ANALYSIS AND COMPARISON OF POPULAR TECHNIQUES FOR OPTICAL CHARACTER RECOGNITION

Nicholas Redmond

nicholas.john.redmond@gmail.com

ABSTRACT

Optical character recognition has been studied in depth for decades, and numerous techniques have been developed to improve performance of recognition systems, but little work has been published to outline the strengths and weaknesses of various techniques implemented within a complete character recognition system. This experiment attempted to analyze a multitude of popular techniques in all major steps of the optical character recognition process and determine which techniques are well-suited for enterprise applications, as well as outlining techniques that increase application complexity while providing minimal increases in system performance. Neural networks were shown to be more effective as a classification algorithm than other popular methods for general-purpose applications, with an average recognition rate of 97.50%. In addition, the recognition rate of the system implemented for this experiment was shown to be drastically altered after changing the feature extraction techniques used by the system.

KEYWORDS

OCR analysis, feature extraction, segmentation, classification, post-processing, enterprise systems

1. INTRODUCTION

The field of optical character recognition (OCR) has been one of active research since the 1970s. Much work has been done to advance the accuracy and capabilities of OCR systems, and commercial OCR software can now recognize printed English characters on clean backgrounds with over 98% accuracy [1]. Many techniques have been employed over the past few decades to improve the accuracy and robustness of OCR systems, and dozens of feature extraction and classification methods have been developed to optimize recognition [2]. That being said, there has not been much active research done on the comparison of currently available OCR techniques in a single working system.

The majority of research conducted in OCR has targeted a single step of the OCR process, such as feature extraction. Many experts in the field, such as Perez et al. [3], make comparisons between different techniques for a single OCR step with the intent of outlining the

most optimal technique. By contrast, this paper aims to compare a variety of implementations of multiple steps of the OCR process in a single system. By analyzing the relationships of multiple steps in an OCR system, a comparison may be made to determine between different combinations of popular techniques in an effort to determine optimal system solutions.

While a comprehensive analysis of multiple OCR steps may provide information regarding optimal solutions, it might also provide insight as to certain OCR techniques that are not well fit for a given system. Many OCR methods, especially regarding feature extraction, are highly powerful and can reliably recognize a variety of character sets. However, more powerful methods are often more computationally expensive and can be unnecessary in a simple OCR system [3]. Systems working on printed English fonts have been able to achieve at least 70% percent accuracy for some time now, and applying novel techniques for recognition that increase computation time may not be an effective approach [1]. By performing an analysis of popular techniques in a working OCR system, this paper may attempt to outline techniques and methodologies that are sufficient for systems without a need for near-perfect recognition or complex character analysis.

Though analyses and comparisons of popular OCR methods have already been performed successfully, little research has been done to analyze the abilities of different methods in an enterprise OCR system. The majority of research performed on popular techniques, such as that conducted by Saha et al. [4], has been conducted in a controlled environment without much variance. While this provides a more accurate representation of the theoretical abilities of the analyzed techniques, it does not entirely represent the performance of those techniques in a working application of OCR. In this paper, an attempt is made to analyze and contrast current OCR practices and methods in an enterprise-type application, which may provide important information regarding the capabilities of different methods in an enterprise scenario.

2. SEGMENTATION METHODS

Perhaps the most challenging aspect of character-based OCR is that of segmentation. The aim of OCR segmentation is to divide the text contained in an image into individual characters for recognition. While segmentation is a vital step of character-based systems, extensive research has not been published to document an optimal solution. As such, a variety of segmentation methods exist and are employed in working applications.

2.1 – X-Y Cut

Printed text in images often follows a simple layout, where lines of text are separated horizontally and characters rarely overlap. The X-Y cut algorithm assumes this layout and segments images in a top-down procedure [5]. The image to be translated is first split into lines based on whitespace. Horizontal slices of the image are taken at every pixel down the length of the image, and slices with the least amount of black pixels are considered to be line breaks [5]. After segmenting the image into lines, each line is broken down into characters in a similar

fashion. Vertical slices are formed across each line, and areas with little or no black pixels are regarded as breaks between characters.

While the X-Y cut is sufficient for printed text without much noise, it encounters problems when attempting to dissect handwritten text or text contained in vintage publications [5]. Since the algorithm divides text based on whitespace, it cannot properly distinguish touching characters or images with substantial imperfections.

2.2 – Topological Segmentation

Often the images translated by an OCR system contain touching characters, which can be difficult to segment using whitespace-based algorithms. Topological segmentation overcomes this by segmenting characters based on the structural features of the text [6]. After segmentation into lines, each line of text is scanned horizontally and possible segmentation points are calculated based on known structural features, such as character endpoints [6]. More possible segmentation points can be calculated by different methods, such as determining the contour or skeletonized representation of a line of text and analyzing the result for possible starting and ending points of characters [7]. After all possible segmentation points are calculated, the list of possible points is filtered based on a variety of approaches, such as selecting points with the lowest pixel density, where pixel density is defined as the number of black pixels found in a vertical slice of a line.

3. FEATURE EXTRACTION METHODS

Feature extraction is arguably the most important step of OCR because it defines how characters will be represented in a system. OCR classification algorithms often require a fixed input size, which means a method must be implemented to translate all possible images into feature vectors of a single normalized length, where a feature vector is a list of values that represent a character image. Feature extraction aims to accomplish such a task while minimizing within-class pattern variability and maximizing between-class pattern variability [2].

Many feature extraction techniques exist with various advantages and disadvantages, but most techniques fall into one of two categories, which are word-based OCR and character-based OCR. The techniques discussed in this paper are centered on character-based OCR, which follows the process of segmenting an image into individual characters and feeding the resulting character images into various feature extraction algorithms.

3.1 – Straight Pixel Values

Perhaps the most straightforward approach to feature extraction, straight pixel extraction translates each pixel in a binary image to a single value in the feature vector for the image. Using straight pixel values is by far the most simple feature extraction algorithm in terms of mathematical complexity, but cannot provide any complex information about the image for use in classification.

3.2 – Chain Codes

Chain code features aim to extract information from images about the lengths and directions of character strokes. In this experiment a simple chain code implementation is used, where the image is skeletonized and then scanned from top-left to bottom-right until the first black pixel that has at least one neighbor is found. After finding the starting point of the character, the algorithm proceeds to follow the skeleton of the image by searching for black pixel neighbors. The neighbor above the current pixel is checked first, followed by all seven remaining neighbors in a clockwise fashion. When a black pixel neighbor is found, a numerical representation of the direction traversed is added to the chain code for the image. After traversing the entire length of the image, the chain code is normalized to a fixed length by omitting chain code directions that have a length of one and converting the chain code into normalized frequencies. This implementation is based on one discussed by Izakian et al. [8], and is described in detail throughout their report.

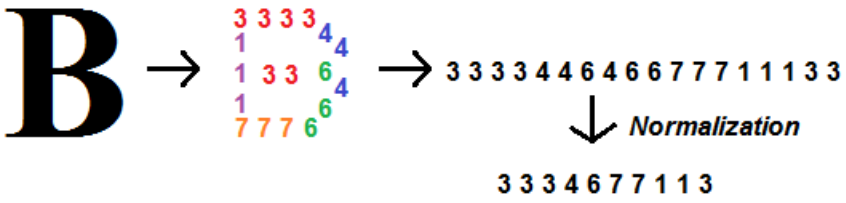

Figure 1. Calculation of the chain code for the letter "B". The final chain code is normalized to a length of 10.

3.3 – Feature Points

Often times the image of a character can be distorted or convoluted to the point that many basic feature extraction algorithms presented with the image will fail to extract useful features. Structural features, or feature points, can often be useful in overcoming distorted images by extracting topological features that represent the structure and geometry of the given character images [9]. However, feature points can also add to computation time because their implementation often requires character images that have been preprocessed in some way, such as by being skeletonized [9].

Two basic feature points are extracted in this experiment. The first feature point describes the locations of character endpoints, which are defined as the locations where the stroke of a character ends without arriving at a junction with one or more other strokes. The second feature point describes character T-junctions, which are the intersections of multiple strokes within a character. Many other feature point extraction techniques exist, such as ones that

extract character loops and locations of convexity or concavity, and possess various advantages and disadvantages depending on the character set being evaluated.

3.4 – Hu Moments

In image processing, a moment is a value representing the degree to which a figure tends to lean on a given line. In other words, a moment describes information regarding the shape of a particular image with respect to the center of mass for the image. In 1962, Hu introduced seven moments that can be used as features in OCR [2]. These moments are invariant to image translation and scale, and are semi-invariant to rotation, and thus can be applied to a variety of applications [2]. The general equation for moment calculation, which does not provide invariance to translation or scale, is defined as

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y)$$

where the sum is taken over all pixels in the image, and the values *x* and *y* correspond to x-y values of the image [2]. The values *i* and *j* represent the order and repetition of the moment respectively, which represent the degree of specificity with which to analyze the image. To achieve Hu's invariant moments from the above formula, a series of steps is taken as outlined by Trier et al. [2]. Invariant moments provide useful information about character images, but the act of making them invariant can cause substantial information loss, thus making the moments less powerful as features [2].

3.5 – Zernike Moments

Zernike moments provide a means of orthogonal feature extraction, which means changing the rotation of a character image will not change the magnitude of the Zernike moments calculated for that image [10]. In addition, Zernike moments contain more precise information about images than Hu's invariant moments, and thus can provide higher performance when dealing with more complex character sets, such as ones containing handwritten data. However, Zernike moments are neither scale nor translation invariant, so images must be normalized by scale to contain the same amount of black pixels, and all images must be centered on their centroid, which is defined as the center of mass of a binary image [10]. For the sake of brevity, the entire implementation of Zernike moments is not discussed in this paper, but Khotanzad et al. [10] have published an in-depth explanation of the process used to calculate Zernike moments.

3.6 – Zoning

Zoning is a simple, straightforward approach to feature extraction that is relatively invariant to image scale and skew [9]. Zoning is accomplished by first dividing the image in question into overlapping or non-overlapping zones, where each zone is a subsection of the original image. After dividing the image into zones, the pixel density for each zone is calculated

and added to the feature vector [9]. In this experiment, 16 non-overlapping zones are used, which are created by dividing the image into 4 rows and 4 columns. An example is illustrated in Figure 2.

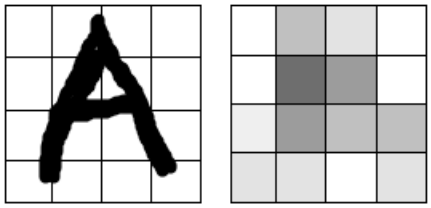

Figure 2. Pixel intensities for the letter "A". Image is divided into 16 zones.

3.7 – Height-to-Width Ratio

As the name suggests, this feature is a simple ratio of the height of an image compared to the width of the image. Before the ratio is calculated, whitespace is cropped away from the image in question until the character inside the image extends to the image boundary in all four cardinal directions.

3.8 – Skeleton Vectors

In theory, every character is made of strokes, which have a direction and magnitude, and each different character has a different set of strokes. Skeleton vectors attempt to utilize this information by extracting vectors from image data. To extract vectors, the image of a character is first converted to a skeleton, which is a very thin representation of the character. The Stentiford Algorithm was used in this experiment for image thinning because it is relatively simple to implement and provides good results for images that do not contain a large amount of curves, such as English characters.

After the image skeleton is obtained, it can be analyzed to determine the direction in which the strokes of the character are traveling, as well as the length of the character strokes [11]. The vector information for the strokes can then be normalized to a feature vector in a variety of ways. A popular method, which is discussed by Tawde et al. [11], involves dividing the image into zones and calculating the most prominent vectors in each zone.

3.9 – Image Symmetry

Character image symmetry is defined as the degree to which a character is symmetrical. Image symmetry can be calculated either vertically or horizontally, and is normalized to be a correlation value between 0 and 1, where values closer to 1 represent better symmetry [12].

Symmetry-based feature values are based on the structure of characters and will not inherently be resistant to changes in skew, rotation, or translation.

3.10 – Profiling

Profiling is the extraction of the vertical and horizontal profiles of an image at different points. Profiles measure the distance between the edges of an image and the edges of a character within that image. To extract profiling features from an image containing a character, the image must first be cropped of whitespace so that the character fills the image entirely. After cropping, the profiling features are extracted in a normalized fashion. One method used to normalize profiling-based feature extraction involves defining a set number of locations along the height and width of the image at which to measure the profile values [11]. Such locations could include points above, below, and exactly at the center of the image. Another method used requires all images to be normalized to a single size, at which point profiling values will be taken at pixel down the side and across the top of the image. This experiment utilizes the former method.

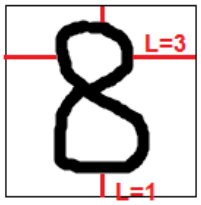

Figure 3. Profiling for the digit "8". This image is not cropped of whitespace, and the values for the left and top profiles are not shown.

3.11 – Crossings

Crossings refer to the number of times a specified line segment, either vertical or horizontal, crosses a character inside an image [11]. Crossings can be calculated either for each pixel of a scale-normalized image, or at a specified number of points down the side and across the top of an image. This normalizes the feature vector produced by calculating character crossings. Based on their nature, crossings are semi-invariant to skew and completely invariant to scale, but are not invariant to translation or rotation.

4. CLASSIFICATION METHODS

After feature extraction is used to obtain feature vectors from images, a classifier is applied to the feature vectors in an attempt to classify each vector as a particular character. Many classification algorithms developed over the past few decades can be applied directly to OCR, and each algorithm has advantages and disadvantages relating to time efficiency and scalability. However, the basis of every classification algorithm is the same, which is to take in feature vectors as input and provide an output that corresponds to a particular character.

4.1 – Neural Network

Neural networks are widely used in OCR because of their classification speed and ability to classify a wide range of data sets. Neural networks use a supervised learning approach, in which a network learns to classify characters based on a training set, which is a collection of images with labels that name the character contained in the image [9]. During training, the training images are converted to their respective feature vectors and presented to the neural network, and the structure of the network is then updated based on the errors made during classification.

In this experiment, a multi-layer perceptron (MLP) neural network is used for classification, and the neural network is trained using the Back-Propagation Algorithm. In back-propagation, the difference between the desired output and the actual output for a particular training image is calculated, and the weights in the network that are responsible for the error are adjusted after all training images have been evaluated [9]. This process is repeated until the error rate of the neural network is at an acceptable level.

4.2 – Euclidean Distance

Euclidean distance is a measurement of the difference between two images *A* and *B* as a function of the physical alteration, or distance, required to transform image *A* into image *B* [13]. As the true Euclidean distance value for two images is often computationally too expensive to determine, a variety of algorithms have been introduced to estimate Euclidean distance within the time complexity $O(m \times n)$, where m is the width of both images and n is the height [13]. This experiment employs a simple Euclidean distance estimation algorithm, which calculates the mean squared difference of the feature vectors in both images. For each feature in the feature vector the two images, feature *A* is compared to feature *B* and the difference between the two feature vectors is increased depending on the difference between the features being compared.

As a classification algorithm, Euclidean distance compares an image to be classified with a training image set of known characters. The Euclidean distance between the image in question and each training image is calculated, and the character with the smallest Euclidean distance is returned as the output.

4.3 – Decision Trees

Unlike most classification algorithms, decision trees do not classify inputs all in one step, but rather make decisions as to which classes a particular input could and could not belong. Each decision made by a tree narrows the possible classes to which an input could belong, and multiple decisions are made until a final class can be determined. Decision trees are popular as classification techniques because they are easy for humans to understand, and because they can classify inputs very quickly [14].

In most OCR applications that use decision trees as classifiers, each decision made by a decision tree is in regard to a single feature in a given feature vector. On the contrary, this experiment uses a decision tree implementation based on one proposed by Parvin et al. [14], where a neural network is used at each node of the tree to make decisions based on entire feature vectors.

5. POST-PROCESSING METHODS

Although many current OCR systems have respectable accuracies when applied to printed English fonts, no system to date has been able to translate text contained in images with perfect accuracy. That being said, classification mistakes are inevitable within any OCR system and must be dealt with accordingly. OCR translation results can be edited and proofread by humans, but a variety of automated systems can also be implemented to remove OCR mistakes in a practice referred to as post-processing.

5.1 – Spell Checking

Spell checking algorithms read in text translated by an OCR system word-by-word, and correct each word that is not contained in a dictionary. This experiment uses a modification of the Levenshtein Distance algorithm, which calculates the number of edits required to translate a misspelled word into each word contained in a dictionary (where an edit can be the removal, addition, or substitution of a single letter) [15]. After the Levenshtein Distance has been calculated for each word in the given dictionary, the word with the smallest number of edits necessary replaces the misspelled word.

However, calculating the Levenshtein Distance for each word in a dictionary containing hundreds of thousands of entries can become very computationally expensive, so in this experiment only words within a certain length of the misspelled word will considered, as proposed by Lalwani et al. [15].

5.2 – Common OCR Mistake Removal

Many characters, such as ‘I’ and ‘1’, have similar appearances in a variety of fonts and can be difficult for most OCR systems to distinguish. As a result, such similar characters characters can often be confused multiple times during the translation of a single image. In this experiment, a system is employed in combination with a dictionary in an attempt to remove common OCR mistakes made by similar characters. For each misspelled word in a translated

text, the characters in the word are replaced by characters that may have similar appearances in multiple fonts. If the resulting word is contained within the dictionary the original word is replaced.

6. RESULTS

6.1 – Feature Extraction

In the first phase of this empirical study, a MLP neural network is trained with back-propagation on a training set of 496 Arial font characters, which are of various sizes. A variety of feature extraction combinations are tested against an image containing 243 characters and the recognition rate is recorded for each combination, where the recognition rate measures the number of characters correctly classified out of the total number of characters in the image. The X-Y cut algorithm is used for segmentation, and no post-processing techniques are applied.

Table 1 shows the results of the classification attempts with the training time required to achieve 0.05 mean squared error on the training set. The recognition rate is also included as a percentage. According to the results, a combination of skeleton vectors, zoning, and feature points performed the best while the combination of chain codes, profiles and crossings performed the worst. Combinations that included vectors but no chain codes outperformed all combinations that included chain codes but no vectors, and no combination that included chain codes obtained a recognition rate higher than 50%. Combinations that did not include vectors or chain codes were tested, but most were not included in Table 1 because the neural network could not be successfully trained to the desired error rate of 0.05.

Feature Extraction Methods	Training Time (in seconds)	Recognition Rate
Vectors, Zoning, Feature Points	165	77.8%
Vectors, Feature Points	154	77.0%
Vectors, Crossings, Feature Pts.	190	70.8%
Vectors, Hu Moments, Feature Pts.	220	70.4%
Vectors, Profiles, Feature Pts.	260	70.4%
Vectors, Zoning, Crossings, Feature Pts.	143	65.0%
Vectors, Zoning	266	55.6%
Chain Code, Feature Pts.	142	44.9%

Vectors, Chain Code	156	40.3%
Vectors, Chain Code, Feature Pts.	183	37.9%
Chain Code, Profiles, Crossings	167	22.6%

Table 1. Comparison of performance for various feature extraction techniques.

6.2 – Segmentation

Two segmentation algorithms, X-Y cut and topological segmentation, are tested in this experiment. An image containing 415 characters of various font sizes (ranging from 12-point to 36-point) is segmented by both algorithms, and the number of segmentation mistakes made by each algorithm is recorded. A segmentation mistake occurs either when two or more separate characters are not segmented, or when a single character is incorrectly segmented into two or more characters.

Based on the results shown in Table 2, topological segmentation achieved a lower segmentation error than X-Y cut segmentation. Both algorithms achieved a segmentation error rate under 5%.

Segmentation Method	Segmentation Error Rate
Topological Segmentation	0.96%
X-Y Cut	2.89%

Table 2. Segmentation errors for methods used.

6.3 – Classification

MLP neural networks, Euclidean distance, and decision trees are tested as classification algorithms in the third phase of this experiment. First, two images containing 174 characters in 24-pt font are segmented into individual characters using topological segmentation. The first image contains Arial font characters, and the second image contains characters in Times New Roman font. After segmentation, each character is classified by one of the three algorithms. The process of segmentation and classification is then repeated for the remaining classification algorithms and the recognition rates are recorded, where the recognition rate is the percentage of characters correctly classified out of the total number of characters given to the algorithm. Straight pixel values, profiling, vectors, and crossings are used together for feature extraction, and no post-processing methods are employed.

Classification Method	Font Family	Recognition Rate
Neural Network	Arial	**98.85%**
Neural Network	Times New Roman	94.83%
Euclidean Distance	Arial	97.74%
Euclidean Distance	Times New Roman	63.22%
Decision Tree	Arial	94.83%
Decision Tree	Times New Roman	81.40%

Table 3. Recognition rates achieved by classification algorithms.

Table 3 outlines the results of the classification tests. The neural network achieved the highest recognition for both images, and the network also achieved the highest average recognition with an average recognition rate of 97.50%. Euclidean distance outperformed the decision tree on the image containing Arial font, and the decision tree performed better on Times New Roman font. Interestingly, all three classification algorithms performed better on Arial font than on Times New Roman font, even though an equal amount of training images was provided for both fonts.

6.4 – Post-Processing

Common OCR mistake removal and spell checking are tested together in this experiment as a single post-processing algorithm. Four separate images are tested; the first two images each contain 243 characters in Arial font of different sizes, and the last two images contain 92 characters in Times New Roman font of different sizes. Euclidean distance is used as a classification technique, and straight pixel values are used for feature extraction. After the image is segmented using the X-Y Cut algorithm and the characters are classified the recognition rate is recorded, and the result is then run against the aforementioned post-processing algorithm. After post-processing the recognition rate is recorded again, which is measured as a percentage of characters correctly classified.

The recognition rates for all four images are presented in Table 4. In two of the four images, the recognition rate increased after performing post-processing, while the recognition rates regarding the other two images remained the same.

Font Family	Recognition Rate (after classification)	Recognition Rate (after post-processing)
Arial	**86.8%**	**94.7%**
Arial	92.2%	92.2%
Times New Roman	89.1%	89.1%
Times New Roman	**95.7%**	**97.8%**

Table 4. Recognition rates achieved before and after post-processing.

7. DISCUSSION

This paper reviews a multitude of techniques for various steps in the OCR process with the intent of identifying both optimal solutions and solutions that might not perform well in an

enterprise-type application. The results of this experiment are a reminder of the many variables and procedures that may influence the performance of an OCR system.

The results regarding the feature extraction tests indicate the acute changes in recognition rates that can occur when altering the features used in a particular system. Based on this experiment, skeleton vectors are a very likely candidate for optimal feature extraction in an enterprise application. Zoning also appears to have promising success as a feature extraction technique, and when applied to vectors the recognition rate for the system used was always above 70%. This may be due to the idea that zone-based feature extraction techniques, such as zoning and vectors, are intrinsically scale-invariant and can be applied to images that have not been normalized. Zoning and vectors may also have optimal scalability properties, meaning they can be applied to a variety of character sets without much need for change within a single OCR system. Chain codes performed quite poorly in this experiment, although that may not be a sign that chain codes cannot be used for adequate feature extraction. Rather, a better normalization algorithm may be needed to properly apply chain codes to fonts of various sizes. In addition, chain codes are a relatively new method in the field of OCR, and more research may be needed before they can be used as a feature extraction technique in an enterprise application.

The popular claim is one that neural networks can be applied easily to a wide variety of OCR applications relative to other classification methods, and the results of this experiment support that idea. The neural network used was able to outperform both Euclidean distance and MLP-based decision trees on multiple fonts, which indicates that the neural network has potential to be an optimal choice as a classification algorithm in a complete OCR system. The statement must be made, however, that the true recognition capabilities of the classification algorithms described in this paper were not evaluated, and neural networks may not be the best choice when used in combination with different segmentation methods or feature extraction techniques. That being said, this experiment supports neural networks as a reasonable classification choice for an enterprise-type system, regardless of the OCR techniques used for other steps in the process.

For segmentation, the topological-based method was shown to be superior to the X-Y Cut algorithm in terms of accuracy. This displays the notion that segmenting characters based on structural or topological features will likely yield better accuracy than by using areas of whitespace as the primary basis for segmentation. That being said, the X-Y Cut algorithm is simpler and may be sufficient for certain applications.

While segmentation, feature extraction, and classification all play a vital role in the success of a complete OCR system, post-processing appears to be less important. While employing post-processing techniques will likely increase the overall recognition performance of a particular system, there is no evidence to show that the choice of post-processing techniques will radically alter the recognition capabilities of that system. However, more research is needed

to provide a definitive answer as to the extent of the role that post-processing plays in enterprise OCR applications.

Lastly, the complexity of various OCR techniques was shown to have an impact on the performance achieved within a complete system. Multiple highly complex feature extraction methods, including Zernike moments, were unable to be used successfully trained during experimentation, which reveals the relationship that complexity demonstrates with system performance. While complex feature extraction techniques may enjoy respectable performance advantages in controlled experiments, the same techniques can be difficult to implement properly within complete OCR systems. Furthermore, less complex feature extraction methods, such as skeleton vectors, have proven to be sufficient for the recognition of printed English fonts.

To conclude, optical character recognition remains a complex process in the field of computer science, and little evidence that certain methodologies are optimal or superior has been presented. That being said, certain techniques can be applied to a wide variety of enterprise-type OCR applications with reasonable success, while other techniques have proven to be too complex to be implemented in a general-purpose OCR system, even though they may be useful in specific applications.

REFERENCES

1. Rose Holley. *Analyzing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitization Programs. D-Lib Magazine, Vol. 15, No. 3.* 2009.
2. Oivind Due Trier, Anil K. Jain, and Torfinn Taxt. *Feature Extraction Methods for Character Recognition – A Survey. Pattern Recognition, Vol. 29, No. 4.* pp. 641-662. 1996.
3. Juan-Carlos Perez, Enrique Vidal, and Lourdes Sanchez. *Simple and Effective Feature Extraction for Optical Character Recognition. http://citeseerx.ist.psu.edu.* Date unknown.
4. Sandeep Saha, Nabarag Paul, Sayam Kumar Das, and Sandip Kundu. *Optical Character Recognition using 40-point Feature Extraction and Artificial Neural Network. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 4.* pp. 495-502. April 2013.
5. Song Mao and Tapas Kanungo. *Empirical Performance Evaluation of Page Segmentation Algorithms. http://citeseerx.ist.psu.edu.* 1999.
6. R. L. Hoffman and J. W. McCullough. *Segmentation Methods for Recognition of Machine-printed Characters. IBM Journal of Research and Development, Vol. 15, No. 2.* pp. 153-165. March 1971.
7. Richard G. Casey and Eric Lecolinet. *A Survey of Methods and Strategies in Character Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 7.* pp. 690-706. July 1996.
8. H. Izakian, S. A. Monadjemi, B. Tork Ladani, and K. Zaminifar. *Multi-Font Farsi/Arabic Isolated Character Recognition Using Chain Codes. World Academy of Science, Engineering, and Technology, Vol 19.* pp. 67-70. July 2008.
9. M. Zahid Hossain, M. Ashraful Amin, and Hong Yan. *Rapid Feature Extraction for Optical Character Recognition. http://arxiv.org.* June 2012.
10. Alireza Khotanzad and Yaw Hua Hong. *Invariant Image Recognition by Zernike Moments. Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 5.* pp. 489-497. May 1990.
11. Gaurav Y. Tawde and Jayashree M. Kundargi. *An Overview of Feature Extraction Techniques in OCR for Indian Scripts Focused on Offline Handwriting. International Journal of Engineering Research and Applications, Vol. 2, No. 1.* pp. 919-926. February 2013.

12. Vivek Shrivastava and Navdeep Sharma. *Artificial Neural Network based Optical Character Recognition. Signal and Image Processing: An International Journal, Vol. 3, No. 5.* pp. 73-80. October 2012.

13. A. Meijster, J.B.T.M. Roerdink, and W.H. Hesselink. *A General Algorithm for Computing Distance Transforms in Linear Time. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 2.* pp. 265-270. February 2003.

14. Hamid Parvin, Hosein Alizadeh, and Behrouz Minaei-Bidgoli. *A New Divide and Conquer Based Classification for OCR. Convergence and Hybrid Information Technologies.* pp. 426. March 2010.

15. Mahesh Lalwani, Nitesh Bagmar, and Saurin Parikh. *Efficient Algorithm for Auto Correction Using N-gram Indexing. International Journal of Computer and Communication Technology, Vol. 2, No. 7.* pp. 23-27. 2011.