TJHSST Senior Research Project
Devlopment of a Generic Font OCR
Second Quarter Research Paper
2007-2008

Nathan Harmata

January 23, 2008

Abstract

OCR (Optical Character Recognition) is a very practical field of Computer Science. Since the late 1980's, researchers have been developing systems to identify text from non electronic text sources, like pictures or papers. The use of OCR systems has spanned from making books in Braille to sorting mail by zip code by United States Post Office.

Keywords: OCR, Image Processing

1 Introduction

The goal of this project is to create an application that can read text from electronic picture files. One of the main focuses will be developing a generic way to recognize different fonts, rather than hardcoding in definitions for specific fonts. Although OCR is by no means a "new" field, it has still yet to be fully explored. There are very few OCR applications readily available to the public, and even the ones that are free of charge are lacking in performance and consistancy.

For now, an application that can read and recognize "simple" pictures, i.e. ones with minimal headers and only text, will be developed. If that

is successful, more advanced techniques to handle headers and 'background noise,' or erroneous markings in an image, will be used. The algorithms used to actually analyze text will be developed completely from strach, and will be made to work irrespective of the font of the input image.

2 Background

OCR systems have been around since the late 1980's. Still, they are not widely available or used by the public. The results from a review of the free ones on the Linux operating system are not very promising. [1] Although most of them had measured accuracies above 94 percent, that is not good enough. The one commercial product tested, Aspire OCR, was accurate only 91.5 percent of the time. The most likely industry standard, Tesseract, is also one of the oldest OCR systems. The review measured it to have an accuracy rate of 99 percent. Development on it started in 1985 and it is still used as the OCR engine for Ocropus, Google's textual analysis application. It is unlikely that this project will be able to achieve similar success, but the goal is to be on par with the current OCR options.

3 Procedures

The current version works as follows:

The application accepts a picture file as input. The input file must be a png image of text. The current version accepts images of only text. Methods will eventually be implemented to deal with text that is actually part of a larger picture. [4] The picture is first boxed to remove excess whitespace along the border. Then, the program finds the locations of horizontal lines on whitespace inside the new image and pairs them so that paragraphs and the spaces between paragraphs become separated. A similar method is used to parse paragrahs into lines. Spaces between words are handled by comparing the size of an actual space to the average size of the spaces between letters. Lines are broken into words, which in turn are broken into letters. The idea of this is to then have individual pictures of each letter, which can be interpreted as a graph of pixels.

Each character is then compared to a database so that the best match

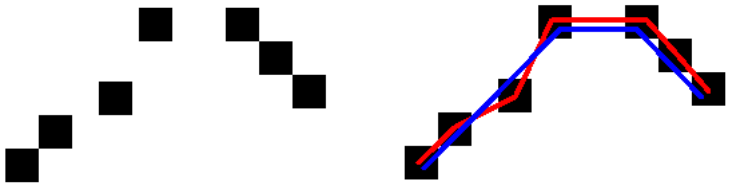

Figure 1: Visualization of the 'Slope Field' algorithm. To the top is some input image. In the image on the bottom, the red lines represent the intermediate line segments and the blue lines represent the final line segments after execution of the algorithm.

can be determined. The database was generated from the results of a testing program (see Testing). It contains 'definitions' of each letter in the English alphabet per my own algorithm. Starting with an image of a single letter as mentioned above, it works as follows:

1. The image is broken into portions, referred to as 'sectors', that pass the vertical line test. This 'Sector Parsing' is done by finding the locations of the optimal cuts.

2. Each sector image is converted into a two-dimensional array of pixels. Pixels that are whitespace and thus not text are removed, meaning that the array is comprised of only 1's and 0's.

3. Each of these arrays of pixels is then converted into a SlopeField. This is done by first averaging together horizontal groups of pixels into a single pixel, thus getting rid of unnecessary data. Then, starting with the lower left pixel, line segments between contiguous pixels are formed and their slopes calculatted. Contiguous line segments with slopes similar in sign and magnitude are paired together, by treating keeping the starting point of the first one and the end point of the second one. The slope is this new line segment is re-calculated, and the process continues. The result is a collection of line segments or, more simply, a collection of vertices that form the line segments.

Letters are defined by a 'Sector Vector,' which contains the number of sectors, the total number of line segments in all the sectors of the letter, and the sign of the slope of the first line segment. The comparison is performed by finding the element in the database whose distance to the 'Sector Vector' representation of the image is a minimum. A scalar is applied to certain attributes of the Sector Vector to give them more weight. For example,

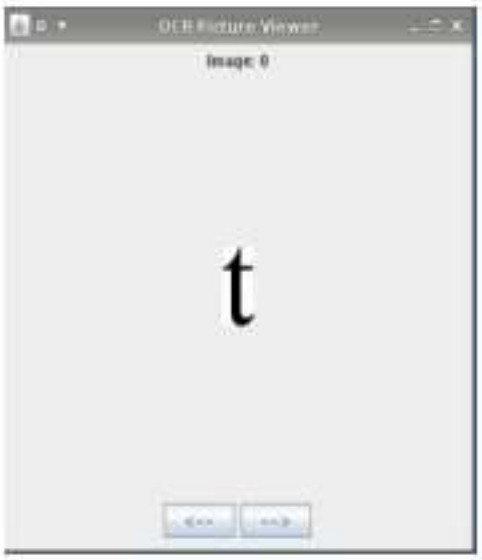

Figure 2: The simple viewer interface.

testing has shown that the number of sectors is a very good indicator of the value of a letter. The following expression is used to find the magnitude of the different vector between Sector Vectors A and B.

For now, the closest two matches for each letter are considered. Using those possibilities, all possible permuations can be generated. These can then be compared to a dictionary reference or to some grammar verification system to assess to validty of the translation of the word. Even simply using a dictionary, as is being done in the current iteration of the OCR system, has been shown to vastly improve preformance. [3]

$$\sqrt{\sum_{i=1}^{n}(scalar_i * (attribute_i A - attribute_i B)^2)} \tag{1}$$

3.1 Testing and Results

1. A user-friendly interface for viewing the pictures of parsed words has already been developed. This allows for manual visual detection of errors in letter parsing.

2. A generic testing program was developed to assess the results of the addition of new methods. It simply runs the current image

Pattern	Total Frequency	Average Frequency
-1 6	4	0
-1 5	4	0
-1 4	14	2
-1 3	14	2
-1 2	24	4
-1 1	20	4
-1 0	32	6
1 1	2	0
1 2	3	0
1 3	4	0
1 4	6	1
1 6	2	0
1 9	1	0

Figure 3: Tested relationship (a).

transformation algorithm on every letter in the alphabet for several fonts, which were picked for their different attributes:

(a) Arial

(b) Comic Sans MS

(c) Courier

(d) Helvetica

(e) Luxi Sans

The results were ouputted to a file and a separate program analyzed them. Three relationships were determined:

1. The total frequency of each pattern (e.g. attributes of SectorVector) and the average frequency of that pattern in each font.

2. For each pattern, a list of matching letters.

3. For each letter, a frequency table of each pattern.

The following computer languages, algorithms and programs are being used.

3.2 Software

1. The OCR system is written entirely in Java.

Pattern		
Segments*1st Slope	**# of Sectors**	**Matching Letters**
-3	3	a
-2	3	e o
-2	5	g
-1	3	c d q
-1	4	s z
0	1	l n r
0	2	I t
0	3	f j
1	1	h m u
1	3	p
2	1	v
2	3	x y
4	3	b k
5	1	w

Figure 4: Tested relationship (b).

2. Java's ImageIO class is used for picture input and output.

3. Java's BufferedImage class is ued to handle pictures.

3.3 Algorithms/Programs

1. KolourPaint is being used to make picture files for input and to precisely view pictures.

2. My own algorithm which transforms an image of a letter into a collection of line segments (see Procedures) is used.

3. The current working version of the OCR system is called 'OCR Manager.' It uses the methods outlined in the 'Procedures' section and goes from the input of an image file to the output of all the possible matching words. For example, the input shown in Figure 6 generates the following output:

```
beak
boob
book
keek
kook
```

Letter	Number of Segments * 1st Slope														
	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
a	1	2	1	-	-	-	-	-	-	1	-	-	-	-	-
b	-	1	-	2	-	-	-	-	-	2	-	-	-	-	-
c	-	-	-	2	2	-	-	1	-	-	-	-	-	-	-
d	-	-	-	-	3	1	-	-	-	1	-	-	-	-	-
e	-	-	3	2	-	-	-	-	-	-	-	-	-	-	-
f	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-
g	-	-	-	1	2	-	-	-	-	-	-	1	-	-	-
h	-	-	-	1	2	1	-	-	1	-	-	-	-	-	-
i	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-
j	-	-	-	1	1	3	-	-	-	-	-	-	-	-	-
k	1	-	1	3	-	-	-	-	-	-	-	-	-	-	-
l	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-
m	-	2	2	-	-	-	-	-	-	-	-	-	-	-	1
n	-	-	-	-	3	-	-	1	1	-	-	-	-	-	-
o	-	2	3	-	-	-	-	-	-	-	-	-	-	-	-
p	-	1	1	-	-	-	-	-	1	2	-	-	-	-	-
q	-	-	-	-	3	1	-	-	-	-	-	1	-	-	-
r	-	-	-	1	-	3	-	1	-	-	-	-	-	-	-
s	-	1	-	3	-	-	1	-	-	-	-	-	-	-	-
t	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-

Figure 5: Tested relationship (c).

Figure 6: Example input for OCR Manager.

4 Conclusions

A lot of progress has been made since the first iteration of the OCR system. The original version was based off of direct comparisons to a cache, meaning that only text of the font that was cached could be read. The current, version, however has been designed to work with any font. This is done by comparisons to a cache of generic letter definitions, created by my own algorithm.

Most of the remaining work will be devoted to improving these methods. Perhaps the best way to do so is to either improve the current algorithms or to develop new ones, such that the relationship shown in Figure 4 is strengthened. This relationships measured how 'spread out' the results are; i.e. how different one letter is from another. The fewer letters in each group, the better, but at the same time, the more complex the comparison mechanism, the more devastating errors casued by 'noise' can be.

Further work will also have to be done to improve the current methods for the detection and removal of noise. There are various methods that can be used to accomplish this. [2] Overall, the successes the current version has had shows that, with improvement, it will be a viable way to implement an OCR system.

References

[1] Austin Acton. A review of free optical character recognition software, 2007.

[2] Faisal Shafait, Joost van Beusekom, Daniel Keysers1, and Thomas M. Breuel. Page frame detection for marginal noise removal from scanned documents, 2007.

[3] Kazem Taghva, Julie Borsack, and Allen Condit. An expert system for automatically correcting ocr output, 1994.

[4] Victor Wu, R. Manmatha, and Edward M.Riseman. Finding text in images, 1997.