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 this guy [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 this dude [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.

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.

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.

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.

FEATURE EXTRACTION METHODS

References (sort of)

1. OCR Accuracy Rates
2. Advanced Feature Extraction
3. Simple Feature Extraction for OCR
4. 40 Point Feature Extraction for Neural Networks
5. Analysis of Character Segmentation Algorithms
6. Topological Segmenting of Touching Characters
7. More Segmentation Methods