition for TIGER, DLG and LLS file formats.

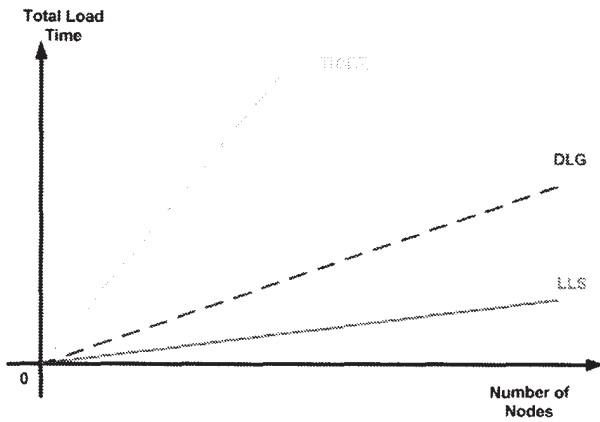


Figure 4 : Theoretically predicted Total Load Time as a function of the number of nodes.

We have obtained experimental measurements that support our theoretically predicted Total Load Time dependency on the number of nodes using the test data set #3. Figure 5 shows our measurements and linear trends, where the points correspond to data sets with boundaries for 1, 2, 3, 4, 10, 15, and 24 Illinois counties. These supporting measurements for “Total Load Time” and “Load RAM Required” were calculated by averaging three runs to load the ZCTA, Block Group (BG), Census Tract (CT), and County boundaries for each data set. The total number of nodes and the number of unique nodes were measured (a) by counting nodes inside of our software developed for loading LLS and DLG files, and (b) by summing end points and shape points for TIGER files according to the accompanying TIGER documentation. While TIGER files do not contain any duplicate points, LLS duplicate points were found using a hash table in our software.

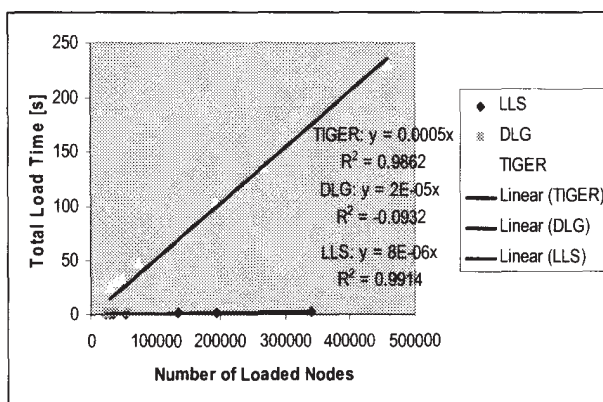


Figure 5 : Total Load Time vs. Number of Nodes for 1, 2, 3, 4, 10, 15, and 24 counties with a best-fit line.

According to Figure 5 and based on our test data set #3, the total loading time for TIGER files is approximately 40 times slower than for LLS files, and

the total loading time for DLG files is about 2.5 times slower than for LLS files. We collected measurements for only 1, 2, 3 and 4 county aggregations in the case of DLG format because the data preparation is very time consuming.

5.3 DLG and LLS Comparisons

The DLG optional and LLS (or ESRI Shapefile) formats specify boundaries over an area. Both formats have geographic information that allows the boundaries to be geo-referenced with other data sources. The formats differ in how the data is structured and stored.

The first primary difference between DLG and LLS is that DLG is stored in an ASCII format, while LLS is stored in a binary format. DLG files are comprised of ASCII characters organized into fixed-length logical records of 80 characters. When loading a DLG file, all data contained within must be converted to native data types. For example, a coordinate is stored as the ASCII characters “4598829.0” in the file. This must be read in and converted to its numeric value. ESRI Shapefile, on the other hand, stores the data as a series of bytes that can be quickly converted to a data type. For the previous example, the value “4598829.0” would be stored as 8 bytes that can be directly converted into a numeric value. However, it may be necessary to reverse the order of the bytes to account for the byte order (little or big endian). The reading (and possible reversing) of bytes for a shapefile is far simpler than the ASCII-to-native transformation needed for DLG.

This primary difference in representation (ASCII vs. binary) greatly affects the loading times of the two approaches. Each entry in a DLG soil database must be read individually, and then converted to a numeric value. This is the most time-consuming operation when loading the data, typically over 25% of the loading time of a DLG file. Loading ESRI Shapefile, however, is much quicker. It is simply reading a series of bytes from a file, with little conversion needed. This quickness comes at the price of a larger file size for the ESRI Shapefile. In an examination of one county, the DLG data needs approximately 79 MB of disk space uncompressed, 23 MB compressed. The ESRI Shapefile, on the other hand, needs 90 MB of disk space when uncompressed, and 65 MB when compressed. These results are summarized in Table 1 and Table 2. The difference in compressed sizes between the two encodings is attributable to their physical representations. DLG data contains fixed-length records with white space between elements to maintain the fixed length. This white space is insignificant and can be easily compressed. On the other hand, all binary data in an ESRI Shapefile are significant and cannot be easily compressed.

The second difference between DLG and LLS is the way how the data in a file are structured. DLG format uses nodes, lines, and areas to define its polygons. In

each of the SSURGO DLG datasets examined so far, nodes have not been used to define lines or areas. The lines are a series of coordinate values, and the areas have a list of the lines that make up the area. On the other hand, LLS format lists the bounding box and the points for each boundary contained within it. DLG format makes more efficient usage of space; areas that share lines will both reference the same line, while in a shapefile, each coordinate, including coordinates shared between different boundaries, is explicitly listed. In addition, this difference makes it necessary to first read all the lines in a DLG file before reading in the areas, because the areas are made up of a list of the lines. The lines have to be kept in a lookup table, and areas cannot be fully processed until all lines have been read.

The consequence of the second difference between DLG and LLS is that different data structures have to be used when loading these files. Our goal is to have one ShapeObject that contains all the polygons in a soil database. DLG format gives no hint as to how many points will be needed to store all the polygons in the DLG file. Furthermore, it does not give the bounding box for each polygon. In contrast, ESRI Shapefile stores these values so that it is possible (a) to pre-compute the space requirements needed and (b) to allocate arrays to hold the data when loading a Shapefile. With DLG, however, it would only be possible to pre-compute the sizes by reading in all data files twice. One time to determine the sizes, and one time to actually read in the data. In addition, the bounding box for each polygon is not stored in DLG, and must be found while reading in the coordinates of each area. This requires comparisons for each coordinate to find the bounding box. In our implementation, expandable arrays (or vectors) were used so that the files only had to be read in once. Then, once fully read, the data are copied into an array in the ShapeObject, of the exact size needed. The problem with this approach is that when the copy is made, two arrays must exist in memory. The first will be the array that contains the vector data. The second will be the new ShapeObject array to copy the contents of the vector into. This causes the memory requirements of DLG-3 files to balloon to twice the total necessary size in the worst case, when copying all the individual points of all the polygons into one ShapeObject.

The third difference between DLG and LLS is related to georeferencing information. SSURGO DLG files are stored as quarter-quadrangles. Each quadrangle represents 7.5 minutes of a degree of longitude and latitude. It is necessary to load 64 individual files to represent a one degree block. ESRI Shapefile does not need to be represented this way. However, Shapefiles could be stored in this way, if desired. All coordinates in SSURGO DLG files are stored in UTM format. This causes problems when geo-referencing the boundaries in I2K because the state of Illinois is located in both UTM zone 15 and UTM

zone 16. The solution was to immediately translate the UTM coordinates to latitude and longitude. Over 29% of the time to load a SSURGO DLG file was spent in the conversion from UTM coordinates to latitude and longitude. Each DLG file contains the UTM zone in the header information. ESRI Shapefile normally contains latitude and longitudinal geo-referencing information. No conversion was required when loading the shapefile in I2K. A potential drawback of the ESRI Shapefile format is that there is not a standard way to define the projection used in for the coordinates. DLG has a value in the header to signify if UTM or Albers projection is used. Also, some of the projection parameters are stored in the header of a DLG. Shapefiles, on the other hand, do not store projection information. This information could be stored with the meta data for a shapefile, but it is not required. This makes it difficult to distribute shapefiles with geo-referencing information other than standard latitude and longitude.

5.4 DLG and TIGER Comparisons

DLG and TIGER offer similar methods to encode vector data. TIGER's use of an edge with shape points corresponds directly to DLG's use of lines and coordinates. Likewise, a TIGER polygon is comprised of a series of edges, and a DLG area is made up of a series of lines. This provides a compact, human-readable representation of the vector data.

The two formats differ in the type of data that are encoded. DLG format typically encodes one layer of data in a file, such as the soil types used by SSURGO. Other layers, such as water boundaries, are encoded in separate files. This scheme introduces some redundancy between the layers. Layers are unrelated to one another, and any shared boundaries will be specified in each layer. For example, a soil layer encoded as a DLG may have boundaries defined along a river. A layer containing bodies of water may share the same boundaries, but the points will be specified again because the soil layer is unrelated to the body of water layer in DLG. TIGER format, on the other hand, groups all edges together, regardless of layer. The different metadata files are used to determine which edges to use. This format allows for less redundancy.

Polygons are retrieved very differently by the DLG and TIGER loaders. DLG format specifies the exact boundaries for each polygon. A list of lines defines the exact border of a polygon, and the lines are in the proper sequence. Since the lines appear in the proper sequence, the polygon can be quickly constructed after all line retrieval. In contrary to DLG format, the boundaries stored in TIGER format must be found programmatically. Each edge is labeled with the polygons that appear on the left and right of the edge. To construct a polygon A, you must first find all edges that border the polygon A. The edges only define the

end points of each edge, and not the order in which the edges should be connected. So the boundary of polygon A must be constructed programmatically by comparing the end points of each edge. Thus, the TIGER polygon construction is far more complex and time-consuming than the DLG polygon construction.

5.5 TIGER and LLS Comparisons

One can derive TIGER and LLS comparisons from the description provided in Sections 2, 3 and 4 that compare DLG and LLS, and TIGER and DLG formats. Since the experimental tradeoff evaluations of TIGER and LLS are summarized in Table 2, we devoted this section to the implementation of TIGER to LLS conversion.

The underlying principle of the conversion process from TIGER/Line files to ESRI Shapefiles could be compared to sorting points according to the order of boundary edges. This is illustrated in Figure 4. In reality, the conversion process begins by loading the raw TIGER/Line files into 2-D table-like data structures by making use of manually developed meta data files. Since the TIGER/Line files are fixed-width encoded flat files, meta data is necessary to define the indices of the first and last characters for each attribute in the lines of the flat file. This information, the attributes' names, and their type (integer, floating point number, string, etc) come from meta data files provided by the Census Bureau. The final piece of information contained in the meta data file is a "Remove Column" field, which dictates whether or not the attribute will be dropped from the table as it is read in. Attributes that are not used during the processing are removed early on for the sake of memory efficiency. The meta information for each Record Type is stored in a comma-separated-value (csv) file, which can easily be parsed into a table object, then accessed in that form by the routine that parses the main data file.

Once the TIGER/Line data are in the form of tables, they are streamed through a complex system of procedures, including conversion to several intermediate data structures, before being inserted into Hierarchical Boundary Objects (HBoundary) [7]. Each HBoundary represents one type of region (county, census tract, etc) for a single state. It can also be viewed as one master list of boundary points that all boundaries reference by pointers. The RAM memory savings of HBoundary versus ShapeObject for each point that is shared by two counties, two census tracts, and two block group boundaries is 30 bytes. For the state of Illinois, this optimization translated into a 38% reduction in memory usage (16.45 MB versus 26.64 MB).

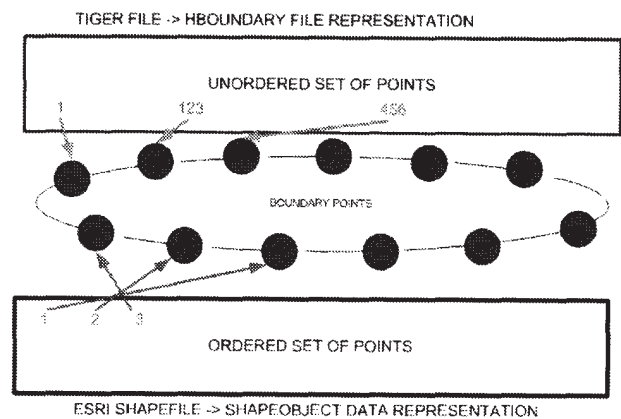


Figure 6: The TIGER/Line to ESRI Shapefiles conversion of boundary representation can be viewed as a transformation from an unordered set of points to a clock-wise ordered set of points.

Finally, the HBoundary object is converted into LLS format by constructing all polygons. The resulting LLS format file was tested by loading it into the commercial ArcExplorer software package [3]. For our experimental tradeoff evaluations, we extracted only a selected subset of Census boundary records from the Census 2000 TIGER/Line files. Thus, it is hard to evaluate loading RAM requirements for TIGER and other two formats since the HBoundary object contains all hierarchical boundaries and their associated information, while the converted LLS file contains only four types of boundaries (counties, ZCTAs, blocks and tracts) and extracted information about region names, neighboring regions to each boundary, and an internal point of each region.

6 Summary

In this paper we have investigated the storage and retrieval efficiency tradeoffs between the ESRI Shapefile (LLS), DLG, and TIGER formats. LLS files will provide the fastest method for boundary retrieval (40 times faster than TIGER and 2.5 times faster than DLG). All boundaries are stored in a binary format for quick retrieval. This speed comes at the price of file size. Each boundary in a LLS file contains all the points that make up the boundary. This introduces storage redundancy (between 70% and 180% redundancy in our experiments) since boundaries can be shared between different polygons. Digital Line Graphs reduce the amount of redundant data. This reduction is tempered by the need for more retrieval processing per boundary. The TIGER format further reduces the amount of data. TIGER format is the most compact representation that comes at the cost of the highest boundary retrieval requirements. Detailed information about these results can be found in Reference [12].

Our goal was to evaluate numerically the trade-

offs between storage and boundary retrieval requirements for the three vector files. The measurements about “Total Load Time”, “Load RAM Required” and “Hard Disk” as a function of “Number of Loaded/Unique Nodes” were used as our metric to demonstrate the trade-offs. Our measurements support the existing knowledge about the choice of a file format depending on the data content that is mapped to boundary overlaps. However, there are other metrics that might affect institutional decisions as well, and were not included in this study. We could enumerate a few metrics, such as (1) a cost of storage media and RAM, (2) a cost of software development to support complex file formats, (3) a preservation of storage media, (4) an availability of software tools for ingesting and processing certain file formats, or (5) an open source implementation of software tools that would allow tracking discrepancies in file format interpretation (loading) and replication (writing). While we did not quantify the additional possible metrics, we have made the following observations. First, numerous software tools support the ESRI Shapefile format whereas not many tools work with Digital Line Graphs or TIGER files. Second, the amount of time we have spent implementing the LLS, DLG and TIGER file format loaders was increasing in the order of the listed file formats. We hypothesize that the increase is almost linear but it becomes quadratic as the file format is too complex to track and eliminate software bugs. Finally, the cost of storage and RAM has been rapidly decreasing over the last decade. We could not foresee the future technological advancements of storage media that would favor one file format over another.

7 Future Directions

One would like to incorporate the effects of computer clusters and mass storage systems on the storage versus boundary retrieval efficiency tradeoff evaluations for LLS, TIGER and DLG data structures. Our tradeoff study thus far has been in an isolated workstation environment. The results of our tradeoff study could differ when a very large cluster or mass storage system is used. We have identified several directions that further research could take.

First, investigate the effect of computer clusters on boundary retrieval efficiency assuming distributed or centralized locations of a large number of boundary files. The benefit of using a computer cluster would come from parallelization of loading and boundary reconstruction tasks.

Second, empirical results and theoretical analyses from our research thus far have shown file size (computer storage size) to be related to the amount of overlap between boundaries. The usage of a mass storage system will add to the time needed to load bytes from a file.

Another component of mass storage systems and

computer cluster environments is the Input/Output (I/O) bandwidth and I/O schemes. While the relationship between I/O bandwidth and boundary retrieval efficiency is straightforward (linear dependency), there are a few questions to ask about I/O schemes. For instance, can more efficient I/O schemes be used to improve boundary retrieval? Would message passing interface input/output (MPI-IO) have any effect? What would be the bottlenecks?

Finally, our ultimate goal is to understand multiple effects of electronic vector files on the archival process. We could mention just a few effects, such as vector file format, data organization and representation, algorithmic parallelization, scalability of vector file loading in terms file size and centralized or distributed file locations, software re-usability, computer platform dependency, computer cluster environments, I/O bandwidth and I/O schemes, and mass storage systems.

Acknowledgements

This research was supported by a National Archive and Records Administration (NARA) supplement to NSF PACI cooperative agreement CA #SCI-9619019. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, the National Archives and Records Administration, or the U.S. government.

References

- [1] Campbell, James B. Introduction to Remote Sensing, Second Edition. The Guilford Press, New York. 1996.
- [2] Miller, Catherine L. TIGER/Line Files Technical Documentation. UA 2000. U.S. Department of Commerce, Geography Division, U.S. Census Bureau.
<http://www.census.gov/geo/www/TIGER/TIGERua/ua2ktgr.pdf>
- [3] ArcExplorer, ESRI web site: <http://www.esri.com>
- [4] ESRI Shape file, File Format Specification, <http://www.esri.com/library/whitepapers/pdfs/shapfile.pdf>
- [5] Bajcsy P. et. al., “Image To Knowledge”, documentation at web site:
<http://alg.ncsa.uiuc.edu/tools/docs/i2k/manual/index.html>.
- [6] Alumbaugh T.J. and Bajcsy P., “Georeferencing Maps with Contours in I2K”, ALG NCSA technical report, alg02-001, October 11 2002.
- [7] Groves P., S. Saha and P. Bajcsy, “Boundary Information Storage, Retrieval, Georeferencing and Visualization,” Technical Report NCSA-ALG-03-0001, February 2003.
- [8] “Data To Knowledge”, software documentation at

web site:

<http://alg.ncsa.uiuc.edu/tools/docs/d2k/manual/index.html>

- [9] "Digital Line Graph References" prepared by the Office of Information Technology,
<http://www.oit.ohio.gov/SDD/ESS/Gis/DigitalLineGraphs.aspx#Documentation>
- [10] Digital Line Graphs from 1:24,000-Scale Maps Data Users Guide I,
<http://www.geodata.gis.state.oh.us/dlg/usrguide/usrguide.htm>
- [11] "US GeoData, Digital Line Graphs" prepared by the U.S. Department of the Interior and the U.S. Geological Survey,
http://www.usgsquads.com/downloads/factsheets/usgs_dlg.pdf
- [12] Clutter D. and P.Bajcsy. "Storage and Retrieval Efficiency Evaluations of Boundary Data Representations for LLS, TIGER and DLG Data Structures," Technical Report NCSA-ALG-04-0007, October 2004.

Demonstration and Poster Abstracts



ABBYY Software House OCR, Forms Processing and Data Capture Technology Review

This abstract will provide an overview of ABBYY's applicable technologies for possible SDIUT05 product demos.

ABBYY Overview

ABBYY is a world leader in the development of recognition, data capture and linguistic technologies. Leading products include the award-winning FineReader line of OCR/ICR/OMR/Barcode recognition software and SDK, the FormReader line of form-processing solutions, and FlexiCapture Studio for data-capturing from various semi-structured documents and forms.

ABBYY FineReader OCR and Fraktur XIX

ABBYY's award-winning FineReader OCR technology converts static paper documents and PDF files into manageable electronic data.

ABBYY FineReader Engine is the most comprehensive data capture and document recognition / conversion SDK. It includes all the technologies needed for developing state-of-the-art intelligent data capture, document recognition, and document conversion systems. In addition to OCR, ICR, OMR and barcode recognition, it also offers image pre- and post-processing, document layout analysis, PDF conversion, and forms processing (for both fixed and semi-structured forms) technologies.

ABBYY FineReader Engine 7.1 offers all of the essential data capture and document conversion tools in a single toolkit. As a result, developers and integrators benefit from working with a single and efficient environment. Developers no longer need to outsource different technologies, learn different programming environments or deal with multiple licensing agreements. In addition, FineReader Engine's modular approach in its run-time licenses ensures that developers only pay for the technology they use in the end application.

ABBYY FineReader Engine is a common platform for ABBYY's recognition technology. FineReader Engine covers all the core technology functions that are inside of ABBYY's flagship products including ABBYY FineReader OCR, ABBYY FormReader and ABBYY FlexiCapture Studio. It also has additional recognition functions specifically designed for international conversion or capturing projects, such as Fast Mode Recognition, Document Analysis for Invoices, and CJK OCR (Recognition of Chinese, Japanese, and Korean).

FineReader Engine is flexible enough to support development of an application of any architecture and scale. Developers can create a client workstation designed from scratch, build a server-based solution, or integrate FineReader technologies into an existing application.

The First Omnifont OCR Fraktur: Ancient Text Recognition without Extensive Training

In January ABBY announced the availability of the first omnifont OCR software for recognition of “Fraktur” or Black Letter print, ABBYY FineReader XIX. More than two years in development, ABBYY FineReader XIX is designed to read ancient texts using “Fraktur” or Black Letter print and those printed between 1600 and 1938. It supports German, English, French, Italian and Spanish languages. FineReader XIX reads Fraktur, Schwabacher and many types of Textura (Gothic) fonts including Roman-type characters no longer in use such as the elongated “s” used in early English or French texts.

Until now, recognition systems have required many hours or days of training to recognize key words and symbols. ABBYY designed FineReader XIX to be an omnifont system for Fraktur and old European language recognition, allowing users to scan and read documents using Fraktur scripts and convert them into digital text with minimal training and dictionary work. ABBYY achieved this by developing a special classifier for Fraktur and special language dictionary models for Fraktur together with ABBYY’s partner ATAPY software. With the aim of delivering high levels of accuracy, ABBYY FineReader XIX has been trained using hundreds of sample documents, including 10 dictionaries and more than 105 books published between 1808 and 1930. In addition, 31,000 pages of text from different sources were used in a test base. Special recognition dictionaries using historical printed dictionaries representing the orthography of the 19th centuries for German were also developed.

ABBYY FormReader

FormReader, ABBYY’s powerful data capture and forms processing solution, is designed to extract information from paper forms and transfer captured data to databases or information management systems.

ABBYY FormReader is an easy-to-use form-processing application designed to extract information from paper forms and transfer captured data to databases or information management systems. FormReader is designed to recognize 177 machine-print and 16 handprint languages, checkmarks and barcodes. It supports XML, multiple file formats and databases as well as allows customizable data export through Automation. FormReader’s built-in form/template designers help reduce efforts in form definition dramatically and increase machine readability and data recognition. A user-friendly interface, including “Scan&Read”, guides users through steps of form processing -- scanning, template matching, recognition, verification and export. ABBYY FormReader offers organizations a fast-deployment and cost-effective solution without requiring extensive IT resources.

ABBYY FlexiCapture Studio Powerful Data Extraction for Semi-Structured Forms and Documents

ABBYY FlexiCapture Studio, backed by ABBYY’s latest FlexiCapture technology, is an add-on option for both the ABBYY FormReader line of form processing solutions and the

ABBYY FineReader line of OCR/ICR/OMR/barcode recognition to process semi-structured forms and documents. With FlexiCapture Studio and FormReader or FineReader Engine, users can automatically recognize information from various semi-structured forms and documents. It allows system developers to integrate this powerful and flexible tool to build or customize their own automated data-capture applications for processing semi-structured forms and documents.

Unlike other semi-structured forms processing technologies, FlexiCapture offers a GUI-based design environment versus a script-based one. This allows users to create a logical description of any semi-structured form or document, then use that description, or FlexiLayout, with an enabled capture application to simplify and expedite the processing task. With added support for new layout formats and pre-processing functions, the latest version of FlexiCapture further increases accuracy in analyzing and reading multiple form layouts, making it an ideal tool for processing multiple types of semi-structured forms and documents.

VERUS

A Middle Eastern Language OCR

Steven G. Schlosser
NovoDynamics, Inc.
123 N. Ashley St. STE 210
Ann Arbor, MI 48104
steve@novodynamics.com

Demonstration Abstract

VERUS is a recently released standalone OCR application for Middle Eastern languages. Four languages, Arabic, Farsi, Dari and Pashto, are recognized by the system as is embedded English and French. The software allows the user to scan individual pages or entire documents from within the application. Degraded pages are automatically cleaned in batch mode without a need for page-by-page parameter adjustment. Page skew and incorrect orientation (e.g., upside down) are automatically corrected. The software can automatically determine the language of a page to circumvent the need for manual pre-sorting. Pages selected for recognition are converted to Unicode text and can be saved in RTF format. Input formats include TIF, JPEG and PDF.

VERUS Professional includes an Application Programmer Interface (API) or SDK for users who wish to integrate the software into a third-party solution. The API allows access to intermediate results for use in controlling a workflow application. Examples of such uses might be integration of cleaned images into a workflow application's interface to improve readability during exploitation, or use of the language identification information to determine proper routing to a linguist. Many other options are also available to meet alternative application requirements. In addition to the output formats of VERUS, the Professional edition includes PDF with hidden text as an option. This enables users to create Middle Eastern language document repositories that can be searched with keywords and key phrases using Adobe Reader.

The VERUS demonstration covers the following typical sequence of software operations: loading an image or workspace, selecting language options, recognition, viewing and editing of output text, display and editing of text zones, export of text and saving of image and text files, and searching PDF with hidden text files.

NovoDynamics, Inc. is an Advanced Image Discovery company located in Ann Arbor, Michigan that specializes in processing, analyzing and retrieving critical information from both pristine and degraded sources. The company currently sells three products, ArborScript™ CRS, ArborScript™ ES and VERUS. ArborScript™ CRS is an SDK for Middle Eastern language OCR for Arabic, Farsi, Dari, and Pashto. ArborScript™ ES is a document exploitation system for Middle Eastern languages that supports image capture, OCR, full-text indexing, and search and retrieval. VERUS is an advanced Middle Eastern language OCR with an intuitive graphical user interface. NovoDynamics' products are differentiated by their superior accuracy and their ability to successfully process large volumes of "real-world" documents, such as smudged faxes, poor quality photocopies, and documents with stains. As a result of NovoDynamics' accomplishments and technology, In-Q-Tel, a venture capital firm funded by the CIA, has made a significant investment in the company and has selected NovoDynamics as an In-Q-Tel portfolio company.

IDG: A Business Information Extraction, Management, and Routing Front-End for Content Management Systems

Vikas Krishna, Savitha Srinivasan, Neil Boyette, Isaac Cheng,
Jeffrey Kreulen and Tapas Kanungo
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
{vikas,savitha,isaacc,nboyette,kreulen,kanungo}@us.ibm.com

ABSTRACT

Content management systems are becoming the standard tool for storing, organizing, and managing corporate information. One of the driving forces behind this is compliance with government regulation. The other driver is pure economic – it is less expensive to manage and manipulate information electronically via content management systems than have it spread over various forms of information systems.

While content management systems are efficient in managing information once it is in the system, getting the information into the content management system is a manual, time-consuming, error-prone, and expensive process. In this article we describe the Intelligent Document Gateway (IDG), which is an information extraction, management and routing front-end for content management systems. IDG can be tailored to extract information from various media and formats, including document images.

Keywords

Information extraction, routing, OCR, business transformation, content management, workflow.

1. INTRODUCTION

Government regulations like HIPAA, Sarbanes-Oxley, Patriot Act, etc., are requiring companies to migrate their corporate information into electronic content management systems. A typical content management (CM) system stores, organizes, manages, and searches information in various formats like relational databases, web pages, voice and email messages, FAX documents, power point and PDF documents, instant messages, etc. Besides the basic functionalities, CM systems are also responsible for providing a) records management, which is responsible for destroying documents as soon as the government-specified retention period is over, or until a time specified by the business retention policy, b) encryption and access control, c) workflow specification, which organizes tasks into a sequential order, and d) storage and archival.

Today CM systems have become quite crucial for businesses to function. However, the CM market is quite fragmented with products like IBM Content Manager, Filenet, Documentum/EMC, InterWoven, Stellent, Mobius, OpenText, etc., each with its proprietary data representation and information exchange formats. Open standards like JSR 170, and their subsequent adoption by the software vendors will eventually make CM systems modular, interoperable, and standardized.

The Achilles heel of CM systems, however, is data entry. While CM systems can handle information that is already in the system, populating CM systems with various types of data is currently out of the scope of CM systems, and is done manually. Manual data entry can be laborious, time consuming, error-prone and expensive. For example, any logic associated with the inbound and outbound document flow as well as the indexing is commonly hard-coded in specific applications. This makes it difficult to identify and change the document-routing logic as required by dynamic business scenarios.

In this article we describe the architecture of the Intelligent Document Gateway (IDG) that works as a front end to the IBM Content Manager and allows users to ingest new information into IBM CM by applying user-specified business rules to the information and then routing the information to the appropriate repository in IBM CM. Metadata is extracted from the incoming documents (e.g. form images) or messages using rules that are authored by the user. The rules are expressed in spreadsheets enabling business users to author and update rules using a familiar paradigm. Updated rules are deployed into a rule engine that handles rule validation and execution. Rule generation is automated using automatic generation of XPath expressions and business rules are validated using XML schema validation. This system is currently deployed in various IBM divisions and potential applications being considered are: citizenship and immigration services, import compliance, commercial invoice management, customs and border protection, etc.

2. RELATED WORK

The notion of routing FAXes has been proposed in the OCR research literature by many researchers (e.g. [L01]). However, these systems focus only on the identification of names and addresses of people, and do not address the extraction of various other business variables like contract amount, point of contact, etc. and CM system issues like retention, encryption, etc. In the commercial world, Kofax distributes Ascent Capture software that extracts metadata from document images. The output can also be integrated into various CM systems including the IBM CM system. However, Ascent Capture extracts text only; application of business logic to the extracted has to be achieved outside the system. Furthermore, the system is setup to work in a batch mode and does not have the dynamic/realtime updating and authoring facilities.

Document management with workflow has been prevalent for several decades and many document management systems incorporate this feature. Aalst et al [AVK01] developed a

workflow language named XRL (eXchangeable Routing Language) for supporting cross-organizational processes. XRL also uses XML for the representation of process definitions and Petri nets for its semantics. Since XRL is instance-based, workflow definitions can be changed on the fly and sent across organizational boundaries. However, these features make cross-organizational workflows susceptible to errors. They also present XRL/Woflan, a software tool using state-of-the-art Petri-net analysis techniques for verifying XRL workflows. The tool uses XSLT (Extensible Style Language Transformations) to translate XRL specifications to a specific class of Petri nets to determine whether the workflow is correct.

In the area of Business Rule Processing (BRP), OMG issued a RFP for Business Semantics of Business Rules [OMG]. The proposal aims to enable business people to define rules in their business languages, describe artifacts for their business domains, and make rule definitions clear, unambiguous and portable to other representations. Domain knowledge is usually encoded in rules and an inference engine makes decisions based on these rules [DHW94]. OPS5 [BFKM85] is one of the earliest expert system rule languages developed at Carnegie-Mellon University. It uses a forward chaining inference engine and programs search working memory to find matching rules to execute. Other well-known expert systems include Java Expert System Shell (JESS) [JESS] from Sandia National Labs, and Agent Building and Learning Environment (ABLE) [BSPMD02] from IBM. Our work is based on ABLE as our rule processing engine.

3. IDG SYSTEM ARCHITECTURE

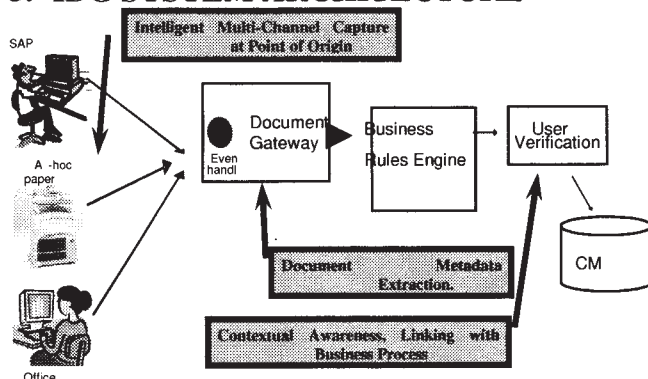


Figure 1. System Architecture

In Figure 1 we have outlined the basic architecture of our system. The intelligent document gateway (IDG) receives messages from the Enterprise Service Bus (ESB) via Java Messaging Services (JMS) and extracts business information from the messages. These information chunks are then either used to either trigger business rules, or are themselves information chunks that get moved to certain locations in the content management system. The user has ability to verify the business logics that are triggered, and the system also allows the user to author new business rules or to modify existing ones. During the execution of the rules, the actual input parameters come from each purchase order, which is an XML document. Rule actions are embedded in XML, compressed, and sent to an appropriate outbound ESB; they can vary broadly depending on the customer and location of a purchase order.

3.1 Information Routing Architecture

As shown in Figure 2, the architecture provides the Intelligent Document Gateway with the flexibility to work with any document routing logic and any inbound document type (e.g. ASCII, XML, or form images). Both the business logic and the structure of inbound documents are externalized by the gateway in standard formats. The business logic is externalized as a comma-separated value (CSV) file. This allows business users to easily author and edit it with a familiar spreadsheet tool.

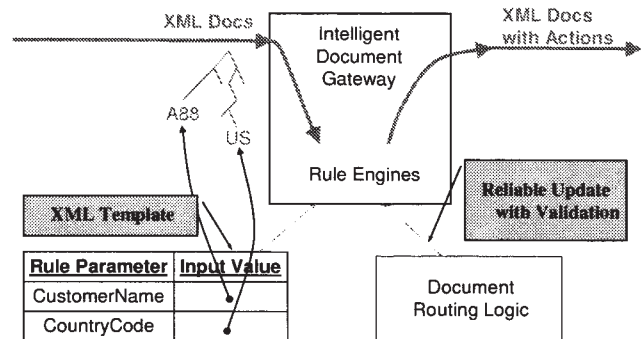


Figure 2. Document Routing Architecture

Each rule can have a number of input parameters, whose values are to be determined at runtime. The link between each parameter name and the location of its runtime value in each inbound document is externalized as a XML template file per document type. This allows the gateway to work with any inbound document type which in turn can have any internal document structure.

3.2 Document Routing Authoring

Business users can author and edit the routing rules with a familiar spreadsheet tool, as shown in Figure 3. In particular, business users do not need to write any computer programs, macros, or scripts. The knowledge base format contains just the essential elements of the business domain. The rule format supports logical operators (AND, OR, NOT), pattern matching, and set operations (membership and complement). For instance, a business user can specify that a country code should not be in the set {UK, SE}.

RuleId	CustomerId	Logic	CustomerName	CountryCode	Action	CSRs	Bcc
1	3MDenmat	AND1000142282				Email: ruh@uk.ibm.com	custsup@ibm
2	ABBUSCA	AND100001	"ABB"			Email: smoeller@u.b2beron@us.ibm.com	custsup@ibm
3	ABBUSCA	AND100001	"ABB"	UK SE		Email: ABBORDER@fr.ibm.com	custsup@ibm
4	ADPCSR	AND100004405				Email: donissmit@u.b2beron@us.ibm.com	custsup@ibm
5	AetnaUSC	AND100001	"Aetna"	US		Email: dickpunk@us.ibm.com	comlebuff@custsup@ibm
6	AlcoaCSR	AND100001497				Email: alcoa@us.ibm.com	b2beron@us.ibm.com
7	AMEXC	SRAND10000C	"American Express"			Email: dkuhman@kgluvc@us.ibm.com	custsup@ibm

Figure 3. Document Routing Logic Authoring

3.3 Reliable Update with Validation

Any routing rule can be updated while the Intelligent Document Gateway is in service. The gateway receives updates in a staging area, where it checks for syntactic and semantic errors. If the new logic has a problem, the gateway tries to identify the rules that might have caused the problem. It then notifies the business users of the problem by email, along with any metadata from the inbound document to help them understand the problem. The updates are automatically delivered to the Intelligent Document Gateway from a secure repository via an encrypted channel with

public-key authentication to protect customers' privacy without sending the password over the network.

Behind the scenes, the knowledge base is processed by the scripting engine of ABLE. This allows our architecture to also externalize the powerful rule-set interface of ABLE for advanced document engineers. Each rule set can contain any number of rule blocks where each rule contains a number of input parameters. Each parameter is bound to a value from an inbound XML document at runtime based on an XML template which defines the linkage between each parameter and the location path to an attribute or an element in an inbound XML document.

3.4 XML Invoice Image Template Authoring

We provide an Eclipse plug-in called the Document Gateway Architect, for designers to author and edit XML templates for extracting fields from images of invoice documents as shown in Figure 4. The template is used to extract fields, which are processed by ABBYY OCR to extract symbolic text.

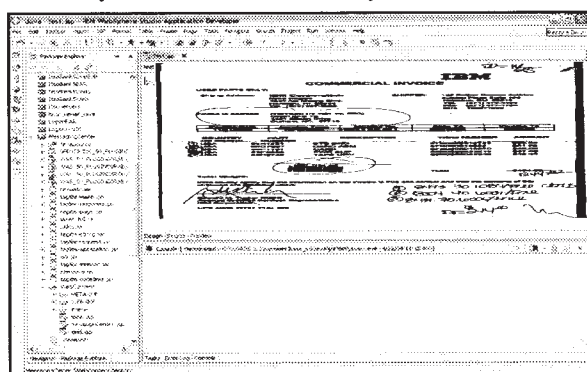


Figure 4. XML Invoice Image Template Authoring and OCR

Business processes are organized in Eclipse projects along with the business rules in them. To edit business rules, a user simply double-clicks on a rule file to launch Excel within Eclipse. The Document Gateway Architect validates business processes with XML Schema definitions. The gateway uses the standard XSLT engine to process the XML template to provide business rules with the runtime values of input parameters.

Authoring or editing an XML template does not require any knowledge of XML or XPath. With the Document Gateway Architect, a designer imports an inbound document into a business process. The tool then displays a familiar expandable tree view for the designer to browse the XML document, as illustrated in the top half of Figure . When the designer selects the attribute or element that contains the input value of a business rule, the tool automatically generates the correct XPath expression.

4. IMAGING CHALLENGES

Currently we are working on many challenges that arise during information extraction from scanned document images. Some of these include:

- Information extraction from geometric forms where the fields of the form are always at the same physical location in the original form template but are skewed during the scanning process [KH99, KK02].
- Information extraction from boiler plate forms that have running prose, but certain entities in the document

are replaced with new strings (e.g. names and project duration in contractual documents) [KM03].

- Extraction of handwritten entries in geometric forms.
- Extraction of information of faxed geometric and boiler-plate forms that have very low quality due to multiple re-faxings.
- Multilingual extraction for international transactions (e.g. immigration, import and export, customs, etc.) that require multilingual text recognition.
- Benchmarking dataset for OCR-based information extraction.

5. CONCLUSION

We described the architecture of Intelligent Document Gateway (IDG), which is an information extraction, management and routing front-end for document management systems. We also described some of the challenges we are currently facing in extracting information from various document image sources. This system is currently deployed in various IBM divisions and we are currently working on extending it to various applications in customs, citizenship and immigration, border protection, etc.

6. REFERENCES

1. [AVK01] W. M. P. van der Aalst, H. M. W. Verbeek, and A. Kumar. *XRL/Woflan: Verification and Extensibility of an XML/Petri-net based language for inter-organizational workflows*. In Proceedings of the 6th INFORMS Conference on Information Systems and Technology, 2001.
2. [BSPMD02] J. P. Bigus, D. A. Schlosnagle, J. R. Pilgrim, W. N. Mills, and Y. Diao, "ABLE: A Toolkit for Building Multiagent Autonomic Systems," *IBM Systems Journal*, Vol. 41.No.3, pp. 35071, 2002.
3. [BFKM85] L. Brownston, R. Farrell, E. Kant, and N. Martin, *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*. Reading, Massachusetts: Addison-Wesley, 1985.
4. [DHW94] U. Dayal, E. N. Hanson, and J. Widom, "Active Database Systems," in *Modern Database Systems: The Object Model, Interoperability, and Beyond*, W. Kim, Ed.: Addison-Wesley, September 1994, pp. 434-456.
5. [JESS] E. Friedman-Hill, *Jess in Action*: Manning Publications, July 2003.
6. [KS04] V. Krishna and S. Srinivasan, Towards Smarter Documents. In Proceedings of CIKM 2004.
7. [OMG] "Business Semantics of Business Rules (BSBR) RFP," OMG Business Enterprise Integration Domain Task Force June 2003.
8. [L01] L. Likforman-Sulem, "Name block location in facsimile images using spatial/visual cues," *Proc. ICDAR*, pp. 680-684, Sept. 2001.
9. [KH99] T. Kanungo and R. M. Haralick, "A closed-loop methodology for generating character groundtruth for scanned images," *IEEE PAMI*, vol. 21, pp. 179-183, 1999.
10. [KK02] D. Kim and T. Kanungo, "Attributed point matching for automatic groundtruth registration," *IJDAR*, vol. 5, pp.47-66, 2002.
11. [KM03] T. Kanungo and S. Mao, "Stochastic language models for style-directed physical layout analysis of documents," *IEEE PAMI*, vol. 12, pp. 583-596, 2003.

Fast skew and *slant* correction for Arabic written word or line

Essam M. Zaki
Sakhr Software USA, Inc.

Mohamed El-Adawi
Helwan University, Faculty of Engineering

Abstract

Word or line processing is important stage in OCR (Optical Character Recognition.) or ICR (Intelligent Character Recognition.) systems. Here, we propose two algorithms appropriate for this stage. The first one corrects the skewing of word (line). The second removes the slant from printed or handwritten word (line). The skewing detection algorithm collects the first foreground pixel in every image column then use linear regression to find the best fitting line, the slope of the line is the skew angle. The slant detection algorithm uses the connected component analysis, the slant angle is calculated by finding the edged image and removing horizontal lines then finding the slant histogram for connected components, the maximum peak in slant histogram will be the slant angle. The algorithms have been tested for printed and handwritten words (lines) of Arabic and English languages. The algorithms provide high accuracy and fast results. The overall accuracy of OCR or ICR systems have been improved after using skew and slant correction.

Evaluation Workshop Abstracts

Performance Evaluation of Multilingual Document Exploitation Systems

Steven G. Schlosser
NovoDynamics, Inc
123 N. Ashley St. STE 210
Ann Arbor, MI 48104
steve@novodynamics.com

Abstract

This presentation identifies and discusses key forms of performance evaluation called for by multilingual document exploitation systems. The architecture and capabilities of ArborScript™ ES are used to highlight typical information requirements of exploitation system test data. RML, an XML-based recognition markup language used by ArborScript ES, is offered as an example of how comprehensive groundtruth can be structured. In addition, common practical issues in formulating groundtruth are described.

Issues with Automatic OCR Evaluation and Metrics

Kristian J. Concepcion

7525 Colshire Drive – M/S H305

McLean, VA 22102

Abstract

In this presentation, we discuss the issues that arose while semi-automatically creating a degraded Arabic corpus for OCR testing, and the subsequent weakness of current OCR metrics to report on the complexity of the documents to be processed. Using a hybrid of manual and automatic techniques, we were able to create clean and degraded versions of our test corpus, with many controlled variables (dpi, font, point size, font face). Character accuracy and word accuracy were then calculated against the ground truth. However, while character accuracy and word accuracy are useful measures of the performance of an OCR engine, they are neither descriptive of how the OCR engine handles document features such as images, watermarks, or logos, nor do they differentiate between the cause of extraneous characters that appear in the output. While these shortcomings are partly due to a lack of ground truth standards for how to ground truth a complex document, they also occur because these accuracies cannot encapsulate the numerous issues that arise from processing real-world documents.

Evaluation Issues in ImageRefiner

Kristen Summers Eugene Borovikov

CACI

{ksummers,eborovikov}@caci.com

Abstract

ImageRefiner is an application that uses machine learning to select enhancement methods to apply to document images, for the purpose of improving OCR accuracy. In addition to requiring OCR evaluation for both the training phase and testing the system, it raises additional evaluation issues. These include: appropriate evaluation of selected enhancement methods for particular document images; evaluation of segmentation of images into regions for separate enhancement; and evaluation of the generality of the system, including both the effects of a training set that imperfectly represents the application data, as is often unavoidable in operational environments, and the effects of heterogeneity within that training set.

The Sporadic Nature of OCR Evaluations

Tapas Kanungo

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
kanungo@us.ibm.com

ABSTRACT

The Optical Character Recognition (OCR) community has had a long history: the OCR problem is one of the oldest pattern recognition problems that researchers have worked on, and commercial OCR products have been around for decades. However, a long history does not imply maturity. The maturity of a field is typically reflected in, amongst other things, the existence of a regular objective evaluation of the state of the art. For example, i) the processor design community has been running the SPEC benchmarks for decades; ii) the database community has been evaluating database performance using various TPC benchmarks; iii) the information retrieval community has been running the TREC evaluations for a very long time, and iv) the speech community has its regular evaluations. However, in the case of OCR, we can not claim that we have had continued evaluations over decades. The UNLV evaluations of OCR systems were held for five years. Arabic and multilingual system evaluations were held at University of Maryland for two years. There was an OCR track at TREC one year, and more recently, the VACE program had video OCR evaluations. On the positive side, the community has produced various resources over the years: i) the UNLV evaluation datasets and tools, ii) the UW dataset and tools, iii) the UMD groundtruthing tool PSET, the logo dataset, and the Viper tool. At this evaluation workshop, we should i) identify how we can establish a long running evaluation program, ii) agree on what kinds of OCR problems and evaluations are necessary for the government, academia and industry, and iii) identify what organization is best positioned to run the evaluations on a yearly basis.

Ground Truth Representation used in Testing and Optimization of the Optical Word Recognition System

**Mike Ladwig
Northrop Grumman Corporation**

Abstract

This paper describes the ground truth representation and associated software tools used in testing and optimizing the Optical Word Recognition system. Optical Word Recognition (OWR) technology addresses key elements in the process of analyzing scanned document page images using an optical correlation approach. OWR applications have focused on document triage - the rapid identification of high value documents based on content such as signatures, graphical logos, and keywords. In these applications, OWR is used to identify high-value items rather than perform general content conversion. Because of this difference, OWR testing and optimization uses a ground truth representation oriented towards the representation of metadata about the page as whole and interesting objects found on it. This representation is used throughout the suite of software used to test and optimize OWR.

Metrics for Word Spotting

Sargur Srihari

CEDAR, University of Buffalo

ABSTRACT

Central to searching a document repository is a method for finding a query word in each of the documents. In the case of scanned handwritten documents there is uncertainty in being able to find words since their shapes can vary greatly. Thus there is a level of uncertainty in word spotting-- unlike in the case of electronic text where words can be found with high accuracy by using string matching algorithms. In this paper we propose measures for the evaluation of a word spotting system for handwritten documents. It consists of measures for line segmentation, word segmentation and word matching. The use of these measures in evaluating the performance of the CEDARABIC system in spotting Arabic words is illustrated.

Evaluating with Informational Confidence

Stefan Jaeger
David Doermann

Institute for Advanced Computer Studies
University of Maryland,
College Park, MD 20742, USA
jaeger@umiacs.umd.edu

Abstract

A fair comparison is the key to a meaningful evaluation of different approaches to the same problem. In order to ensure a fair evaluation, however, it is not sufficient to concentrate solely on the recognition rates alone. We need to take into account the confidence values, which indicate how confident a classifier is in its output results. Confidence values usually play an important role in post-processing and are essential to finding rejection thresholds or ranking documents according to a given objective function. For instance, a classifier that assigns the correct label to a test pattern with high confidence should be identified as being superior to another classifier giving the same correct label but with lower confidence.

We are promoting an information-theoretical technique to meet these requirements. In particular, we assume that each confidence value conveys information that is directly related to the performance of its classifier in the application domain. In an evaluation step, we can normalize each confidence value so that it equals its informational content. These newly computed confidence values, called *informational* confidence values, will then replace the old, raw confidence values. The normalization technique makes comparison of confidence values straightforward and even allows combination of multiple approaches into a single integrated system. Informational confidence values have already shown their effectiveness in several pattern recognition problems, such as character recognition or script identification.

Author Index

-A-

Ablavsky, Vitaly.....49
Agam, G. 23
Al-Onaizan, Yaser 103
Argamon, S......23

-B-

Babu, Pavithra 123
Bajcsy, Peter 211
Bhole, Chetan.....123
Boyette, Neil.....231
Borovikov, Eugene 79, 243
Borsack, Julie.....39

-C-

Chang, Richard.....187
Chen, K. 31
Cheng, Isaac 231
Clutter, David 211
Concepcion, Kristian J......241

-D-

Doermann, David.....31, 251

-E, F-

El-Adawi, Mohamed 235
Farooq, Faisal 89, 103
Femiani, John C......65
Fisher, Francis.....187
Frieder, O. 23

-G-

Gantz, Donald T......173, 197
Govindaraju, Venu.....89
Grossman, D. 23

-H-

Haritaoglu, Esin Darici 55
Haritaoglu, Ismail.....55
Henderson, Thomas C......157
Holden, Steve. 49

-I, J-

Jaeger, Stefan.....31, 251

-K-

Kanungo, Tapas 231, 245
Kreulen, Jeffrey.....231
Krishna, Vikas.....231

-L-

Ladwig, Mike.....247
Lewis, D. 23
Lin, Yun.....11
Lopresti, Daniel 111

-M-

MacRostie, Ehry.....15
Marcus, Kevin.....187
Miller, John J......197
Monnier, Camille 49

-N-

Nagy, George 111
Nartker, Tom.....39
Natarajan, Premkumar 15

-O, P-

Phielipp, Mariano 65
Price, Robert.....97

-Q, R-

Razdan, Anshuman.....65

-S-

Sarkar, Prateek 167
Saund, Eric.....167

<i>Schlosser, Steven G.</i>	139, 229, 239
<i>Seales, W. Brent</i>	11
<i>Shalev, Michael</i>	71
<i>Snorrason, Magnus</i>	49
<i>Spitz, A. Lawrence</i>	133
<i>Sridharan, Karthik</i>	89
<i>Srihari, Sargur</i>	123, 249
<i>Srinivasan, Harish</i>	123
<i>Srinivasan, Savitha</i>	231
<i>Summers, Kristen</i>	243

-T-

<i>Taghva, Kazem</i>	39
<i>Turner, Joi</i>	187
<i>Turner, Mark</i>	79

-U, V-

<i>Valenzuela, Imelda</i>	225
<i>Vincent, Luc</i>	9
<i>Vogt, Robert C.</i>	139

-W-

<i>Walch, Mark A.</i>	173, 197
-----------------------------	----------

-X, Y, Z-

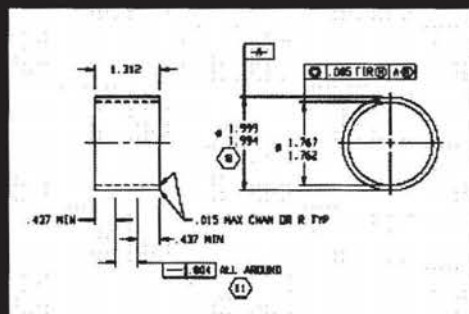
<i>Xu, Chimiao</i>	157
<i>Yaghi, Jim</i>	133
<i>Zaki, Essam M.</i>	235
<i>Zavorin, Ilya</i>	79
<i>Zhu, Guangyu</i>	31
<i>Zukas, Anthony</i>	97

String candidate	Linguistic score	Gloss	Details
Alshen almshryfn (الشبن المشرفين)	12	The young experts	valid words, correct phrase structure - well formed agreement
Alshen almshryfn (الشبن المشرفين)	23	ungrammatical (*experts the window)	valid words, incorrect grammatical context
Alshen almshryfn (الشبن المشرفين)	-	word discarded	invalid word

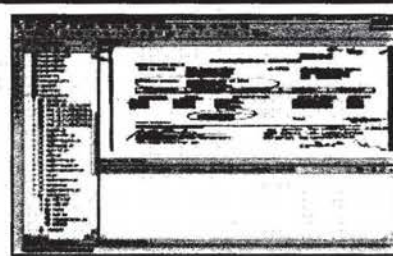
Processing of Arabic Documents



Document Analysis Applications



Cross Domain Applications



Demonstrations



SDIUT'05

2005 Symposium on Document
Image Understanding Technology

Marriott
Inn and Conference Center
College Park, Maryland
November 2-4, 2005

