

Pictographic Matching: A Graph-based Approach Towards a Language Independent Document Exploitation Platform

Mark A. Walch
The Gannon Technologies Group
1000 North Payne Street
Alexandria, Virginia 22314
011-703-373-1962
mwalch@gannontech.com

Donald T. Gantz, Ph.D.
George Mason University
4400 University Drive
Fairfax, Virginia 22030
011-703-993-1511
dgantz@gmu.edu

ABSTRACT

In this paper, we introduce the concept of Pictographic Matching as a tool for document exploitation across multiple languages. The primary technology supporting Pictographic Matching uses graph-based pattern matching to detect the *signature* of words contained in images of documents. This signature takes the form of graphs created by the line forms used to construct written words. These graphs can be matched to known graphs representing alphabetic characters, groups of characters and character components. And, since graphs are intrinsic to the structure of written language, graph-based searching promises to support a common platform capable of handling a multiplicity of languages. The conceptual roots for Pictographic Matching are found in Graph Theory. Pictographic Matching is implemented through a multi-step workflow drawing upon a variety of analytical techniques. The objective of this paper is to provide a high-level overview of Pictographic Matching as it exists in its current implementation.

Categories and Subject Descriptors

D.1.1 [Applicative Programming]:

E.2 [Data Storage Representations]:

General Terms

Algorithms, Documentation, Languages

Keywords

Searching, Triage, Optical Character Recognition, OCR, Optical Word Recognition, OWR

1. INTRODUCTION

Graph Theory is a branch of Mathematics that focuses on representing relationships as line diagrams containing nodal points and the linkages among the points. In graph terminology, the nodes are referenced as *vertices* and the links as *edges*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

HDP'04, November 12, 2004, Washington, DC, USA.
Copyright 2004 ACM 1-58113-976-4/04/0011...\$5.00.

Graphs are a handy and effective way to represent written language since they 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 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 and geometric information sufficient to replicate complete written forms. Graphs are the foundation of written language. A search methodology that focuses on graphs offers the potential to permit searching that encompasses several languages.

Figure 1 illustrates the simple conversion of writing into a graph. In this illustration, the pen strokes from the original word become the *edges* within the graph and the points at which the strokes cross become the *vertices*.

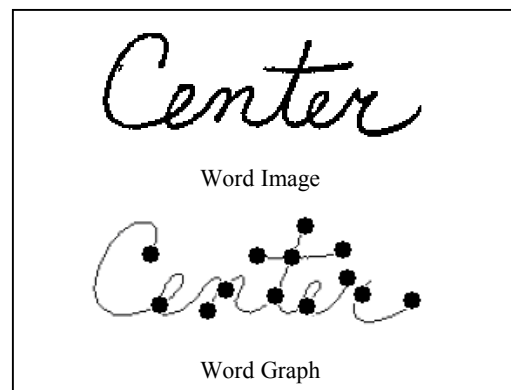


Figure 1: Comparison of written and graph forms of word “Center”

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.

Graph topology can be encapsulated in a single numeric code. 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 quantifications for characters and words contained in the images of documents.

Pictographic Matching is a multi-stage process that harnesses the data contained within graphs as a means of searching documents. This process presumes that documents are scanned into electronic images in a conventional format. Transforming the content of these images into searchable graphs encompasses a step-by-step procedure that (1) detects written objects within the images and converts these objects into a series of graphs, (2) matches these unknown graphs against templates based on known graph topologies, (3) decodes the unknown graphs into known forms by comparing their feature vectors against feature vectors of known graphs representing characters and words, (4) stores the decoded data in a format highly supportive of searching and (5) searches the stored data.

Specific methodologies involved include techniques relying on Isomorphic Graph Matching, Discriminant Analysis and Dynamic Programming. Isomorphic Graph Matching identifies topologies of known graphs such as alphabetic characters embedded in graphs of unknown words. Once graphs have been identified and isolated, a technique employing Discriminant Analysis classifies the graphs as the appropriate alphabetic characters, parts of characters or groups of characters. At the time a search is performed a technique that incorporates Dynamic Programming methods matches the collection of classified graphs taken from an unknown, imaged word against a known search term of interest.

Steps 1, 2 and 3 identified above represent the true invention that distinguishes Pictographic Matching from Optical Character Recognition (“OCR”) and other methods supporting document searching. Steps 4 and 5 represent the current implementation of Pictographic Matching that distinguishes its application from OCR-based approaches. Regarding Steps 4 and 5, Pictographic Matching could be implemented in a fashion similar to that found in most OCR systems. That is, it could be used to generate text from images. However, the nature of results from the Pictographic Matching process, particularly the finding of reasonable alternative character choices, supports an output format with more dimensionality than would be possible through a pure text output. Pictographic Matching produces results in the form of a matrix where alternative character candidates can be considered at the time searching is performed.

On the ensuing pages, this paper presents a descriptive overview of each of these five steps. By necessity, each step is discussed as a high level overview without delving into the considerable mathematical and algorithmic detail that makes each step possible. More complete technical detail can be found by reviewing the pending U.S. Patent entitled “System and Methods for Source

Language Pattern Matching” (Walch) which discusses the concepts herein presented in considerably more detail.

Pictographic Matching can be used to detect individual key words and combinations of words in documents, for concept detection and document triage. Pictographic Matching can also be used for mining concepts from documents by *driving* the search effort through specially designed lexicons. Since Pictographic Matching addresses the underlying geometry and structure of written language, it promises not to be bound by language barriers. It is capable of handling both printed and handwritten material in segmented (printed English), cursive (English and Arabic) and pictoform-based languages (Korean, Chinese, etc.).

Since it transcends usual language constraints, Pictographic Matching technology offers the potential for creating a Language Independent Document Exploitation Capability (“LIDEC”) in support of the Intelligence Community. The LIDEC will enable U.S. Intelligence Agencies to eliminate the current backlog in extracting critical information from massive volumes of collected and captured documents by bringing documents online in a searchable form earlier than currently possible. Also, because it deals directly with the shape and structure of writing, Pictographic Matching offers the ability to extract biometric measures from handwriting bringing a completely new dimension to document review.

2. CONCEPTUAL FRAMEWORK

Pictographic Matching shares common roots with Optical Character Recognition, but it is a fundamentally different approach. Pictographic Matching looks for the *Pictographic Signature* of words in their native form. In this context Pictographic Matching is somewhat similar to Optical Word Recognition (“OWR”), but differs significantly from OWR in its ability to evaluate the actual characters that comprise the unknown words.

Pictographic 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.

Another aspect of Pictographic Matching that is distinctive from OCR is its output data format. OCR is typically used to convert images of documents into text. Pictographic Matching 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 and 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 Matching takes the form of a *Results Matrix* rather than 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 Matching technology 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 information available at exactly the time it is needed.

Pictographic Matching empowers document searching by making much relevant information available while the actual search is being performed. 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.

3. IMPLEMENTATION

Pictographic Matching starts with images of documents. The images are parsed into individual objects representing words, characters and other items of interest. 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 character, a group of characters or a word. Because of the abundance of measures available, classification can be sharpened by focusing on the specific set of measures that separates each particular character 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.

In summary Pictographic Matching is accomplished through five principal stages:

Stage 1: Preliminary Graph Segmentation

Stage 2: Isomorphic Graph Matching

Stage 3: Graph Alignment and Classification

Stage 4: Results Matrix Construction

Stage 5: Word Searching

Prior to these steps it is first necessary to scan documents and to load them into the Pictographic Matching system. Once scanning is accomplished, a variety of techniques can be performed to improve image quality. This reference to scanning and image enhancement is presented for informational purposes only to establish the context for the implementation of Pictographic Matching.

Following is a step-by-step descriptive account of the major stages comprising Pictographic Matching.

3.1 Pictographic Matching Stage 1: Preliminary Graph Segmentation

Preliminary graph segmentation is accomplished by use of a *Flexible Window* which is an algorithm that extracts graphs from an image. The Flexible Window has two functions. First, it locates all objects in the image that are likely words. Second, it crosses each of these words in an *inchworm-like* fashion extracting graphs as it proceeds. Techniques for finding word objects within documents are well established in OCR systems and are not discussed in detail within this paper. However, the technique for extracting graphs is a key component of Pictographic Matching and is illustrated in Figure 2. This figure shows how character combinations are isolated in a cursive or connected word through use of the Flexible Window.

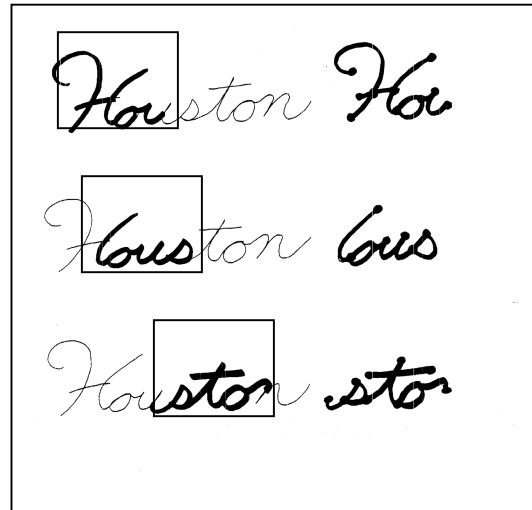


Figure 2 Examples of character strings isolated using the concept of the Flexible Window

The Flexible Window need not provide perfect character segmentation. It only must divide connected strings of characters into smaller segments likely to contain one or more characters. Separated characters, such as those found in hand print, are already segmented, but the Flexible Window is still useful for grouping parts of characters that may inadvertently be segmented by unintentional breaks in the characters. As graphs are isolated by the Flexible Window, they can be further analyzed and decoded using Isomorphic Graph Matching (Stage 2) and Graph Alignment and Classification (Stage 3).

3.2 Pictographic Matching Stage 2: Isomorphic Graph Matching

The computational engine for Pictographic Matching 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 3 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 differences in angles and distances make them look different.

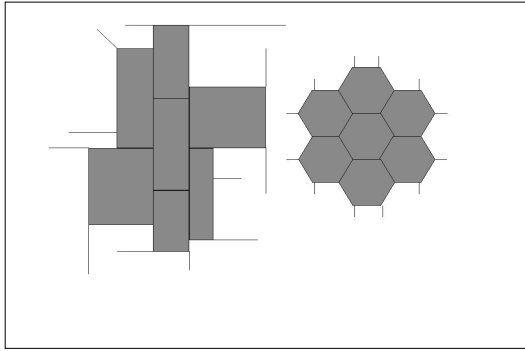


Figure 3 Illustration of two Isomorphic Graphs with different features

Graph isomorphism is a critical concept underlying Pictographic Matching. In fact, Pictographic Matching 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 by the Flexible Window algorithm. The foundation for this database is graph isomorphism. As previously discussed, Figure 2 shows sample letter combinations embedded as graphs in a larger word graph that could be extracted by the Flexible Window during Preliminary Graph Segmentation. Figure 4 indicates the relationship between these extracted graphs and some known reference graphs for the selected characters “H”, “o”, “u”, “s” and “t”. In Figure 4, the left column shows extracted graphs and the right column shows the reference graphs for the selected characters. The middle column indicates how the reference graphs are embedded in the extracted graphs. The embedded relationship shown in the middle column of Figure 4 is the relationship that can be captured in a database that maps a set of known graphs against a series of unknown graphs. Such unknown graphs would be extracted from imaged documents.

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 only necessary to consider planar graphs. Planar graphs are graphs that can only exist in a plane, such as lines on a page of paper.

Within the collection of planar graphs, the database can be built containing all the non-isomorphic forms of such graphs. The order of a graph indicates the number of vertices in the graph. The following table shows the number of non-isomorphic planar graphs up to order eleven.

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 Order 11 can be pre-calculated and all instances of known character graphs can be mapped to these pre-calculated models. This pre-calculating process creates the database of graphs.

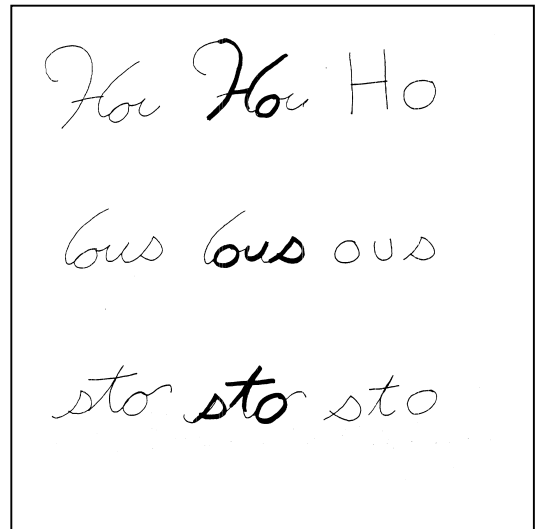


Figure 4 Examples of characters embedded in extracted graphs

Table 1: Cumulative Listing of Embedded Graphs

Number of Vertices	Cumulative Number of unique embedded planar graphs
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

Order 11 appears to be a reasonable model both for English and Arabic languages. An evaluation of character graphs contained in a publicly disseminated reference data collection provided by the National Institute of Science and Technology (“NIST”) shows that for written English over 99.8 percent of characters as actually written create graphs of Order 11 or less. Similarly, analysis by the authors of databases of cursive English (100 writers) and Arabic (100 writers) found the equivalent percent of all written characters to produce graphs less than or equal to Order 11.

The fact that there are 17,449,299 theoretically possible graph forms that will contain over 99.8 percent of all known graph forms provides the practical foundation for building the mapping database. Creating this database presents another challenge. However, it 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 identify each unique graph topology and this key can be used for linkage 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. This database of mappings actually accomplishes two major steps required for character and word recognition: Segmentation and Classification. *Segmentation* involves dividing a monolithic object, such as a cursive word, into its constituent forms. *Classification* occurs as a label is assigned to each form that is isolated through the Segmentation process.

Segmentation and Classification are independent, sequential stages in a typical OCR process. Pictographic Matching *bundles* both Segmentation and Classification activities into a single action that is *table driven*. That is, the calculations that support these activities are performed on a one-time basis and stored as tables in a database. This combined Segmentation/Classification process captures both the topology and geometry of an embedded graph and determines whether this embedded graph resembles the graph form of any known character. This *self-segmenting* capability significantly distinguishes Pictographic Matching from OCR techniques.

As graph topologies are matched, another benefit emerges in the form of graph alignment. That is, for the matched topologies, the corresponding graph components—edges and vertices—are aligned. Because the graphs are aligned, their feature vectors can also be aligned.

Although graph isomorphism is a very important ingredient for Pictographic Matching, it must be stressed that isomorphism alone is insufficient for recognizing 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. The process for evaluating these measurements is found in the next step of the Pictographic Matching process: Graph Alignment and Classification.

3.3 Pictographic Matching Stage 3: Graph Alignment and Classification

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 5 shows two features that are part of a character graph's feature vector.

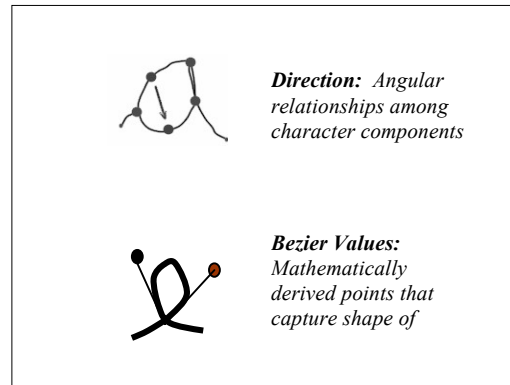


Figure 5 Examples of two types of features contained in graphs: Direction and Bezier Values

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 6 shows graph alignment between two versions of the letter "W" from two separate handwriting samples.

The isomorphic graph matching process ensures that the two versions of the character "W" 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.

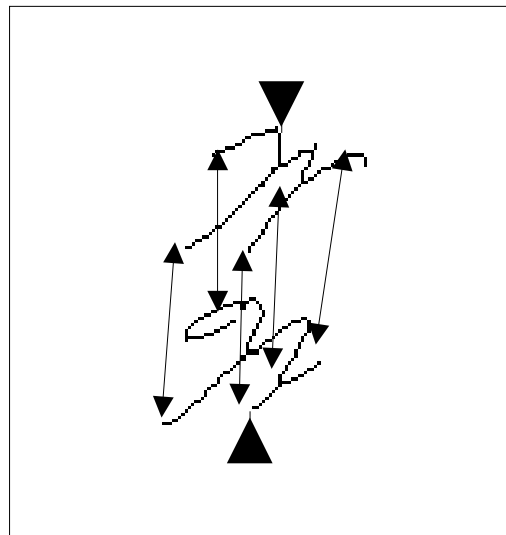


Figure 6 Isomorphic graph alignment from independent handwriting samples

Figure 7 shows two aligned graphs with selected vertices labeled: A, B, C, A', B' and C'. Because of this alignment, all pieces of the feature vector applicable to one graph are mirrored in the second graph. For instance, the curvature of the edge extending from vertex A to vertex C can be compared directly with the vertex extending from vertex A' to vertex C'. The angular direction from vertex A to vertex B can be compared directly to the direction from vertex A' to vertex B'. The same type of relationships hold for any other combination of distances, directions, or other attributes. In this way, the feature vectors can be perfectly aligned.

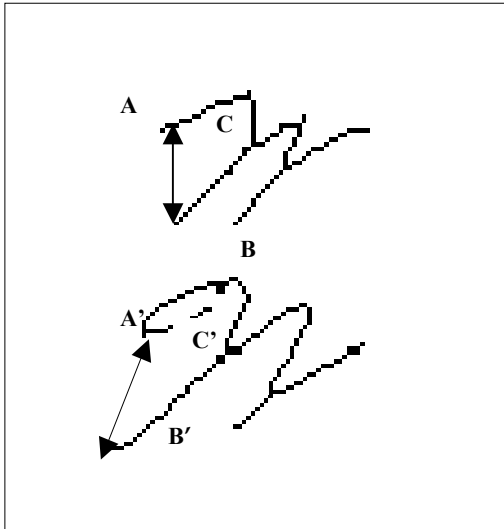


Figure 7 Isomorphic graph alignment of features

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 classified as letters, combinations of letters or words.

Classification takes place by decoding the aligned feature vectors. The feature vector from an unknown character graph is decoded by evaluating it against the feature vectors of known graphs that were modeled on identified characters.

Considerable data are available in the feature vectors of graphs. Even a relatively simple graph can generate a vector with over a hundred measurements. This wealth of data is well suited for a classification 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 for different characters with similar graph topologies to identify the specific set of measures that distinguishes one from the other.

To permit classification through Discriminant Analysis, a modeling process is performed. Modeling involves training on data for each unique graph isomorphism for each letter of the alphabet. During this training process, a specific set of variables is isolated for each form of each character. These variables are actually very specific measurements contained in the large set of measurements available for the graph. The variable set isolated through Discriminant Analysis can be considered as the *Alphabetic Kernel* for that character form. The Alphabetic Kernel is defined as the specific set of measures that will reliably distinguish that particular character from all other characters with the same topology. This set is unique to each form of each character. Figure 8 provides a graphical illustration of a full set of measures for one isomorphic representation for a lower case “a” contrasted with the set of measures constituting the Alphabetic Kernel. Through the modeling process, Alphabetic Kernels are generated for all forms of all characters. These kernels provide the basic data to decode unknown graphs into their appropriate characters.

As unknown graphs are extracted from a document they are first matched against reference database of graphs. That is, they are compared against the subset of reference graphs that shares exactly the same topological structure. Once this match is complete, Alphabetic Kernels, in the form of specific subsets of measurements, are extracted from the unknown graph and compared against the Alphabetic Kernels of the graph models. The known graph that best matches the unknown graph on the specific measurements constituting the Alphabetic Kernel is considered the best character match to the unknown graph. Those that are not the best fit, but are close fits, are considered as alternative possibilities.

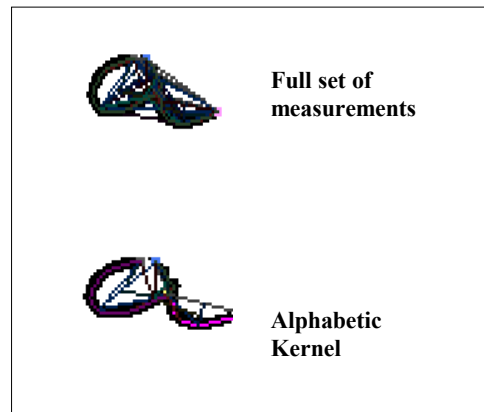


Figure 8 Sample Alphabetic Kernel for letter one isomorphism of letter “a”

Classification based on Alphabetic Kernels focuses on those places where individual characters are most different. Alphabetic Kernels permit the distinction between very similar character forms. One sample of such a case is shown in Figure 9 depicting the “evolution” of form from a lower case “a” to a lower case “u”.

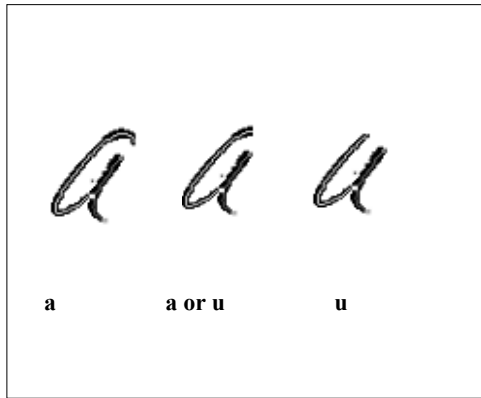


Figure 9 Sample indeterminate forms for letters “a” and “u”

Very slight and subtle differences often distinguish individual characters. Outside the presence of an actual word to establish context, these character forms are and should be considered indeterminate. Another cause for indeterminacy is rooted in embedded forms. For instance, a single stroke character such as “I” occurs in almost every complex character. The upper case “E” often has “F”, “L” and “I” embedded in it. Some of these embeddings can be addressed through feedback. That is, if a complex character is identified, it is not necessary to evaluate every simple form which may be embedded inside it. However, sometimes such a decision cannot be made with reliability and a preferred method is to retain the alternative solutions for later evaluation.

Built into Pictographic Matching is the concept of *Planned Indeterminacy*. Simply stated, Planned Indeterminacy permits the consideration of multiple competing candidates as results. Multiple recognition results can be stored and evaluated at a later time such as when a particular search term is specified. These alternative outcomes distinguish Pictographic Matching from traditional OCR techniques that force a single outcome from the recognition process.

The concept of maintaining results in a planned, but the indeterminate form, requires a data structure tailored to support this concept. This data structure can be found in the Results Matrix.

3.4 Pictographic Matching Stage 4: Results Matrix Construction

The Results Matrix is a tool for identifying characters embedded within a cursive word with their actual physical locations. The matrix contains the following information.

1. Character name as classified
2. Physical coordinates in image
3. Confidence Score from classification: how far the classifier found the features from the unknown character to be from known characters

Figure 10 illustrates the concept of the Results Matrix:

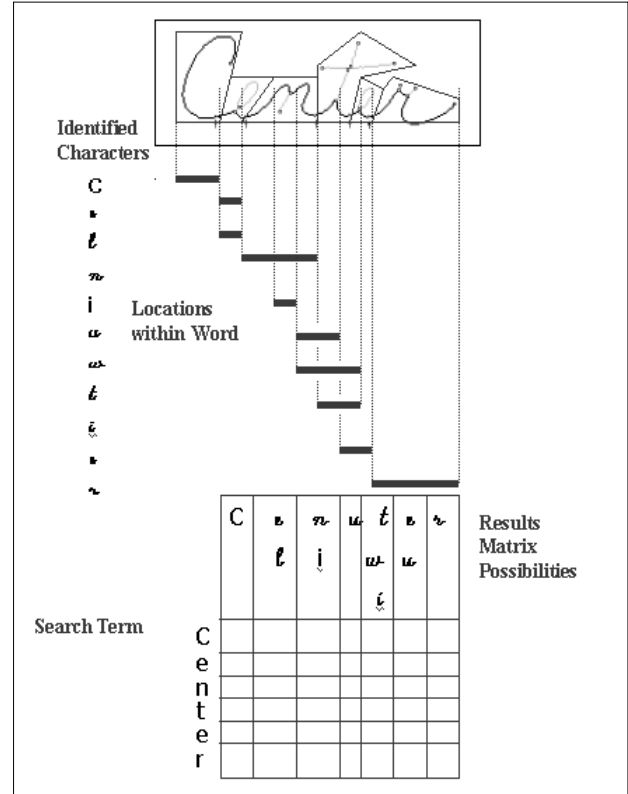


Figure 10 Example of individual character mapping into Results Matrix

The purpose of the Results Matrix is to maintain recognition data in a form of *Planned Indeterminacy*. That is, the recognition results, including likely character matches and their attendant confidences, are maintained as data without producing word results. As previously stated, this aspect of its functionality distinguishes Pictographic Matching technology from traditional OCR. Within the Results Matrix, different character results can compete for the same position and some character results can *span* across others. Pictographic Matching retains the data in a format that keeps considerable information available to support searching. At the time a search is performed, the contents of the Results Matrix are evaluated using dynamic programming techniques that attempt to match the search term with the characters in the Results Matrix.

3.5 Pictographic Matching Step 5: Word Searching

Within Pictographic Matching, word matching only occurs when a specific search term is known and the actual word matching takes place by mapping the contents of the Results Matrix against the prescribed search term. This mapping is performed using methods based on Dynamic Programming. Dynamic Programming is a well-documented method for comparing patterns in character strings. The Results Matrix can be considered to be one string and the search term can be considered to be the other string. One difference from traditional dynamic programming techniques is that the string represented by the Results matrix can take a number of alternative forms, reflecting the alternative character possibilities for each

position. This represents an extension over traditional Dynamic Programming approaches. A detailed discussion of Dynamic Programming techniques is beyond the scope of this paper.

4. PROCESS SUMMARY

The current implementation of the Pictographic Matching workflow operates through Graph-based pattern matching supported by techniques incorporating Discriminant Analysis (for classification) and Dynamic Programming (for string matching). Pictographic Matching is accomplished in five steps with Steps 1 through 4 building a data structure to support the search and Step 5 encompassing the actual search.

Critical aspects of Pictographic Matching include:

1. A reference set of valid character forms is collected and labeled. These characters are converted into graphs and maintained as a language specific reference.
2. A database is built from this reference collection. This database contains information regarding both the topology and geometry of the reference characters. The topology is encapsulated into a unique key that reflects the unique graph structure of each reference character. This key measures graph isomorphism. Geometry is contained in a feature vector for numerous measurements contained within each character graph. A selected subset of measurements is contained in the database. The combination of graph topology and the distilled subset of geometric measures is referenced as the Alphabetic Kernel for each individual character. The Alphabetic Kernel represents those particular measurements that will distinguish an individual character from all others.
2. When a new document collection is to be searched, it is first scanned and *loaded* into the Pictographic Matching system. Loading takes the form of converting the content of the documents into graphs, matching the document-based graphs with the reference graphs. This match takes place by matching unknown graphs with known graphs using the graphs' Alphabetic Kernels. As matches are made, the unknown graphs are classified and the results recorded in a matrix that maintains detailed information about the unknown graphs and the known reference characters that they match. This matrix contains the basic data to support searching.
3. Searching is accomplished by specifying a search term or group of terms. As terms are specified, their characters are matched against items in the matrix. As character group sequences from the matrix match character group sequences within the search term, they are flagged as possible matches.
4. Since Pictographic Matching allows alternative matches to a specified search term, the output pool includes multiple documents most likely to contain the search term. The distinction between truly responsive documents and false positives in the output pool can be established using the following criteria.
 - A. Coverage: The percentage of all documents containing the search term that are present in the output.

- B. Confidence: The fraction of output documents having the specified coverage.

5. The possible matches are filtered to eliminate false positives. This filtering is currently accomplished by human review incorporating a *Directed Workflow* process. Directed Workflow is a method for incorporating the skills of trained operators in the review process while highly leveraging their effectiveness. Operators are automatically presented with screen images of words and their corresponding results and are asked to select those items responsive to the specified search terms. Using this proven technique, operators can review 50 to 80 transactions per minute—or 3,000 to 5,000 transactions per hour.

It must be stressed that by focusing on the underlying structure and geometry of written language, Pictographic Matching offers the ability to cross traditional language barriers within a single functional platform.

5. CROSSING LANGUAGE BARRIERS

Pictographic Matching is currently implemented in English with an Arabic Prototype currently under development. In terms of appearance, written Arabic and English appear to have little in common. English is written left to right and Arabic right to left. They do not share common characters. Arabic is principally connected. English has both cursive and printed forms. Except for capitalization, written English characters tend to hold their form wherever they occur within a word. Arabic characters change form depending on their location and use. English uses virtually no ligatures—multiple characters combined into a single form. Written Arabic is heavily reliant on ligatures.

Given all these apparent differences, between written forms of English and Arabic, their underlying graph structures are remarkably similar. The following Table shows the usage of the “top ten” character graphs from a group of 100 English writers compared with a similar sample from a group of 100 Arabic writers.

Table 2: Graph Usage Comparison for English and Arabic

English Graph Rank	Arabic Graph Rank
1	3
2	1
3	4
4	8
5	7
6	12
7	9
8	5
9	2
10	15

This table shows that the ten most commonly used graphs used in handwritten English are within the fifteen most commonly used graphs in Arabic. As previously discussed, graph usage is a major component of the Alphabetic Kernel concept employed by Pictographic Matching.

Clearly, English and Arabic have numerous differences. However, they also have a critical common denominator in terms of their underlying graph structure. Pictographic Matching focuses on the

common characteristics of the two languages. It is the authors' belief that all written languages have measurable graph-based constructs that can be captured by Pictographic Matching. And, this commonality among written language offers far reaching opportunity for Pictographic Matching as a language independent document exploitation technology.