A NEW ADAPTIVE EDGE ENHANCEMENT ALGORITHM FOR COLOR LASER PRINTERS

Manoj Kumar Masthan Reddy[(1)], *Vladimir Misic*[(1, 2)], *Eli Saber*[(1)], *Jeff Trask*[(3)]

1 - Department of Electrical Engineering, Rochester Institute of Technology, Rochester, NY.
2 - Radiation Oncology, Thomas Jefferson University, Philadelphia, PA.
3 - Hewlett Packard Company, Boise, ID.

ABSTRACT

This paper presents a novel algorithm for improving quality of edges in printed text. The algorithm is designed to add pixels at selected edge locations post 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 image document.

Index Terms- Text Edge Quality, Edge Rendering.

1. INTRODUCTION

Text edge quality is a critical factor in judging print quality of images and documents. Various methods have been proposed for edge quality enhancement. One of the early methods to address this problem uses anti-aliasing wherein two adjacent pels in the oblique direction are detected and a corrective pixel is inserted to smooth the line [1]. This technique is not very effective for regions containing complex contours. Template matching is another widely used technique for print quality enhancement [2, 3]. However, this approach requires predefined regions/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 [4]. Braica [5] states a method of enhancement by increasing the contrast at the edge.

This paper 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 paper presents a general overview of the algorithm and illustrates its performance on a specific printer, namely Printer X.

The rest of the paper is organized as follows. Section 2 presents the experimental setup (the pipeline) used. Section 3 describes the dynamic threshold generation algorithm. Results are presented in Section 4 and conclusions are drawn in Section 5.

2. EXPERIMENTAL SETUP

The experimental setup is presented in Fig. 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* represents the continuous color image, and *outdump* refers to the halftoned, uncorrected image. The modified pipeline, on the other hand, mimics the original (see Fig. 1b) with one main exception: the *outdump* 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.

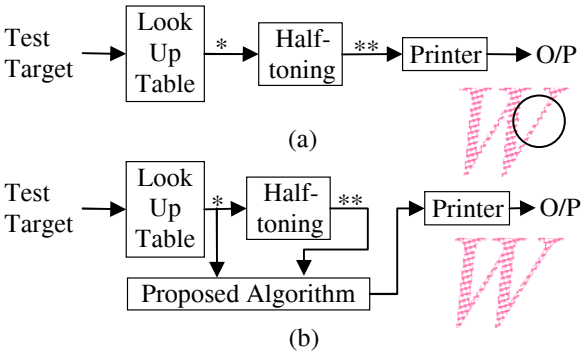

Figure 1: Experimental Setup of the printer. (a) Original print pipeline (b) Modified print pipeline.
* - Indump (I_{in}) ** - Outdump (I_{out})

3. 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, our 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 details of the pre-processing stage are presented below.

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 input 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)* or skips it (case *false*). A detailed flowchart of the algorithm is shown in Figure 2.

3.1 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 2-4 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 belong to *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:

(i) $I_{in}(i, j) - I_{in}(i, j+1)$, *(iii)* $I_{in}(i, j) - I_{in}(i-1, j)$,
(ii) $I_{in}(i, j) - I_{in}(i, j-1)$, *(iv)* $I_{in}(i+1, j) - I_{in}(i, j-1)$.

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).

The *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:

$$edge\ difference > 0 \quad (1)$$

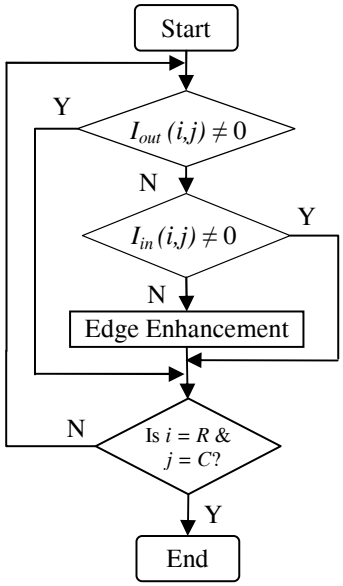

Figure 2: Flowchart of the algorithm at any pixel *(i,j). R* and *C* are the number of rows and columns in I_{out} respectively.

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 3.2). A pixel is inserted at a given location only when the *edge difference* is greater than *T*. The condition described in Eq. 1 is thus replaced (upgraded) by a stronger criterion:

$$edge\ difference > T \quad (2)$$

To avoid unnecessary calculation of the threshold *T*, both criteria (Eq. 1 and Eq. 2) are handled as separate steps in the algorithm, yielding a higher throughput. Hence, only when both conditions (1) and (2) are true, a corrective edge

pixel is inserted in I_{out}. The complete flowchart is presented in Fig. 3. The procedure utilized to compute the dynamic threshold is detailed below.

3.2 Dynamic Threshold

The dynamic threshold is a number that is calculated real-time depending on the spatial location of the pixel. The main purpose of computing the dynamic threshold in Eq. (2) is to ensure that only selective edge pixels are restored to retain the visual quality of the character as mentioned above. This threshold is generated as follows:

$$T = (a * i + b * j) \bmod (c) \tag{3}$$

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 c-1 (ideally, c = 256). 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 Eq. (4) was used to derive the parameters a, b and c. The corresponding initial conditions for the series are defined in Eq. (5):

$$Tr_n = Tr_{n-1} + Tr_{n-2} + Tr_{n-3} \ (for\ n>2) \tag{4}$$
$$Tr_0 = 0,\ Tr_1 = 1,\ Tr_2 = 1 \tag{5}$$

Any three consecutive terms derived from Eq. (4) can be used as parameters a, b and c. However, such choice of numbers might not be optimal for efficient hardware implementation. Therefore, a normalization (followed by a rounding) of $a, b,$ and c by the factor of $\lambda = 256/c$ is suggested.

An analysis of the region circled in Fig. 1 is presented in Fig. 4. The gray level of the character was 136 on the magenta plane. Fig. 4(a) is the corresponding indump, I_{in} after the character has been processed through the look up table (see Fig. 1 for details). Fig. 4(b) is the corresponding outdump, I_{out} and Fig. 4(c) is the modified image. It can be noted from Fig. 4(c) that the algorithm only enhances the edge at certain specific points as discussed in Section 3.1 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. The performance of the algorithm on the complete character can be seen in Fig. 5.

4. RESULTS AND DISCUSSIONS

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 Fig. 5 and Fig. 6. As seen in Fig. 5, the algorithm gives better edge quality on the rightmost and the center bars of the letter. Similar edge improvements are also observed (see Fig. 6) 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 Fig. 6b).

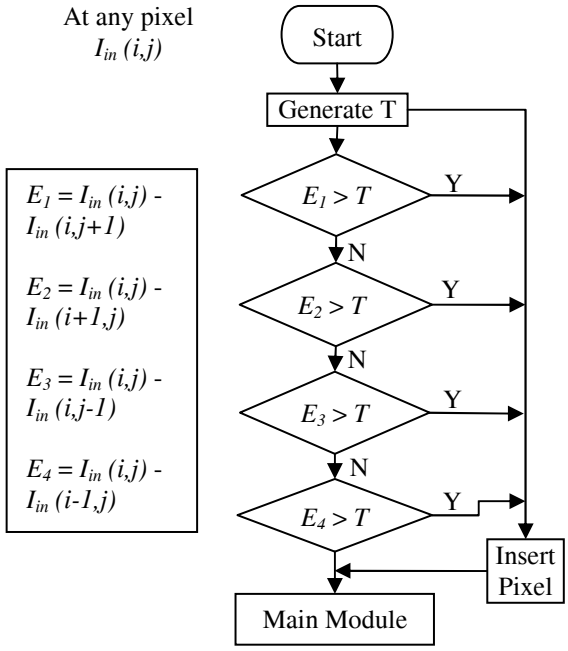

Figure 3: *Edge Enhancement* module.

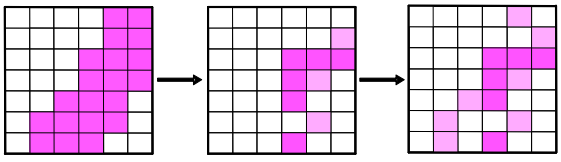

Figure 4: Magnified portion of rightmost bar of alphabet 'W' used in Fig.1. Left to right: I_{in}, I_{out}, *corrected character.*

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 Fig. 7. For this font, the cyan plane was more affected than the other planes as shown in Fig. 7. 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 5 degrees. As can be seen from Fig. 8, 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.

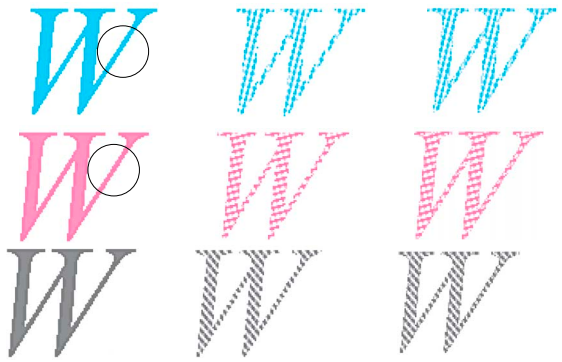

Figure 5: From left to right: I_{in}, I_{out}, *corrected character.* Font-face: Times New Roman, Font-size: 10. Gray level: 136 on the respective plane(s).

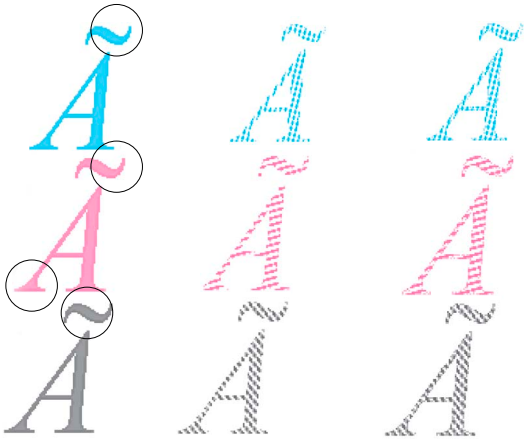

Figure 6: From left to right: I_{in}, I_{out}, *corrected character.* Font-face: Times New Roman, Font-size: 12. Gray level: 119 on the respective plane(s).

5. CONCLUSIONS

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.

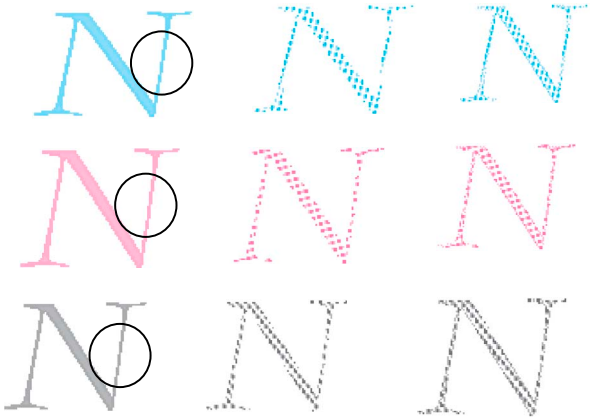

Figure 7: From left to right: I_{in}, I_{out}, *corrected character.* Font-face: Garamond, Font-size: 12. Gray level: 85 on the respective plane(s).

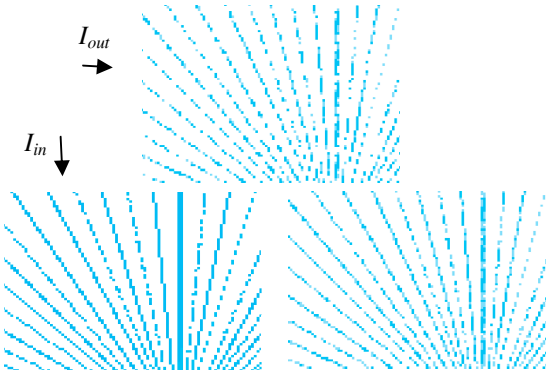

Figure 8: Cyan lines at various angles (steps of 5 degrees). I_{in}, I_{out}, *corrected character.*

ACKNOWLEDGEMENTS

This work was funded by the Laser Printing Division of Hewlett Packard Company. The authors wish to thank Mr. Guru Prashanth Balasubramanian for his review comments.

6. REFERENCES

[1] S.Yonezawa, T. Kawakami, T. Shimada, Y. Chida, "Apparatus for forming a character out of a pattern of separate display picture elements" *US Patent No. 4079367, Mar. 1978.*

[2] M. Yao, M. T. Stevens, M. R. Parker, "Text and image quality enhancement" *US Patent No. 6987588, Jan. 2006.*

[3] M. D. Lund, "Pixel image edge-smoothing method and system" *US Patent No. 5650858, July 1997.*

[4] B. A. Clayton, "System and method for scaling and enhancing color text images" *US Patent No. 7046390, May 2006.*

[5] P. Braica, "Edge detection and sharpening process for an image" *US Patent No. 7068852, June 2006.*

[6] V. Misic, P. Anderson, "Algebraic Masks for color halftoning" *Proc. of SPIE, Vol. 566, pp. 441-448, 2005.*