Estimating toner usage with laser electrophotographic printers, and object map generation from raster input image

Lu Wang
Purdue University

Follow this and additional works at: http://docs.lib.purdue.edu/open_access_dissertations
Part of the Electrical and Electronics Commons

Recommended Citation
http://docs.lib.purdue.edu/open_access_dissertations/384

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Lu Wang

Entitled
Estimating Toner Usage with Laser Electrophotographic Printers, and Object Map Generation From Raster Input

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

JAN P. ALLEBACH

MARY L. COMER

EDWARD J. DELP

MARY L. COMER

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University’s “Policy on Integrity in Research” and the use of copyrighted material.

JAN P. ALLEBACH

Approved by Major Professor(s): ________________________________

______________________________

Approved by: M. R. Melloch ________________________________ 12/08/2014

Head of the Department Graduate Program Date
ESTIMATING TONER USAGE WITH LASER ELECTROPHOTOGRAPHIC PRINTERS, AND OBJECT MAP GENERATION FROM RASTER INPUT IMAGE

A Dissertation
Submitted to the Faculty
of
Purdue University
by
Lu Wang

In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

December 2014
Purdue University
West Lafayette, Indiana
ACKNOWLEDGMENTS

This thesis is the result of many experiences I have encountered at Purdue from dozens of remarkable individuals who I wish to acknowledge.

First and foremost, I would like to gratefully and sincerely thank my major advisor, Prof. Jan P. Allebach, who has been a tremendous mentor for me. I would like to thank you for encouraging my research, giving me the moral support, and providing the research assistantship during my PhD study. For everything you have done for me, Prof. Allebach, I thank you. I would also like to thank my committee members, Dr. Dennis A. Abramsohn, Prof. Mireille Boutin, Prof. Mary L. Comer, Prof. Edward J. Delp, and Dr. Thomas W. Ives for their advice and careful guidance.

I would like to express my highest gratitude to my dad, mom, and husband, Chen. Their support, patience, and unwavering love are undeniably the bedrock where my life has been built. I thank them for their faith in me, and I appreciate that I always have my family to count on when times are rough.

I would also like to thank my EISL colleagues for their support. I praise their enormous amount of help on my research and study throughout the years. More importantly, they shared the delightful and hard times with me at Purdue, providing unending support and friendship that I needed.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>SYMBOLS</td>
<td>vi</td>
</tr>
<tr>
<td>ABBREVIATIONS</td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>xii</td>
</tr>
<tr>
<td>ESTIMATING TONER USAGE WITH LASER ELECTROPHOTOGRAPHIC PRINTERS</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Laser electrophotographic printer architecture</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Implementation of the two-stage predictor</td>
<td>4</td>
</tr>
<tr>
<td>1.3.1 Implementation of the first stage: absorptance predictor</td>
<td>5</td>
</tr>
<tr>
<td>1.3.2 Implementation of the second stage: toner usage mapping</td>
<td>15</td>
</tr>
<tr>
<td>1.4 Construction of the two-stage predictor</td>
<td>16</td>
</tr>
<tr>
<td>1.4.1 Construction of the first stage: absorptance predictor</td>
<td>16</td>
</tr>
<tr>
<td>1.4.2 Construction of the second stage: toner usage mapping</td>
<td>26</td>
</tr>
<tr>
<td>1.5 Evaluation of the two-stage predictor</td>
<td>31</td>
</tr>
<tr>
<td>1.5.1 The 3 × 3 absorptance predictor and the weighted hamming distance predictor</td>
<td>31</td>
</tr>
<tr>
<td>1.5.2 Evaluation of the first stage: predicted scanned images</td>
<td>32</td>
</tr>
<tr>
<td>1.5.3 Evaluation of the second stage: Cross-validation experiment</td>
<td>36</td>
</tr>
<tr>
<td>1.5.4 Comparisons between the two-stage predictor and current toner usage prediction approach</td>
<td>37</td>
</tr>
<tr>
<td>1.6 Conclusion</td>
<td>38</td>
</tr>
<tr>
<td>OBJECT MAP GENERATION FROM RASTER INPUT IMAGE</td>
<td>40</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>40</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.2  The New Imaging Pipeline with OMGA</td>
<td>46</td>
</tr>
<tr>
<td>2.3  Object Map Generation Algorithm</td>
<td>48</td>
</tr>
<tr>
<td>2.3.1 Detect strong edges</td>
<td>53</td>
</tr>
<tr>
<td>2.3.2 Use connected component (CC) to find and label CSECs and CSEICs</td>
<td>55</td>
</tr>
<tr>
<td>2.3.3 Classify CSECs (symbol or non-symbol)</td>
<td>56</td>
</tr>
<tr>
<td>2.3.4 Classify CSEICs (symbol or non-symbol)</td>
<td>59</td>
</tr>
<tr>
<td>2.3.5 Edge linking</td>
<td>60</td>
</tr>
<tr>
<td>2.3.6 Use CC to find and label CSWEIC</td>
<td>66</td>
</tr>
<tr>
<td>2.3.7 Classify CSWEICs (vector or non-vector)</td>
<td>66</td>
</tr>
<tr>
<td>2.3.8 Integrate classifications</td>
<td>68</td>
</tr>
<tr>
<td>2.3.9 Erode around raster and symbol</td>
<td>68</td>
</tr>
<tr>
<td>2.4  Hardware implementation</td>
<td>71</td>
</tr>
<tr>
<td>2.5  Experimental Results</td>
<td>72</td>
</tr>
<tr>
<td>2.6  Conclusion</td>
<td>76</td>
</tr>
<tr>
<td>3    CONTRIBUTIONS OF THIS DISSERTATION</td>
<td>78</td>
</tr>
<tr>
<td>3.1  Contribution of Part 1</td>
<td>78</td>
</tr>
<tr>
<td>3.2  Contribution of Part 2</td>
<td>79</td>
</tr>
<tr>
<td>LIST OF REFERENCES</td>
<td>80</td>
</tr>
<tr>
<td>VITA</td>
<td>86</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>3×3 absorptance predictor look-up table.</td>
</tr>
<tr>
<td>1.2</td>
<td>The algorithm for weighted hamming distance predictor.</td>
</tr>
<tr>
<td>1.3</td>
<td>Absorptance ranges and their corresponding toner usage.</td>
</tr>
<tr>
<td>1.4</td>
<td>The printed absorptance value recorded for each configuration array.</td>
</tr>
<tr>
<td>1.5</td>
<td>The algorithm for searching the parameters of the weight hamming distance predictor.</td>
</tr>
<tr>
<td>1.6</td>
<td>Original absorptance ranges selection and their variance inflation factors (VIFs).</td>
</tr>
<tr>
<td>1.7</td>
<td>The 3×3 absorptance predictor and the weighted hamming distance predictor.</td>
</tr>
<tr>
<td>1.8</td>
<td>Average weights with standard deviations derived from the cross validation for the toner usage mapping.</td>
</tr>
<tr>
<td>1.9</td>
<td>Comparison between our new two-stage predictor and the pixel-counting algorithm.</td>
</tr>
<tr>
<td>2.1</td>
<td>Object types and their corresponding color codes in the object map, halftone screens, and color maps.</td>
</tr>
<tr>
<td>2.2</td>
<td>List of parameters used in OMGA.</td>
</tr>
<tr>
<td>2.3</td>
<td>Classification performance of the OMGA when applied to the image library.</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure | Page
--- | ---
1.1 Overview of the two-stage toner usage predictor. | 3
1.2 Architecture of a typical electrophotographic printer [38]. | 4
1.3 Illustration of the absorptance predictor. (a) A digital continuous-tone monochrome image example of the letter “C” with gradient color. (b) PWM file generated by (a). (c) Quantized PWM image. (d) The 5×5 predictor window with pixel indices indicated. The pixel in blue is the one under estimation. (e) 5×5 absorptance predictor LUT. (f) Predicted scanned image of letter “C”. (g) Actual printed and then scanned image of letter “C”. | 7
1.4 Histogram of absorptance values in the 5×5 LUT. | 10
1.5 Illustration of the weight with respect to each pixel’s position in the 5×5 window in calculating the weighted hamming distance between two configurations. The hamming distance between the two example configurations is 18. | 11
1.6 Procedure for developing the 5×5 absorptance predictor. | 18
1.7 The five training pages for the 5×5 LUT. (a) Training page 1: letters in Arial with 16 point size. (b) Training page 2: letters in Times New Roman with 12 point size. (c) Training page 3: mixed lines. (d) Training page 4: Complex. (e) Training page 5: Bridge. | 19
1.8 A clipped portion of training page 4, ”Complex”. Illustrate the fiducial marks and the analysis area (blue area), where pixel’s printed absorptance is collected for training purpose. | 21
1.9 Two examples from the 35 test pages and their corresponding histograms of the predicted scanned images: (a) the test page with 12 point Arial random text with 1.15 line spacing, and (b) DNA. | 27
1.10 Predicted scanned page and the real scanned page of Toy Store . | 34
1.11 Zoom-in picture of predicted scanned page and the real scanned page of Toy Store . | 35
2.1 Illustration of the object type definition. (a) Example page content. (b) The corresponding object map (red: raster; blue: symbol; green: vector). | 42
<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>43</td>
</tr>
<tr>
<td>2.3</td>
<td>45</td>
</tr>
<tr>
<td>2.4</td>
<td>47</td>
</tr>
<tr>
<td>2.5</td>
<td>48</td>
</tr>
<tr>
<td>2.6</td>
<td>49</td>
</tr>
<tr>
<td>2.7</td>
<td>50</td>
</tr>
<tr>
<td>2.8</td>
<td>51</td>
</tr>
<tr>
<td>2.9</td>
<td>54</td>
</tr>
<tr>
<td>2.10</td>
<td>57</td>
</tr>
<tr>
<td>2.11</td>
<td>58</td>
</tr>
<tr>
<td>2.12</td>
<td>61</td>
</tr>
<tr>
<td>2.13</td>
<td>62</td>
</tr>
<tr>
<td>2.14</td>
<td>63</td>
</tr>
<tr>
<td>2.15</td>
<td>64</td>
</tr>
<tr>
<td>2.16</td>
<td>65</td>
</tr>
<tr>
<td>2.17</td>
<td>67</td>
</tr>
<tr>
<td>2.18</td>
<td>69</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>2.19 Illustration of the ice crystal and branches shapes and the smoothed object map after erosion.</td>
<td>70</td>
</tr>
<tr>
<td>2.20 Object map generated by OMGA for Page 2 in the image library.</td>
<td>73</td>
</tr>
<tr>
<td>2.21 Object map generated by OMGA for Page 57 in the image library.</td>
<td>74</td>
</tr>
<tr>
<td>2.22 Object map generated by OMGA for Page 63 in the image library.</td>
<td>75</td>
</tr>
</tbody>
</table>
SYMBOLS

$q_i$  
(pixel value in the quantized image

$a^{\text{predict}}$  
(predicted pixel absorptance value

$w_j^{3\times3}$  
(weight, the contribution of a pixel with quantized level $j$ to the center pixel in the $3 \times 3$ window

$w_{\text{center}}^{\text{ring}}$  
(weight for the center pixel in the $5\times5$ window used in the weighted hamming distance predictor

$w_{\text{middle}}^{\text{ring}}$  
(weight for the middle ring pixels in the $5 \times 5$ window used in the weighted hamming distance predictor

$w_{\text{outer}}^{\text{ring}}$  
(weight for the outer ring pixels in the $5 \times 5$ window used in the weighted hamming distance predictor

$d_R$  
(threshold used to define similar configurations in the weighted hamming distance predictor

$w_t^{\text{tu}}$  
(weights for the second stage, toner mass mapping

$a^{\text{scan}}[m,n]$  
(\text{absorptance value of the scanned image at the pixel with coordinates [m,n]

$d$  
(weighted hamming distance between two configurations

$Re\text{Error}$  
(relative error between the prediction and the average measured value

\text{std. dev.}$  
(standard deviation

$m_{\text{experiment}}$  
(measured toner usage from splitting and weighing experiment

$T_{S,\text{edge}}$  
(threshold to detect strong edges

$T_{W,\text{edge}}$  
(threshold for weak edge points in edge linking

$T_{\text{hwl}}$  
(lower boundary for \text{height} and \text{width} of symbol CSEC

$T_{\text{hwu}}$  
(upper boundary for \text{height} and \text{width} of symbol CSEC
$T_{hwrl}$ lower boundary for \textit{height/width} of symbol CSEC

$T_{hwru}$ upper boundary for \textit{height/width} of symbol CSEC

$T_{N_{CSEC}}$ upper boundary for the number of pixels of symbol CSEC

$T_{EM_{CSEC}}$ lower boundary for the average value of edge magnitude of symbol CSEC

$T_{L,area}$ threshold to determine large or small area

$T_{VR}$ threshold for $r_{\text{rough},CSEIC}$ to differentiate between vector and non-vector

$T_{SR}$ threshold for $r_{\text{rough},CSEIC}$ to differentiate between symbol and non-symbol

$G_{x,i}$ horizontal derivative approximation of the input RGB image for channel $i$, $i = R, G, B$

$G_{y,i}$ vertical derivative approximation of the input RGB image for channel $i$, $i = R, G, B$

$EM$ edge magnitude value

$N_4(p)$ 4-neighbors connectivity

$N_8(p)$ 8-neighbors connectivity

$\land$ logical conjunction operator

$\lor$ logical disjunction operator

\textit{height} number of rows between the rst and last pixel of the CSEC in vertical direction

\textit{width} number of columns between the most left and the most right pixel of the CSEC in horizontal direction

$N_{CSEC}$ number of pixel of the CSEC

$EM_{\text{avg}}$ average value of the edge magnitude of a CSEC

$\mathbb{1}(\cdot)$ indicator function

$r_{\text{rough},CSEIC}$ roughness ratio of a CSEIC

$N_{CSEIC}$ number of pixels of a CSEIC

$N_{image}$ number of total pixels in the input image
$r_{\text{rough,CSWEIC}}$ roughness ratio of a CSWEIC

$N_{\text{CSWEIC}}$ number of pixels of a CSWEIC
ABBREVIATIONS

EP       ElectroPhotographic
OPC      Organic Photo Conductor
PWM      pulse width modulation
LUT      Look-Up Table
5×5 LUT  5×5 absorptance predictor LUT
WHD      weighted hamming distance
LASSO    least absolute shrinkage and selection operator
VIF      variance inflation factor
RMSE     root-mean-square error
PN       process neutral
KN       K-only process
PDL      page description language
PCL      Printer Command Language
PDF      Portable Document Format
RIP       raster image processor
JPEG     Joint Photographic Experts Group
URF      Unicode Transformation Format
OCR      optical character recognition
OMGA     object map generation algorithm
CSEC     Connected strong edge components
CSEIC    connected strong edge interior components
CSWEIC   connected (strong+weak) edge interior components
SE       strong edge map
CC       connected component
CCA connected component analysis
ABSTRACT


Accurate estimation of toner usage is an area of on-going importance for laser, electrophotographic (EP) printers. In Part 1, we propose a new two-stage approach in which we first predict on a pixel-by-pixel basis, the absorptance from printed and scanned pages. We then form a weighted sum of these pixel values to predict overall toner usage on the printed page. The weights are chosen by least-squares regression to toner usage measured with a set of printed test pages. Our two-stage predictor significantly outperforms existing methods that are based on a simple pixel counting strategy in terms of both accuracy and robustness of the predictions.

In Part 2, we describe a raster-input-based object map generation algorithm (OMGA) for laser, electrophotographic (EP) printers. The object map is utilized in the object-oriented halftoning approach, where different halftone screens and color maps are applied to different types of objects on the page in order to improve the overall printing quality. The OMGA generates object map from the raster input directly. It solves problems such as the object map obtained from the page description language (PDL) is incorrect, and an initial object map is unavailable from the processing pipeline. A new imaging pipeline for the laser EP printer incorporating both the OMGA and the object-oriented halftoning approach is proposed. The OMGA is a segmentation-based classification approach. It first detects objects according to the edge information, and then classifies the objects by analyzing the feature values extracted from the contour and the interior of each object. The OMGA is designed to
be hardware-friendly, and can be implemented within two passes through the input document.
1. ESTIMATING TONER USAGE WITH LASER ELECTROPHOTOGRAPHIC PRINTERS

1.1 Introduction

When using a desktop/workgroup printer, unexpected low levels of toner or unacceptable print defects due to inaccurate prediction of end of life behavior in the midst of a single long print job or series of print jobs, can be undesirable, especially if the user has no replacement consumables on hand. A more predictive end-of-life indicator can help the user anticipate this situation, thereby improving the overall customer experience. Thus, there is a great need for a means to accurately estimate the usage of consumables when printing pages with a wide range of content.

With laser electrophotographic printing, the task of estimating toner usage is made particularly challenging by the influence of the state of neighboring pixels in the digital page on the amount of toner that is printed at a given printer-addressable pixel on the output page [1] [2]. This influence is due to the overlap in the laser spot profile from pixel-to-pixel [3], the complex field effects that govern transfer of toner from the developer roller to the Organic Photo Conductor (OPC) drum [4], and further spreading of toner during transfer and fusing to the media [5].

The net result is that accurate estimation of toner usage must take into account not only the aggregate gray value in the digital image to be printed, but also the spatial arrangement of the pixels over which this aggregate gray value is distributed. While existing pixel-counting approaches attempt to do this in a limited way, these methods are too simplistic to yield the level of accuracy that is desired. For example, U.S. Pat. No. 5,349,377 [6] estimates the consumption of toner by analyzing the frequency rate of 1’s and 0’s in the halftone image, and calculating weighting factors for different types of images. U.S. Pat. No. 6,810,218 [7] discloses a method which
calculates a ratio of transition count/pixel count using a pixel transition count that can account for disparate toner mass consumption caused by different types of images such as text/line, half tone and solid area. U.S. Pat. No. 6,356,359 [8] implements the estimation based on a reduced resolution bit map, which is typically faster than a process that counts each and every pixel within the datastream of a full-resolution image. U.S. Pat. No. 7,720,397 [9] discloses a method which accounts for the “diffusion effect” caused by non-adjacent pixel groups. The system determines a proximity factor that is indicative of the number of independent pixel groups within each set of eight adjacent pixels, and then calculates toner usage based on the proximity factor and a pixel count for the page. However, these approaches are not sufficiently comprehensive to account for the complicated inter-pixel interaction.

We propose a new two-stage approach to the estimation of toner usage that yields much more accurate results. As shown in Figure 1.1, the input to the proposed predictor is the halftone image with pulse width modulation (PWM) [10]. PWM technology provides subpixel addressability in the laser scan direction, so that the laser beam may be turned on or off at fractional positions within a pixel. We use halftone image instead of continuous-tone image as the input to our two-stage predictor, because the PWM values directly instruct where and how much the toner should be. Before the first stage, the PWM image is derived from the digital continuous-tone image, using corresponding halftone algorithm. In the first stage, we predict the measured absorptance on the scanned page based on the PWM file in a local window. This model is inspired by work on modeling printed halftones in liquid, electrophotographic printing [11] [12]. In the second stage, we assign a weighting to the predicted absorptance value at each pixel, and accumulate these weights to predict the overall toner usage on the printed page. The weights are chosen by least-squares regression on the toner usage measured with a set of printed pages.

The remainder of this part is organized as follows. Section 1.2 introduces the electrophotographic printer architecture and reviews the imaging pipeline used in the laser, electrophotographic printer technology. Section 1.3 describes the absorptance
Fig. 1.1. Overview of the two-stage toner usage predictor.

predictor and the toner usage mapping. There we discuss in detail the implementation of the two predictors. In Section 1.4, we introduce the construction approaches of this two-stage predictor. In Section 1.5, we present the error metric of the two-stage predictor from the cross validation, and compare its accuracy and stability with the pixel-counting algorithm. Finally, conclusions are drawn in Section 1.6.

1.2 Laser electrophotographic printer architecture

As shown in Figure 1.2, the electrophotographic process in the laser printer involves six basic steps: charging, exposure, development, transfer, fusing, and cleaning. The printing process begins after the printer controller receives page data and creates the raster image. The organic photoconductor drum (OPC) is positively charged by a charged roller until exposed to light. A pulsed laser beam scans the specific locations on the OPC drum surface with the help of a rapidly spinning polygon mirror. Light photons incident on the drum’s surface improve the conductivity in the region making the drum locally discharge in the area. The charges of the OPC surface are discharged according to the target image producing the latent image. Positively charged toner particles are then adhered to the discharged surface of the OPC from the developer roller. This developed image is transferred onto the paper through a charged transfer roller. The page is then passed through the fuser and pressure roller, and the toner
particles are pressed onto the paper, so that an input digital document is printed on the paper. The residual toner is removed by a cleaning blade that scrapes across the OPC surface to prepare for the following print [16].

Fig. 1.2. Architecture of a typical electrophotographic printer [38].

1.3 Implementation of the two-stage predictor

Since in the EP process, each of the C, M, Y, and K channels is treated separately [13], our pipeline is also separately implemented for each channel. That is why the input is a monochrome image, and the output is the corresponding toner usage of that channel. The proposed approach is further discussed below using the K channel as an example. The halftone algorithm embedded in our target printer generates 256-level PWM files. Here, each pixel value of the PWM image for K channel represents the
exposed duration of that pixel under the laser beam in the black cartridge, with 0 for non-exposure and 255 for fully exposure.

1.3.1 Implementation of the first stage: absorptance predictor

5×5 absorptance predictor

We develop the model for a particular target laser, electrophotographic printer, printing at 600 dpi. Assuming that the printed absorptance of a certain printed pixel is influenced by the distribution of its surrounding pixels in PWM image, we estimate the absorptance by first analyzing its 5×5 neighborhood in the digital image, then looking up the corresponding absorptance in a Look-Up Table (LUT) [14]. Figure 1.3 demonstrates how we implement the predictor for a letter “C” with gradient color. Its digital continuous-tone image is shown in Figure 1.3 (a). First, the PWM image is converted to a 16-level uniformly quantized image by preserving the highest 4 bits of each pixel value and omitting the lowest 4 bits, as shown in Figure 1.3 (b) and (c). In this gradient example, horizontal bandings appear due to the quantization in the 16-level PWM image. The quantized image contains much fewer 5×5 neighborhood configuration possibilities than the PWM image. Next, prediction process is performed pixel-by-pixel throughout the entire quantized image. Figure 1.3 (d) shows the 5×5 predictor window. We assign each pixel in the window an index ranging from 1 to 25. Pixel 13 is the one whose printed absorptance we wish to estimate. Every 5×5 dot configuration has a unique hexadecimal code array, \( \{q_1, q_2, \ldots, q_{25}\} \), where \( q_i, \quad i = 1, \ldots, 25 \), is the pixel value in the quantized image. For example, the hexadecimal code array of the pixel under processed in Figure 1.3 (d) is:

\[
\{0F00 0F00 20F2 00C0 0F00\}.
\] (1.1)

Search such a code array in the 5×5 absorptance predictor LUT (5×5 LUT) as shown in Figure 1.3 (e), and find the corresponding absorptance value stored in the table. Each absorptance value in the 5×5 LUT is an average over thousands of observations.
for a certain configuration collected from printed and then scanned training pages. These absorptance values are integers ranging from 38 to 249, where 38 represents page white and 249 stands for black. Figures 1.3 (f) and (g) show, respectively, the predicted scanned image of the letter “C” from the absorptance predictor and the actual scanned image captured with an Epson 10000 XL scanner at 1800 dpi.

The absorptance predictor has $4 \times 25$ state variables in total. If they were fully used, they could represent $2^{100}$ unique dot configurations. However, since the model is designed for the EP process, limited number of dot configurations will be encountered in practice, among which only several thousand configurations occurs frequently. This fact greatly reduces the storage and computational expense. In programming, the $5 \times 5$ LUT is developed using binary tree structure [15], which makes the searching more efficient and fast.
Fig. 1.3. Illustration of the absorptance predictor. (a) A digital continuous-tone monochrome image example of the letter “C” with gradient color. (b) PWM file generated by (a). (c) Quantized PWM image. (d) The $5 \times 5$ predictor window with pixel indices indicated. The pixel in blue is the one under estimation. (e) $5 \times 5$ absorptance predictor LUT. (f) Predicted scanned image of letter “C”. (g) Actual printed and then scanned image of letter “C”.

<table>
<thead>
<tr>
<th>Index</th>
<th>Configuration hexadecimal code array</th>
<th>Absorptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000000 00000 00000 00000 00000</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>FFFFF FFFFF FFFFF FFFFF FFFFF</td>
<td>245</td>
</tr>
<tr>
<td>3</td>
<td>F200F 00FC0 00F20 FC00F F200F</td>
<td>229</td>
</tr>
<tr>
<td>4</td>
<td>00FC0 00F20 FC00F F200F 00FC0</td>
<td>212</td>
</tr>
<tr>
<td>5</td>
<td>00F20 FC00F F200F 00FC0 00F20</td>
<td>208</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>28834</td>
<td>0FF9 FFFFF FFFFF FFFFF FFFFF</td>
<td>248</td>
</tr>
</tbody>
</table>
Due to the way in which we generate the 5×5 LUT, there might be a small number of dot configurations that occur during prediction but which were not encountered during the training, and are thus not included in the 5×5 LUT. Experimentally, we have observed that for a letter-size page full of random text, the chance is about 0.0024%; for an arbitrarily chosen image, the chance would generally be less than 1%. For those configurations, their estimated printed absorptance values are obtained by the following two approaches.

**3×3 absorptance predictor**

We count the number of pixels corresponding to each quantized level within the 3×3 neighborhood in a histogram manner. Denote such numbers as $n_j$, $j = 0, \ldots, 15$. Then the absorptance for the center pixel is the weighted sum performed according to the 3×3 LUT as shown in Table 1.1:

$$a^{\text{predict}} = \sum_{j=0}^{15} n_j \cdot w^{3\times3}_j,$$

(1.2)

where $w^{3\times3}_j$ is the weight, in other words, the contribution of a pixel with quantized level $j$ to the center pixel. Again, $w^{3\times3}_j$ are obtained based on data collected from training pages. This 3×3 absorptance predictor achieves acceptable prediction accuracy with efficient computation work, making it a desired algorithm to be implemented in real products.

<table>
<thead>
<tr>
<th>Quantized levels</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight ($w^{3\times3}_j$)</td>
<td>14.42</td>
<td>15.73</td>
<td>16.47</td>
<td>17.20</td>
<td>17.93</td>
<td>18.67</td>
<td>20.52</td>
<td>21.13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quantized levels</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight ($w^{3\times3}_j$)</td>
<td>24.87</td>
<td>26.82</td>
<td>28.34</td>
<td>31.06</td>
<td>33.80</td>
<td>34.22</td>
<td>33.78</td>
<td>36.55</td>
</tr>
</tbody>
</table>
**Weighted hamming distance predictor**

A more sophisticated but time-consuming approach is the weighted hamming distance (WHD) predictor. This approach is also inspired by the Shepard’s method [17] [18]. Given a configuration not existed in the 5x5 LUT but occurred in prediction process, we search for the closest several configurations defined by the weighted hamming distance [19] in the 5x5 LUT. Then the absorptance of this unknown configuration is calculated with a weighted average of absorptance values of its closest several configurations. This predictor does not involve any regression modeling the complex influences caused by neighbor pixel values, and all the absorptance values in the 5x5 LUT come from the measurement of printed and then carefully scanned training pages, not the manipulated means, so that its prediction result is much closer to the truth than the 3x3 absorptance predictor.

Another important fact is that the 5x5 LUT consists of configurations with absorptance value covers the whole range from 38 to 249, which can be observed from Figure 1.4, the plot of the histogram of absorptance values in the 5x5 LUT. And each absorptance value corresponds to at least dozens of configurations. Therefore, the predictor will have a great chance to find similar configurations from the 5x5 LUT given an unknown configuration.

**Weighted hamming distance**  Hamming distance [20] between two configurations is the sum of the absolute value of the differences between the 25 pairs of corresponding quantized pixel value. However, differences from inner pixel pairs lead to greater variance to the absorptance value of the center pixel than the same amount of differences from outer pixel pairs [21]. Therefore, such a difference is adjusted by certain weight associated with its position. As illustrated in Figure 1.5, there are 3 kinds of pixels regarding different positions in the 5x5 neighborhood: the center pixel itself, the 8 middle-ring pixels, and the 16 outer-ring pixels. They are assigned to weights,
Fig. 1.4. Histogram of absorptance values in the $5 \times 5$ LUT.

$w_{\text{center}}^{\text{ring}} = 3$, $w_{\text{middle}}^{\text{ring}} = 4$, and $w_{\text{outer}}^{\text{ring}} = 1$, respectively. The weighted hamming distance between two configurations is computed by:

$$d = \sum_{i=1}^{25} |q_{i,1} - q_{i,2}| \cdot w_{i}^{\text{ring}},$$

(1.3)

where $q_{i,1}$ and $q_{i,2}$ are quantized pixel values of the $i$th pixel in configuration 1 and configuration 2, respectively. $w_{i}^{\text{ring}}$ is the adjusted weighted corresponding to the position (center, middle, or outer) of the $i$th pixel. For instance, the weighted hamming distance between the two example configurations in Figure 1.5 is:

$$d = |3 - 3| \times 1 + |3 - 2| \times 1 + |0 - 2| \times 4 + |3 - 0| \times 3 = 18.$$  

(1.4)

**Shepard’s method**  Weighted hamming distance helps to determine the similarity between the given configuration and configurations in the $5 \times 5$ LUT. However, using
the closest one only would not be a wise choice. Because on the one hand, empirical areal data always comes with inaccuracy of measurement; on the other hand, the absorptance value of the closest one needs to be adjusted to become closer to the given configuration [22].

The similarity is examined by a threshold \( d_R = 65.5 \). Only configurations in the LUT with calculated weighted hamming distance \( d_k \) smaller than \( d_R \) are regarded as similar configurations, where \( k = 1, \ldots, N_{5\times5} \), and \( N_{5\times5} \) is the number of configurations in the \( 5\times5 \) LUT, which is 28834 in our case. Then the predicted printed absorptance is determined by the weighted average of the absorptance values of those similar configurations, where the weight \( w_k^{\text{shepard}} \) is proportional to the square of the inverse of \( d_k \).

A special situation we might encounter in practice is that the smallest value of \( \{d_k\} \), \( d_{\text{min}} \), is greater than \( d_R \). In this extreme case, no configuration in the LUT
is considered to be similar as the given configuration. We then directly adopt the absorptance value corresponding to this minimum distance as the estimation value.

The algorithm for the weighted hamming distance predictor is further described in Table 1.2.
Table 1.2
The algorithm for weighted hamming distance predictor.

- $k = 0$
- while $k < N_{5 \times 5}$
  - $k \leftarrow k + 1$
  - Compute the weighted hamming distance between the given configuration and the $k$th configuration in the $5 \times 5$ LUT, $d_k$.
- Search for the minimum value of $\{d_k\}$, $d_{\text{min}}$.
- if $d_{\text{min}} < d_R$
  - while $k < N_{5 \times 5}$
    * $k \leftarrow k + 1$
    * Update the weight $w_k^{\text{shepard}}$ as:
      $$w_k^{\text{shepard}} = \begin{cases} 
      \left(\frac{d_R - d_k}{d_k}\right)^2 & d_k < d_R \\
      0 & d_k \geq d_R 
      \end{cases}$$
  else if $d_{\text{min}} \geq d_R$
  - while $k < N_{5 \times 5}$
    * $k \leftarrow k + 1$
    * Update the weight $w_k^{\text{shepard}}$ as:
      $$w_k^{\text{shepard}} = \begin{cases} 
      0 & d_k \neq d_{\text{min}} \\
      1 & d_k = d_{\text{min}} 
      \end{cases}$$

(Continued on next page)
Table 1.2
The algorithm for weighted hamming distance predictor.
(continued from previous page)

- Estimate the absorptance of the given configuration as:

\[ a_{\text{predict}} = \frac{\sum_{k=1}^{N_{5 \times 5}} w_{k}^{\text{shepard}} \cdot a_{k}}{\sum_{j=1}^{N_{5 \times 5}} w_{j}^{\text{shepard}}}, \]

where \( a_{k} \) is the absorptance value corresponding to the \( k \)th configuration in the \( 5 \times 5 \) LUT.
1.3.2 Implementation of the second stage: toner usage mapping

The printed and then scanned image demonstrates the consequences of all of the complicated effects within each device-addressable cell on the digital image, which is what we have benefited from the first stage. On the other hand, based on the printed absorptance value, we are able to tell the toner consumption by knowing the relationship between them in advance. We assume the toner consumption won’t change for printed absorptance values in appropriately chosen ranges. Table 1.3 presents the amount of toner to be aggregated when a single-pixel area displaying certain absorptance value is observed. Denote such value as \( w_{tu}^q \), \( q = 1, ..., N_{\text{range}} \), where \( N_{\text{range}} \) is the total number of ranges listed in Table 1.3. Thus, given the predicted printed absorptance values from the first stage, we map them to corresponding \( w_{tu}^q \) respectively, and then integrate millions of pixels’ weights together; the result is the toner usage to print one copy of the digital continuous-tone image. The process can be represented by

\[
M_{\text{toner}} = \sum_{q=1}^{N_{\text{range}}} n_q \cdot w_{tu}^q,
\]

(1.5)

Where \( n_q \) is the number of pixels with predicted printed absorptance value in range \( q \).

\( w_{tu}^q \) does not stand for the toner consumption for that pixel, but is proportional to it. One might observe from Table 1.3 that ranges [0, 37] and [250, 255] map to zero. It is because the absorptance predictor does not provide absorptance values less than 38, or greater than 249. For the other ranges associated with zero weight, the reason is they are not necessarily needed for predicting toner usage. Those ranges, in fact, contain redundant information for estimating the toner usage. In other words, strong multicollinearity exists among the original set of ranges, which may lead to an unstable and less accurate predictor model. The printed page consists of millions of dots, clustered or separated. Each of them would have solid center, where toner particles assemble the dot, as well as the edge, where tone scattered around. Therefore, the printed absorptance must cover values the whole possible range, from 38 to 249,
even for those predicted by our first stage predictor. Appropriately selected ranges not only provide information for the overall page, but also result in more accurate regression results. Simplifying the complexity of the predictor not only makes the model more robust, but also reduces the memory requirement and the amount of time spent on computation [36]. The least absolute shrinkage and selection operator (LASSO) [23] [24] helps us determine the representative ranges (ranges with non-zero weights). Then \{w^\text{lin}_q\} are trained using the least squares method.

An interesting result concluded from Table 1.3 is that middle level absorptance value, such as values in the range [180, 184], contributes more toner per pixel than high level absorptance value, such as values in the range [230, 239]. It is consistent with the known fact that edges absorb more toner per pixel than thoroughly printed blocks [25]. For instance, two black rectangular blocks with the same area printed on the page; but one is square, and the other is an elongated rectangular. According to Table 1.3, the second one consumes more toner for its much longer edge. This result corrects a misconception people always tend to believe: printing trivial intervals within blocks saves toner. The truth is more toner would be attracted along the intervals, and even covers them, so that we would barely see those intervals.

1.4 Construction of the two-stage predictor

1.4.1 Construction of the first stage: absorptance predictor

5×5 absorptance predictor

The essence of developing the predictor is to obtain the 5×5 absorptance LUT, which lists all pairs of dot configuration and corresponding absorptance that may occur in the test pages. Figure 1.6 shows the procedure to obtain the LUT. There are three kinds of training pages with different contents: text, lines and images. as shown in Figure 1.7. These 5 training pages are halftone files printed with target printer at 600 dpi and scanned with the Epson Expression 10000XL scanner at 1800 dpi.
Table 1.3
Absorptance ranges and their corresponding toner usage.

<table>
<thead>
<tr>
<th>Absorptance</th>
<th>$w_{q}^{t_{u}} \times 10^{-10} g$</th>
<th>Absorptance</th>
<th>$w_{q}^{t_{u}} \times 10^{-10} g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 39]</td>
<td>0</td>
<td>[160, 179]</td>
<td>0</td>
</tr>
<tr>
<td>[40, 54]</td>
<td>-28.80</td>
<td>[180, 184]</td>
<td>62.27</td>
</tr>
<tr>
<td>[55, 69]</td>
<td>26.12</td>
<td>[185, 214]</td>
<td>0</td>
</tr>
<tr>
<td>[70, 84]</td>
<td>-8.64</td>
<td>[215, 224]</td>
<td>31.00</td>
</tr>
<tr>
<td>[85, 119]</td>
<td>0</td>
<td>[225, 229]</td>
<td>36.77</td>
</tr>
<tr>
<td>[120, 129]</td>
<td>33.14</td>
<td>[230, 239]</td>
<td>15.74</td>
</tr>
<tr>
<td>[130, 154]</td>
<td>0</td>
<td>[240, 249]</td>
<td>22.82</td>
</tr>
<tr>
<td>[155, 159]</td>
<td>31.47</td>
<td>[250, 255]</td>
<td>0</td>
</tr>
</tbody>
</table>

Training pages with text are about random text chosen from the 52 letters of the alphabet including all small and capital letters from “a” to “z” and “A” to “Z”. The meaningless words on these random text files are generated by the software, “Monkey Random Text Generator” [26]. Letters are written in Times New Roman typeface with 12 point size on training page 1, and in Arial with 16 point size on training page 2. Training page 3 contains horizontal and vertical lines with widths randomly changing from 1-pixel-wide to 10-pixel-wide. The spacing between two lines is 50-pixel-wide, so that it is large enough to preserve the appearances of those lines. Training pages 4 and 5, named as “Bridge” and “Complex”, respectively, are images often used in image quality test.
Fig. 1.6. Procedure for developing the $5 \times 5$ absorptance predictor.

<table>
<thead>
<tr>
<th>Index</th>
<th>Configuration hexadecimal code array</th>
<th>Absorptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00000 00000 00000 00000 00000</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>FFFFFFF FFFFFFF FFFFFFF FFFFFFFF</td>
<td>245</td>
</tr>
<tr>
<td>3</td>
<td>F200F 00FC0 00F20 FC00F F200F</td>
<td>229</td>
</tr>
<tr>
<td>4</td>
<td>00FC0 00F20 FC00F F200F 00FC0</td>
<td>212</td>
</tr>
<tr>
<td>5</td>
<td>00F20 FC00F F200F 00FC0 00F20</td>
<td>208</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>741131</td>
<td>AFFFF FFFFF FFFFF FFFFF FFFFFF</td>
<td>247</td>
</tr>
</tbody>
</table>
Fig. 1.7. The five training pages for the $5 \times 5$ LUT. (a) Training page 1: letters in Arial with 16 point size. (b) Training page 2: letters in Times New Roman with 12 point size. (c) Training page 3: mixed lines. (d) Training page 4: Complex. (e) Training page 5: Bridge
Figure 1.8 is a clipped portion of training page 4. Fiducial marks [27] are printed every 265 printer pixels to address the alignment problem, and to locate pixels when we analyze the scanned training page. Those fiducial marks are $5 \times 5$ black squares placed in the middle of the $25 \times 25$ white space, which helps to ensure that fiducial marks do not affect the printed absorptance of the content. But for pixels eligible to be used in the $5 \times 5$ LUT construction, a farther distance is needed. As the blue area shown in Figure 1.8, the space outside those $50 \times 50$ squares centered by each fiducial mark is called the analysis area, where every pixel’s printed absorptance contributes to the training datum. Also, we eliminate a 100-pixel-wide margin from the analysis area. More specifically, for each pixel in the blue area, we compute its configuration array, measure its printed absorptance, and record the printed absorptance according to its configuration array as illustrated Table 1.4. There are 21 rows and 15 columns of fiducial marks on the training page. Therefore, there are about $2.8 \times 10^7$ pixels to be analyzed for a 6200 pixels by 4800 pixels training page.

Since the test page is printed at 600 dpi and scanned at 1800 dpi, every printer pixel corresponds to $3 \times 3$ (1800/600 $\times$ 1800/600) scanner pixels. The scanned test page is calibrated to make it more consistent with the appearance of the printed page. Specifically, the calibrated scanner gray values are proportional to 1 minus the CIE Y (luminance) value scaled to lie between 0 and 1.

### Table 1.4

The printed absorptance value recorded for each configuration array.

<table>
<thead>
<tr>
<th>Configuration code array</th>
<th>Record of the printed absorp. value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000 00000 00000 00000 00000</td>
<td>37 38 37 40 \ldots</td>
</tr>
<tr>
<td>FFFFF FFFFF FFFFF FFFFF FFFFF</td>
<td>240 247 239 241 244 \ldots</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>AFFFF FFFFF FFFFF FFFFF FFFFF</td>
<td>246 243 233 241 \ldots</td>
</tr>
</tbody>
</table>
In the analysis step, the calibrated image is binarized by Otsu’s method [28] to generate a binary mask that generally indicates the location of the fiducial marks and letters. Then a much more accurate estimation of the centroids of the fiducial marks is found based on the spatial distribution of toner absorptance in the binary mask region. The horizontal and vertical centroids of the $i$th fiducial mark are given by

$$C_{x,i} = \frac{\sum_{[m,n] \in D_i} m \cdot a^{\text{scan}}[m, n]}{\sum_{[m,n] \in D_i} a^{\text{scan}}[m, n]}, \quad (1.6)$$

and

$$C_{y,i} = \frac{\sum_{[m,n] \in D_i} n \cdot a^{\text{scan}}[m, n]}{\sum_{[m,n] \in D_i} a^{\text{scan}}[m, n]}, \quad (1.7)$$

where $D_i$ is the region of $i$th fiducial mark in the binary mask, and $a^{\text{scan}}[m, n]$ is the absorptance value of the scanned image at the pixel with coordinates $[m,n]$. For
a given 265×265 fiducial-mark block, there is a fiducial mark at each corner of the block, which gives us two pairs of fiducial marks in the scan direction. The center of the fiducial block is determined by averaging the estimations from the two pairs respectively. Usually, this center does not fall right on an intersection point of the scanner-pixel grid. So we perform a sub-scanner-pixel method as follows to locate each printer pixel. The printer pixels in the analysis area are located by expanding from the center, which means segmenting every 3×3 scanner-pixel area around the center point, until we acquire all the 200×200 printer pixels.

Next, the absorptance of each analyzed printer pixel is measured by averaging the absorptance in the corresponding 3×3 scanner-pixel area. We then record the absorptance according to the pixel’s configuration code array. The estimated printed absorptance of the central pixel for a certain 5×5 pattern is given by the sample absorptance of the corresponding scanned 3×3 scanner-pixel cells averaged over all the occurrences of that pattern. These estimates are put into a LUT as shown in Figure 1.3 (e). Finally, only configurations occurred more than 50 times are preserved, because too few occurrences might enlarge the influence of inaccurate measurement. Also, those minor configurations would hardly be encountered in practice.

3×3 absorptance predictor

The 3×3 absorptance predictor is developed by linear regression based on the 5×5 absorptance LUT. The parameter we aim at is $w_{j}^{3×3}$, $j = 0, ..., 15$, the contribution of a pixel with quantized level $j$ to the center pixel. We count the number of pixels corresponding to each quantized level within the 3×3 neighborhood for configurations listed in the 5×5 LUT. Denote such numbers as $n_{(k,j)}$, where $k = 1, \ldots, N_{5×5}$, is the number of configurations in the 5×5 LUT. The corresponding absorptance value is denoted as $a_{k}^{5×5}$. The linear regression equation is given as:
The solution of the above equation gives out the values we are interested in for \( w_{j}^{3 \times 3} \).

**Weighted hamming distance predictor**

As discussed in section 2.1.3, there are 4 parameters to be determined: \( w_{\text{ring}}^{\text{center}}, w_{\text{ring}}^{\text{middle}}, w_{\text{ring}}^{\text{outer}}, \) which are weights used to adjust the hamming distance according to pixel position respectively, and \( d_{R} \), which is the upper bound for the acceptable length of hamming distance between two configurations. However, one of the three weights can be set as 1, as long as the other two weights and \( d_{R} \) are chosen appropriately. Let \( w_{\text{outer}}^{\text{ring}} \) equal to 1.

The four parameters are trained based on datum in the 5×5 absorptance LUT. We use the first 4000 most frequently occurred configurations as data base to predict the following 2000 configurations. The goal is to obtain \( w_{\text{ring}}^{\text{center}}, w_{\text{ring}}^{\text{middle}}, \) and \( d_{R} \), that achieve minimum average relative error, where the average relative error is computed by:

\[
ARE_{rr} = \frac{1}{2000} \sum_{p=40001}^{6000} \left| \frac{a_{p}^{\text{predict}} - a_{p}^{5 \times 5}}{a_{p}^{5 \times 5}} \right| \times 100%,
\]

where \( a_{p}^{5 \times 5} \) is the absorptance of the \( p \)th configuration listed in the 5 × 5 LUT, and \( a_{p}^{\text{predict}} \) is its corresponding predicted absorptance by the weighted hamming distance predictor. \( w_{\text{center}}^{\text{ring}} \) and \( w_{\text{middle}}^{\text{ring}} \) cannot be set too large, otherwise, the influence of the outer-ring pixel would be override. Experimentally, both \( w_{\text{center}}^{\text{ring}} \) and \( w_{\text{middle}}^{\text{ring}} \) should be smaller than 10.

We can derive a rough upper bound for \( d_{R} \) also. Assume the quantized pixel value is a random variable with uniform distribution of the integer value from 0 to 15.
Then the expected value of the hamming distance of two pixels is 5.3 with standard deviation of 3.8. Suppose $w_{center}^{\text{ring}}$ and $w_{middle}^{\text{ring}}$ are set to be the maximum value 10, and pixels are independent and identically distributed within the $5 \times 5$ window. Thus the expected value of the weighted hamming distance would be:

$$Ed = 5.3 \cdot w_{center}^{\text{ring}} + 5.3 \cdot w_{middle}^{\text{ring}} \cdot 10 + 5.3 \cdot 16.$$  \hspace{1cm} (1.10)

The corresponding standard deviation would be:

$$Stdd = 3.8 \cdot w_{center}^{\text{ring}} + 3.8 \cdot w_{middle}^{\text{ring}} \cdot 10 + 3.8 \cdot 16.$$  \hspace{1cm} (1.11)

Therefore the rough estimate would be that $d_{R}$ is smaller than

$$d_{R} < Ed - Stdd = 1.5 \cdot w_{center}^{\text{ring}} + 1.5 \cdot w_{middle}^{\text{ring}} \cdot 10 + 24.$$  \hspace{1cm} (1.12)

When $w_{center}^{\text{ring}}$ and $w_{middle}^{\text{ring}}$ are assigned the maximum value 10, $d_{R} < 159$.

The function of the average relative error with respect to the four parameters is not continuous, nor in certain tendency, but we assume it is unimodal within small variation of the parameters [30]. We apply the Golden section search [29] starting from all the possible integer combinations of $w_{center}^{\text{ring}}$, $w_{middle}^{\text{ring}}$, and $d_{R}$ respectively. Each time, a local minimum solution is obtained. Then the optimizer of this problem is the set of value that results in the smallest local minimum average relative error. Table 1.5 explains the algorithm of this optimization problem in detail.
Table 1.5
The algorithm for searching the parameters of the weight hamming distance predictor.

1. \( w_{\text{center}}^{\text{ring}} = 1 \), \( w_{\text{middle}}^{\text{ring}} = 1 \), \( n = 1 \)

2. for \( w_{\text{center}}^{\text{ring}} < 11 \)
   - for \( w_{\text{middle}}^{\text{ring}} < 11 \)
     - \( d_R = 1 \)
     - \( \text{Maxd}_R < 1.5 \cdot w_{\text{center}}^{\text{ring}} + 1.5 \cdot w_{\text{middle}}^{\text{ring}} \cdot 10 + 24 \)
     - for \( d_R < \text{Maxd}_R \)
       - \( \text{ARErr} = 1 \), \( \text{arerr} = 0 \)
       - while(\( |\text{ARErr} - \text{arerr}| > 0.01 \))
         - \( \text{ARErr} \leftarrow k \)
         - Search for the minimizer \( D_R \) in \([d_R - 1, d_R + 1]\) using the Golden search given \( w_{\text{center}}^{\text{ring}} \) and \( w_{\text{middle}}^{\text{ring}} \).
         - Search for the minimizer \( W_{\text{center}}^{\text{ring}} \) in \([w_{\text{center}}^{\text{ring}} - 1, w_{\text{center}}^{\text{ring}} + 1]\) using the Golden search given \( d_R \) and \( w_{\text{middle}}^{\text{ring}} \).
         - Search for the minimizer \( W_{\text{middle}}^{\text{ring}} \) in \([w_{\text{middle}}^{\text{ring}} - 1, w_{\text{middle}}^{\text{ring}} + 1]\) using the Golden search given \( d_R \) and \( w_{\text{center}}^{\text{ring}} \).
         - Update \( \text{arerr} \) using current optimizer.
       - \( \text{ARErr} \leftarrow \text{arerr} \)
       - \( n \leftarrow n + 1 \)

3. Find the set of parameters corresponding to the smallest value in \( \text{LMS} \), and assign them to \( w_{\text{center}}^{\text{ring}} \), \( w_{\text{middle}}^{\text{ring}} \), and \( d_R \), respectively.
1.4.2 Construction of the second stage: toner usage mapping

We build this mapping based on 35 training pages, which contain 3-pixel-wide to 10-pixel-wide horizontal and vertical lines files, 10 random text files with different typefaces, point sizes, and line spacings, and 9 arbitrarily chosen images. Figure 1.4.2 shows two examples of them: the 12 point Arial random text with 1.15 line spacing and the image named “DNA”.

We measure the toner consumed by printing 100 copies of each test page to reduce the experimental error. The measurement is implemented by splitting and weighing the black cartridge [31] each time before and after the 100-copy printing job. Denote the toner usage per single page as $m_{\text{toner}_i}, i = 1, \ldots, 35$.

The pixel-by-pixel absorptance of the 35 printed and scanned test pages are predicted by the absorptance predictor. We then calculate the histograms of these predicted scanned images. Figure 1.4.2 shows the histograms of the two example test pages. The printed absorptance is expected to be related to the toner mass. We assume that the mapping can be regarded to be constant within reasonably divided absorptance ranges. The set of ranges is arbitrarily chosen at first as shown in Table 1.6. We ignore ranges [0, 37] and [250, 255), because the absorptance predictor does not provide absorptance values less than 38, or greater than 249.
Fig. 1.9. Two examples from the 35 test pages and their corresponding histograms of the predicted scanned images: (a) the test page with 12 point Arial random text with 1.15 line spacing, and (b) DNA.
Table 1.6
Original absorptance ranges selection and their variance inflation factors (VIFs).

<table>
<thead>
<tr>
<th>Range Index</th>
<th>Abs. range</th>
<th>VIF</th>
<th>Range Index</th>
<th>Abs. range</th>
<th>VIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[38, 39]</td>
<td>46.88</td>
<td>15</td>
<td>[170, 174]</td>
<td>2.50 × 10^7</td>
</tr>
<tr>
<td>2</td>
<td>[40, 54]</td>
<td>5.73 × 10^5</td>
<td>16</td>
<td>[175, 179]</td>
<td>4.40 × 10^7</td>
</tr>
<tr>
<td>3</td>
<td>[55, 69]</td>
<td>1.06 × 10^6</td>
<td>17</td>
<td>[180, 184]</td>
<td>1.80 × 10^6</td>
</tr>
<tr>
<td>4</td>
<td>[70, 84]</td>
<td>1.34 × 10^6</td>
<td>18</td>
<td>[185, 189]</td>
<td>7.52 × 10^6</td>
</tr>
<tr>
<td>5</td>
<td>[85, 99]</td>
<td>3.78 × 10^6</td>
<td>19</td>
<td>[190, 194]</td>
<td>1.61 × 10^7</td>
</tr>
<tr>
<td>6</td>
<td>[100, 119]</td>
<td>1.08 × 10^8</td>
<td>20</td>
<td>[195, 199]</td>
<td>5.00 × 10^7</td>
</tr>
<tr>
<td>7</td>
<td>[120, 129]</td>
<td>1.96 × 10^8</td>
<td>21</td>
<td>[200, 204]</td>
<td>4.72 × 10^7</td>
</tr>
<tr>
<td>8</td>
<td>[130, 139]</td>
<td>1.85 × 10^7</td>
<td>22</td>
<td>[205, 209]</td>
<td>2.51 × 10^7</td>
</tr>
<tr>
<td>9</td>
<td>[140, 144]</td>
<td>6.01 × 10^7</td>
<td>23</td>
<td>[210, 214]</td>
<td>1.45 × 10^8</td>
</tr>
<tr>
<td>10</td>
<td>[145, 149]</td>
<td>1.22 × 10^7</td>
<td>24</td>
<td>[215, 224]</td>
<td>8.77 × 10^6</td>
</tr>
<tr>
<td>11</td>
<td>[150, 154]</td>
<td>2.97 × 10^7</td>
<td>25</td>
<td>[225, 229]</td>
<td>1.98 × 10^6</td>
</tr>
<tr>
<td>12</td>
<td>[155, 159]</td>
<td>1.27 × 10^7</td>
<td>26</td>
<td>[230, 239]</td>
<td>314.42</td>
</tr>
<tr>
<td>14</td>
<td>[165, 169]</td>
<td>1.01 × 10^7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We can examine the degree of multicollinearity by using the variance inflation factor (VIF) [32]. Let \( h_{\text{range}_j}^i \) denote the number of pixels with absorptance in range \( j \) of test page \( i, \ j = 1, \ldots, 27 \). In our case, the VIF of the \( j \)th range is defined as

\[
VIF_j = \frac{\sum_{i=1}^{35} (h_{\text{range}_j}^i - \bar{h}_{\text{range}_j})^2}{\sum_{i=1}^{35} (h_{\text{range}_j}^i - f_{\text{range}_j}^i)^2},
\]

where

\[
\bar{h}_{\text{range}_j} = \frac{1}{35} \sum_{i=1}^{35} h_{\text{range}_j}^i,
\]

\[
f_{\text{range}_j}^i = \sum_{k=1, k \neq j}^{27} \alpha_k \cdot h_{\text{range}_k}^i.
\]

\( f_{\text{range}_j}^i \) is the predicted value of \( h_{\text{range}_j}^i \) using \( h_{\text{range}_k}^i, \ k = 1, \ldots, 15, \ k \neq j \). \( \alpha_k \) can be determined by the ordinary least squares method. Table 1.6 shows the VIF value of each range for the original set of ranges. A VIF value that exceeds 10 is often regarded as an indicator of strong multicollinearity [33]. According to this criterion, except the last one, all of the other original ranges are highly correlated with each other.

The algorithm we use to select ranges is the Least Absolute Shrinkage and Selection Operator (LASSO) [34]. LASSO is able to find a minimal set of ranges needed to maintain an accurate prediction model. LASSO starts with the ordinary least square regression (OLSR) in the following equation, which includes all the original ranges in the calculation [35].

\[
\begin{bmatrix}
    h_{\text{rang}_1}^1 & h_{\text{rang}_2}^1 & \cdots & h_{\text{rang}_{27}}^1 \\
    h_{\text{rang}_1}^2 & h_{\text{rang}_2}^2 & \cdots & h_{\text{rang}_{27}}^2 \\
    \vdots & \vdots & \ddots & \vdots \\
    h_{\text{rang}_1}^{35} & h_{\text{rang}_2}^{35} & \cdots & h_{\text{rang}_{27}}^{35}
\end{bmatrix}
\begin{bmatrix}
    w_{\text{range}_1} \\
    w_{\text{range}_2} \\
    \vdots \\
    w_{\text{range}_{27}}
\end{bmatrix}
= 
\begin{bmatrix}
    m_{\text{toner}_1} \\
    m_{\text{toner}_2} \\
    \vdots \\
    m_{\text{toner}_{35}}
\end{bmatrix}
\]

(1.16)

Let \( WH_{\text{initial}} \) represent the absolute sum of the coefficient vector \( w \) in the above equation, i.e., \( WH_{\text{initial}} = \sum_{k=1}^{27} |w_{\text{range}_k}| \). Now, we add a constraint \( \sum_{k=1}^{27} |w_{\text{range}_k}| < t \)
to the OLSR, and solve the constrained least square regression. In the process, we gradually reduce the tuning parameter $t$ from $WH_{\text{initial}}$ to zero. As $t$ is reduced, more and more of the less important coefficients $w_{\text{range}k}$ are shrunk to zero. The ranges with non-zero coefficients can be selected as a candidate set of the explanatory variables for the prediction model. Every time $t$ is reduced, a candidate set will be generated. We record those candidate sets for further examination.

Sevenfold cross-validation is performed to estimate the accuracy of the models regressed with all the candidate sets from the LASSO process. Consider that a candidate set is to be examined. The 35 test pages are divided into 7 groups with 5 pages in each group. Six groups are used as training pages to build up the toner mapping model, and the remaining group serves as test pages. We compute the root-mean-square error (RMSE) between the measured toner usage and the predicted toner usage for the pages in the testing set. We repeat this process seven times until each group has been used exactly once as the testing data. We calculate the average value $\text{mean}_{\text{RMSE}}$ of the seven RMSEs, and their standard error $\text{std}_{\text{RMSE}}$. Assume the minimum average RMSE among all candidate sets is $\text{min}_\text{mean}_{\text{RMSE}}$ and its corresponding standard error is $\text{min}_\text{std}_{\text{RMSE}}$. We determine the final optimal set of ranges according to the one-standard error rule: the set consisting of the least number of ranges with $\text{mean}_{\text{RMSE}}$ less than $\text{min}_\text{std}_{\text{RMSE}}$ above $\text{min}_\text{mean}_{\text{RMSE}}$ is chosen as an optimal set. The model trained on the optimal set has fewer attributes and also is comparable in accuracy to that obtained for OLSR.

Once the set of representative ranges is obtained, the toner mass mapping weights $\{w_{\text{tu}q}\}$ are trained by applying the ordinary linear regression to the set.
1.5 Evaluation of the two-stage predictor

1.5.1 The $3 \times 3$ absorptance predictor and the weighted hamming distance predictor

As discussed in Section 2 and 3, both the $3 \times 3$ absorptance predictor and the weighted hamming distance predictor help to predict printed absorptance for configurations not existed in the $5 \times 5$ LUT. However, the WHD algorithm is expected to perform better, because it searches for similar configurations, and the printed absorptance does not vary significantly with subtle changes in the pattern. While the $3 \times 3$ predictor represents all the complicated inter-pixel interaction with only 16 parameters, and it overlooks the significance of the pixel position in the $3 \times 3$ window, not mention the $5 \times 5$.

Quantitative evaluation of the two approaches is given by Table 1.7. Both of the two predictor are first constructed as described in Section 3.1.2 and Section 3.1.3 with the $5 \times 5$ LUT. Recall that, in Section 3.1.1, configurations that occur less than 50 times in the training pages are eliminated from the $5 \times 5$ LUT, because they might be especially linked with certain training pages. We choose those encountered more than 40 times as the testing configurations for the two predictor, so that their measured absorptance value are more reliable. There are in total about five thousand such configurations. Relative error between the prediction and the average measured value for the testing configurations are computed by

$$ReError_{\text{abs}} = \frac{a_{\text{predicted}} - a_{\text{measure}}}{a_{\text{measure}}} \times 100\%,$$

(1.17)

The average relative error between the prediction and the average measured value are presented in Table 1.7.

Table 1.7 confirms our conjecture that the WHD predictor produces more accurate results. The average absolute $ReError_{\text{abs}}$ is only 2.05%; considering one standard deviation above it, most of the absolute $ReError_{\text{abs}}$ are smaller than 5.45%, which are very close to the true values. And this small difference generally would not cause
Table 1.7
The $3 \times 3$ absorptance predictor and the weighted hamming distance predictor.

<table>
<thead>
<tr>
<th></th>
<th>$3 \times 3$ predictor</th>
<th>WHD predictor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum / maximum $ReError_{\text{abs}}$ (%)</td>
<td>-98.36 / 120.94</td>
<td>-44.33 / 45.10</td>
</tr>
<tr>
<td>Mean $ReError_{\text{abs}}$ (%) $\pm$ std. dev. (%)</td>
<td>3.27 $\pm$ 19.02</td>
<td>-0.53 $\pm$ 4.69</td>
</tr>
<tr>
<td>Mean abs. $ReError_{\text{abs}}$ (%) $\pm$ std. dev. (%)</td>
<td>7.45 $\pm$ 15.72</td>
<td>2.05 $\pm$ 3.45</td>
</tr>
</tbody>
</table>

a change to absorptance range in the second stage. WHD predictor is much superior to the $3 \times 3$ predictor for all the error metrics listed in the table. But the $3 \times 3$ predictor is still acceptable based on fact that the average $ReError_{\text{abs}}$ and absolute $ReError_{\text{abs}}$ are less than 10%. Another important reason is that the regular halftone patterns generate limited number of possible $5 \times 5$ configurations, and most of the frequently occurred configurations have been included in the $5 \times 5$ LUT. So that the non-existent pattern predictor contributes a small amount of toner usage to the total prediction.

One obvious advantage of the $3 \times 3$ predictor is the fast computational speed. It merely requires 9 add operations. But the WHD predictor works much more complicatedly. In practice, we need to consider carefully to balance between the accuracy and the computational expenses.

1.5.2 Evaluation of the first stage: predicted scanned images

The first stage predictor is examined in this section with the test page, named “Toy Store”, in which the predictor is challenged by the mixed contents of text and images. We use the WHD predictor to estimate those configurations not found in the $5 \times 5$ LUT. Figure 1.5.2 illustrates both the predicted scanned page and the page scanned under 1800 dpi. The two images exhibit very unnoticeable visual differences with normal rendering resolution. The gray value of both text and images
are estimated correctly compared with the real scanned image; even for those “tricky areas”, such as the wrinkles on the boy’s cloth, and the children’s hair, the predictor preserves extraordinarily rich details. The predicted images looks more smooth and uniform over the background, because the estimate process filters out the variance in absorptance values for the same configuration. Also, the printer and scanner always produce noises, especially on images.

Figure 1.5.2 shows two zoom-in parts of the predicted scanned image and the real scanned image, one for the text, and the other for picture of the girl’s dress. For the estimated text image, the edge of strokes are predicted as transition from letters to background, which is consistent with the scanned page. The clip portion of the cloth is a great example where halftone patterns are clearly preserved. Both examples demonstrate the accuracy and resemblance of the predicted image to the real scanned file.
Fig. 1.10. Predicted scanned page and the real scanned page of Toy Store.
Fig. 1.11. Zoom-in picture of predicted scanned page and the real scanned page of Toy Store.
1.5.3 Evaluation of the second stage: Cross-validation experiment

Sevenfold cross-validation is performed in order to evaluate the performance of our new two-stage predictor. We examine the same 35 test pages described in the previous section. The 35 test pages are divided into 7 groups each with 5 representative pages. Six groups are used as training pages to establish the two-stage predictor; the remaining group serves as test pages. The process is repeated seven times until each group has been used exactly once as the test set. Table 1.8 shows the average weights and their standard deviations (std. dev.) over the 7 training sets. All of the ranges yield small standard deviations less than 10% of their corresponding weights. In fact, except the first three ranges, all of the other ranges achieve small standard deviations less than 3%, which indicates superior stability and robustness of the two-stage predictor.

<table>
<thead>
<tr>
<th>Absorptance</th>
<th>$w_{q}^{\text{in}} \pm \text{std. dev.} , (\times10^{-10} g)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[40, 54]</td>
<td>-29.40 ± 1.46</td>
</tr>
<tr>
<td>[55, 69]</td>
<td>26.17 ± 1.37</td>
</tr>
<tr>
<td>[70, 84]</td>
<td>-8.64 ± 0.81</td>
</tr>
<tr>
<td>[120, 129]</td>
<td>33.41 ± 0.96</td>
</tr>
<tr>
<td>[155, 159]</td>
<td>31.47 ± 0.77</td>
</tr>
<tr>
<td>[180, 184]</td>
<td>62.17 ± 0.87</td>
</tr>
<tr>
<td>[215, 224]</td>
<td>30.81 ± 0.62</td>
</tr>
<tr>
<td>[225, 229]</td>
<td>36.42 ± 0.88</td>
</tr>
<tr>
<td>[230, 239]</td>
<td>16.09 ± 0.59</td>
</tr>
<tr>
<td>[240, 249]</td>
<td>22.82 ± 0.23</td>
</tr>
</tbody>
</table>
1.5.4 Comparisons between the two-stage predictor and current toner usage prediction approach

To further examine our proposed predictor, we compare it with the current toner usage prediction approach, pixel-counting algorithm. Table 1.9 demonstrates the accuracy of them. Here, the relative error is defined as

\[ ReError_{tu} = \frac{m_{predicted} - m_{experiment}}{m_{experiment}} \times 100\%, \]  

(1.18)

where \( m_{predicted} \) will be calculated with both of the two approaches, and \( m_{experiment} \) is the toner consumption measured through the split and weigh experiment. For each test page in the data set, we compute the relative error for both the cross validation experiment with the two-stage predictor and the pixel-counting algorithm. Table 1.9 shows the mean value, mean absolute value, and standard deviations of the relative error over the 35 test pages.

<table>
<thead>
<tr>
<th></th>
<th>Two-stage predictor</th>
<th>Pixel-counting algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum / maximum</td>
<td>-4.75 / 6.69</td>
<td>-15.68 / 33.13</td>
</tr>
<tr>
<td>( ReError_{tu} ) (%)</td>
<td>-0.01 ± 2.21</td>
<td>5.57 ± 11.12</td>
</tr>
<tr>
<td>Mean absolute ( ReError_{tu} ) (%)</td>
<td>1.58 ± 1.54</td>
<td>10.48 ± 6.69</td>
</tr>
</tbody>
</table>

The two-stage predictor exhibits significantly better accuracy based on all measures. The large absolute value of both the minimum and maximum relative error for the pixel-counting algorithm are inherited from the prediction of the images. Images are often more challenging to be estimated, because the spatial arrangement of pixels tends to be especially complicated. However, the pixel-counting algorithm neglects lots of inter-pixel interaction, so that fails the prediction. Our two-stage predictor
yields excellent performance over all of the 35 test pages. The mean relative error is a factor of 557 less for the two-stage predictor than it is for the pixel counting algorithm, which indicates a much lower average bias close to zero in the predictor, while the pixel-counting algorithm is generally over estimated. The mean absolute relative error is a factor of 6.6 less for the two-stage predictor; and the standard deviation in this statistic is a factor of 4.3 less for the two-stage predictor. This suggests that the two-stage predictor will provide better accuracy on average, and will do it consistently across many different types of content.

1.6 Conclusion

In this work, we have proposed a two-stage toner usage predictor for laser, electrophotographic printers. The predictor is designed to achieve a higher accuracy of toner consumption estimation than current prevalent algorithms, and then enhance customer experience when use laser, electrophotographic printers.

The predictor consists of two stages, an absorptance predictor which determines the absorptance from printed and scanned pages, and a toner usage mapping which converts the absorptance into toner usage. We discussed the implementation and the construction for both the two stages. In the first stage, the predicted scanned page accounts for the influence of the state of neighboring pixels, which correlates the digital page with the toner usage better than the pixel-counting method which assumes the toner usage is linearly increasing with the digital pixel values. In the second stage, the multicollinearity between the histogram of the predicted scanned page and the toner usage is reduced by LASSO. Prediction based on the selected absorptance ranges yields better stability and consistency in the results.

Two alternative substep prediction strategies are introduced for the "non-existent patterns" problem. The $3 \times 3$ absorptance predictor only involves several add and multiply operations so that it can be implemented efficiently while the satisfactory prediction accuracy is preserved. The weighted hamming distance predictor is a much
more sophisticated algorithm, therefore produces accurate results that much closer to the truth more than the 3 \times 3 absorptance predictor.

The performances of the two stages are separately evaluated. We examined the proposed predictor on 35 arbitrarily chosen test pages, and compared it with the existing approach, the pixel-counting algorithm. Our two-stage predictor yields a much more accurate knowledge of toner consumption and a more consistent predicting ability than the pixel-counting algorithm.
2. OBJECT MAP GENERATION FROM RASTER INPUT IMAGE

2.1 Introduction

When using a laser electrophotographic (EP) printer, undesirable print artifacts may occur such as fine pitch banding, streaks, and mottle. In some circumstances, those artifacts are the results of the laser EP printer’s unstable operations, which include fluctuation in the angular velocity of the organic photoconductor (OPC) drums, and the irregular charging, development, and transfer processes [39] [40]. Researches have shown that smooth areas are more susceptible to print artifacts compared to detail areas, because detail information would mask the appearance of print artifacts [41]. Lowering the screen frequency is considered to be an effective method to reduce visible artifacts in smooth areas. However, lower screen frequency also restricts the ability of halftone process to reproduce the detail information on a page. To address this dilemma, the object-oriented halftoning [38] is proposed, where different halftone patterns are applied to different areas on the page according to their object types, and the seamless halftoning algorithm is proposed to remove the boundary artifacts.

Halftone screen frequency is not the only variable that should be adjusted for different object types. The color map, which is used in the transformation from the RGB color space to the CMYK color space, would also influence the appearance of the page printed [37]. Two types of color maps are considered here, process neutral (PN) and K-only process (KN). PN uses the four-color ink set (cyan, magenta, yellow, and black) to reproduce neutral colors, while KN only uses the black ink. Different color appearances are perceived when the two color maps are applied respectively. Generally speaking, PN smoothes the transition in gradient from highlight to shadow,
and blends the boundary of two adjacent regions very well. But PN may introduce color misregistration and halo artifacts in the smooth area [42]. KN gives a sharper image, which is ideal for symbols, but would generate contour or gloss changes in the transition from neutral gray to color regions [43].

Based on the characteristics of each halftone screen and color map, the object-oriented halftoning strategy is designed as follows [37]. Three object types are defined: raster, symbol, and vector. Raster includes areas such as photos, pictures, and graphics, which consists of high-frequency contents and frequent color changes. Symbol denotes small areas with sharp contours and smooth interiors, for example, text and logos. Vector refers to large smooth areas like background. A typical page content and its corresponding object map are illustrated in Figure 2.1. We apply high frequency screen and PN to raster areas, apply high frequency screen and KN to symbols, and apply low frequency screen and KN to vector areas. Table 2.1 summarizes the correspondence between each object type and its optimal halftone screen and color map.

<table>
<thead>
<tr>
<th>Object type</th>
<th>Raster</th>
<th>Symbol</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color code</td>
<td>Red</td>
<td>Blue</td>
<td>Green</td>
</tr>
<tr>
<td>Halftone Screen</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Color map</td>
<td>PN</td>
<td>KN</td>
<td>KN</td>
</tr>
</tbody>
</table>

In order to implement the object-oriented halftoning, an object map is absolutely needed. When printed using the PC, the document is usually encoded in a page description language (PDL), such as PostScript, Printer Command Language (PCL), and Portable Document Format (PDF), by the printer driver [44] [46]. The PDL specifies how each object on the page should appear in a higher level than the bitmap
Fig. 2.1. Illustration of the object type definition. (a) Example page content. (b) The corresponding object map (red: raster; blue: symbol; green: vector).
For example, text and graphics would be described in terms of abstract graphical entities such as line, circle etc., rather than in terms of device pixels [44]. In a typical printing process as illustrated in Figure 2.2, those vector digital information is then converted into a high-resolution raster image by the raster image processor (RIP) [47] [48]. The object type information can be extracted through this rasterization process as well. However, the object type information obtained may be incorrect due to the mislabeling occurred when the document is generated. Chen [37] et al proposed an object map correction algorithm which aimed at the most frequently occurred misclassification, vector area being labeled as raster by mistake, but the other types of misclassifications remain unsolved.

Fig. 2.2. The imaging pipeline (with object-oriented halftoning) for the laser electrophotographic (EP) printer.

Besides the incorrect object map, we may encounter a more thorny scenario where the object map is not available at all. Mobile devices are now gradually supplanting the PC as primary compute devices for a large fraction of the public. They have substantially all of the computing capabilities desired, including printing. Yet, mobile printing technique is different from the traditional driver-based approach for the PC. In order to avoid the heavy burden to maintain a plethora of print files and connection methods for each unique pair of mobile device and printer, the printer-specific driver is abandoned, and a single platform consensus enabling industry-wide compatibility is established [68]. Take AirPrint [69] for example, the consensus designed for operating systems, iOS and OS X. The image formats specified by AirPrint to send and accept
print jobs between the mobile device and printer are raw Joint Photographic Experts Group (JPEG), Unicode Transformation Format (URF), and PDF (by default). If the printer cannot process PDF, URF and raw JPEG would be considered in turn as fallback. However, URF and raw JPEG do not encode any object type information, so that the object map would not be generated in this case. Therefore, an algorithm that generates accurate object map from the raster input image directly would be greatly appreciated.

Various kinds of document segmentation and object classification algorithms have been developed, and have been utilized in many applications, such as object-oriented rendering [49], document retrieval [50] [51], document compression [53], and optical character recognition (OCR) [52] [57]. [58] and [59] provide review and evaluation on existing page segmentation techniques. However, an unique algorithm distinguished from most of current methods is in great need in the perspective of the specific purpose and restriction in our application.

First, the definition of our object types is different from traditional classification categories [54], where objects are generally classified into text, image, and background [55] [56]. It is the visual appearance of each object that drives the classification not its content. For example, the tea cup graphic in Figure 2.3(a) would be labeled as symbol instead of raster, because KN preserves the sharpness of the contour; and we are inclined to print the wall in the photo illustrated in Figure 2.3(c) with vector mode rather than raster mode, because high frequency screen would result in visual artifacts in areas as smooth as the wall.

Secondly, most page layout analysis algorithms classify a document into box-wise regions [60] [61], or even make decisions block-by-block [62]. However, classification decision need to be carefully drawn with respect to every pixel in our case. The boundary between two objects has to be perfectly cut along the contour. A slight shift of the boundary would lead to problems such as color mismatch, spurious edges, and "jaggies" [38].
Fig. 2.3. Two example objects and their optimal object maps (red: raster; blue: symbol; green: vector), where the object classification is different from traditional definition.
Thirdly, global methods which require simultaneous access to the entire page image and multiple visits to each pixel would be impossible to be applied for hardware implementation [63–66]. A hardware-friendly strip-based algorithm has access to image data only one strip at a time, and never revisits previously processed strips [74]. Also, the fewer passes the document is processed through, the more the algorithm would be appreciated. These restrictions on computational complexity and memory requirements trigger the exploration for efficient and effective segmentation approaches.

In this chapter, we propose an efficient object map generation algorithm (OMGA) based on the raster input image. It segments the document into objects defined in Table 2.1, so that the object oriented halftoning can be applied, and then improves the overall print quality. The algorithm first detects objects based on two hierarchical edge maps. Objects are then classified according to several representative features, such as the sharpness of contour and the smoothness of the interior. The class labels of the edge pixels need to be adjusted before integrated into the final object map. Our algorithm is designed to be hardware-friendly, and can be implemented within two passes through the input document.

The rest of this part is organized as follows. In Section 2.2, we introduce the new imaging pipeline which incorporates the OMGA with the object-oriented halftoning. Section 2.3 presents the detailed description of each step in the OMGA. Section 2.4 proposes a strip-based hardware implementation algorithm that can be implemented within two passes through the input document. In Section 2.5, we provide object map examples generated by the OMGA, and evaluate the performance of the algorithm. Finally, conclusions are drawn in Section 2.6.

2.2 The New Imaging Pipeline with OMGA

The imaging pipeline need to be modified when the OMGA is taken into consideration. Figure 2.4 shows the new imaging pipeline with the OMGA and the
object-oriented halftoning algorithm for the laser EP printers. The document to be printed is translated to the image format that can be recognized by the printer. When print with the PC, the translation is usually done by the printer driver and the document is expressed in the form of a PDL; but with mobile devices, such a format is specified by the printing protocol. The RIP further interprets the translated document and renders it as a continuous-tone RGB color space raster image. The corresponding object map can be obtained by applying our proposed OMGA to the continuous-tone RGB image. Then, with the object map, we are able to implement the object-oriented halftoning algorithm. First, the continuous-tone RGB image is transformed by the color channel conversion block into a continuous-tone CMYK image according to different color maps indicated by the object map. Then, still under the instruction of the object map, we apply different halftone screens to the continuous-tone CMYK image to generate the halftone CMKY image. The transition region between two adjacent halftone textures is blended to reduce the boundary artifacts, which is referred as the seamless halftoning approach [38]. Finally, such a halftone image is sent to the marking engine, and a hardcopy document is printed.

Fig. 2.4. The new imaging pipeline with OMGA and object-oriented halftoning for the laser electrophotographic (EP) printer.
2.3 Object Map Generation Algorithm

The object map generation algorithm (OMGA) is inspired by the fact that every object consists of one contour and one interior. It is the properties of the contour and the interior that determine which halftone screen and color map to be chosen. In other words, object type is determined by the features extracted from the contour and the interior. To isolate objects from each other, we detect edges.

The block diagram of the OMGA is shown in Figure 2.5. There are ten steps in total. First, a strong edge map which indicates the salient edges is generated based on the continues-tone RGB input image. Connected strong edge components (CSECs) are found and labeled in Step 2. CSECs are then classified in Step 3 into either symbol or non-symbol, because edges of raster and vector do not exhibit distinguished properties. On the other hand, we can also identify connected strong edge interior components (CSEICs) based on the strong edge map (Step 4). At this stage, all the symbol objects can be detected, but not necessarily the raster and vector objects. So with the classification results of CSECs, we are able to separate symbol CSEICs from non-symbol CSEICs (Step 5). In Step 6, the gaps between strong edges are fixed by the edge linking operator, which generates the (strong+weak) edge map. Now, all of the raster and vector objects are isolated. Similar as Step 4, connected (strong+weak)
Fig. 2.6. Illustration of the object map generation algorithm for the input example image, Step 1, and Step 2.
Fig. 2.7. Illustration of the object map generation algorithm from Step 3 to Step 6.
Fig. 2.8. Illustration of the object map generation algorithm from Step 7 to Step 10.
edge interior components (CSWEICs) are detected in Step 7, and then classified into vector or non-vector. Step 9 integrates the classification results of Step 3, 5, and 8. However, there may still be some misclassifications around symbol and raster objects. So in the final step, Step 10, we perform erosion operator to remove the extra contour of symbol and raster objects. The result of Step 10 is the final output of the OMGA, the object map of the raster input document. Figure 2.6, 2.7, and 2.8 illustrate an example of the input image and the output of each step. Table 2.2 lists the denotation and value of each parameter used in OMGA. Detailed information of each step is provided as follows.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Denotation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{S_{edge}}$</td>
<td>Threshold to detect strong edges</td>
<td>10000</td>
</tr>
<tr>
<td>$T_{W_{edge}}$</td>
<td>Threshold for weak edge points in edge linking</td>
<td>500</td>
</tr>
<tr>
<td>$T_{hwl}$</td>
<td>Lower boundary for height and width of symbol CSEC</td>
<td>10</td>
</tr>
<tr>
<td>$T_{hwu}$</td>
<td>Upper boundary for height and width of symbol CSEC</td>
<td>3000</td>
</tr>
<tr>
<td>$T_{hwrl}$</td>
<td>Lower boundary for height/width of symbol CSEC</td>
<td>0.005</td>
</tr>
<tr>
<td>$T_{hwru}$</td>
<td>Upper boundary for height/width of symbol CSEC</td>
<td>60</td>
</tr>
<tr>
<td>$T_{N_{CSEC}}$</td>
<td>Upper boundary for the number of pixels of symbol CSEC</td>
<td>10000</td>
</tr>
<tr>
<td>$T_{EM_{CSEC}}$</td>
<td>Lower boundary for the average value of edge magnitude of symbol CSEC</td>
<td>110000</td>
</tr>
<tr>
<td>$T_{L_{area}}$</td>
<td>Threshold to determine large or small area</td>
<td>1%</td>
</tr>
<tr>
<td>$T_{VR}$</td>
<td>Threshold for $r_{\text{rough}_{CSWEIC}}$ to differentiate between vector and non-vector</td>
<td>0.5</td>
</tr>
<tr>
<td>$T_{SR}$</td>
<td>Threshold for $r_{\text{rough}_{CSEIC}}$ to differentiate between symbol and non-symbol</td>
<td>0.1</td>
</tr>
</tbody>
</table>
2.3.1 Detect strong edges

The contour of an object retains salient information which is a concise and useful resource for computer vision or image processing tasks. In general, contour perception and edge detection approaches require complex computational effort, especially global information. Prevalent algorithms, such as Canny edge detector [70], Kirsch operator [71], and Laplacian of Gaussian [73], are usually too complicated to be implemented in strip-based manner. We utilize the Sobel operator [72] as the edge detection method to extract edge information.

Sobel operator approximates the derivatives in pixel values by the convolution of the image and two 3×3 kernels for horizontal and vertical directions respectively. Apply Sobel operator to each of the three channels of the input RGB image, and the horizontal and vertical derivative approximations are:

\[
G_{x,i} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \ast I_i \tag{2.1}
\]

and

\[
G_{y,i} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \ast I_i \tag{2.2}
\]

where \( i = R, G, B \), so that \( I_i \) is the corresponding channel of the input image; \( \ast \) denotes the 2-dimensional convolution operator. Edge magnitude value is then computed as:

\[
EM = \sqrt{\frac{1}{3} \sum_{i=R,G,B} (G_{x,i}^2 + G_{y,i}^2)} \tag{2.3}
\]

Edge magnitude represents the changes in gradient, therefore greater value indicates more salient edge. Only strong edges are of interest at this stage, because they are always related to the contours of objects. Strong edges help to separate regions
and detect objects. Pixels with edge magnitude value greater than $T_{S,\text{edge}}$ belong to strong edges, and the rest pixels are strong edge interiors. $T_{S,\text{edge}}$ is determined such that all symbol contours in the training pages are classified as strong edges. The strong edge map (SE) is defined as:

$$SE = \mathbb{1}(EM > T_{S,\text{edge}})$$  \hspace{1cm} (2.4)$$

where $\mathbb{1}(\cdot)$ is the indicator function. An example of the strong edge map can be found in Figure 2.6(c).

Figure 2.9 shows an example of the input image and its strong edge map where each pixel is presented as a small square. There are four objects illustrated with different colors in Figure 2.9(a). In Figure 2.9(b), white pixels are strong edge pixels, and black pixels are interiors of the objects according to the procedures aforementioned. The strong edges detected are usually wider than the contours. The extra width is eliminated in Step 10. Each interior is enclosed by a strong edge, and such a pair composes an object. But some objects correspond to a strong edge only, such as object 4.
2.3.2 Use connected component (CC) to find and label CSECs and CSE-ICs

In Step 2, we apply the connected component analysis (CCA) [74] to find contiguous subsets of the strong edge pixels, and then label these connected strong edge components (CSECs). The same as Step 2, connected strong edge interior components (CSEICs) are found and labeled by CCA in Step 4. Fast and efficient CC algorithms for the hardware-based embedded system have been proposed [75,76], and CCs can be computed using a single-pass algorithm.

Common choices for neighborhood in CCA are 4-neighbors and 8-neighbors. For a point $p$ with coordinates $[m,n]$, its 4-neighbors, $N_4(p)$, and 8-neighbors, $N_8(p)$, are defined respectively as

$$N_4(p) = \{[m,n-1],[m-1,n],[m+1,n],[m,n+1]\}.$$  \hspace{1cm} (2.5)

and

$$N_8(p) = \{[m-1,n-1],[m,n-1],[m+1,n-1],[m-1,n],$$

$$= [m+1,n],[m-1,n+1],[m,n+1],[m+1,n+1]\}. $$  \hspace{1cm} (2.6)

Two strong edge pixels are connected if they are from the 8-neighbors of each other, while strong edge interior pixels are connected based on the 4-neighbors. Such a strategy insures that each CSEIC is enclosed by only one CSEC, and each CSEIC with closed contour is well separated. For example, the input image with two objects and its strong edge map in Figure 2.10(a) and 2.10(b). As shown in Figure 2.10(c), with 8-neighbors, the strong edge pixels compose one integrated CSEC. But with 4-neighbors, as shown in Figure 2.10(d), the contour of object B is divided into two CSECs, so that the interior of the object is enclosed by two CSECs. Figure 2.10(e) and 2.10(f) illustrate the connectivity for interior pixels. With 4-neighbors, object B is separated from object A by the strong edge. However, with 8-neighbors, the two objects are merged together, even though object B has closed contour. Another
example of the results of Step 2 and Step 4 can be found in Figure 2.6(d) and Figure 2.7(b) respectively, where CSECs and CSEICs are shown in pseudo color.

2.3.3 Classify CSECs (symbol or non-symbol)

Symbol objects are printed with high frequency halftone screen and KN color map, because the contour of symbol is very sharp. The contours of raster and vector do not form a stereotype. Most likely, they are smooth and less sharper than symbol contours, but exceptions exist. Another distinction between the symbol and non-symbol objects is that symbols are always small areas, which restricts the size of symbol CSECs. Examples of symbol CSEC and non-symbol CSEC are given in Figure 2.11. We decide whether a CSEC is symbol using the following five criterions:

\[(a). \ T_{hwl} < height < T_{hwu} \land T_{hwl} < width < T_{hwu}\]
\[(b). \ T_{hwrl} < height/width < T_{hwru}\]
\[(c). \ N_{CSEC} < T_{N.CSEC}\]
\[(d). \ EM_{ave} > T_{EM.CSEC}\]
\[(e). \ (height < 3 \lor width < 3) \land N_{CSEC} < T_{S.CSEC}\]  

where \(\land\) and \(\lor\) are the logical conjunction and disjunction operators respectively; \(height\) is the number of rows between the first and last pixel of the CSEC in vertical direction, and \(width\) is the number of columns between the most left and the most right pixel of the CSEC in horizontal direction as shown in Figure 2.11(b). \(N_{CSEC}\) denotes the number of pixels of the CSEC. \(EM_{ave}\) is the average value of edge magnitude of this CSEC. The parameters used in the criterions are further described in Table 2.2.

Criterion (a), (b), and (c) constrain the size and shape of the CSEC. If the height and width are both less than \(T_{hwu}\), and the number of pixels of the CSEC is less than \(T_{N.CSEC}\) as well, then such a CSEC encloses a small area. Also objects that are too small are also eliminated from symbol class to avoid the noise. Criterion (b) insures
Fig. 2.10. Examples of CSECs and CSEICs with 4-neighbors and 8-neighbors.
Fig. 2.11. Examples of symbol CSEC and non-symbol CSEC.
that such an object is not elongated to much. $T_{hwrl}$ and $T_{hwru}$ are chosen to be loose constraints, since the shape of an object is not the dominant factor to determine the object type in our application. Criterion (d) controls the sharpness of the CSEC. The average edge magnitude value of symbol contour should be greater than $T_{EM,CSEC}$. Criterion (e) aims at tiny CSECs which usually occur when the symbol contour is blurred slightly as illustrated in Figure 2.11(b). If some tiny CSECs do not belong to symbol, they will be corrected in Step 10. A symbol CSEC satisfies criterions (a), (b), (c), and (d) simultaneously, or criterion (e):

$$
\{Symbol\; CSEC\} = \{CSEC | (\mathbb{1}(a) \land \mathbb{1}(b) \land \mathbb{1}(c) \land \mathbb{1}(d)) \lor \mathbb{1}(e)\} \quad (2.8)
$$

where $\mathbb{1}(\cdot)$ is the indicator function. Figure 2.7(a) shows the result of Step 3, the classified CSECs, where CSECs in blue are symbol CSECs, and CSECs in red are non-symbol. Symbol CSECs do not necessarily belong to symbol objects. Misclassified symbol CSECs can be corrected by Step 10.

### 2.3.4 Classify CSEICs (symbol or non-symbol)

In Step 5 we classify the CSEICs obtained from Step 4 into symbol and non-symbol. At this stage, symbol interiors are isolated by strong edges, but some raster and vector areas are still connected due to the gaps between strong edges. Those edge gaps will be fixed in Step 6. So we search for symbol interiors first.

The same as vectors, symbol interiors are smooth without intense color changes; however, symbols are distinguished from vectors by the size. Symbols are smaller objects compared with vectors. We can differentiate between symbol and raster by the roughness of their interiors. The high-frequency contents in raster leads to a much greater roughness ratio value. The roughness ratio of a CSEIC, $r_{rough,CSEIC}$, is defined as:

$$
r_{rough,CSEIC} = \frac{\sum_{[m,n] \in CSEIC} \mathbb{1}(EM[m,n] > T_{W,edge})}{N_{CSEIC}}
$$

(2.9)
where $N_{CSEIC}$ is the number of pixels of the CSEIC; $EM[m,n]$ is the edge magnitude value of the pixel with coordinates $[m,n]$. Furthermore, a CSEIC is not symbol if its contour CSEC is classified as non-symbol.

To summarize, we classified a CSEIC based on its size, roughness, and contour CSEC:

$$\{Symbol \ CSEIC\} = \{CSEIC \mid (1(N_{CSEIC} < T_{L,area} \cdot N_{image})$$
$$\land 1(\text{rough}_{CSEIC} < T_{SR})$$
$$\land 1(\text{contour} \ CSEC \in \text{symbol})\}$$

(2.10)

where $N_{image}$ is the number of total pixels in the input image. $T_{L,area}$ is a threshold that defines the large area. Currently, we choose 1% of the number of total pixels as the boundary between large and small areas. $T_{SR}$ is threshold for $r_{\text{rough},CSEIC}$ to differentiate between symbol and non-symbol. Detail information of $T_{L,area}$ and $T_{SR}$ is given in Table 2.2. Figure 2.7(c) demonstrates the classification result of CSEICs which derived from the results of Step 3 and Step 4.

### 2.3.5 Edge linking

As discussed in Section 2.3.1, symbol contours all belong to strong edges. However, as shown in Figure 2.12(c) and Figure 2.12(d), small pieces of the strong edges of vector or raster objects extracted from Step 1 might be missing, so that those CSECs do not necessarily produce closed object boundaries. In Step 6, the break strong edges which are denoted as weak edges are fixed by edge linking operator based on local knowledge.

The pixel where a strong edge terminated is referred to as the end point. The end points are potential locations where the gaps exist. Extracting the end points relies on the information of the 3×3 neighborhood. Assuming the pixel under investigated has coordinates $[m,n]$, check if it satisfies the following three criterions for the end point:
Fig. 2.12. Examples of the results of Step 6, edge linking.
(a). $SE[m, n] = 1$

(b). $SE_{\text{sum}}[m, n] < 6$, $SE_{\text{sum}}[m, n] = \sum_{k=-1}^{1} \sum_{l=-1}^{1} SE[m + k, n + l]$

(c). At least one of the four pairs of diagonal pixels in the 3x3 neighborhood are different (Strong edge pixel or interior pixel) \( (2.11) \)

where $SE$ is the strong edge map. Criterion (a) states that an end point is a strong edge pixel. In criterion (b), the number of strong edge pixels in the 3x3 neighborhood should be less than 6; otherwise, such a point is surrounded by quite a few strong edge points, and it is a common scenario more likely to happen in the middle of the strong edges, not the terminal. Criterion (c) ensures that the strong edge can be extended at the point under investigation. Figure 2.13 illustrates the four pairs of diagonal pixels in the 3x3 neighborhood by red. According to the three criterions, Figure 2.14 gives four examples for the end points and four examples for the non-end points.

Explore starting from the end points, so that the missing pieces of strong edges are fixed. Interior pixels around the end point are relabeled as weak edge pixels if they satisfy the linking criterion.
Fig. 2.14. Examples of end points and non-end points. The center pixel is investigated by the three end point criterions. White: strong edge pixel; Black: strong edge interior pixel.
**Linking criterion**: for each of the four pairs of diagonal pixels in the $3 \times 3$ neighborhood, if only one pixel is a strong edge point, then check if the edge magnitude of the other one is greater than $T_{W_{\text{edge}}}$. where $T_{W_{\text{edge}}}$ is a threshold value smaller than $T_{S_{\text{edge}}}$. For example, corresponding to the four end points examples in Figure 2.14(a), the red pixels in Figure 2.15 are potential gap points. Apply the linking criterion to them, and link them if their edge magnitude values are greater than $T_{W_{\text{edge}}}$.

There is no distinction regarding to the capacity of strong edge pixels and weak edge pixels. They are equally treated in the end point criterions and linking criterion. Edges can be extended from weak edge points as well. Assuming $T_{W_{\text{edge}}}$ equals 50, Figure 2.16 displays the procedures of edge linking step by step. Pixels are processed in raster scan order in this example. Figure 2.12(e) and Figure 2.12(f) show the results of edge linking for the given two input images. Breakpoints of the strong edges become connected after edge linking, so that more objects are separated out. However, some spurious edges and noise are also detected due to edge linking. The ice crystal or branches in Figure 2.12(f) can be eliminated by Step 10, Erode around symbols and raster. Because of the spurious edges and noise, symbol edges are extracted before edge linking, so that they are not mixed with the weak edges.
Fig. 2.16. Procedures of edge linking step by step. Assume $T_{W,\text{edge}}$ equals 50. Pixels are labeled with the edge magnitude value, and are processed in raster scan order.
2.3.6 Use CC to find and label CSWEIC

As more objects are detected after edge linking, in Step 7, we apply CC to find and label the connected (strong+weak) edge interior components (CSWEICs) similarly as Step 4. In order to separate each CSWEIC with closed contour, 4-neighbors is chosen instead of 8-neighbors when define the connectivity. Figure 2.8(a) shows an example of the labeled CSWEICs in pseudo color.

2.3.7 Classify CSWEICs (vector or non-vector)

Edge linking fixes the boundaries for vector and raster. Isolated vector and raster objects are included in the CSWEIC set. Vector interiors are distinguished from symbols by the size, and raster by the roughness, as discussed in Section 2.3.4. Classify the CSWEICs into vector and non-vector according to the aforementioned rules. The roughness ratio of a CSWEIC, \( r_{\text{rough,CSWEIC}} \), is defined as:

\[
r_{\text{rough,CSWEIC}} = \frac{\sum_{[m,n] \in \text{CSWEIC}} \mathbb{1}(EM[m,n] > T_{W\text{edge}})}{N_{\text{CSWEIC}}} \quad (2.12)
\]

where \( N_{\text{CSWEIC}} \) is the number of pixels of the CSWEIC; \( EM[m,n] \) is the edge magnitude value of the pixel with coordinates \([m,n]\). The classification rule can be summarized as:

\[
\{\text{Vector CSWEIC}\} = \{\text{CSWEIC} \mid (\mathbb{1}(N_{\text{CSWEIC}} > T_{L\text{area}} \cdot N_{\text{image}}) \land \mathbb{1}(r_{\text{rough,CSWEIC}} < T_{VR}))\} \quad (2.13)
\]

Again, \( N_{\text{image}} \) is the number of total pixels in the input image. \( T_{L\text{area}} \) is a threshold that defines the large area, currently chosen as 1%. \( T_{VR} \) is the threshold for \( r_{\text{rough,CSWEIC}} \) to differentiate between vector and non-vector. Detail information of \( T_{L\text{area}} \) and \( T_{VR} \) is given in Table 2.2. Figure 2.8(b) demonstrates the classification result of CSWEICs where green represents vector CSWEICs.
Fig. 2.17. Symbols and vector areas obtained from Step 3, Step 5 and Step 8. Green: vector; Blue: symbols; Black: non-symbol or non-vector.
2.3.8 Integrate classifications

Until Step 8, some parts of the input document are classified as symbol and some are labeled as vector. Since there are only three classes, symbol, vector and raster, the rest of the page belongs to raster. In Step 9, we integrate the classification from previous steps and generate a draft object map.

Areas labeled as symbol are inherited from the results of Step 3 and Step 5, and areas labeled as vector are obtained from Step 8. Combine the results from Step 3, Step 5, and Step 8. An example of the combination result for the input document given in Figure 2.6 is shown in Figure 2.17. Relabel areas other than vector and symbol as raster. The result is illustrated in Figure 2.8(c).

2.3.9 Erode around raster and symbol

There are several problems remain unsolved with the draft object map of Step 9. First of all, as discussed in Section 2.3.1, strong edges detected are wider than the contour. Since strong edges are either classified as symbol in Step 3, or relabeled as raster in Step 9, we need to eliminate the extra width for symbol and raster. Secondly, some CSECs are misclassified as symbol in Step 3. The classification of CSECs does not take the object type of the corresponding interior into consideration; consequently, the contours of raster and vector objects may be labeled as symbol if the edges are shape enough. Correction of such kind of mistakes can be done with the erosion operator. Thirdly, edge linking produces the ice crystal and branches shapes which are the results of noise as shown in 2.12(f).

To address the three problems mentioned above, we apply the erosion operator to the result of Step 9, the draft object map. The three object types are sorted in priority order from the highest level to the lowest as:

\[ \text{symbol} > \text{raster} > \text{vector}. \]  

(2.14)
Fig. 2.18. Illustration of the misclassified symbol edges and the corrected object map with erosion.
Fig. 2.19. Illustration of the ice crystal and branches shapes and the smoothed object map after erosion.
For every pixel, if one or more neighbor pixels in the 3×3 window centered at that pixel belong to classes with lower priorities, relabel it to the class with the lowest priority. The structure element, a 3×3 window, corresponds to the mask size of the Sobel operator, which is also a 3×3, so that the extra width of the edges can be exactly removed. For the second issue listed above, the misclassified symbol edges can be eroded by its interior which is prioritized with lower level, since symbol is the highest level. Figure 2.18 shows an example where the boundary between a raster object and a vector object is labeled as symbol by mistake at first, but then corrected by the erosion operator. Furthermore, the erosion operator removes any ice crystal and branches shapes narrower than 3-pixel wide. As illustrated in Figure 2.19, the jagged edges are smoothed after being eroded.

2.4 Hardware implementation

The OMGA is designed to be hardware-friendly. Each of the ten steps in the procedure utilizes only the local information in the 3×3 neighborhood, which is necessary for strip-based implementation.

Also, the OMGA can be implemented within as few as two passes through the input document. In the first pass, the input document is processed in raster scanner order from top to the bottom. When encounter an unprocessed pixel, we first calculate the edge magnitude value, determine if it is a strong edge pixel, and then use the CCA to connect it to either strong edge component or strong edge interior component. Meanwhile, we can also perform the edge linking procedure, so that the CSWEICs are extracted along with CSECs and CSEICs. We can store the properties of these three kinds of components, such as the number of pixels, height, width, and roughness ratio, when we process them. The maximum number of each kind of components under processed is half of the number of pixels in a row of the input document. After the last pixel of a component is processed, the class type of the component can be determined. Label the last pixel with the classification of the component. In the
second pass, we process the document from bottom to the top. We would encounter the last pixel of a component first, and then extend the classification of the last pixel to the entire component as processing towards the top. At the same time, the erosion operator is implemented on pixels being labeled.

2.5 Experimental Results

The object map generation algorithm is tested on an image library consisting of 92 RGB files which cover a variety of page contents, such as article, travel brochures, magazines, advertisement, and posters. These test pages are obtained from the Purdue University e-Archives and Hewlett-Packard Company. The ideal object map is generated so that each object on the page is correctly classified into raster, vector, or symbol categories.

Apply the OMGA to the image library. Figure 2.20 shows an example of Page 2 in the image library with its ideal object map and the object map generated by the OMGA. Almost every object on the document is classified exactly the same as the ground truth, except that the two raster areas are slightly incomplete by the boundary. However, the two missing parts are white regions which are connected with the white background. Such a flaw would not affect the print quality, because white regions are not printed with toner in the process. Figure 2.21 shows another example of the result of the OMGA, the Page 57 in the image library. Text are mixed with the background picture in this case, which makes it very challenging for the object detection and classification. However, the OMGA yields an excellent performance by generating an identical object map as the ground truth. The text on the page is extracted accurately without being blended with the background. The Object map of Page 63 is illustrated in Figure 2.22. The five raster areas in the middle are blurred on the contours, and the boundaries fade gradually to the background. Thanks to the edge linking operator, the boundaries are extracted continuously without gaps. As
Fig. 2.20. Object map generated by OMGA for Page 2 in the image library.
Fig. 2.21. Object map generated by OMGA for Page 57 in the image library.
Fig. 2.22. Object map generated by OMGA for Page 63 in the image library.
a result, the five pictures are separated from the background without missing parts, except the white regions connected to the background.

To better evaluate the algorithm, we now provide statistics of the classification performance. Quantify the performance into three levels rating as Excellent, Satisfactory, and Unsatisfactory. If more than 98% area of the entire page is classified correctly according to the ground truth, the object map would be an Excellent result; if the correct area is less than 98%, but more than 80% of the entire page, such a classification result belongs to Satisfactory level; if more than 20% of a page is classified with mistake, the object map would be unsatisfactory to the user. Apply such a rating rule to the results of the 92 documents in the image library. Table 2.3 lists the performance evaluation statistics of the OMGA. 92.4% of the results are above the Satisfactory level, among which 52.2% are Excellent meaning that minor errors occurred on the object maps. Overall, the OMGA achieves accurate classification results by yielding the low unsatisfactory rate.

Table 2.3
Classification performance of the OMGA when applied to the image library.

<table>
<thead>
<tr>
<th>Performance level</th>
<th>Denotation</th>
<th>Number of documents (the percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excellent</td>
<td>&gt; 98% of the area is correct</td>
<td>48 (52.2%)</td>
</tr>
<tr>
<td>Satisfactory</td>
<td>&gt; 80% of the area is correct</td>
<td>37 (40.2%)</td>
</tr>
<tr>
<td>Unsatisfactory</td>
<td>&lt; 80% of the area is correct</td>
<td>7 (7.6%)</td>
</tr>
</tbody>
</table>

2.6 Conclusion

In this work, we present a hardware-friendly and low-complexity object map generation algorithm (OMGA), and use it for the object-oriented halftoning to improve print quality. Three objects types (raster, vector, and symbol) are defined based on their unique characteristics, and different halftone screens and color maps are apply
to them when print. The OMGA classifies the objects in the page content and generates the object map, so that the object-oriented halftoning can be applied. The new imaging pipeline with the OMGA for laser EP printers is introduced.

In the OMGA, the edge information is extracted first, and objects are separated and detected according to the strong and weak edges. Classification of each connected component is determined according to their properties, such as the number of pixels and smoothness. Since the edges detected are usually wider than the contours and noise existed due to the edge linking step, we modify the draft object map with erosion operator. The ten steps of the OMGA can be implemented within two passes through the input document. It is proved to be hardware-friendly by the strip-based implementation.

The OMGA is evaluated on the image library of 92 RGB document with a variety of image contents. Three examples of the object map generated are given and compared with the ground truth. The experimental results demonstrate the good performance of our algorithm by showing that the OMGA achieves satisfactory results for 92.4% of the pages in the image library. The object maps of 52.2% of image library quality the Excellent level which means that 98% of the area is classified correctly by the OMGA.
3. CONTRIBUTIONS OF THIS DISSERTATION

3.1 Contribution of Part 1

In Part 1, we introduced a two-stage toner usage predictor for laser, electrophotographic printers. We illustrated how to construct each of the two stages in the predictor. The predictor was evaluated on 35 arbitrarily chosen test images, and the results surpassed those of the pixel-counting algorithm.

Recapitulating, this work had made the following contributions.

• We introduced a novel toner usage estimation algorithm which took into account the spatial arrangement of the pixels over which the aggregate gray value was distributed. Toner usage estimation method based on this fact achieves better performance than those considering the gray value linearly.

• The two-stage predictor enhanced customer experience when use laser, electrophotographic printers by offering competitive toner usage estimation accuracy, and gave better control to the remained toner in the cartridges.

• We proposed two alternative substep prediction strategies for stage one, the absorptance predictor. The 3×3 absorptance predictor and the weighted hamming distance predictor satisfy the requirement for either computation expense and accuracy.

• In contrast to prior methods, our approach picked ranges of absorptance value that highly correlated with toner assumption using LASSO operator. Remove the redundant gray value information increased the stability and consistency of the predictor.
3.2 Contribution of Part 2

In Part 2, we presented a hardware-friendly and low-complexity object map generation algorithm. It performed page content classification based on three object types: raster, vector, and symbol. Edge information and the properties of the interiors are used to determine the class of each object.

The contributions of Part 2 can be summarized as followed.

- An new image pipeline which incorporated the OMGA and the objected-oriented halftoning method was proposed. The OMGA generated object maps which were utilized in the objected-oriented halftoning. The image quality could be greatly improved using this pipeline.

- The OMGA is excellent in respect to the classification performance. It yields a classification accuracy of 92.4% on the test image library which ensured the implementation of the object-oriented halftoning algorithm.

- Different with transitional page content classification methods, the OMGA defined object types according to the sharpness of the edges and the smoothness of the interiors other than their contents. The new definitions matched the goal of the object-oriented halftoning, therefore, leaded to better performance of the whole pipeline.

- The OMGA is hardware dedicated. It used local information in the 3×3 only, so that strip-based implementation was achieved. The algorithm is designed to be hardware-friendly and low-complexity. The hardware implementation scheme was illustrated in Part 2.

- The classification result was addressed to each individual pixel, in contrast to most page layout analysis algorithms where a document is classified into box-wise regions. The boundaries between two objects were perfectly cut along the contours. It cast light on applications where objects were extracted completely in the object map.
LIST OF REFERENCES


VITA

Lu Wang received her B.S. degree in Electronic Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2010. She is currently pursuing her Ph.D. degree in Electrical and Computer Engineering at Purdue University. Her current research interests include digital image processing, color science, and digital imaging.