A new adaptive edge enhancement algorithm for color laser printers

Manoj Kumar Reddy

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
A NEW ADAPTIVE EDGE ENHANCEMENT ALGORITHM FOR
COLOR LASER PRINTERS

by

Manoj Kumar M. Reddy

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

Approved by:

Prof. ____________________________________________
Thesis Advisor – Dr. Eli Saber

Prof. ____________________________________________
Thesis Committee Member – Dr. Sohail A. Dianat

Prof. ____________________________________________
Thesis Committee Member – Dr. Eric Peskin

Prof. ____________________________________________
Electrical Engineering Department Head – Dr. Vincent Amuso

Department of Electrical Engineering,
College of Engineering,
ROCHESTER INSTITUTE OF TECHNOLOGY,
Rochester, New York
November 2007
Thesis Author Permission Statement

Thesis Title: A New Adaptive Edge Enhancement Algorithm for Color Laser Printers
Author: Manoj Kumar M. Reddy
Degree: Master of Science
Program: Electrical Engineering
College: Kate Gleason College of Engineering

I understand that I must submit a print copy of my graduate thesis to the RIT Archives, as per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make my thesis in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Print Reproduction Permission Granted:

I, ______________________________, hereby grant permission to the Rochester Institute of Technology to reproduce my print thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: ________________________________ Date: _____________
ACKNOWLEDGMENTS

My intention before documenting this thesis was to skip this section all together; but as the document took shape, I cannot resist writing this part. First and foremost I would like to thank my parents and my brother for being together and supporting every decision of mine; I wouldn’t be here but for them. I would like to thank my teachers who are responsible for guiding me all this way by imparting their knowledge and experience so that I could avoid their pitfalls. First Ms. S. Subramanian who laid my foundations in mathematics; Ms. L. Prasanna for all her guidance during secondary school that helped me become a better writer. I whole-heartedly thank Mr. S. Varaprasad for all the encouragement and support during undergraduate school. I learnt an invaluable lesson from him that failure is not a sign of lack of ability; but of not fighting hard enough.

At R.I.T., I would first like to thank the E.E. department for allowing me to pursue my masters’ here; Dr. E. Saber not only his support and trust but also for cultivating the discipline to approach and solve problems. Many thanks to Dr. V. Misic for all the long discussions during my thesis, listening and challenging ideas thus keeping the creative factory running all throughout. Heart felt wishes to Hewlett Packard (H.P.), Boise for funding my thesis and to Jeff Trask; my technical liaison at H.P. who constantly monitored and kept the solutions practical for commercial usage. Finally, I would like to thank Dr. E. Peskin for an exhaustive review of my thesis. If you happen to read any section or view any flowchart and feel “Wow!!, that’s really clear” its mostly his hand there. I would like to take the opportunity to thank Dr. N. Rao, C.I.S & Dr. P. Reddy, C.S. for their unconditional support and everyone at the racquet club apartments and at the imaging lab who supported me one way or the other at R.I.T. and made this place as close to home as it could possibly be.

- Manoj Kumar M. Reddy
# TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................5

ABSTRACT ....................................................................................................................6

Chapter 1 INTRODUCTION .............................................................................................7
  1.1 Motivation.............................................................................................................7
  1.2 Contributions .......................................................................................................7
  1.3 Thesis organization ..............................................................................................8

Chapter 2 BACKGROUND AND LITREATURE REVIEW ...........................................9
  2.1 Literature review .................................................................................................9
  2.2 Background .........................................................................................................12

Chapter 3 PROPOSED ALGORITHM .............................................................................15
  3.1 Proposed algorithm ............................................................................................16
  3.2 Module 1 ............................................................................................................16
  3.3 Module 2 ............................................................................................................17
  3.4 Dynamic threshold .............................................................................................19
  3.5 Demonstration at pixel level ...............................................................................22

Chapter 4 RESULTS AND DISCUSSIONS.................................................................24

Chapter 5 CONCLUSIONS AND FUTURE WORK ....................................................28

REFERENCES ............................................................................................................31
LIST OF FIGURES

1a Experimental setup ................................................................. 15
1b Modified experimental setup .................................................. 15
2. Proposed algorithm ................................................................. 17
3. Order of threshold comparison .................................................. 18
4. Edge enhancement module ....................................................... 20
5. Effect of different thresholds to insert pixels .............................. 21
6. Pixel values of image used in section 3.5 ...................................... 22
7. Magnified portion of an alphabet explained ................................. 23
8. Result 1 ............................................................................ 25
9. Result 2 ............................................................................ 25
10. Result 3 ........................................................................... 26
11. Result 4 ........................................................................... 26
12. Result 5 ........................................................................... 27
13. Result 6 ........................................................................... 27
14. Result 7 ........................................................................... 27
15. Result 8a ........................................................................... 28
15. Result 8b ........................................................................... 28
ABSTRACT

This thesis presents a novel algorithm for improving quality of edges in printed text. The algorithm is designed to add pixels at selected edge locations after halftoning. The extent of the correction is proportional to the “strength” of the edge, as determined by comparing the local differences in a four-pixel neighborhood to a dynamically generated threshold. The process is computationally efficient and requires minimal memory resources. The performance of our proposed algorithm is clearly demonstrated on several characters and lines. While the algorithm aims to improve the quality of printed text (edges), it is possible to extend its application to improvement of any edge identifiable in an image document.
Chapter 1. INTRODUCTION

1.1 MOTIVATION

Text edge quality is a critical factor in judging print quality of images and documents. Depending on the printing technology and halftoning algorithm, some printers have problems in edge rendering. This is especially visible at the character edges. Some characters appear to be broken or choppy, or even some parts of characters (serifs) are completely missing. We developed an algorithm that improves the quality (appearance) of the printed edges without changing the original halftoning pattern(s).

This thesis presents a new algorithm for improving the quality of edges produced by electro-photographic (laser) printers. The extent of the correction is determined by comparing the local difference in a first-order neighborhood to a dynamically generated threshold. The algorithm is explained in detail and its performance is illustrated on a specific printer, namely Printer X.

1.2 CONTRIBUTIONS

Present technology uses character preprocessing (anti-aliasing) to smooth character edges prior to halftoning. Sometimes the character pixels are labeled as a “text” and not halftoned but rather matched to a library of existing printing patterns (e.g., horizontal line, left horizontal corner, etc.). Some of the algorithms used to improve text edges indiscriminately add edge pixels, thus significantly affecting the overall output quality.

We propose a post-processing algorithm that selectively adds toner (print pixels) at the edge of the processed tokens (e.g., characters) proportional to the local difference between processed
pixel location and its neighbors. A two dimensional pseudo-random number field is utilized to generate the threshold values. This method is an extension of a one dimensional hashing table indexing (also can be considered as a creation of the index in a two dimensional hash table). This is a novel way to generate threshold values. Additionally, threshold matrices used in prior work to render characters have an absolute reference point (i.e. zero), rather than a relative reference point as proposed here.

To summarize, the proposed algorithm is dynamic and computationally efficient and can be directly integrated into any print pipeline because of its post-processing nature. The algorithm has been chosen to be implemented in the upcoming 2007 series of Hewlett Packard color laser printers.

1.3 THESIS ORGANIZATION

The rest of the document is organized as follows: Chapter 2 presents the background and literature review. Chapter 3 describes the proposed algorithm. Results are presented in Chapter 4, and conclusions are drawn in Chapter 5.
CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

2.1 LITERATURE REVIEW

Various methods have been proposed for edge quality enhancement. One of the early methods to address the problem of edge quality enhancement uses anti-aliasing wherein two adjacent pixels in the oblique direction are detected and a corrective pixel is inserted to smooth the line [1]. Shirasaka [2] uses a similar technique wherein staircase regions are detected and notched. This is not a very effective approach for regions containing complex contours. Template matching is another widely used technique for print quality enhancement, wherein the image region is scanned in the piece-wise order (windows) and compared against a defined set of patterns to be rectified [3, 4, 5]. However, this approach requires predefined regions and conditions that can add to the processing time depending on the document complexity. Simpler yet effective techniques first convert the text to black and white and then reconvert it as described by Atkins [6]. Braica [7] states a method of enhancement by increasing the contrast at the edge.

Techniques that (slightly) manipulate image content for better quality are explained in [8] wherein the intensity of a pixel is chosen depending on the distance between the center of the pixel and the edge of the image. Ort [9] proposes a technique for shifting pixels by half a position while printing diagonal elements. A more robust algorithm was proposed by Sommer wherein the position of the pixel relative to the contour determines whether it needs to be shifted or not.

Text edge quality has been approached from other perspectives in the print industry. Shamir and Rappoport [10] approach the problem of text quality enhancement by examining the typographical attributes related to printing and propose a mathematical approach to bridge the
gap between displayed and printed text. This approach, however, is a system by itself, and practical realization into a pipeline would need enhancements and changes throughout the pipeline. Watanbe [11] introduces a technique for text smoothening by first extracting the contour of a dot pattern and performing interpolation and a corner rounding process to enhance the quality of the edge. The paper also presents a hardware apparatus that is geared towards performing these operations. This approach however is more favorable for display applications rather than printing (refer to Uphaus, et al. [12] for a simple two pixel font technique for enhancing text quality in display applications). Another algorithm for contour extraction of text data can be found in [13]. Haruki et al. [14] propose an algorithm for extracting joint points from the source image and reconstructing images (generating fonts) based on the extracted features. In a similar direction, Hu and Herch [15] propose a font description system that describes characters as assemblies of parameterizable shape components. Various combinations of the global parameters are utilized to obtain different variations of the character system. This approach is completely typographical in nature. Hussain and Zalik [16] approached modeling of font outlines using splines. This analysis is not always good for modeling fonts because of the nature of splines and their compatibility with peripheral display devices, since the former is continuous in nature and the latter is rather discrete. Miao et al. [17] and Zhu et al. [18] utilize the Gabor transformation [39] to identify font characters based on a guidance system. Miao’s work is geared mainly towards the Chinese typeset wherein character recognition is regarded as an important part during the rendering stage. This method, however, requires prior knowledge of certain parameters from font typesetting in order to form the guidance system. Similar work in font classification utilizing feature extraction and neural networks has been performed by Jung et al. [19] utilizing typographical attributes. Wada and Hagiwara [20] propose a system that not
only creates fonts but automatically adapts itself according to the user inputs based on a built-in evaluation module. However, Wada’s system is focused more towards Japanese typeset and requirements. Also, the evaluation system is based on fuzzy decisions that make the algorithm more iterative and time consuming.

It is also important for rendering devices to be backed by good hardware capabilities. Many approaches have been proposed for enhancing the printers’ abilities to produce higher quality output. *Multi-level printing* (MLP) is a technique utilized to improve image quality by introducing extra gray levels. This enables the printer to print more states of color in a unit area without changing the dot size. For example, instead of the printer having just having two states *on* and *off*, the printer is empowered, for example, with *off*, *mid-off*, *mid-on* and *on* states at each pixel location. Clearly, the printer’s performance will be better. Details of MLP are described in [21].

*Thermal inkjet edge smoothing* (TES) [22] is a method of smoothing jagged edges wherein the input image data is first scaled to a higher resolution and then the smoothing operation is performed. The end apparatus, however, requires a specialized hardware unit to scale images to a higher resolution, since such operations in software may lead to sub-optimum performance. Another approach in hardware modification of printers for better edge performance is varying the quantity of ink deposited on the printed medium. These algorithms alter the timing of firing the ink jets and are often referred to as “depletion” techniques [23, 24, 25, & 33].

Another approach for enhancing text quality is by the use of *tokens*, i.e., certain regions in an image are labeled (assigned *tokens*) and treated separately for better results. Yamada *et al.* [26] propose an algorithm similar to this idea for enhancing text quality by performing *multi-resolution halftoning*, wherein, black and colored image data are halftoned and rendered
separately to the print engine. Bacon [27] and Schatz [28] also use the token approach, wherein, bad pixels along the edge are assigned to certain gray levels (also called pels). This information from the pels is then interpreted within a certain neighborhood for taking a print/no-print decision.

To summarize, the scope of edge quality is well determined by the following two factors:

- Edge Detection
- Hardware capabilities of the rendering engine.

Edge detection has long been and continues to be an area of active research since its applications are manifold in areas such as segmentation, shape matching, etc. Similarly, hardware abilities have evolved greatly over the years. Finally, the choice comes down to choosing an algorithm or system that best suits the printer line, the halftoning technique, and other commercial aspects of the product line.

2.2 BACKGROUND

All halftoning dots are a collection of pixels for simulating a contone appearance.

- Henry Kang, Digital Color Halftoning

Any print, photographic, or display equipment can distinguish, represent, and reproduce only a certain set of colors, also called the color gamut of the system [38]. The choice of color range is generally application oriented and varies from vendor to vendor. If any color cannot be represented by the gamut, that color is said to be out of gamut. The most commonly used color space in printing industry is cyan, magenta, yellow, key [usually Black] (CMYK). The CMYK color space is a subtractive color model where the desired colors are obtained by subtracting/filtering various degrees of the primary colors. For example, the intensity of red in the CMYK color space is dictated by controlling/filtering the amount of cyan on the paper, since
cyan is a complement of red. In this way, all the combinations of colors in the gamut are generated. Another color space that is common is red, green, blue (RGB). RGB, unlike CMYK, is an additive color space where the desired color is obtained by combining/emitting various degrees of the primary colors. RGB color space is utilized mainly for display applications such as monitors, liquid crystal display (LCD) systems, etc.

Halftoning is a process that converts continuous tone (contone) images to an image with a fixed number of levels while simulating the visual quality of the contone (because the human eye performs the local spatial averaging). The simplest form of halftoning can be described by the threshold operation, where the output of at a pixel location \((i, j)\) is defined by:

\[
HT_{out}(i, j) = \begin{cases} 
1 & CT(i, j) \geq T_{screen}(i, j) \\
0 & CT(i, j) < T_{screen}(i, j)
\end{cases} \quad \text{....(1)}
\]

\(HT_{out}\) is the halftone output of the contone image \(CT\). \(T_{screen}\) is the threshold value at location \((i, j)\) on the half tone screen/ mask. This screen is a periodic replication of the halftone cell, a group of threshold values that are computed and arranged taking into consideration various factors such as the physics of light, the human visual system, etc. A good summary of the factors required to construct a good halftone cell can be found in [29, 30].

There are two main types of masks used in halftoning, dispersed masks and clustered masks. The former is more suitable for use in ink-jet printers and the latter in laser printers because of the printers’ ability/inability to print certain combination of dots.

In the case of color images, each plane is halftoned separately and placed on one another to produce a multi-plane (color) output. The method used to ‘pile’ the individual halftones over one another is called halftone dot placement. There are three types of dot placement techniques:

- **Dot on Dot**: All primary color dots overlap each other.
- **Dot off Dot**: Primary color dots are placed adjacent to each other without overlapping.
- *Rotated Dot*: Primary color screens are rotated by various angles during halftoning. There is partial overlapping.

*Dot off Dot* placement is mainly used in monitor displays. *Rotated Dot* placement is most widely used in printing systems, as it is less sensitive to printing defects such as mis-registration and banding.

The choice of halftoning is also determined by factors such as:

- Amplitude Modulation / Frequency Modulation.
- Bilevel / Multilevel Halftoning.
- Shape of the dots - *Elliptical, Round* or *Square*. Elliptical is the most commonly used shape.
- Size of screening cells – large cells mean more ink absorption (color quality), but less detail; whereas small cells mean more detail and less ink absorption.

A good description of the above-mentioned factors can be found in [31, 32, 34].

There are many halftoning techniques prevalent in the industry, and this science has evolved greatly over the years. Halftoning still continues to be an area of active research; a comprehensive study of various halftoning models and techniques can be found in [35].
Chapter 3. PROPOSED ALGORITHM

The experimental setup is presented in Figure 1. Figures 1a and 1b illustrate the original and modified printing pipeline employed in our experiments. In the original pipeline, the test target is first processed through a look-up table that converts it from its current color representation, such as RGB, to the printer CMYK domain. This continuous CMYK image is then halftoned and the resulting binary image is printed. The CMYK images before and after halftoning are referred to as the indump and outdump respectively. Indump ($I_{in}$) represents the continuous color image, and outdump ($I_{out}$) refers to the halftoned, uncorrected image. The modified pipeline shown in Figure 1b, on the other hand, mimics the original with one main exception; ($I_{out}$) is re-routed through our proposed algorithm prior to printing. In essence, our algorithm is designed to intercept the halftoned output, assess and improve the edge quality and then release the raster for printing, yielding a higher quality output. It is to be noted that $I_{in}$ has values ranging from 0-255 on all the respective planes, whereas $I_{out}$ has a restricted number of levels dictated by the halftone screen of the printing process.

![Diagram of experimental setup](image)

Figure 1: Experimental Setup of the printer. (a) Original print pipeline (b) Modified pipeline. * - Indump ($I_{in}$)  ** - Outdump ($I_{out}$)
3.1 PROPOSED ALGORITHM

During the process of halftoning, the number of gray levels in the indump ($I_{in}$) is reduced from 256 to a fixed number in the outdump ($I_{out}$) as dictated by the halftone screen of the printer. This generally results in edges being rendered with sub-optimal quality. To overcome this shortcoming, the proposed algorithm is designed to selectively insert pixels at edges in $I_{out}$ to enhance the perceived edge quality, without removing any of the original edge pixels. Hence, the existing halftone prior to the modification is not disturbed but augmented as needed to improve the quality of the output. To this effect, the algorithm first pre-processes $I_{out}$ in order to locate the relevant edges and determines the pixels/locations that are candidates for enhancement. The block diagram of the proposed algorithm is shown in the Figure 2 below. Details of the pre-processing module follow.

3.2 MODULE 1: PRE-PROCESSING TO RETAIN EXISTING PATTERN

At each location in $I_{out}$, the pixel value is compared to zero. If the result is $false$ (not zero), the algorithm proceeds to the next pixel in the raster order. However, if the result is $true$, the pixel in question is either a white space (non character) or was not rendered to the proper level during halftoning. This issue is resolved by comparing the gray level at the same location in $I_{in}$ with zero. Depending on the result of the comparison, the algorithm either proceeds to the edge enhancement (case $true$) module or skips it (case $false$). The notable feature is that this procedure is independent of the image raster rendering order. This module chooses the pixels to be enhanced by further modules in the algorithm and hence is a very significant contribution to the algorithm. Figure 2 on the adjoining page shows the block diagram of the algorithm.
3.3 MODULE 2: ADAPTIVE EDGE ENHANCEMENT

The next objective of our proposed algorithm is to introduce pixels that enhance the edge quality. However, a few practical issues arise during this phase. First, there is a restriction on the gray level of the corrective pixels that can be used (determined by the halftoning output, generally between two to four levels). This brings up the issue of retaining the visual quality of the alphabet after modification—the edge pixels should be inserted in such a manner that the end user should not notice any saturation shift.
After the pre-processing stage, it can only be deduced that an output pixel is missing. Thus, it is imperative to properly recognize the spatial location of the observed pixel: *out-of-character, on the edge*, or *within the character*. This is necessary since a pixel can lie *within* a given character and the existing halftoning algorithm may have decided not to render it, in which case it is not to be disturbed. The spatial localization of the pixel is calculated by utilizing the following procedure:

For a pixel at location \((i, j)\) with the intensity value \(I_{in}(i, j)\), the following terms are computed:

\[
\begin{align*}
(i) & \quad I_{in}(i, j) - I_{in}(i, j+1), \\
(ii) & \quad I_{in}(i, j) - I_{in}(i, j-1), \\
(iii) & \quad I_{in}(i, j) - I_{in}(i-1, j), \\
(iv) & \quad I_{in}(i, j) - I_{in}(i+1, j).
\end{align*}
\]

For ease of presentation, the terms \((i), (ii), (iii)\) and \((iv)\) are generalized to a single term ‘*edge difference*’ that refers to the difference between a pixel and its immediate neighbors (top, bottom, left, or right).

This *edge difference* provides pertinent information about \(I_{in}(i, j)\) with respect to its neighbors and its relative impact in the local neighborhood. If it is less than or equal to zero, the pixel either lies within the character or on the lighter side of the edge, in which case corrective pixels need not be added. If it is greater than zero, it can be ascertained that the pixel lies on the darker side of the edge, and, as a result, a corrective pixel has to be introduced to improve the edge quality. This condition is expressed as:

\[
\text{edge difference} > 0 \quad \text{....(2)}
\]

\[
\begin{array}{c|c|c}
X & 4 & X \\
\hline
2 & X & 1 \\
\hline
X & 3 & X \\
\end{array}
\]

Figure 3: Order of comparison of thresholds in the mask after computing the *edge difference*.
Once it has been decided that a corrective pixel needs to be introduced, the actions that follow must ensure that the overall visual quality of the character is not significantly altered post correction. Hence, pixels are inserted only on the darker side of corresponding edges. Also, corrective pixels are inserted quasi-randomly along the edge and not at all available edge locations. This objective is attained by a comparison of the edge difference and a non-negative dynamic threshold, \( T \) (detailed in Section 3.4). A pixel is inserted at a given location only when the edge difference is greater than \( T \). The condition described in Equation 2 is thus replaced (upgraded) by a stronger criterion:

\[
edge \ difference > T 
\]

\[
\ldots(3)
\]

To avoid unnecessary calculation of the threshold \( T \), both criteria (Equation 2 and Equation 3) are handled as separate steps in the algorithm, yielding a higher throughput. Hence, only when both conditions (2) and (3) are true, a corrective edge pixel is inserted in \( I_{out} \). Figure 3 above shows the order of the comparison of the edge difference value to a dynamically generated threshold in order to insert the corrective edge pixel. Dynamic threshold generation and corresponding principles involved are detailed in the upcoming section. The complete flowchart of the edge detection and enhancement algorithm is presented in Figure 4.

### 3.4 DYNAMIC THRESHOLD

A hash table, or a hash map, is a data structure that enables efficient data look up. It consists of an array of objects that are arranged as defined by the hashing function. The objective of the hashing function is to ensure that the probability of different but adjacent inputs (keys) leading to the same object (output) is close to zero. In order to ensure this, a function is defined to separate adjacent values by a large number, i.e., place values that are close to each other at farther ends of
the array limits. Hence, the hashing function is a very decisive part of the hash table performance.

At any pixel \( I_{in}(i, j) \)

\[
E_1 = I_{in}(i, j) - I_{in}(i, j+1)
\]

\[
E_2 = I_{in}(i, j) - I_{in}(i+1, j)
\]

\[
E_3 = I_{in}(i, j) - I_{in}(i, j-1)
\]

\[
E_4 = I_{in}(i, j) - I_{in}(i-1, j)
\]

Figure 4: Edge enhancement module

We propose a dynamic threshold generation function that is based on the above-mentioned principle. The objective of this function is to generate threshold values that are well separated from each other. The output values of the function are in the range \([0 - 255]\). This is done in order to retain the visual quality of the print target post-modification.

The dynamic threshold is a number that is calculated in real-time depending on the spatial location of the pixel. The main purpose of computing the dynamic threshold in Equation (4) is to ensure that only selective edge pixels are restored to retain the visual quality of the character as mentioned above. Figure 5(b) below shows the effect of using a constant threshold on a half-toned character “W” represented by Figure 5(a). Figure 5(c) shows the effect of using a
dynamically generated threshold on the same character in 5(a); it can be seen that output 5(c) is clearly better than 5(b) in terms of the enhanced edge quality.

The dynamic threshold $T$ is generated as follows:

$$T = (a \times i + b \times j) \mod (c) \quad \ldots (4)$$

where $i, j$ are the pixel locations in $I_{in}$ and $a, b, c$ are derived from mutually prime numbers. $T$ is a number between 0 and 255. It should be noted that any method that yields well-separated mutually prime numbers can be used for this purpose. In the experiments performed, the Tribonacci Series as defined by Equation (5) is used. The corresponding initial conditions for the series are defined in Equation (6):

$$Tr_n = Tr_{n-1} + Tr_{n-2} + Tr_{n-3} \quad (for \ n > 2) \quad \ldots (5)$$

$$Tr_0 = 0, Tr_1 = 1, Tr_2 = 1 \quad \ldots (6)$$

Any three consecutive terms derived from Equation (5) can be used as parameters $a, b, c$. However, such choice of numbers might not be optimal for efficient hardware implementation. Therefore, a normalization (followed by a rounding) factor of $\lambda$ is suggested.

Figure 5: Effect of using thresholds to insert corrective pixels along the edge. Part (a) shows the outdump $I_{out}$, part (b) shows the modified output using a constant threshold value and part (c) shows the corrected character using the proposed threshold.
3.5 DEMONSTRATION AT PIXEL LEVEL

An analysis of the region circled in Figure 1 is presented in Figure 6 and Figure 7. Figure 6 below shows the pixel values on the magenta plane during all the stages and Figure 7 shows these pixel values as an image. The gray level of the character is 136 on the magenta plane. Figure 7(a) is the corresponding indump $I_{in}$ after the character has been processed through the look-up table (see Figure 1 for details). Figure 7(b) is the corresponding outdump $I_{out}$, and Figure 7(c) is the modified image. It can be noted from Figure 7(c) that the algorithm only enhances the edge at certain specific points as discussed in Section 3.4 to avoid the visual effect of “edge thickening.” By doing so, the visual quality of the character is also preserved, resulting in a much higher quality of rendered edges.

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>160</th>
<th>152</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>160</td>
<td>152</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>164</td>
<td>156</td>
<td>160</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>164</td>
<td>160</td>
<td>168</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>160</td>
<td>152</td>
<td>164</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>160</td>
<td>156</td>
<td>156</td>
<td>168</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>164</td>
<td>160</td>
<td>168</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>170</td>
<td>170</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>85</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>77</th>
<th>215</th>
<th>99</th>
<th>238</th>
<th>121</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>152</td>
<td>35</td>
<td>175</td>
<td>58</td>
<td>197</td>
<td>80</td>
</tr>
<tr>
<td>228</td>
<td>112</td>
<td>250</td>
<td>134</td>
<td>17</td>
<td>156</td>
</tr>
<tr>
<td>48</td>
<td>187</td>
<td>70</td>
<td>209</td>
<td>92</td>
<td>232</td>
</tr>
<tr>
<td>123</td>
<td>6</td>
<td>146</td>
<td>29</td>
<td>168</td>
<td>51</td>
</tr>
<tr>
<td>199</td>
<td>82</td>
<td>221</td>
<td>105</td>
<td>244</td>
<td>127</td>
</tr>
<tr>
<td>19</td>
<td>158</td>
<td>41</td>
<td>180</td>
<td>63</td>
<td>202</td>
</tr>
</tbody>
</table>

(c)

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>85</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>85</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>170</td>
<td>170</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>85</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>85</td>
<td>170</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>85</td>
<td>0</td>
<td>0</td>
<td>85</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>85</td>
<td>0</td>
<td>170</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(d)

Figure 6: (a) – Indump $I_{in}$, (b) Outdump $I_{out}$, (c) Threshold Table $T$, (d) Modified Output. Red - pixels omitted by halftoning, Green - pixels inserted by proposed algorithm, Blue – pixels that are not modified by the algorithm.
Figure 7: Magnified portion of rightmost bar of alphabet ‘W’ used in Figure 1.

Part (a) shows the indump $I_{in}$, part (b) shows the outdump $I_{out}$, and part (c) shows the corrected character using the proposed algorithm.
Chapter 4. RESULTS & DISCUSSION

The performance of the algorithm was tested on all four color planes at various gray levels and for two different fonts, namely Times New Roman (TNR) and Garamond. Garamond was chosen because the character set is comprised of relatively thinner strokes. The test target was designed to encompass several characters with various sizes that possess all possible gray levels that the printer might encounter during run time.

The first set of test targets was developed using TNR of sizes 10-12 pt., which is the most widely used (and affected) size. The performance of the algorithm on this set can be seen in Figure 8 and Figure 9. As seen in Figure 8, the algorithm gives better edge quality on the rightmost and the center bars of the letter. Similar edge improvements are also observed (see Figure 9) on the more subtle “tilde” and “serif” aspects of characters. This is well demonstrated in the magenta plane where the serif is not well rendered (see Figure 9 – second column).

The next set of test targets was designed by utilizing the Garamond font with various point sizes as well. The performance of our algorithm on this set is shown in Figure 10 and Figure 11. For this font, the cyan plane was more affected than the other planes. To this effect, the algorithm inserts more pixels in the cyan channel than in the other channels during edge enhancement. Note that the observed differences in the original halftoning are a result of the use of rotated screens for the various channels.

The algorithm was also tested on thin lines inclined in steps of five degrees. As can be seen from Figure 12 and Figure 13, the algorithm effectively restores the lost details of the lines. The effect is particularly noticeable when the line has the same inclination as the screen that is used for the corresponding color during halftoning.
The final set of results was obtained by placing text on a colored background rather than a white background as was the previous cases. As can be seen in Figure 14, the algorithm inserts fewer pixels than it used to when the character was on a white background. One last variation was to tilt the character itself and place it against a background. The performance of the algorithm on such a character can be seen in Figure 15, wherein the algorithm retains edge details that were lost during the halftoning stage, particularly when the rotation was close to the halftone screen angles.

Figure 8: From left to right: $I_{in}$, $I_{outs}$, corrected character. Font-face: Times New Roman, Font-size: 10. Gray level: 36 on respective plane(s).

Figure 9: From left to right: $I_{in}$, $I_{outs}$, corrected character. Font-face: Times New Roman, Font-size: 12. Gray level: 119 on the respective plane(s).
Figure 10: From left to right: $I_{in}$, $I_{out}$, corrected character. Font-face: Garamond, Font-size: 12.
Gray level: 85 on the respective plane(s).

Figure 11: From left to right: $I_{in}$, $I_{out}$, corrected character. Font-face: Garamond,
Font-size: 12. Gray level: 34 on the respective plane(s).
Figure 12: Cyan lines at various angles (steps of 5 degrees). $I_{\text{in}}$, $I_{\text{out}}$, corrected character.

Figure 13: Magenta lines at various angles (steps of 5 degrees). $I_{\text{in}}$, $I_{\text{out}}$, corrected character.

Figure 14: Cyan text Against a uniformly varying gradient. Font: Times New Roman $I_{\text{in}}$, $I_{\text{out}}$, corrected character.
Figure 15: Magenta text (rotated 30 degrees) against a uniformly varying gradient.

Font: Times New Roman, Left to right: $I_{\text{in}}$, $I_{\text{out}}$, corrected character.
Chapter 5. CONCLUSIONS & FUTURE WORK

The proposed algorithm is very effective in improving the quality of the edges. Its computational efficiency allows for either hardware or software implementations without disturbing the existing document workflow (i.e., internal printer pipeline). Another significant feature is its self-adaptive correction, i.e., the number of inserted pixels is proportional to the strength of the edge. The algorithm treats light-colored and dark-colored characters differently. In highlights, the algorithm offers alternative renderings that can even recover some of the lost edge details. Use of the dynamic thresholding operation for rendering eliminates the need for user supervision and extends its applicability.

The following are some directly possible enhancements to the proposed algorithm:

1. Adaptively varying corrective pixel value

Presently, as per the described algorithm, only a fixed gray level ($I_{\text{fix}}$) is used in places that require enhancement. However, with the addition of some additional conditional logic, this value can be chosen dynamically as well. The criteria here would be the value of the $I_{\text{in}}$, as expressed the following equations:

\[
I_{\text{fix}} = \begin{cases} 
I_{\text{fix}1} & 0 \leq I_{\text{in}} < T_1 \\
I_{\text{fix}2} & T_1 \leq I_{\text{in}} < T_2 \\
I_{\text{fix}3} & T_2 \leq I_{\text{in}} \leq 255
\end{cases}
\] 

...(7)

where $T_1$ and $T_2$ are pre-selected thresholds and $I_{\text{fix}1} < I_{\text{fix}2} < I_{\text{fix}3}$.

2. Varying the threshold range generated.

Equation 4 presently generates values in the range [0-255]. The number of pixels that are inserted can be increased by introducing a threshold scaling factor ($T_x$) defined as:
where S.F. is the threshold scaling factor. This is a simple way of changing the range of threshold values generated depending on the extent of edge enhancement desired.

3. Enhancing clustering based on local region neighborhood.

As mentioned above, clustering can be introduced in order to increase the number of pixels that are subjected to the same threshold. In such a situation, the corrective pixels will be placed more frequently, thus inserting more pixels. It is also possible to tailor the number of pixels in a cluster by varying certain parameters of Equation 4; thus darker characters (greater gray values) will be subjected to more correction over characters that are lighter, which is preferred.
REFERENCES


[38] General information lookup on: www.wikipedia.org