

Measuring the Robustness of Character Shape Coding

A. Lawrence Spitz and Paul Marks

Document Recognition Technologies, Inc.
459 Hamilton Avenue, Suite 204
Palo Alto, California 94301, USA
{spitz,marks}@docrec.com

Earlier claims of great robustness for the character shape coding process have been largely unsupported. We provide quantitative measures of the sensitivity of the character shape coding process to the text input, production values and image quality and to the complexity of the destination character shape codes. Using this evaluation tool we can tune the character shape coding process in a systematic way and also develop new versions of the shape codes appropriately adapted to particular applications.

1 Introduction

The process of generating Character Shape Codes (CSCs) encodes only the grossest features of character image size and placement, while ignoring the high spatial frequency features that Optical Character Recognition (OCR) relies upon [1][2]. Because character shape coding is a lossy transform resulting in an ambiguous representation of the characters in the document, it is important that robustness be maintained at as high a level as possible lest the representation become meaningless. The robustness required varies with the particular application, but whether that application is word spotting, language identification, information retrieval or support for word recognition, tolerance for a range of production values and types of noise present in document images is quite important.

Reliance only on gross features of characters reduces the susceptibility to significant noise, distortion and low resolution. In enhancing the representation, we must balance the desired increased specificity (reduced ambiguity) of new character shape code versions with the change in susceptibility to font variabilities and image defects.

We measure robustness by synthesizing text images with particular characteristics and subsequently analyzing these images.

Synthesis involves several steps. We describe some statistics of frequent words and of letter juxtaposition in English and the construction of ground truth appropriate for measuring character shape code accuracy in Section 2. In Section 3, we describe the generation of text images based on this truth information and on typesetting font, face and size variations. Section 4 describes differing forms of degradation based on the types of noise commonly found in document images.

Likewise, image analysis has several aspects. Multiple versions of CSCs are briefly described in Section 5. We compare the character shape coded output to the truth data that have been appropriately transformed into CSCs in Section 6, and show how noise affects performance in Section 7 and the effects relevant to Character Shape Coding in Section 8.

In Section 9 we draw conclusions from the current study and lay out some of the outstanding problems that deserve attention in future work in Section 10.

2 Synthesis

Text image synthesis has two components, signal and noise. The signal is that part of the text image that is under the control of the producer and is the result of the combination of character content and production values. Noise is the degradation of the signal and, in this study, is modeled by inputs to a document image defect model.

We measure the performance of character shape coding under a large number of varying, largely independent, conditions. The variables are shown in Table 1.

Table 1: Variables over which we measured character shape coding accuracy.

Variable	Number of values	Variable	Number of values	Variable	Number of values
Font	17	Spacing	15	Blur	4
Face	3 (nominal)	Resolution	3	Threshold	9
Point size	11	Skew	7	Sensitivity	9

Testing all possible combinations would result in the generation of more than 50 million images and thus make a full analysis impossible. We examine each variable while holding the others constant at an arbitrary, but sub-optimal, level. In other words, we investigate the degradation due to the change of one variable when there is already some degradation from other influences. Thus it is the shape of the curve of degradation that is important rather than the absolute values of accuracy as a function of the variable.

2.1 Ground Truth

The truth file starts out with the English sentence “The quick brown fox jumps over the lazy dog” which contains all the letters of the alphabet. The rest of the truth is based on the analysis of a training corpus of 650,000 words. The truth contains the 63 highest frequency words that together comprise 35% of the words in the corpus while retaining capitalization. That is, **the** and **The** are considered separate words.

The 69 most frequent non blank-containing bigrams and the 90 most frequent trigrams follow. These n-grams represent 72% and 36% of the characters respectively. A text image of the truth file is shown in Figure 1.

3 Production Values

We had a high level of control over the typesetting of the test images.

3.1 Fonts

The fonts studied are shown in Figure 2 and named in Table 3. Proportionally spaced, mono-spaced, serif, sans-serif and script-like fonts are included.

3.2 Faces

There are three faces (Roman, Bold, Italic) represented for most fonts. However, not all fonts are present in all faces, and there are combinations of faces such as BoldOblique which are found only in a small number of fonts.

3.4 Horizontal Spacing

On proportionally spaced fonts we start with the designed pair kerning value and add or subtract space to encourage or discourage the touching of character pairs.

On Courier (the only mono-spaced font in our study) we started with the default character pitch and added and subtracted inter-character space from that value.

3.5 Image Generation

The transformation from the ASCII truth information to an image file is a two step process. First the Aladdin Enterprises `gs1p.ps` program, functionally similar to the Adobe UNIX program `enscript`, but modified to give user control over horizontal spacing, is run to produce a PostScript output file. then the PostScript file is rendered into a TIFF bitmap at 300 spi by the Aladdin `gs` (GhostScript) program.

The resultant TIFF images are cropped to within one pixel of the data on all four sides.

4 Image Degradation

We use the Baird image defect model to add different types and degrees of degradation to the synthesized images. Note that we apply the degradation to page images rather than character images as Baird does in his examples. See [5] for a more complete discussion of the types of degradation performed.

The effects of varying several variables are shown in Figures 3 - 7. Threshold and sensitivity are two variables that are not independent. Decreasing threshold and increasing sensitivity both make the image darker.

4.1 Sensitivity



Fig. 3: Increasing the sensitivity behaves much like the reduction of threshold and results in additive noise.

4.2 Threshold

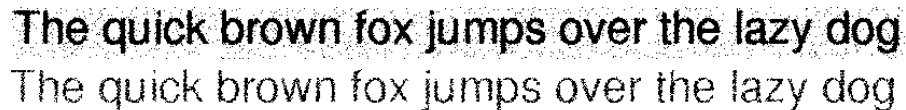


Fig. 4: The effects of low threshold resulting in additive noise and of high threshold resulting in character image degradation.

4.3 Skew

Skew is a common document image defect. Traditional OCRs often recognize individual characters without regard for their positions with respect to the neighboring characters. Character shape coding, on the other hand, is wholly dependent on the knowledge of the baseline and x-line positions in the text line and thus potentially more vulnerable to skew-induced errors than OCR would be. Determining these fiducial lines is a process which must be either tolerant of skew or rely on upstream skew

detection and correction. In our implementation we detect and correct skew before attempting to perform character shape coding.

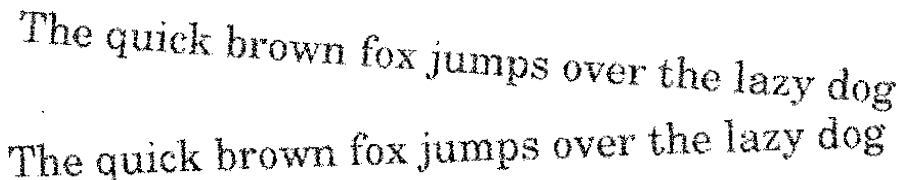


Fig. 5: Image skew of -4 and +2 degrees.

4.4 Blur

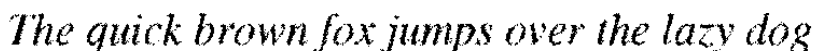


Fig. 6: Blur can cause otherwise separate characters to touch.

4.5 Resolution

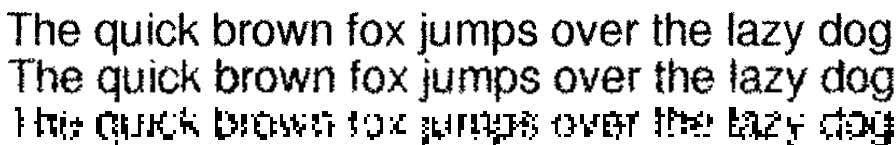


Fig. 7: Re-sampling the basic image at 300, 200 and 100 spi.

5 Character Shape Coding

Character shape coding (V_{0-2}) has been discussed in a number of previous publications [1][2]. In this paper we will add three new versions, two of which (V_3 and V_5) take advantage of the presence of one or more vertical stems in some ascender characters. V_3 lumps all x-height characters together, providing great robustness in poor images. These coding methods are outlined in Table 2.

Table 2: Character shape coding schemes for alphabetic characters.

Characters	V_0	V_1	V_2	V_3	V_4	V_5
amorsuvxwz	x	x	x	x	x	x
n			n			
c		e			c	
e					e	
ACGIOQSTUVWXYZflt	A			A	A	A
HMN				N		N
bhKL				b		b
BDEKR				E		E
PF				P		P
dJ				d		d
j				j		
i				i		
gpqy				g		

However, space restrictions dictate that in this paper we restrict the results of analysis to V_0 (the most basic version) and V_5 (the most complex) except in the instance of overall performance on various fonts and point size.

6 Analysis

The large number of variables, each of which can take on a large number of values, and the multidimensional nature leads to an overwhelming amount of data and considerable difficulty in isolating the effects of individual variables. In order to work around this complexity, we have adopted a method that might be characterized as serial one dimensional analysis which is analogous to a ridge following algorithm.

We ran initial tests on varying point size on three proportional (one serif, one serif italic and one san-serif) fonts and one mono-spaced. After that we held point size constant and varied only the font.

Since, in this paper, we are only concerned with the accuracy of the character shape coding process, not in the accuracy of determining word shape tokens, for instance, we remove blank characters from both the encoded truth and from the test data. Doing so loses information about incorrectly interpolated spaces and missed spaces, but makes the remaining analysis much easier.

The two strings are compared using the Wagner-Fischer algorithm [6]. We collect information on the insertions, deletions and substitutions between the encoded truth and the output of the character shape coder.

7 Overall Results

7.1 Production Values

In Table 3 we show the accuracy of CSC V_0 through V_5 on all 47 fonts. It is interesting to note that V_5 accuracy, though in general less than V_0 accuracy, approaches it for some fonts. This is a good illustration of how the selection of an appropriate CSC version can enhance the shape coding process. Also note that the inappropriateness of V_5 CSCs for italic fonts is obvious.

Column by column comparisons can yield significant insights. For example, comparing V_3 to V_2 shows the degradation in performance due to the difficulty in detecting the southward concavity that allows us to distinguish a “n” from other x-height characters. Likewise V_1 compared to V_0 shows the difficulty of detecting eastward concavity and V_5 and V_3 demonstrate the reliable detachability of the crossbar in an “e”.

Figures 8 and 9 show the effects of varying the synthesis of the images. We start by selecting a single font and varying the point size. As can be readily seen, above 6 pt, the character shape coding process is very accurate. At 10 pt, it is essentially perfect.

Table 3: Character shape coding accuracy as a function of font, sorted by accuracy on V_0 coding

Font/Face	Accuracy (%)					
	V_0	V_1	V_2	V_3	V_4	V_5
Textbook-Bold	100.0	100.0	100.0	100.0	100.0	100.0
NewCenturySchlbk-Roman	100.0	100.0	100.0	100.0	100.0	100.0
Antiqua	100.0	100.0	100.0	100.0	100.0	100.0
Palatino-Roman	100.0	100.0	100.0	99.8	100.0	99.8
Antiqua-Bold	100.0	100.0	100.0	99.8	100.0	99.8
NewCenturySchlbk-Bold	100.0	100.0	99.8	99.8	100.0	99.8
Textbook	100.0	100.0	97.8	99.8	100.0	99.8
Magazine	100.0	100.0	97.8	99.8	100.0	99.8
Helvetica	100.0	100.0	97.8	93.2	100.0	93.2
Handbook	100.0	100.0	97.8	100.0	100.0	100.0
Courier	100.0	100.0	100.0	89.0	100.0	100.0
Magazine-Italic	100.0	97.3	97.3	89.9	100.0	89.9
Handbook-Italic	100.0	83.2	83.2	88.7	100.0	88.7
AvantGarde-Book	99.8	99.8	99.8	92.9	99.8	92.9
Palatino-Bold	99.7	99.7	99.7	99.5	99.7	99.5
Helvetica-Oblique	99.7	99.7	99.7	88.2	99.7	88.2
Helvetica-Narrow-Bold	99.7	99.7	99.7	92.5	99.7	92.5
Helvetica-Narrow	99.7	99.7	97.5	92.6	99.7	92.6
Helvetica-BoldOblique	99.7	99.7	99.7	90.9	99.7	87.9
Helvetica-Bold	99.7	99.7	99.7	99.2	99.7	99.2
AvantGarde-DemiOblique	99.7	99.7	99.7	91.0	99.7	91.0
AvantGarde-BookOblique	99.7	99.7	99.7	88.4	99.7	88.4
Magazine-Bold	99.5	99.5	99.5	99.2	99.5	99.2
Helvetica-Narrow-Oblique	99.5	99.5	99.5	88.1	99.5	88.1
Handbook-Bold	99.5	99.5	99.4	99.1	99.5	99.1
College-Bold	99.5	99.5	92.9	99.2	99.5	99.2
College	99.5	99.7	92.9	99.4	99.7	99.4
Times-Bold	99.4	99.2	92.5	99.1	99.2	98.9
Helvetica-Narrow-BoldOblique	99.4	99.4	99.4	90.6	99.4	87.6
Bookman-Light	99.2	99.2	99.1	99.1	99.2	99.1
AvantGarde-Demi	98.9	98.9	98.9	97.8	98.9	97.8
NewCenturySchlbk-Italic	98.7	84.6	84.6	87.3	96.1	84.7
NewCenturySchlbk-BoldItalic	98.7	84.6	84.6	87.4	96.1	84.7
Antiqua-Italic	98.7	98.7	98.7	87.3	98.7	87.3
Palatino-Italic	98.6	84.9	85.1	87.1	98.6	87.1
Palatino-BoldItalic	98.4	98.4	98.4	87.1	98.4	87.1
Lazurski	98.4	95.8	95.6	98.1	95.8	95.8
Times-Roman	98.3	98.3	98.1	97.8	98.3	97.8
Times-BoldItalic	98.3	84.1	84.1	86.8	98.3	86.8
Bookman-DemiItalic	97.6	97.6	97.6	90.4	97.6	90.4
College-Italic	95.6	95.6	88.8	84.1	95.6	84.1
Textbook-Italic	95.4	92.8	92.9	83.6	95.4	84.0
Bookman-Demi	95.3	95.3	88.5	94.3	95.3	94.2
Bookman-LightItalic	94.8	95.3	95.1	86.9	91.7	82.5
ZapfChancery-MediumItalic	94.5	90.7	84.4	83.0	90.7	79.7
Advertisement	91.4	91.2	84.4	84.3	91.0	84.1
Times-Italic	89.5	72.6	72.6	78.6	89.5	78.3

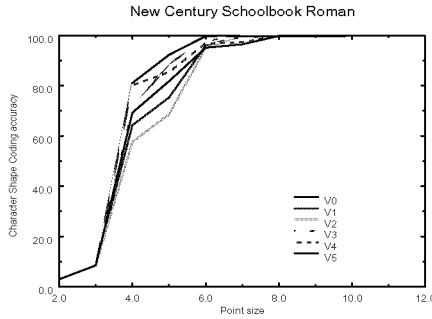


Fig. 8: Effect of point size on character shape coding accuracy.

Referring back to Figure 2, note that the first font shown, Advertisement, is designed without descenders, and contains many glyphs comprising multiple connected components where the canonical form has only one (**g,h,n,m,r,u,w**). Clearly this font is inappropriate for shape coding, but as its name indicates, it was not designed for general purpose running text. It is included here to demonstrate the contrast to more traditional fonts.

Varying the horizontal spacing between characters reveals the susceptibility of the font to the generation of unintended ligatures between character pairs. Note that Courier is extremely robust in the face of tight spacing.

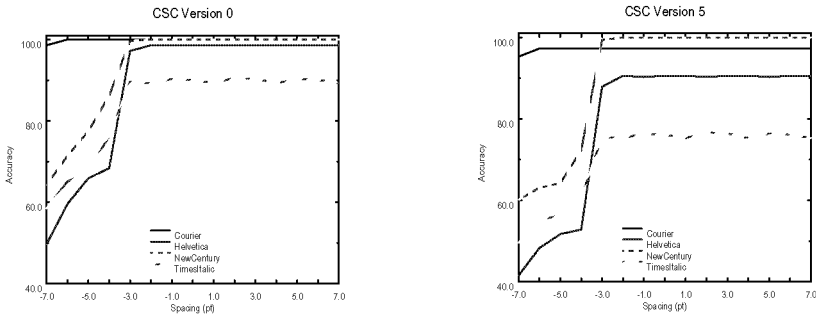


Fig. 9: Spacing (kerning in proportionally spaced fonts) is only important below the threshold where characters start to touch one another.

7.2 Effect of Noise

Figures 10 through 13 demonstrate the effects of adding different types of noise to the signal.

Note that the two CSC versions demonstrate approximately the same response to additive noise induced by a low threshold, while they exhibit different behaviors when a high threshold induces rough or broken characters.

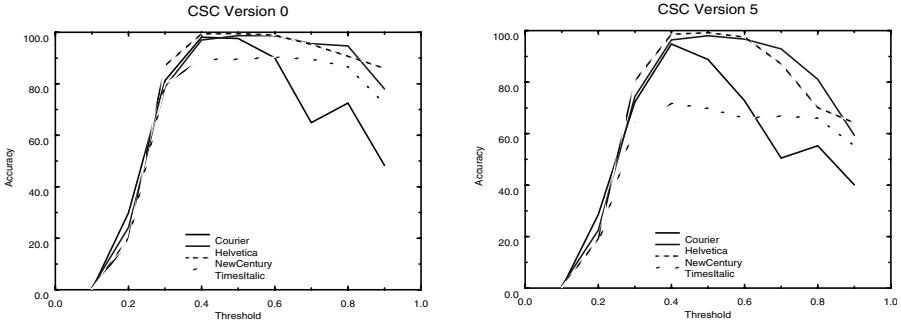


Fig. 10: The effect of varying the threshold on four fonts and two CSC versions

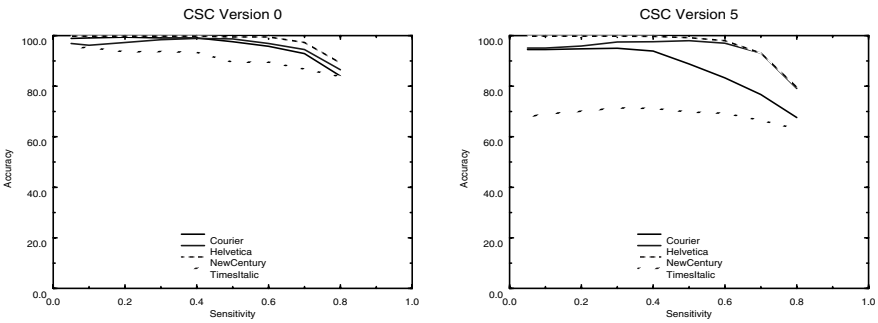


Fig. 11: The effect of varying the sensitivity on four fonts and two CSC versions

The effect of skew on the character shape coding process is minimal, due largely to the skew (and warp) detection and correction incorporated into the process of CSC generation [3].

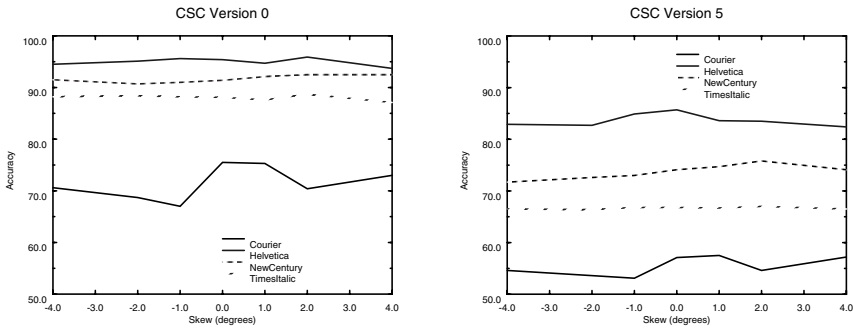


Fig. 12: Skew does not affect accuracy very much.

Courier is much less tolerant of blur than New Century Schoolbook.

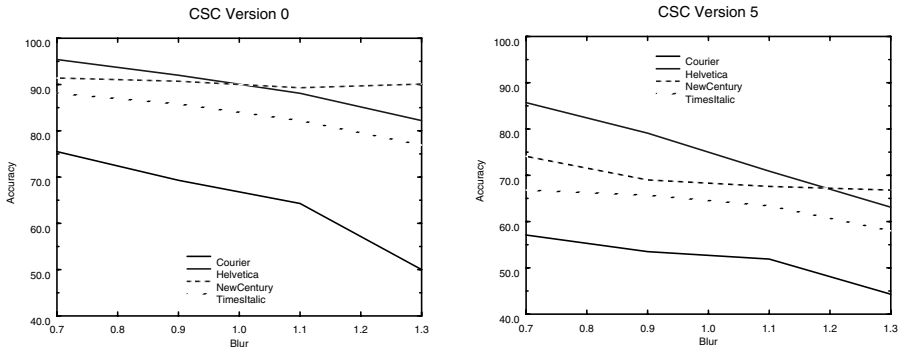


Fig. 13: Increasing blur affects different fonts very differently.

8 Character Shape Coding Results

With the exception of Esakov, *et al.* [4], there are few published studies of the effects on accuracy of an OCR process as a function of image content and quality with a common ground truth. They find non-space deletion errors to be much more common in Times than in Helvetica or Courier. They find that the most frequent substitution errors where the truth character is alphabetic, the detected character is of the same broad shape code class as the truth, i.e. ascenders yield ascenders, x-height yields x-height. An important exception is “r” detected as “t” for Times. “e” is frequently detected as “c” for both Times and Courier. For some versions of our shape coding process, this would constitute an error, for others “e” and “c” map to the same class.

Some characters are more or less susceptible to shape coding errors than others. For the first time we have the tools to examine the systematic errors and to perhaps redesign the shape codes based on actual performance over a wide range of data.

Deletions are rare. Insertions due to broken characters are more common. Substitutions, usually involving failure to detect nominal ascender characters such as “t” as having ascenders, are the most common. Insertions and deletions are more dependent on font and image degradation and less on size. Substitutions become rarer as the point size grows.

V_5 , which is dependent on the detection of stems, yields substitutions of one ascender code for another when the stem finder fails.

We now have the data on which to select an appropriate shape coding version as a function of the font and generalizations about image quality. It is obvious that V_5 is inappropriate for italic (or oblique) fonts because of the lack of vertical stems. Should it become necessary to process large amounts of italic text, a CSC version could be developed to detect the slanted stems.

9 Conclusions

Developing these techniques for evaluation of character shape coding performance has been a useful exercise in supporting the application of this technology to various classes of problems. In addition, the results of this evaluation have been fed back into the character shape coding process in order to find bugs and to suggest ways of extending the shape coding process.

In the process we have developed three new versions of the CSC set and started to develop guidelines for the application of particular versions for particular applications. The most obvious example of this is the development of a CSC version tuned to Courier. This version is suitable for application to typewritten corpora where Courier is the dominant font.

Recognizing the difficulty of assessing commercial character recognizers, we would like to encourage the study of real OCRs and their sensitivity to font and image quality variables.

Of the four fonts studied in depth, New Century Schoolbook Roman is the most robust in terms of tolerance to changes in production values and image degradation.

10 Future Work

None of the high frequency words, bigrams or trigrams used in this study contains a “ff”, “fi” or “fl” character sequence. In our training corpus these character sequences only occurred in 1.3% of the words. However when these sequences do occur and when many proportionally spaced fonts are in use, these character pairs are set as ligatures resulting in a single CSC being generated for two characters of ground truth. Without modification, our current analysis would count this as a deletion error. A more thorough analysis should account for the low frequency occurrence.

The truth on which we tested contains very few capital letters. It contains no punctuation or characters needed to represent languages other than English (e.g.: ä, ô, ñ, ø, å, ß, ç). In order to evaluate text in the languages that use these characters, representative truth files must be developed and images based on this truth must be analyzed.

In this study we ignored the presence or absence of inter-word spaces in the interest of testing only character shape coding accuracy. In the future we will address the problem of word shape token accuracy which will rely on accurate detection of spaces.

Acknowledgments

We appreciate the assistance of Henry Baird in ensuring that our implementation of the image defect model is functionally equivalent to his published model. L. Peter Deutsch modified GhostScript to give us full control over kerning. Daniel Lopresti found a substantive error in an earlier version of this paper and made other useful suggestions.

References

1. A. L. Spitz, "Generalized line, word and character finding", *Progress in Image Analysis and Processing III*. S. Impedovo (ed.), pp. 377-383, World Scientific, (1993).
2. A. L. Spitz, "Text characterization by connected component transformation", *Proc. SPIE*, San Jose, CA, pp. 97-105, (1994).
3. A.L. Spitz, "Analysis of compressed document images for dominant skew, multiple skew and logotype detection", *Comp. Vision Image Understanding*, 70, 3, pp. 321-334, (1998).
4. J. Esakov, D.P. Lopresti & J.S. Sandberg, "Classification and distribution of optical character recognition errors", *Proc. SPIE*, San Jose, CA, pp. 204-216, (1994).
5. H. S. Baird, "Document image defect models", H.S. Baird, H. Bunke, and K. Yamamoto, eds., *Structured Document Analysis*. New York, Springer-Verlag, pp. 546-556, (1992).
6. R. A. Wagner & M. J. Fischer, "The String-to-String Correction Problem", *J. Assoc. Computing Machinery*, 21,1, pp. 168-173, (1974).