5-2018

Digital Image Segmentation and On–line Print Quality Diagnostics

Zuguang Xiao
Purdue University

DIGITAL IMAGE SEGMENTATION AND ON–LINE PRINT QUALITY DIAGNOSTICS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Zuguang Xiao

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Jan P. Allebach

School of Electrical and Computer Engineering

Dr. Amy Reibman

School of Electrical and Computer Engineering

Dr. George T.-C. Chiu

School of Mechanical Engineering

Dr. Fengqing Maggie Zhu

School of Electrical and Computer Engineering

Approved by:

Dr. Venkataramanan R. Balakrishnan

Head of Electrical and Computer Engineering

To my beloved parents, and my wife Yiding.

ACKNOWLEDGMENTS

First I want to express my sincere gratitude to my major Professor Jan Allebach. I first met him in spring 2012 in his image processing class when I was an undergraduate student at Purdue University. He is a knowledgeable and humorous man, and I started to have a keen interest in image processing since then. I stayed at Purdue University for Ph.D. in 2013 and chose him as my major professor without a second thought. During my Ph.D., I worked on several projects under his guidance and financial support. His insight, knowledge, and patience have helped me to become who I am today. I feel very much privileged to able to learn from such an honorable and respectful man like him. I'm also grateful to our HP partners, Eric Maggard and Mark Shaw. They always provide useful feedbacks during our biweekly meetings. Finally, I thank for the companion of my beloved wife Yiding and the mental support of my parents. When I was stressful and in adversity, they were always there for me.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

SYMBOLS

Ts_{edge}	strong edge threshold.
Tw_{edge}	weak edge threshold
G_x, G_y	Sobel operator
G	Gaussian filter
I_x	horizontal gradient image
I_y	vertical gradient image
$*$	2D convolution
k	sensitivity factor for Harris corners
Ptm	feature points coordinates of a master image
Pts	feature points coordinates of a scanned image
θ	skew angle
dx	translation offset of x-axis
dx	translation offset of y-axis
ΔE	the Euclidean distance between two points in CIELab color space.
ΔE_{char}	the average ΔE between two text characters.
$norm$	L2 norm
$Dir_{CMYKfade}$	fade directions of the most faded colors
$Dir_{scanned}$	actual fade direction
$Cart_{predict}$	predicated depleted cartridge
$pCMYK$	The printer CMYK color space based on a color LUT.
$sCMY$	The CMY color space that is directly inverted from the sRGB color space.
o	used as a subscript to indicate it is master.

ABBREVIATIONS

RIP	raster image processor
PQ	print quality
OPC	organic photo conductor
ASIC	application specific integrated circuit
PDL	page description language
EP	electrophotographic
CCL	connected component labeling
CCA	connected component analysis
SEM	strong edge map
CSEC	connected strong edge component
NSEM	non-strong edge map
CNSEC	connected non-strong edge component
NEM	non-edge map
CNEC	connected non edge component
EM	edge magnitude
GUI	graphical user interface
CART	cartridge
SSD	sum of squared differences
LUT	look-up-table
PPI	pixel-per-inch
LPI	line-per-inch
CSF	corner strength function
ISO	International Organization for Standardization
RANSAC	random sample consensus

MLESAC	maximum likelihood estimation sample consensus
PCE	pixel counting estimation

ABSTRACT

Xiao, Zuguang Ph.D., Purdue University, May 2018. Digital Image Segmentation and On–line Print Quality Diagnostics. Major Professor: Jan P. Allebach.

During the electrophotographic (EP) process for a modern laser printer, object-oriented halftoning is sometimes used which renders an input raster page with different halftone screen frequencies according to an object map; this approach can reduce the print artifacts for the smooth areas as well as preserve the fine details of a page. Object map can be directly extracted from the page description language (PDL), but most of the time, it is not correctly generated. For the first part of this thesis, we introduce a new object generation algorithm that generates an object map from scratch purely based on a raster image. The algorithm is intended for ASIC application. To achieve hardware friendliness and memory efficiency, the algorithm only buffers two strips of an image at a time for processing. A novel two-pass connected component algorithm is designed that runs through all the pixels in raster order, collect features and classify components on the fly, and recycle unused components to save memories for future strips. The algorithm is finally implemented as a C program. For 10 test pages, with the similar quality of object maps generated, the number of connected components used can be reduced by over 97% on average compared to the classic two-pass connected component which buffers a whole page of pixels. The novelty of the connected component algorithm used here for document segmentation can also be potentially used for wide variety of other applications.

The second part of the thesis proposes a new way to diagnose print quality. Compared to the traditional diagnostics of print quality which prints a specially designed test page to be examined by an expert or against a user manual, our proposed system could automatically diagnose a customer's printer without any human interference.

The system relies on scanning printouts from user's printer. Print defects such as banding, streaking, etc. will be reflected on its scanned page and can be captured by comparing to its master image; the master image is the digitally generated original from which the page is printed. Once the print quality drops below a specified acceptance criteria level, the system can notify a user of the presence of print quality issues. Among so many print defects, color fading – caused by the low toner in the cartridge – is the focus of this work. Our image processing pipeline first uses a feature based image registration algorithm to align the scanned page with the master page spatially and then calculates the color difference of different color clusters between the scanned page and the master page. At last, it will predict which cartridge is depleted.

1. INTRODUCTION

1.1 Laser Printer Mechanism

This thesis aims to address print quality (PQ) issues for laser printers. When a print job is issued to a laser printer, electrophotographic (EP) process is taking place. The EP process commonly involves six steps [1] which are illustrated in Figure 1.1. In the first step of an EP process, the charge roller contacts the OPC (Organic Photo Conductor) drum to uniformly charges the drum with negative potential. However, at the start, there could be leftover charges on the OPC drum from the previous image. So, the OPC drum first has to be exposed to a LED source to remove any residual electrical charges from the drum surface. During the second step, a laser beam comes in and strikes the surface of the OPC. The areas on the drum that are exposed to the laser will be where the image is formed; this image is referred to as latent image which is invisible to human eyes. The negative charges in these areas are neutralized to get ready to accept toners. In step three, toners on the developer roller are negatively charged first, then pressed on the OPC drum. The toners will adhere to the electrostatic neutral areas on OPC drum that have been struck by the laser beam, and repelled from the areas that are also negatively charged. After this process, the latent image becomes visible on the surface of the drum. A sheet of paper (or other medium like plastic) is rolled between the transfer roller and the OPC drum in step four. The back of the paper is charged with positive charge first by the transfer roller. Then the toners on the OPC drum which carry opposite charges are attracted and transferred onto the paper. The transferred image on the paper is still floated on the paper. To create the permanent image, the paper passes through heated, pressurized rollers to melt the toner onto the page; this is step five. Finally, in step six, the cleaning blade removes residual toners from the surface of the OPC

drum to prepare it for the next image. The waste toners are recycled back to the cartridge.

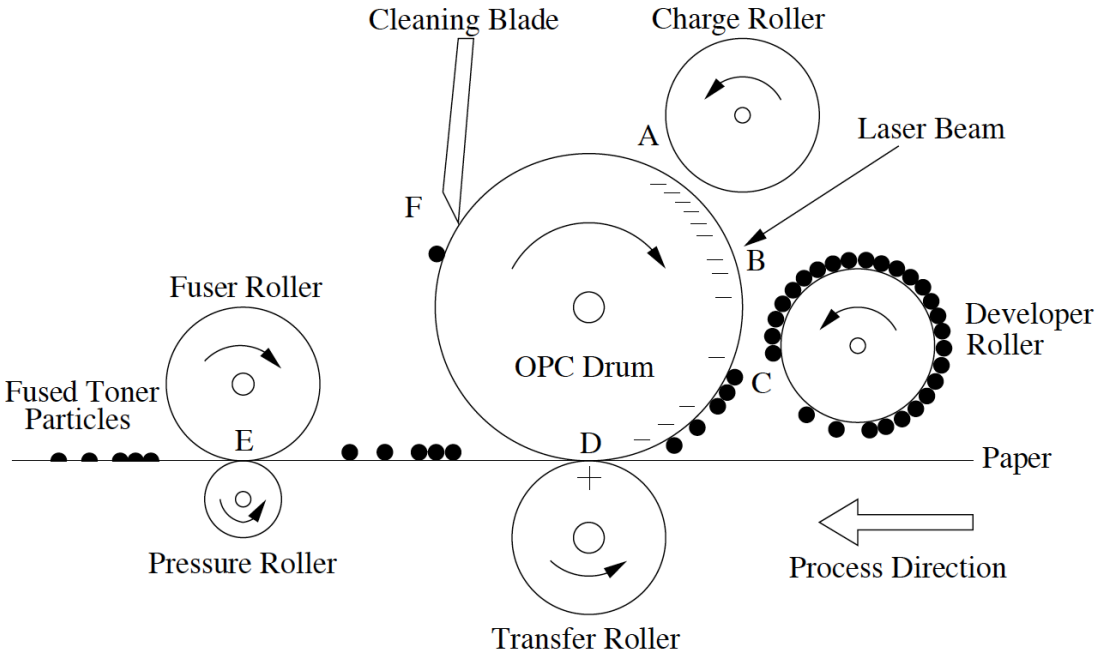


Fig. 1.1. A common laser EP printer mechanism involves: (A) charging, (B) exposure, (C) development, (D) transfer, (E) fusing, (F) cleaning [2].

1.2 PQ Diagnostics

During the EP process, when the operation of a component is non-ideal, or when the component is damaged or contaminated, PQ defects can be caused [3]. There are many commonly seen print defects, such as banding, streaking, ghosting, fading, etc. The appearance of one print defect indicates the abnormal operation of a particular component in the printer. For example, banding is caused when the OPC drum varies its angular velocity with time [4] [5], whereas streaking can be caused by non-uniformity development and transfer of toner particles [6], and the common cause of

ghosting is the OPC drum or the fuser has remained residual toner particles [7] [8]. A challenging problem for customers whose use their printers in their daily life is to diagnose these PQ defects when they appear so that customers can replace the damaged parts. Usually, customers need to resort to the printer user manual or call customer support for help. It can be tough for customers to communicate since they may not know the technical lexicon to describe their problems [9]. To help customers better identify the PQ issues, many printer manufacturers have specially designed test pages stored in the printer memory. These test pages can be easily printed by pushing a button on the printer. Customers can either compare the printed test pages against user manuals or even use some web-based troubleshooting tools to solve the PQ issues by themselves [10] [11] [12] [13]. Most of these test pages are constant-tone, and if there is any PQ issue with the printer, the corresponding print defect will be visibly showing on these printed test pages. Figure 1.2 shows a cropped image of a test page that is provided by our HP partners. A test page with good quality of PQ should have a uniform color. However, We can see that there is a sharp vertical streak across the image. The process direction of this page is parallel to the streak line. Sharp roller bands [14] have almost the same appearance as sharp streaks, and the difference is that bands are perpendicular to the page process direction.

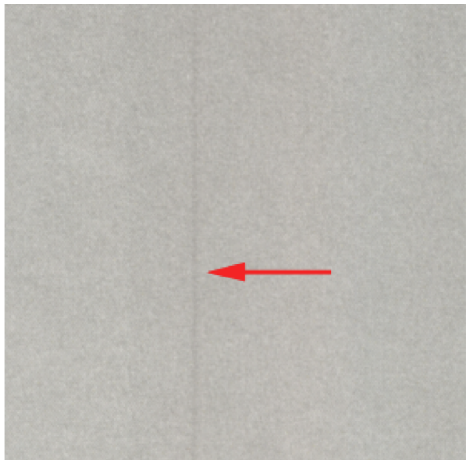


Fig. 1.2. A cropped test page with a streak on it. The paper process direction is along the vertical axis.

1.3 Fading Defect and Detection

Another type of print defect is fading which happens when the cartridge is low on toner. Because there are not enough toners transferred onto the paper from the OPC drum, the printed page may look washed out for black and white print jobs. For color print jobs, a page requires specific amounts of toners from each cyan, magenta, yellow and black cartridge. When one cartridge cannot supply sufficient toner, the reproduced color will be entirely distorted. Even though a page can become faded in some other situations, for example, a printed page that is exposed to different lighting conditions for a long time can exhibit different degrees of fading [15], this work is more focused on the fading caused during the EP process. Compared to banding and streaking defects, fade is not directional, and it is a large area defect. Consequently, the PQ can be severely degraded by fading. Therefore, fading defect detection will be the primary effort of this thesis.

In ISO/IEC 19798:2007 (Method for the determination of toner cartridge yield for colour printers and multi-function devices that contain printer components), fading is defined as a phenomenon in which noticeable reduction in density (increase in light-

ness) uniformity in the bars around sides of the diagnostic page occurs. An example of such diagnostic page is Figure 1.3, where the faded areas are marked by the red arrows. To determine if a diagnostic page is faded or not, a psychophysical experiment is needed. The psychophysical experiment requires imaging scientists/engineers to examine a sequence of printed diagnostics pages; the details of the psychophysical experiment is described in the ISO/IEC 19798:2007 and Yan [16]. In Yan's work, to avoid the costly and time consuming visual examinations by the experts, a machine learning based algorithm is proposed to automate these analysis by predicting the judgments of these expert observers on those diagnostics pages. Ju [17] also developed an algorithm to detect text fading on typical text documents instead of those diagnostics pages. However, her approach requires fiducial marks on the pages so that the faded page can be spatially aligned with the digital originals by using those fiducial marks. The aligned images are then compared.

Since fading defect is related to cartridge conditions, most of the modern laser printers have built-in optical sensors that can monitor the toner levels of the printers. However, such direct measurement can be inaccurate – it is typically only capable of measuring the reporting toner levels in coarse 20% increments, i.e., only toner levels of 100%, 80%, 60%, 40%, and 20% can be detected [18]. What's more, these numbers do not give us any information about how many more pages are we able to print before a page becomes faded. Cartridges will be wasted if they are changed too early. Another problem is that such sensor typically does continually monitor the output images as they are produced. This can result in, for example, during large printing jobs, a low toner condition can go undetected until a significant amount of resource (papers, toners, time and electricity) have been wasted due to the unacceptable PQ [19]. Some pixel counting approaches [20] [21] can yield more accurate results of estimating toner usage of the pages to be printed. Nevertheless, users' decisions of acceptance or fail is made on the printed pages only, and these methods do not address the PQ of the printed pages at all.

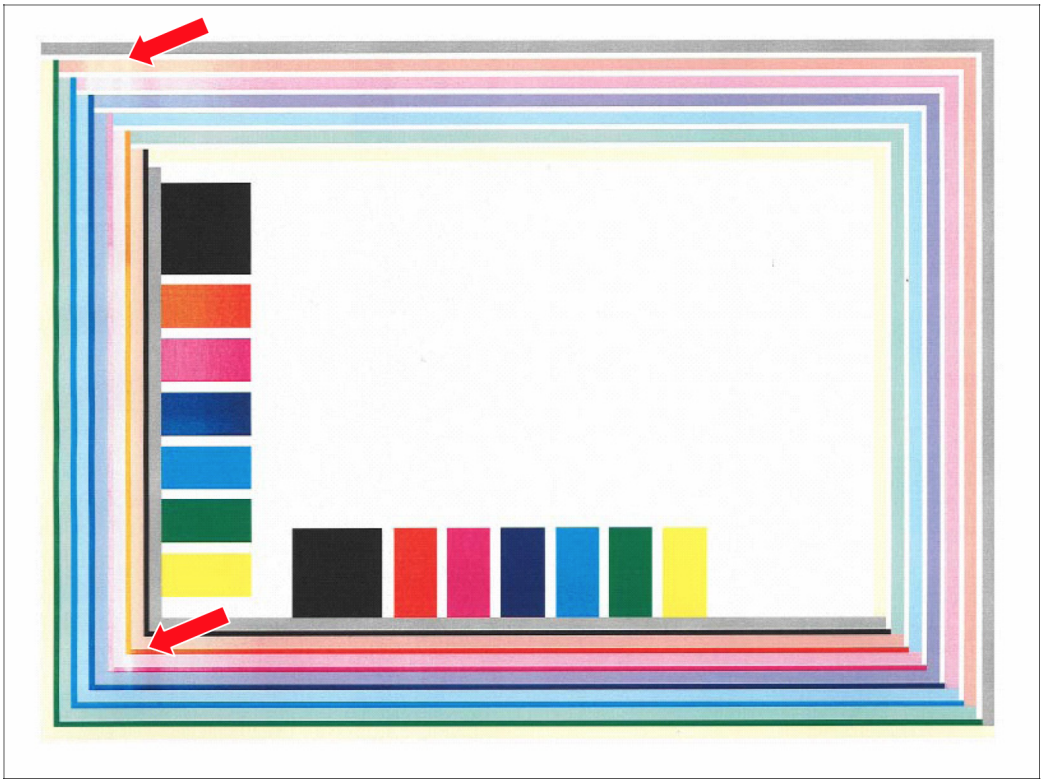


Fig. 1.3. Examples of fade from ISO/IEC 19798:2007(E).

1.4 Printer Imaging Pipeline

Documents sent for printing are typically described using a page description language (PDL), such as PostScript, which is one of the most noted page description languages. These files contain a sequential list of commands, and several operations have to be performed before these files can be printed [22]. A typical imaging pipeline for a laser printer is shown in Figure 1.4. When a print job is issued, the printer driver first translates a user-created digital document into a PDL. The PDL is device independent that describes the structure of a printed page and how each component should be rendered. In general, a printed page is composed of three basic primitives, which are text, geometric figures, and sections of photographs. The created PDL file is next transmitted to a printer buffer [23] [24]. The received PDL file later is raster-

ized by a raster image processor (RIP) which generates a page image; this image is also referred to as a raster image, and each line of the image is a raster line. The page image is continuous toner of an RGB or a gray bitmap. For some complex pages, the RIP requires long times to process, which can become the bottleneck to the overall print speed. Therefore, some RIP tasks are performed on host computers, which have much more processing power than the printers. Since the bitmap is too large, it has to be compressed before being transmitted to the printer, and the compressed raster image is stored in the raster memory of a printer [25]. The raster image which is continuous tone is then converted to a halftone image through a delicate sequence of steps which involve color management and halftoning. Each pixel of the halftone image can be encoded with one bit (for frequency modulation halftone), which turns on or off of a laser beam at a pixel location on the OPC drum during the EP process.

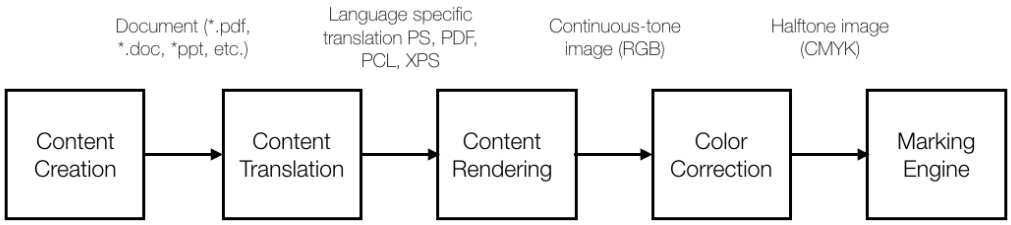


Fig. 1.4. A typical image processing pipeline for laser printer.

1.5 Object-Oriented Halftoning

The spatial frequency of halftone can be represented by line per inch (lpi), and researchers have found that lowering the halftone screen frequency can reduce the visible artifacts in the smooth areas due to the unstable operation of the EP process. But the halftone texture can become more apparent in the detail areas at the same time [26]. One solution proposed by Park et al. [27] is to apply high-frequency screens to the detail areas and low-frequency screens to the smooth areas, which is referred to

as object-oriented halftoning. This approach relies on an object map, which divides a document into the symbol, raster, and vector objects, with symbol and raster objects considered as detail areas that should be rendered with high-frequency screens, and vector objects as smooth areas that should be rendered with low-frequency screens. Besides the halftone screen frequency, the choice of color maps which transform a raster image from sRGB to printer CMYK color space also plays a role in the final PQ [28]. With two types of color maps considered here – process neutral (PN) and K-only process (KN), it is suggested that PN should be applied to the raster objects, and KN to the symbol and vector objects [29].

The challenge for the object-oriented halftoning approach is to obtain a correct object map. It has been found that the object map extracted from the PDL can be incorrect [28]. Chen ea al [28] proposed an algorithm to correct the object maps generated from the PDL. However, for some printers, they cannot process PDF files, and the RIP task is performed on the host computers. When users print a file from a mobile device which does not have a RIP, the only communication between the mobile device and the printer will be JPEG or URF files, and there is no way to extract object maps from these files [29] directly. In Wang's Ph.D. thesis [29], she proposed an algorithm to generate an object map from scratch based on a raster image instead of a PDL file. However, the algorithm is quite complicated and is not suitable for hardware implementation.

1.6 Summary

As a summary of the introductory chapter, there are two main problems introduced here. First is how to effectively and efficiently diagnose PQ issues, especially for fading defect. Second, a hardware-friendly algorithm for generating an object map based on a raster image is needed. This thesis aims to address these problems, and it is organized as follows:

- Chapter 2 propose a hardware friendly and memory efficient algorithm for generating an object map based on a raster image.
- Chapter 3 describe an on-line PQ diagnostic system that does not need any human interference.
- Chapter 5 introduce an image registration algorithm that aligns a scanned image with a raster image for PQ troubleshooting.
- Chapter 6 develop a text fading detection algorithm.
- Chapter 7 Extend the algorithm to allow it to detect fading in non-text regions as well.

This work builds on recent image quality work focused on printer and scanner products that was conducted in our laboratory, and which addressed assessment of page non-uniformity [30] [31] [32] [33] [34] [35], fine-pitching banding [36] [37] [38] [14] [39], ghosting [40], local defects [41] [42], fading [16] [17], scanner MTF [43], and scanner motion quality [44].

2. DIGITAL IMAGE SEGMENTATION [1]

2.1 Introduction

The electrophotographic(EP) process, which is widely used in imaging systems such as laser printers, is susceptible to print artifacts if we render the smooth areas of the image with high-frequency halftone screens. However, applying low-frequency halftone screens over the whole page will restrict the ability to render the fine details [28]. The solution proposed by Park et al [27] is to apply different frequency of screens to different parts of the page – also referred to as object-oriented halftoning. But it requires a correct object map to be generated. With miscellaneous segmented objects in a given image, an object map will classify all the image objects into three categories: raster (pictures or photos), vector (background and gradient) and symbol (symbols and texts). Raster and symbol objects considered as high-frequency objects and vector objects as low-frequency objects. An overall improvement of the print quality can be achieved if symbol and raster objects are rendered with high-frequency screens, and vector objects with low-frequency screens. Although the object map can be extracted from the page description language (PDL) directly, some components may not be correctly classified [28]. To obtain a correct object map from the page image, not the PDL, a new object map generation algorithm has been developed from our lab [29]. This algorithm uses a classic two-pass connected component labeling (CCL) process which requires buffering a whole page of pixels, and it is entirely developed in Matlab. The focus of this work will be describing how can we optimize this algorithm, and proposing new data structure so that the algorithm can achieve memory efficiency and hardware friendliness for an ASIC implementation.

[1]PATENT PENDING: US20170286815A1

In the broad sense, an object map is a matrix of labels, indicating to what type of object each pixel belongs. Figure 2.1(b) shows an example of a object map whose raster image is in Figure 2.1(a). Three types of objects are represented by different color codes in Figure 2.1(b): red for raster objects, blue for symbol objects, and green for vector objects. In the end, we want to render raster and symbol objects with high-frequency of screens, vector objects with low-frequency of screens, to achieve better print quality.

(a) A raster image with mixed contents.

(b) The object map of the raster image (a).

Fig. 2.1. (a) is a raster image and (b) is its object map. In the object map, raster regions are red; symbol regions are blue; vector regions are green.

Different objects in a raster image have different properties: a symbol object is usually small and has sharp edges and smooth interior; a vector object is typically large, and it is smooth; a raster object can be either large or small, and they are always rough. Only two features – the size of an object and the roughness/smoothness of an object are needed to classify a component. To classify all image objects into

the symbol, raster, and vector objects, if we can identify symbol and vector objects, the remaining unclassified objects will be raster objects. Symbol objects can be partitioned into symbol edge objects and symbol interior objects. Accordingly, three binary images – one to find symbol edge objects, one to find symbol interior objects, and one to find vector objects, are generated for connected component analysis.

To be suitable for hardware implementation, the choice of the connected component algorithm requires a small number of passes, no random access, and minimal complexity and memory usage. The classic two-pass connected component [45], which is a two-pass sequential (raster order by convention) scanning process, satisfies all these requirements except for its massive memory consumption. The amount of memory consumption depends on the complexity of the image but is limited to the size of the image. However, if we only process a narrow strip of the image at a time with the classic algorithm, the upper bound of the memory consumption will be reduced to the size of the strip, which is a significant saving. The challenge is how to take care of the discontinuities between every two adjacent strips. To further push down the memory consumption, a label recycling mechanism will be introduced. Besides, we use a union-find data structure with path compression to ensure fast memory access and efficiency for solving label conflicts.

2.2 Methodology

2.2.1 Overall Algorithm

Without considering the notion of strip for now, Figure 2.2 is the block diagram of the overall algorithm. Given an input image, it is first processed by a Sobel operator. The output of the operation is referred to as edge magnitude (EM), which reflect the strength of the gradient along both row and column direction at every pixel. Thresholding the EM image with two different values, Ts_edge (strong edge threshold), and Tw_edge (weak edge threshold), with Ts_edge greater than Tw_edge, can give us three binary images: strong edge map (SEM), non-strong edge map (NSEM), and non-edge map (NEM). To extract object features, connected component analysis and labeling process are performed on each of these three binary images. The extracted components from each binary image will be classified based on their characteristics (size and roughness). In each connected component set, we are only interested in finding a particular type of object: symbol edge objects from the connected strong edge components (CSECs), symbol interior objects from the connected non-strong edge components (CNSECs), and vector objects from the connected non-edge components (CNECs). Combining the classified components, we obtain symbol objects and vector objects. Uniquely label each pixel based on the type of the object it belongs to, then the rest unclassified pixels will be raster. A whole page that consists of these three different labels will be our final object map. Next, we will dive into each step in detail.

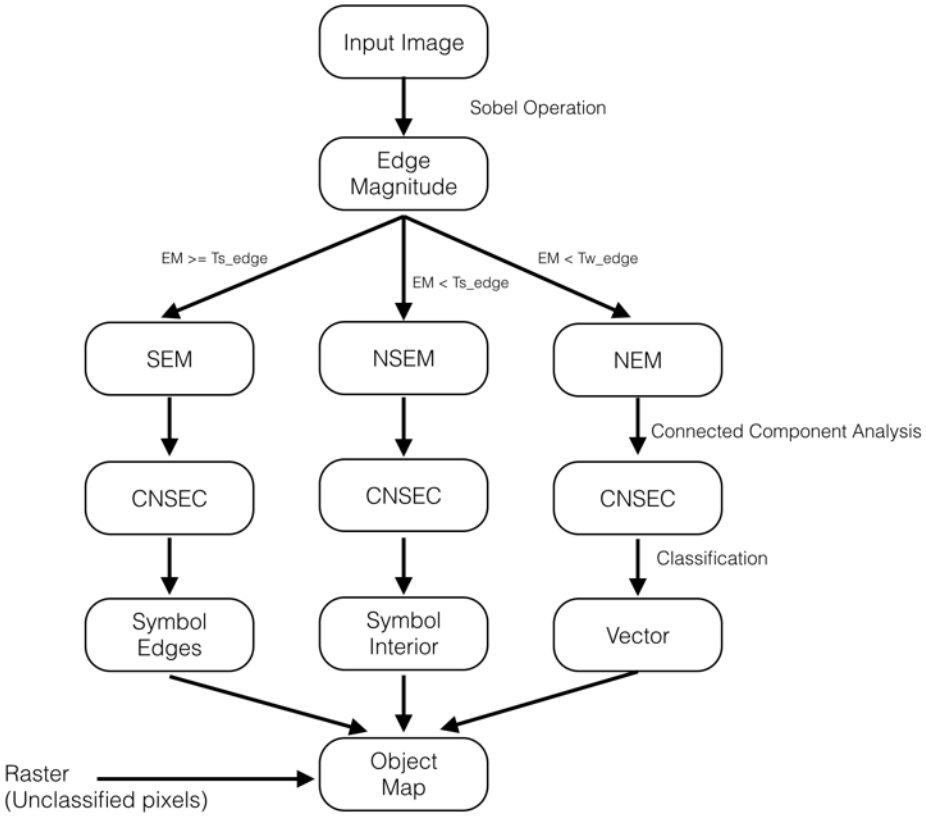


Fig. 2.2. The overall algorithm structure for image segmentation.

2.2.2 Generating Binary Images

To generate a binary image, a conventional method is first to apply an edge detection filter, followed by thresholding. We use a Sobel operator for edge detection because it is simple and easy for implementing hardware speed acceleration [46]. It contains two 3×3 kernel windows – one to detect horizontal gradient, and one to detect vertical gradients:

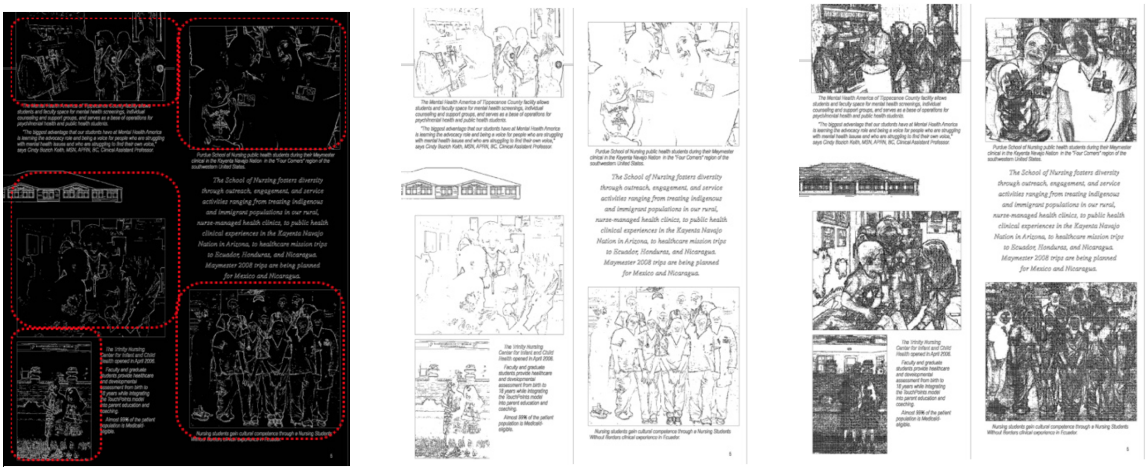
$$Gx = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad Gy = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The edge magnitude (EM) is defined as:

$$EM = \frac{1}{3} \sum_{i=r,g,b} \sqrt{(Gx * image[i])^2 + (Gy * image[i])^2} \tag{2.1}$$

where $*$ denotes a 2D convolution operation.

If we threshold the EM from the Sobel operation with two threshold values, Ts_edge and Tw_edge, we will be able to generate the following three binary images in Figure 2.3 from the input image Figure 2.1(a).

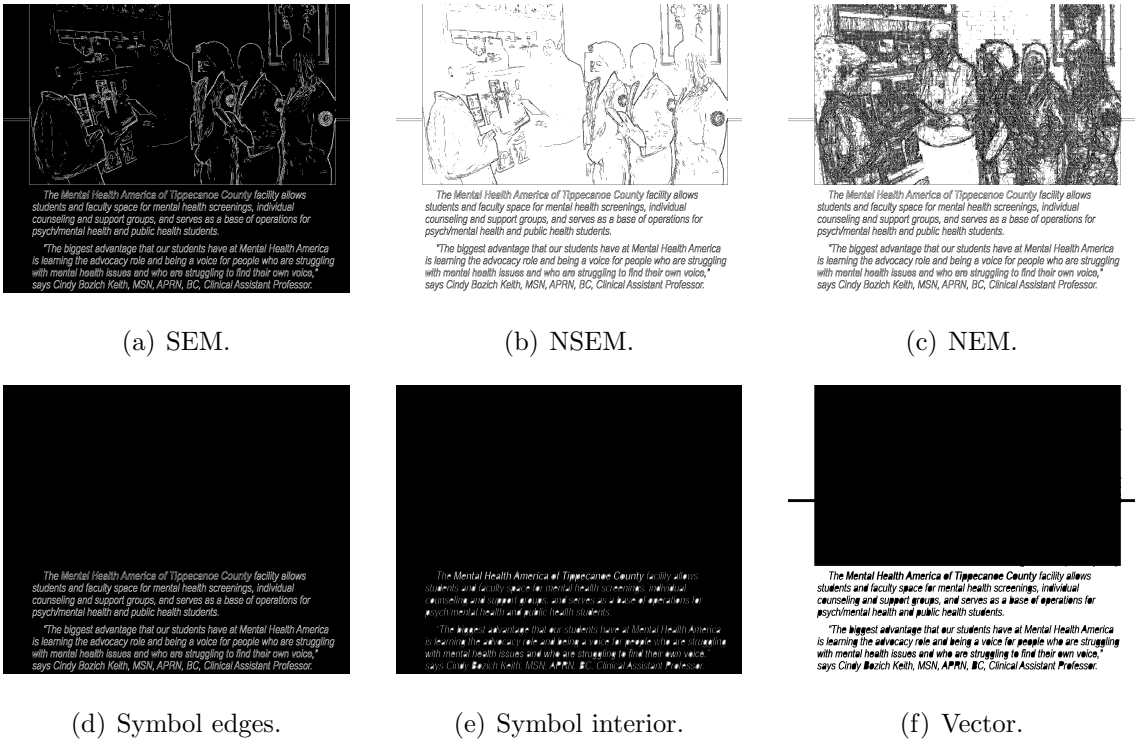


(a) Strong edge map ($EM \geq Ts\_edge$). (b) Non-strong edge map ($EM < Ts\_edge$). (c) Non edge map ($EM < Tw\_edge$).

Fig. 2.3. The three binary images generated from Sobel operator.

In each binary image, only those pixels whose EM satisfy the threshold condition (white pixels) are taken for the next process. From SEM, we can see that the white pixels are strong edge pixels, which contain not only symbol edge pixels but also some raster pixels (the red boxes in Figure 2.3(a)), and we are only interested in finding those symbol edge pixels. For the second binary image NSEM, the white pixels are interior pixels, and we are only interested in symbol interior pixels. For the last binary image NEM, the white pixels are also interior pixels, and our interests are vector pixels only. After classification and relabeling process, which will be introduced later, only

the pixels of our interest will remain, which again are three binary images with white pixels representing pixels of interest for each binary image.



(a) SEM. (b) NSEM. (c) NEM.

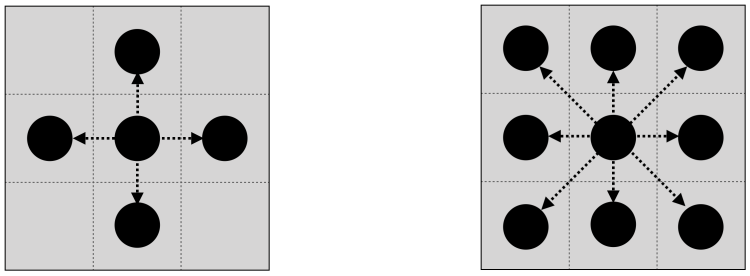
(d) Symbol edges. (e) Symbol interior. (f) Vector.

Fig. 2.4. (a) - (c) are the three input binary images which are generated from edge detections; (d) - (f) are the output binary images after CCL and classifications.

Figure 2.4 shows the result before and after the classification and relabeling process. For visualization purpose, we take only part of the binary images from Figure 2.3(a)–(c), as shown in Figure 2.4(a)–(c). Figure 2.4(d)–(e) are the relabeled binary images with only the pixels of interest remained. The final object map can be generated by combining Figure 2.4(d)–(e). However, before we can classify those binary images in 2.4(a)–(c), connected component analysis and labeling process must be performed, which will be detailed in the next section.

2.2.3 Connected Component Labeling Algorithm

The connected component labeling algorithm is to detect connected regions in a binary image by assigning each disjoint region with a unique label, although color images and data with higher dimensionality can also be processed [47] [48]. The connectivity can be either 4-connectivity or 8-connectivity, as shown in Figure 2.5. For a pixel in the center to be examined, its Northern pixel, Southern pixel, Western pixel, and Eastern pixel are considered as its neighbors for 4-connectivity context. For 8-connectivity context, besides those four pixels, the four diagonal pixels are also considered as its neighbors. For our application, 4-connectivity is used because it is less complex; and there is no significant difference in the performance compared to 8-connectivity for this application. So, 4-connectivity will be used for the remainder of the thesis without further mention.



(a) 4-connectivity. (b) 8-connectivity.

Fig. 2.5. Neighborhood connectivity context.(a) is 4-connectivity neighborhood context and (b) is 8-connectivity neighborhood context.

Figure 2.6 shows an example of a binary image that has been processed by the 4-connectivity CCL algorithm. There are three disjoint foreground regions (white pixels), and each of the joint regions has been uniquely labeled. Two foreground pixels in the same disjoint regions are considered as connected, as they carry the same labels. Sometimes, connected CCL process is also accompanied by extracting

features of each region. The features can be the number of pixels, average pixels value, and bounding box size, for example.

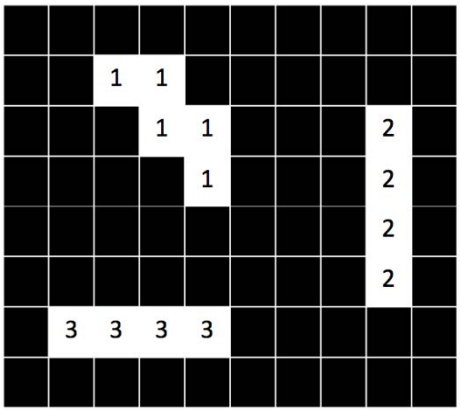


Fig. 2.6. An example of a binary image after being processed with 4-connectivity CCL. White pixels are the foreground pixels that we assign labels to. There are three disjoint white regions. So, three unique labels are assigned.

There are a variety of CCL algorithms, and they can be grouped based on the number of passes or scans of a binary image needed. Concerning the algorithm to be suitable for an ASIC implementation, our options are narrowed down to one-pass or two-pass CCL algorithm. The most popular one-pass CCL algorithm is the seeded region growing approach [49]. However, the algorithm always randomly accesses the binary image to search for connected neighbors, which is not hardware friendly – we want a raster-order scan process instead. Although some other one-pass CCL algorithm [50] are raster order scan process, their functionalities are limited to extracting features only, and can not generate a relabeled image after one pass. One of an option that satisfies all these requirements is the classic two-pass CCL algorithm [51], which will be introduced first and extended to our algorithm later.

2.2.4 Classic Two Pass Connected Component Labeling Algorithm

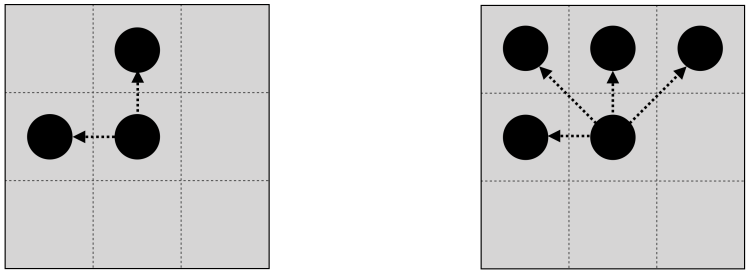
The classic CCL algorithm consists of two passes of processes of a binary image. The algorithm is described in Figure 2.7. The first pass is to assign an initial label to each pixel based on its neighborhood context. The second pass modifies the label of each pixel to its smallest equivalent label. Both first pass and second pass are raster order scans. That implies when we explore the neighborhood context of a pixel, only the neighbors before (in raster order sense) the current pixel are examined to ensure that each pair of neighbors is only checked once. This is shown in Figure 2.8. In Figure 2.8(a), the label to be assigned to the current pixel comes from its Western neighbor or Eastern neighbor only for 4-connectivity neighborhood context. If the current pixel only has neighbor or two neighbors with the same label, then that label is directly used for the current pixel. A difficult situation arises when its two neighbors carry different labels; this is referred to as conflict. When conflict is encountered during the first pass, we let the current pixel takes the smaller label from its two neighbors. At the same time, we record an equivalence between them. Conflict is resolved by modifying all the labels to their smallest equivalent labels during the second pass.

Figure 2.9 shows an example of a binary image after being processed by the classic CCL algorithm with 4-connectivity context. Figure 2.9(a) is the input binary image with white pixels representing the foreground pixels. Figure 2.9(b) contains the labels assigned after the first pass. During the first pass, we are also extracting features, which are now shown here, and recording equivalence if the labels are different in the neighborhood context. In this example, label1–label2, label3–label4, and label5-label6-label7 are equivalent labels, and their equivalences should be recorded for the relabeling process during the second pass. After the second pass, as shown in Figure 2.9(c), each connected region now only has one unique label, which is the smallest label among all the equivalent labels. The output of the Classic two pass CCL algorithm will be a label map that has three unique labels which represent three different disjoint

regions. However, for our application, we are interested in finding only one type of object for each binary image. Hence, we only need two types of labels after the CCL algorithm: Class 1 (C1) represents a pixel belongs to the type of the object of interest, Class 2 (C2) does not. This information can also be represented as a binary image for visualization purpose: white pixels represent C1 pixels, and black pixels represent C2 pixels.

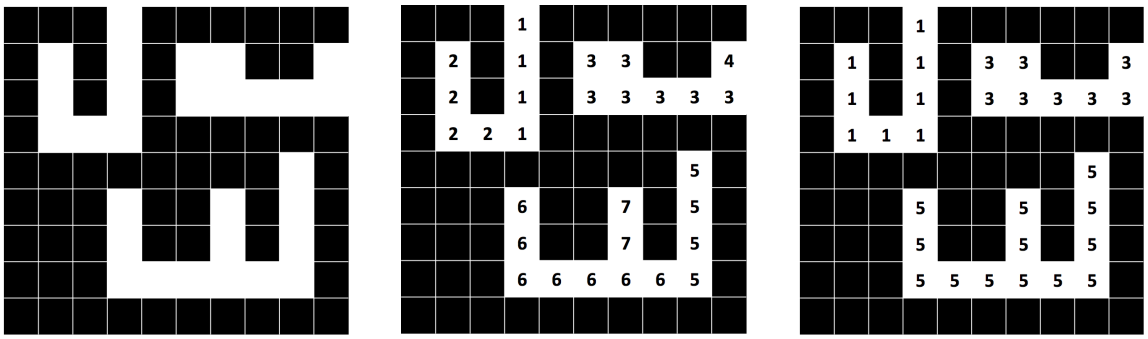
```
1. First Pass - Iterate pixel by pixel in raster order:
if The pixel is foreground (white pixel) then
    Get the neighborhood context of the current pixel
    if There are no neighbors— then
        uniquely assign the current pixel and continue
    else
        Assign the smallest label of the neighborhood
        and store the equivalence
    end
else
    Skip the current pixel
end
2. Second Pass - Iterate pixel by pixel in raster order:
if The pixel is foreground (white pixel) then
    Relabel the pixel with the lowest equivalence label
else
    Skip the current pixel
end
```

Fig. 2.7. Algorithm 1 – The first pass and the second pass of the classic CCL algorithm.



(a) 4-connectivity. (b) 8-connectivity.

Fig. 2.8. Neighbors to be examined for classic CCL algorithm. (a) is 4-connectivity, Eastern and Southern pixels are examined. (b) is 8-connectivity, Southeastern and Southwestern pixels are also examined.



(a) Original binary image. (b) First-pass labeling. (c) Second-pass relabeling.

Fig. 2.9. A example of a binary image (a) after being processed by the first pass (b) and the second pass (c) of the classic CCL algorithm.

2.2.5 Data Structure for Classic CCL Algorithm

During the second pass of the Classic CCL algorithm, as stated in Figure 2.7, each label is modified to its smallest equivalent label. The equivalence of different components can be very complicated. For example, for the bottom component in Figure 2.9(b), there is no prior knowledge about if the three components 5, 6, 7 are connected until the scan reach the last second row of the binary image. However, once

the neighborhood context with different labels in identified, equivalence need to be recorded. Here, we utilize the union-find data structure [52] to store the equivalence. Later, we will see that this data structure can be very efficient for our application.

Union–Find Data Structure

The input to the union-find is a sequence of pairs of integers, where each integer presents an object of some type (connected component in this scenario). We interpret the pair (p, q) as meaning p is connected q which has the following equivalence relations [53].

- Symmetric: If p is connected to q, then q is connected to p.
- Transitive: If p is connected q and q is connected to r, then p is connected to r.
- Reflexive: p is connected to p.

The goal of Union-Find is to partition these integers or objects into equivalent classes or connected components based on the given sequence of input pairs.

The array-based union-find data structure uses an array that is initialized to its own index, as shown in the first row of Figure 2.10. Each array element is an integer or a label of any type of object, and we have 7 components in this case initially. If a pair of integers is given, a union command will be performed to establish the equivalence between these two components. This is done by modifying the element indexed by the larger integer to the smaller integer. For example, for the given input pair (3,4), the 4^{th} element is modified to 3. This can be better visualized as a tree structure shown in Figure 2.11 (a), where the smallest number of a tree – 3 is the root of the tree, also the parent of component 4. For all other components, 1, 2, 5, 6, 7, they are also root components but do not have any child so far. To determine if a component is root or not, we can simply check the union-find array – a component is the root of a tree if its label equals to index. From the second row of the table in Figure 2.10, we can conclude that only component 4 is not a root. Followed from the

first example, when another pair (4,7) is given, instead of making component 7 to be the child of component 4, we can first search the root of component 4 and make the component 7 to be the direct child of the root component, which is component 3. The search of the root component is done by the find command, which will be explained later. As you can see what we have achieved here from Figure 2.11(b), by making component 7 to be the child of component 3 instead of 4, the tree is shortened by 1 level. A shorter tree can in return help us quickly to find the root of a component by traversing less number of nodes. In the union-find array, the 7^{th} element is now modified to 3, as shown in the third row of Figure 2.10. The fourth row is to establish equivalence between component 1 and component 6. From Figure 2.11(c), we can see that now we have two trees with more than 1 element. One challenge is how to merge these two trees together when a union command is called on the pair of children from each tree. In this example, a union command is called on (7,6), where component 7 is one node of a tree rooted at component 3, and component 6 is one node from another tree rooted at component 1. Since component 1 has a smaller label than component 3, so the third element in the union-find array is modified to 1. This also matches with tree interpretations in Figure 2.11(d). Note that the original union-find structure does not require the smaller element to be the parent, such as the weighted union-find data structure [54], which always makes the larger weight root to be the parent and it is considered more efficient. However, maintaining the increasing order from a root component to its leaf components is necessary to adopt this data structure for our CCL algorithm, which will become clearer in our future discussion.

The find command of the union-find structure is to inquiry the root of an element. In the union-find array, array[i] will basically return the parent of component i. For example, array[7] will return 3 according to the last row in Figure 2.10. This will give us component 3, which is the parent of component 7, but not the root of component 7. To find the root of component 7, we can iterate the array (i.e. array[array[array[...]]]) until what the array returns equal to its input, which is component 1.

1	2	3	4	5	6	7	Initialization
1	2	3	3	5	6	7	(3,4)
1	2	3	3	5	6	3	(4,7)
1	2	3	3	5	1	3	(1,6)
1	2	1	3	5	1	3	(7,6)

Fig. 2.10. An example of the union-find structure implemented in an 1D array.

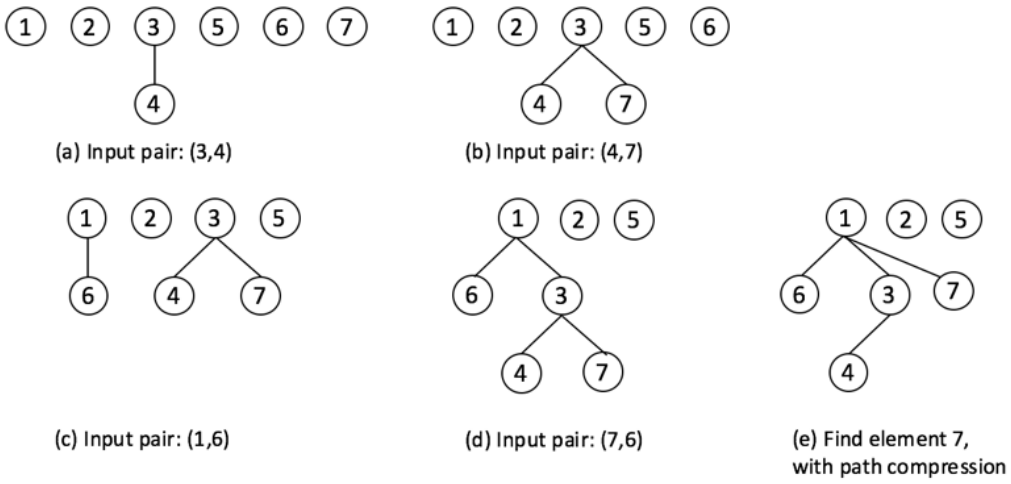


Fig. 2.11. Illustration of the union-find with tree structures. (a)–(d) are aligned with row2–row5 in Figure 2.10

The following code segments show how the union command and find command can be efficiently and compactly implemented in C code. One input to these functions is a union-find array, which has an integer pointer type. It is worth to mention that this array can be any other type instead of integer depending on the application. The minus one inside the bracket is to take into account that the first index is zero for C array, and the labels for assigning to pixels start from one. Two ways of

implementation of the find command are given here. The first way is intuitively more straightforward, which keeps finding the parent of k until k equals to its parent by using a while loop. The second way uses a recursion fashion. Next, we will see that implementing the union command by using a recursion fashion can give us the opportunity to optimize the find algorithm.

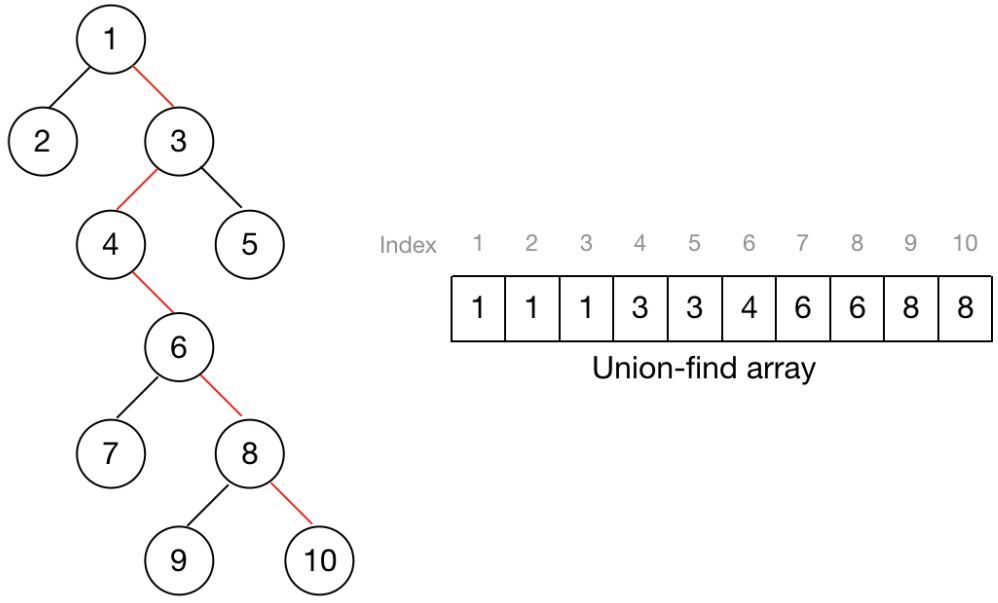
```
// Union two integers
void union_two(int* array, int p, int q){
    int pr = find_root(array, p);
    int qr = find_root(array, q);
    if(pr < qr) array[qr-1] = pr;
    else array[pr-1] = qr;
}
// Find the root of integer k
int find_root(int* array, int k) {
    while(array[k-1] != k) k = array[k-1];
    return k;
}
// Find the root of integer k with recursion
int find_root_recursion(int* array, int k){
    int root = k;
    if (array[k-1] != k)
        root = find_root(array, array[k-1]);
    return root;
}
```

As we introduce the union command, we can see that find command is regularly used when we want to union the children of two trees, and we need to use the find command to find the root of the two trees to establish equivalence. This implies that the performance of the union in a considerable degree depends on the find command. However, the performance of the find command also depends on the tree structure, i.e., the height of the tree, which is maintained by the union command. By shortening the path or the height of the tree, the performance of both find and union can be improved, and this is done by our path compression algorithm. Figure 2.12(a) shows

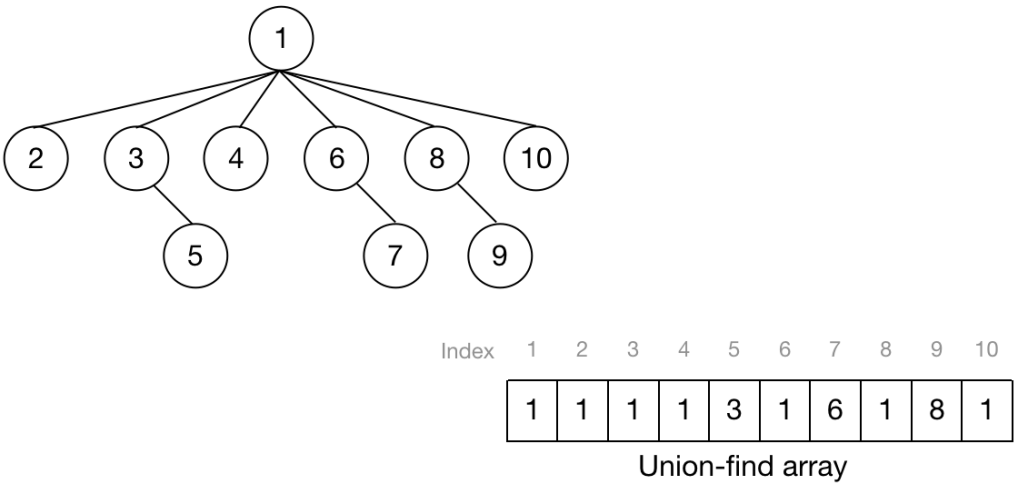
an example where we want to find the root of component 10. The red lines represent the path we have to traverse to find its root. It takes five iterations if we execute our find_root function, which does not look too bad. However, for a high-resolution image that has a lot of complicated structures, trees can grow their height more than we can imagine, and the union-find will become a bottleneck to the overall performance. What's more, it is very likely that the leaf component, e.g., component 10 will be called multiple times by the find command, and each time we are traversing the same long path over and over again. To avoid of this inefficiency, path compression can be integrated into our find algorithm, that is as we are trying to find the root of a node p along a path l_p, every node on the path l_p between node p and its root including p itself will be linked to its root directly. It can be done merely by just inserting one line of code to our find_root_recursion code as follows.

```
int find_root_path_compression(int* array, int k){
    int root = k;
    if (array[k-1] != k)
        root = find_root(array, array[k-1]);
    array[k-1] = root; //Link each node on the path to the root
    return root;
}
```

Figure 2.12(b) shows what the tree will look like if we execute our find command on component 10 and the union-find array in Figure 2.12(a). Not only component 1 will be returned, which is the root of component 10, but also the tree is shortened. Every node on the red path in 2.12(a) is now linked to the root, component 1. Next, we are ready to use this union-find structure to implement our classic CCL algorithm.



(a) Original tree.



(b) Tree after path compression.

Fig. 2.12. An example of path compression. When find the root of node 10 (a), all the nodes along the path from the root to node 10 will become the child of the root (b).

First Pass of CCL Algorithm with Union-Find

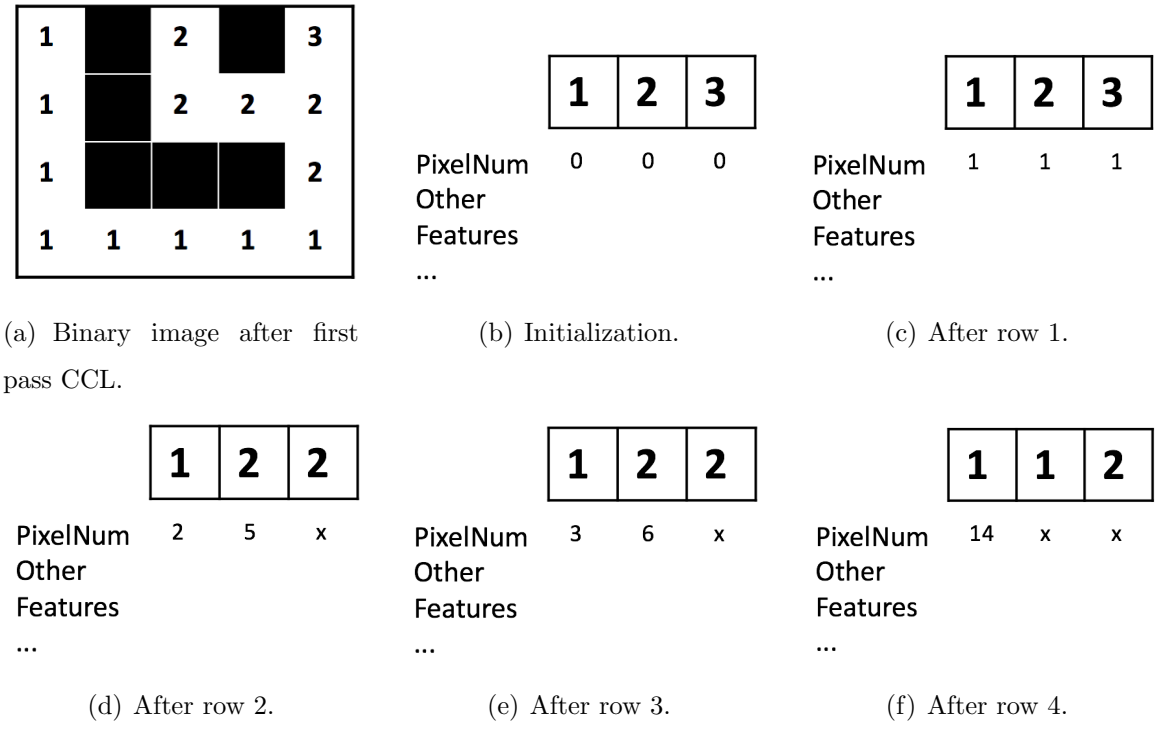
We first create an empty 1-D union-find array, which is referred to as a component array for our application. The component array is initialized to its own index, and each array element is mapped to a component by a pointer that contains a set of features. These features will be used to classify this component into C1 (an object of interest) or C2 (not an object of interest) in the future. As shown in Figure 2.13(b), a component array with three elements is created. The indices of these elements are 1, 2, 3 from left to right, and these elements have been initialized to their indices (the numbers in the array). The feature set is listed below each element. For simplicity, only the feature – the number of pixels, is shown here.

During the first pass labeling, when a new label is needed, we will scan through the component array from left (smallest index) to the right (largest index) until an available element is found. Then its index will be used as the new label, and a corresponding feature set will be created; after that, this element will be set as unavailable. For every white pixel encountered during the first-pass scan, its corresponding feature set in the component array will be updated. As you can see from Figure 2.13(c), after we have swept the first row, the PixelNum feature for each component has been updated, where we have one pixel for each component.

In the situation that an equivalence needs to be recorded, the two labels (Northern and Western neighbors for 4-connectivity) will become the input pair to the union command. It first finds the roots for these two indices and then modifies the root element of the larger index to the smaller index. At the same time, we merge the feature set of the latter one to the former one. Subsequently, we can safely remove the feature set pointed by the larger root, since the feature set of the smaller root now contains all the combined features. In Figure 2.13(d), after the second row has been scanned, the third element becomes two, indicating the third component becomes a child of the second. What's more, the feature set of the third element has been merged to the second and then freed (represented by the "x" symbol). Now the

second element contains 5 pixels, which are actually the number of pixels labeled with 2 and 3 for the first two rows. For the future labeling of a pixel, instead of directly taking the label from its two neighbors, we will use the smaller root label of the two neighbors by performing find command on each neighbor. This again can make the tree shorter by linking future's equivalent components directly to the root.

After the first pass labeling, the component array will contain equivalence relations of all the components, and only each root component carries the combined feature set of all the equivalent components. For the second pass of the CCL, it will change all the labels assigned to the image during the first pass to class labels – C1 or C2. This requires a classification process before we perform the second pass relabeling.



(a) Binary image after first pass CCL.

(b) Initialization.

(c) After row 1.

(d) After row 2.

(e) After row 3.

(f) After row 4.

Fig. 2.13. Illustration of how the component array evolves as we do first-pass labeling row by row.

Classify Component Array

The algorithm of classifying component array is described in Algorithm 2 in Figure 2.14. We scan through the component array from right to left. At each nonempty array element, we first find its root component, and we classify the root component based on its feature set if it has not been classified yet based. The classification result (C1 or C2) is recorded in the feature set. Even though all its offspring components do not have associated feature set anymore, they will carry the same classification result as the root, which can be read from their root component.

```
for Each component in the component array from the
largest index do
    root component ← findRoot (current component);
    if The root component has been classified then
        continue;
    else
        classify the root component and record the
        classification result;
    end
end
```

Fig. 2.14. Algorithm 2 – Classify component array.

Second Pass of CCL Algorithm with Union-Find

Second-pass starts by iterating pixel by pixel of the label buffer in raster order again. At each pixel, we first read the label, then find its root component. We replace each current label with the class label (C1 or C2) that is recorded in the feature set of the root component. The algorithm is stated in details in Figure 2.15. Note that after the second pass, the relabeled map is also a binary image. Also, during this chapter, we are using one binary image for analysis and all the examples. It is important to bear in mind that we have three binary images to be processed.

```
Iterate pixel by pixel in raster order;
L ← read the label of this pixel;
if L ≠ 0 then
    Comp_L ← Component Array [L];
    Comp_Root ← findRoot(Comp_L);
    CL ← read classification result of Comp_Root;
    Replace label L with CL;
else
    Skip the current pixel;
end
```

Fig. 2.15. Algorithm 3 – Second pass of CCL with union-find.

2.2.6 Issues with Classic CCL Algorithm

The main drawback of the classic CCL algorithm is its extensive memory consumption. During the first pass, the assigned labels are stored in a label buffer, and each unique label has associated component in the component array. Referring back to Figure 2.9(b), where we have three disjoint regions, seven labels are used. This will increase not only the length of the component array but also the size of the label buffer since more bits are needed to store a larger label. Consequently, if we allow the first pass continues till the end of the page, the memory to hold the whole component array and all the labels will be tremendous, which is impossible for hardware implementation [55]. However, if we look at the fifth row in Figure 2.9(b), where the first white pixel is assigned with label 5, and components 1-4 have ended in the previous row already. Instead of continuing down to assign labels, if we could classify those ended components and relabel (second pass) those previous pixels, their labels and the corresponding memory will not be needed anymore. Those unneeded memories can be recycled for the following pixels to use. Our strip based CCL algorithm is based on this idea – we only label a strip of the image at a time and classify the previous strip before we start to label the next strip of the image.

2.2.7 Strip Based CCL Algorithm

If we define a strip to be a little bit taller than the height of a regular text character and the same width as the input image, then an input binary image can be divided into many strip regions, as shown in Figure 2.16. The red lines are the strip boundaries that separate every pair of adjacent strips. Now, an object in the binary image can fall into one or several strip regions. We can group all the components based on the number of strips they span. If a component crosses fewer than two strip boundaries, it will be defined as a bounded component, otherwise, it is an unbounded component, like the one at the bottom in Figure 2.16. For the three types of object we have: symbol, raster, and vector, a symbol object is usually small, so it is expected to be a bounded component; a vector object which is usually very large is expected to be an unbounded component. The raster objects can be either bounded or unbounded but can be discriminated based their roughness. The classification is summarized in Figure 2.17. To determine if an object is the of our interest or not in each binary image, we can look at its boundedness and its roughness from its feature set. Also, remember that our classification strategy only tries to identify symbol and vector objects; the remaining unclassified objects are raster objects by default.

For our first pass labeling, it is the same as before, except that we only process and label a strip at a time; at the start, we label the first two strips. Then we classify each root component that does not cross the current strip boundary – the second red line. Figure 2.18(a) indicates that the first two strips have been labeled. The labeling process is also accompanied by extracting features, which are not shown here. These features are stored in the component array and are used to classify all the root components according to the decision criteria in Figure 2.17. Assuming component 1 is classified as class 1, and component 2 as class 2, their classifications are recorded in each root component in the component array, represented by "C1" and "C2". For component 3, since it crosses the current strip boundary, it will not be classified. Instead, we will continue to assign labels and collect features. The

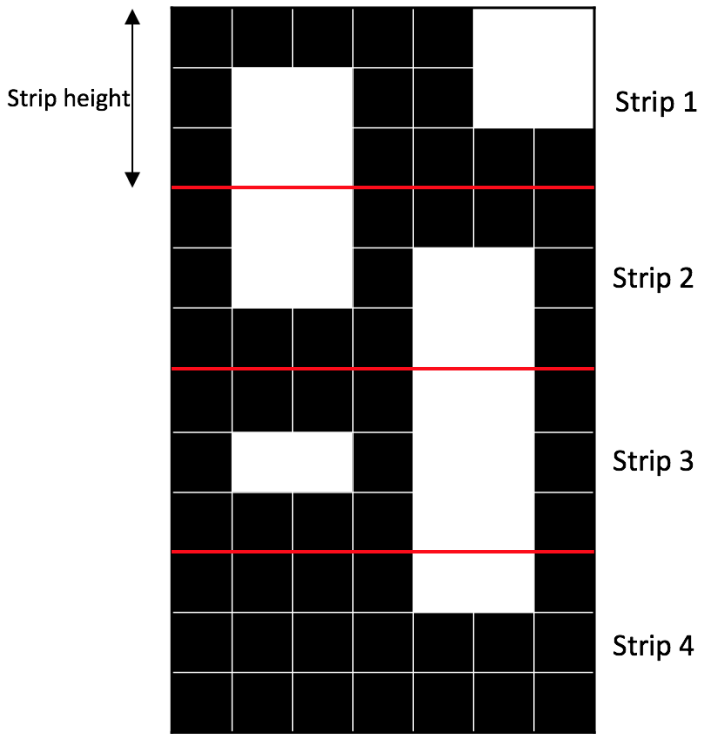


Fig. 2.16. Partition an image to horizontal strips for strip based CCL.

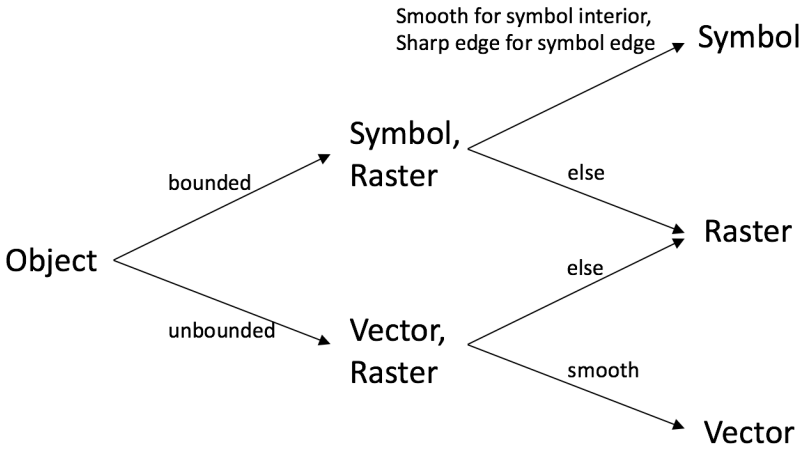
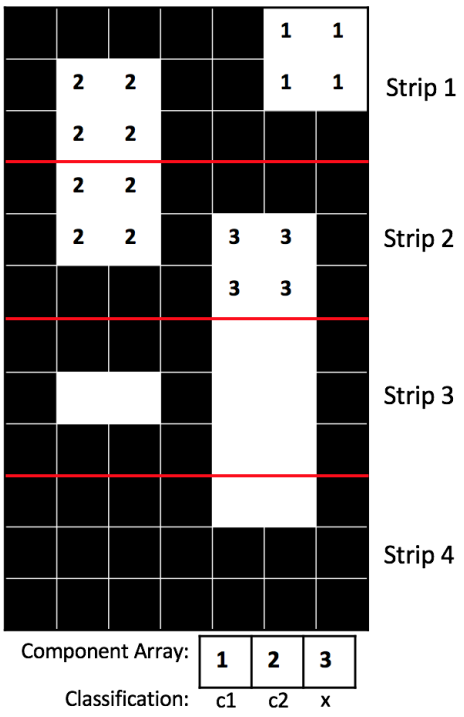


Fig. 2.17. Classification of an objects based on its boundedness and roughness.

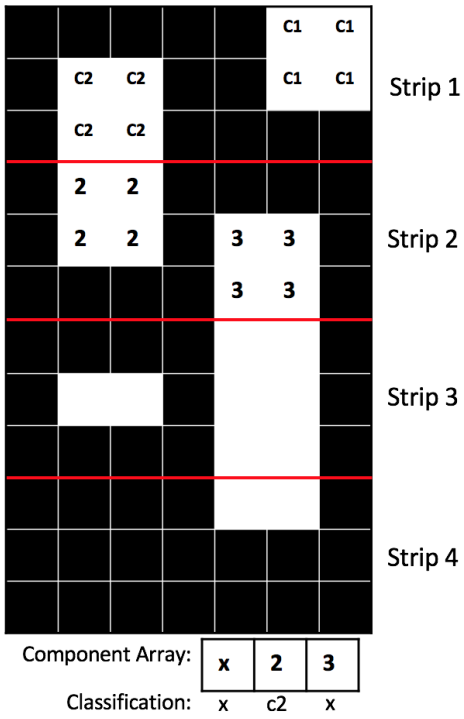
symbol "x" under component 3 represents that the classification of component 3 is still unknown.

The second pass starts right after the classification is done; and we only process the previous strip, which is the first strip in Figure 2.18(b). The reason why we do not process the second strip is that component 3 has not been classified – the pixels with label 3 do not have the associated class labels yet. After the first strip has been processed by the second-pass, as we can see from Figure 2.18(b), all the labels in the first strip assigned during the first pass now have been replaced with their corresponding class labels. What's more important, if a component has ended in the previous strip, such as component 1 in Figure 2.18(b), its memory including its label and the feature set stored in the component array will be recycled. Furthermore, its location in the component array will be marked as available (represented by "x" symbol in the component array). The second pass will be followed by the first-pass labeling of the next strip. In Figure 2.18(c), label 1 and component 1 recycled from the previous component is now used for this new component. The first pass labeling of the current strip and the second pass relabeling of the previous strip will alternate downward with a classification step between each first pass labeling and second pass relabeling until the whole page is processed.

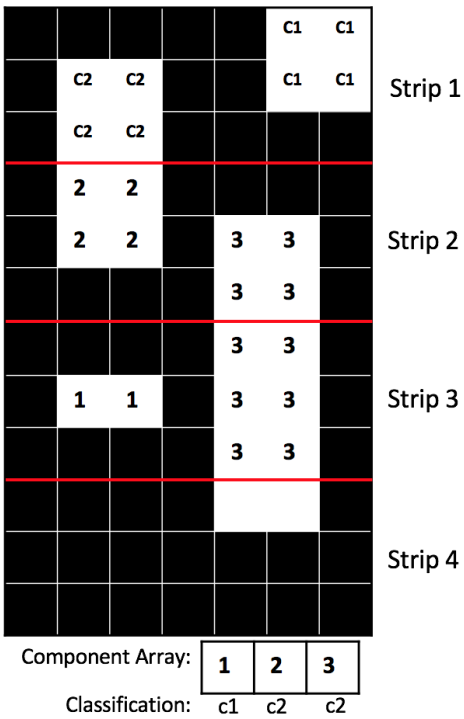
There is a special case, and that is component 3 which is an unbounded component. For an unbounded component, we will force to classify it at the second strip boundary it crosses as shown in Figure 2.18(c). The classification is based on the features collected from the pixels that above this strip boundary. In this example, the bottom two pixels in strip 4 will have no contributions to the classification of component 3 at all. However, during the first pass of strip 4, instead of assigning label 3 to the bottom two pixels, we assign the class labels directly, since component 3 has already been classified. This allows the class labels to carry across the strip boundaries, and prevents label 3 from propagating all the way down the page, and being unable for recycling.



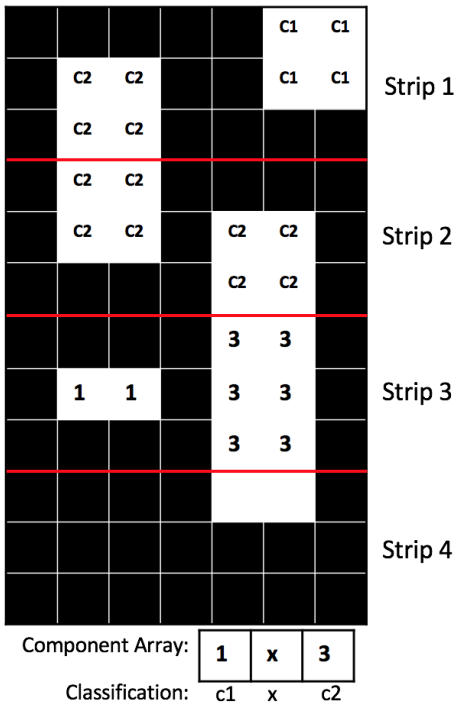
(a) First pass on strip 1 and 2.



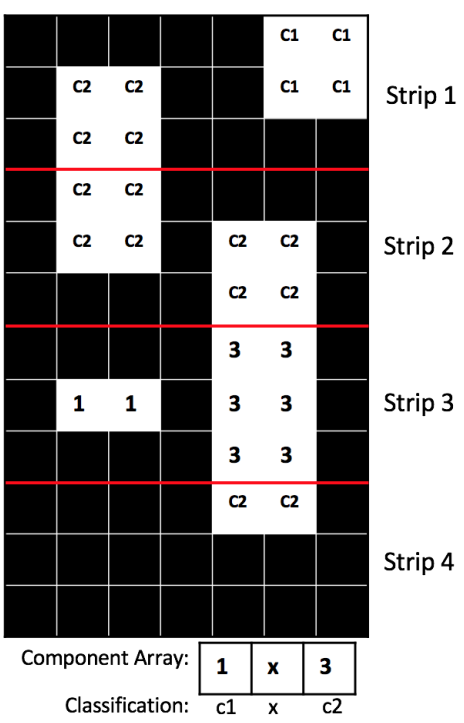
(b) Second pass on strip 1.



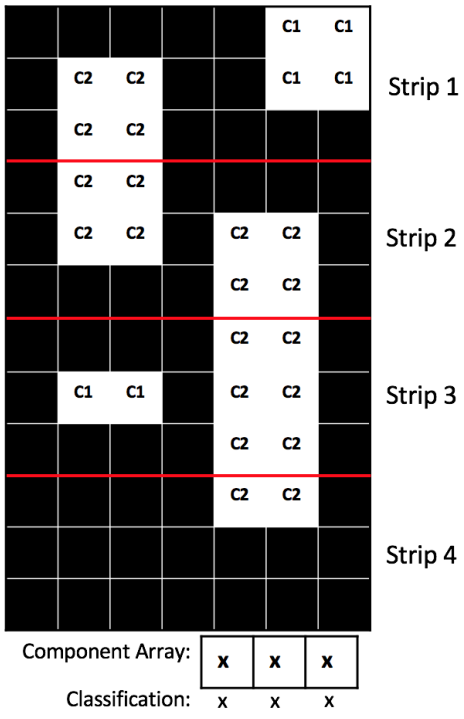
(c) First pass on strip 3.



(d) Second pass on strip 2.



(e) First pass on strip 4.



(f) Second pass on strip 3.

Fig. 2.18. An example of strip based processing.

It is interesting to note that there two types of labels used through the CCL process – first pass label and second pass label. Second pass labels are binary, class 1 or class 2, which can be encoded with only a single bit. First pass labels are represented by numbers, which will grow the required bit depth of the label buffer size if the label number keep increases. This strip based process will limit the life cycle of the first pass labels. If we take a closer look of all the sub-figures in Figure 2.18, we can see that all the first pass labels never span over two strips, which also means that their occupied memory in the component array will not last over two strips. By reusing these labels and memories, this reduces not only the bit depth of the label buffer but also the size of the component array. For the illustration purpose of the previous example where a page is partitioned into multiple strips, only two strip buffers are actually needed. The process is shown in Figure 2.19. The first row and second row

in the table contains all the processes for the first and second label buffer. Each column contains one process, and all the processes proceed sequentially from left to right. The number in the parenthesis is the strip number in the binary image that is being processed. At each processing cycle, only one strip of object map is generated.

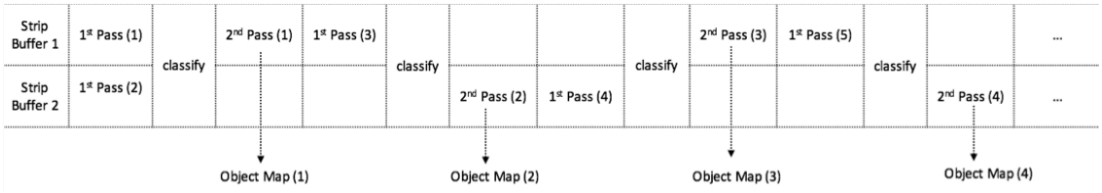


Fig. 2.19. CCL process with two strip buffers. The number inside the parenthesis represents the strip number of an image that is being processed.

2.2.8 Classify Components On-the-fly

Even we recycle unneeded memories at each strip boundary, there are still a lot of small components, such as single-pixel components that are not of interest to us. However, they still need unique labels assigned during the first pass of the CCL and memory in the component array maintained before we reach the next strip boundary. Instead of classifying all the components at the strip boundary, we could classify them on-the-fly. An extra function is added to our CCL algorithm that checks which root components have already ended at each row. This can be done by comparing the maximum vertical coordinate of the root component to a row counter. The row counter keeps track of the number of rows of the image that have been processed. If a root component has ended on the current row, we classify it right away. We make the following decisions on the root component and all its offspring components based on the classification result of the root component:

- Class 2: Recycle all the components in the component array.

- Class 1: Keep all the components in the component array, and record the vertical range.

Note that these decisions are made on the component array; we can not randomly modify the labels in the label buffer. By doing so, we may have a situation that different disjoint regions are mapped to the same component in the component array, since they are assigned with the same label. If we revisit the above decisions, we can conclude that there is at most one class 1 object before we reach the strip boundary, because we keep discarding the component in the component array and reuse the same label until one component is classified as class 1.

To help better understand this algorithm, an example is provided in Figure 2.20. During the first pass of the CCL, we have assigned labels to the first row. Before we continue to assign labels to the second row, component 1 has ended, and it could be classified. Assume it's classified as class 2. Its component, component 1 will be freed from the component array immediately, which means that label 1 becomes available and can be reused. For the new pixel we encounter on row 2, we assign it with label 1. Once it is detected that it has ended after the second row, again, assume it is classified as class 2, then component 1 will be freed and becomes available. For the same reason, we could use label 1 for the new pixel in row 3, and its component ends on row 5. This time, assume it is classified as class 1. Its component, component 1 will be kept in the component array, and the vertical range of component 1 is recorded as [row3, row5].

For the second pass of the CCL, although the first two pixels labeled with 1 are associated with component 1, they are not in the vertical range of component 1, which is from row 3 to row 5. Therefore, they will be directly relabeled with "C2". In contrast, the bottom three label 1 pixels which are in this range will be relabeled with "C1" during the second pass. As we can see, without using this approach, three labels and components in the component array are needed. Now, only 1 label and a single component are needed. This allows us to recycle those unneeded components more frequently before we reach the next strip boundary.

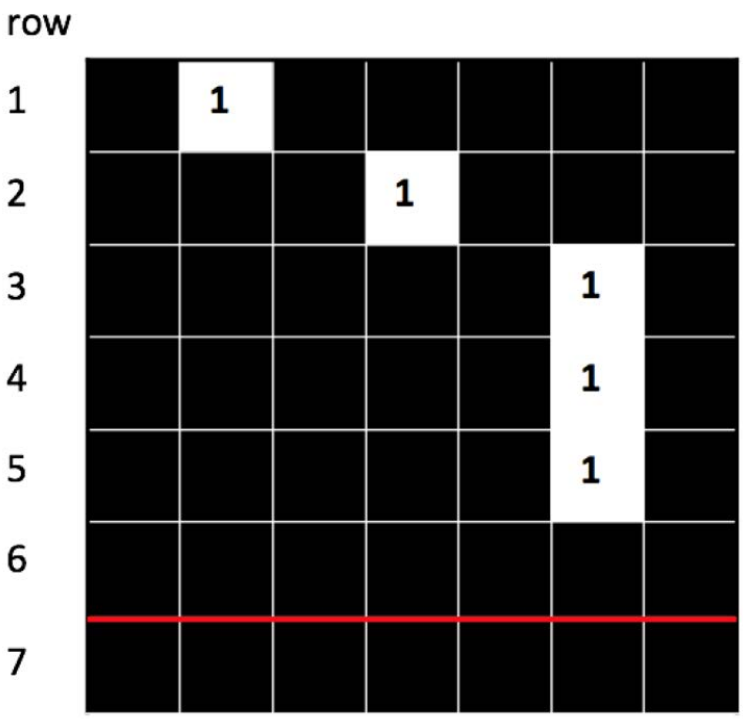


Fig. 2.20. Classify component on-the-fly. If only the bottom component is classified as class 1, then one label and one component are needed.

2.3 System Architecture

Figure 2.21 is a block diagram of the system architecture of our algorithm. The row buffer only buffers three rows of pixels of the input image, because the edge detector uses a 3 × 3 filter. The output of the edge detector is just one row of the binary image. The connected component labeler will read this row of the binary image, and assign a row of labels to the label buffer. At the same time, it extracts the features into component array. For the connected component labeler to assign labels to the next row, it will need the labels from the previous row, which are stored in the label buffer. The labels stored in label buffer will be copied to the corresponding row in the strip buffer. Once the strip buffer has been assigned with a full strip of labels, the controller will stop the row buffer from reading any new pixels, which

complete the first pass of a strip. During this process, the controller also checks ended components and classify the ended components in the component array. After all the components have been classified, the controller unit will make one of the two strip buffers to replace its initial labels with class labels, which are stored in the component array. This completes the second-pass of the strip. The relabeled strip will become a strip of object map as the output. The new processing cycle starts with the controller enabling the row buffer loading a new row of pixels from the input image, and this procedure repeats until all the pixels in the image are processed.

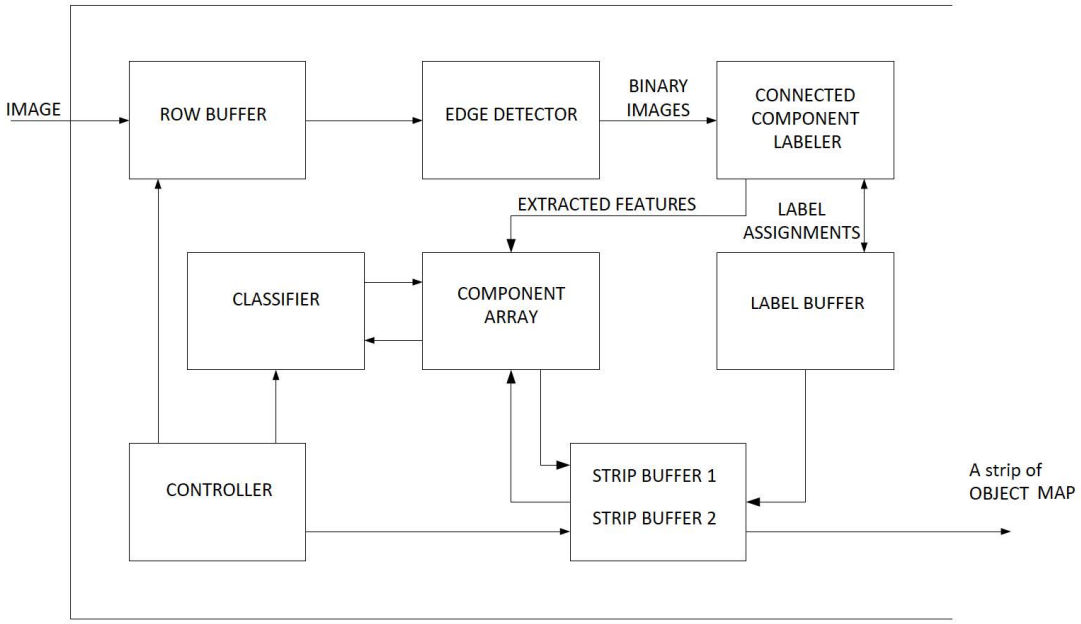


Fig. 2.21. System architecture of the strip-based CCL.

2.4 Experiment Result

The whole algorithm is implemented in C code. The raster pages for testing have 6400 $\times$ 4928 pixels at 600 ppi (pixel-per-inch), and we have only shown three of them in Figure 2.22(a)-(c). The strip is set to a height of 80 pixels. For the object map generated in Figure 2.22(d)-(e), the symbol objects are colored with blue, raster objects with red, and vector objects with green. The object maps look very reasonable if we compare them against their raster pages. The symbol "1956" in Figure 2.22(a) is colored with green in Figure 2.22(d). Although this is a misclassification caused by that this symbol object is unbounded, it is still acceptable for the object-oriented halftoning application. Since if a symbol is too large, it will behave like a vector object: larger and smooth. Therefore, it makes more sense to render those large symbol objects with the low-frequency screens rather than high-frequency screens.

For the classic CCL algorithm, the number of components is bounded by the image size, whereas, for our strip-based CCL algorithm, it is bounded by the strip size. Besides, because we recycle those unneeded components at each row, the number of components we need to maintain is significantly reduced. In Table 2.1, we have summarized the results from 10 test pages. The performance of the raster pages in Figure 2.22(a) are listed in first three rows of the table, and the remaining raster pages are not shown here. The comparison indicates that our algorithm can reduce the number of components by an average of 97.46% compared to the classic CCL algorithm. In other words, our proposed algorithm only needs less than 3% memory of the classic CCL.

Back Row:
Mark Varnau, Jerome Walker, John Gerhart, Gordon Schmitz, Linda (Hodson) Byers, Shirley (Money) Wilson, Max Williams, and Donald Cortrecht

Front Row:
Helen (Gallagher) Obuch, Richard Stevens, Alice (Pishan) Croner, Gwenivere (Hughes) Banker, and James Potts

Class of 1956 50th Reunion

You get facts; not for the sake of knowing the facts, which you will soon forget anyway, but for the sake of learning how to go about getting facts.

You make experiments; not to teach you how to make the same experiments again, but to teach you how to make and test new experiments as you go through life.

You read books; not to memorize their contents, but to analyze their wisdom and discover the means of acquiring wisdom for yourself.

You listen to lectures; not to pass examinations on the information they contain, but to learn how the human mind tackles a human problem.

At every point and at every stage of a general, liberal education, you are active. You are not a trainee, you are not an audience, you are not a receptacle or a storehouse.

You are a learner. The product of liberal education is not learned men, but learning men."

— Quote posted on Dr. Cwalina's bulletin board in 1956

The School of Pharmacy and Pharmaceutical Science's Class of 1956 Reunion was held during October 19-21, 2006. Upon arriving on Thursday, guests attended at cocktail party at Hour Time Restaurant and Lounge in Lafayette. On Friday, alumni retraced their steps around campus and visited some of the many new sites on a highlight bus tour which included the Heine Pharmacy Building, The Chao Center for Industrial Pharmacy and Contract Manufacturing, the Dick and Sandy Dauch Alumni Center, Discovery Park, the Birck Nanotechnology Center, and a behind-the-scenes look at Mackey Arena. Later that evening, alumni gathered for their reunion dinner in the Purdue Memorial Union. Prior to the Purdue vs. Wisconsin football game on Saturday, guests also were invited to attend the President's Council brunch where President Jischke announced a gift made by Charlotte and Stanley (BS 1955) Beck that will be used to construct an outdoor plaza for the Pharmacy building. Special thanks goes to the reunion committee members for helping make this weekend possible: Mark Varnau, Chair, Alice (Pishan) Croner, Dick Kouns, Jim Potts, Gordon Schmitz, and Jerry Walker.

Darolyn (Davis) Quayle was unable to attend the reunion celebration [also see page 23], but was kind enough to send the School one of the lasting impressions during her time at Purdue. "A little blurb was posted on Dr. Cwalina's bulletin board on the right wall as we walked out of the Organic Lab," she recalls. "I remember being impressed with Dr. Cwalina's philosophy on education, so I stood at the bulletin board and wrote it off, and I've saved a typed rendition ever since." She shared that philosophy with her two daughters, and the School is pleased to share it with you now. [see column on the right]

the PURDUE pharmacist

The School of Pharmacy and Pharmaceutical Sciences
Purdue University
Heine Pharmacy Building, Room 104
575 Stadium Mall Drive
West Lafayette, IN 47904-2091

Non-Profit Org.
U.S. Postage
PAID
Purdue University

(a)

Dean's Joint Advisory Council

The Dean's Joint Advisory Council meeting for the Industrial, Minority Advocacy, and Professional Councils was held on September 22, 2006, at the Holiday Inn Select in Lafayette. Members were welcomed by Interim Dean Holly Mason and given a School of Pharmacy and Pharmaceutical Sciences and University update. Update reports were also given for the spring meeting's breakout sessions and The Chao Center. Ron Dollens (BS 1970, HDR 2003), Retired CEO of Guidant Corporation, spoke about "The Future of the Medical Device Industry." Breakout discussions sessions followed. The sessions were divided into four tracks: Alumni, Professional Program, Industrial, and Minority Advocacy Council. The members regrouped to report on the items discussed prior to lunch, where incoming dean Craig Svensson was the guest speaker. Breakout sessions reconvened for the Industrial, Student Roundtable Discussion, and Minority Advocacy Council tracks, and the meeting was adjourned after final breakout group reports were presented. A welcoming reception for Craig and Sue Svensson was held later that evening. The next meeting for the joint councils will take place on Friday, May 18, 2007, at the Purdue Memorial Union.

Craig Svensson addresses the joint council during the lunch buffet

Members of the joint councils listen to Craig Svensson's presentation during lunch

Prof. Steve Byrn with Joseph Mo (PhD 1977)

Chuck Flemming (BS 1957, MS 1964, HDR 1994) and Tom Gerding (BS 1952, MS 1954, PhD 1960, HDR 2002)

Members of the Minority Advocacy Council meet during one of the breakout sessions

the PURDUE pharmacist © FALL / WINTER 2006 23

(b)

TEAM REACH OUT

In addition to their work at the local health clinics, nursing students helped with reconstruction work including painting and roofing.

Nursing students also distributed donations of canned goods, toiletries, clothes, and bedding to families living in FEMA trailers.

Eye Opening Experience

Arriving in the Gulf Coast region was an eye-opening experience for the students, says Professor Lynn Davis, who traveled with the students in May.

Although they saw the news reports during Hurricane Katrina, the cameras failed to capture the extent and widespread devastation that continues to exist in this region.

"The students found out that there is more to disaster response than meeting immediate needs," Davis says.

"The key thing for them to learn from this experience is the nature of catastrophic events and recovery. How do you put your life back together? Each time we go, we look at this."

"With a challenge of this magnitude, flexibility, patience, and incremental assessment, planning, and evaluation are essential," Novak says. "The students learned to set priorities, a key nursing skill."

Jeff Callaway, another May team member, says that during assessments, he saw evidence of the increase in mold-derived respiratory problems. Most of all, as the 2006 hurricane season came near, he saw signs of anxiety and feelings of lack of control.

"Children generally showed signs of hypervigilance and hyperactivity, while the majority of adults manifested with signs and symptoms of depression, lethargy, and/or anxiety," he says.

He observed that the nursing cycle of assessment, nursing diagnosis, care planning and evaluation applies in disaster recovery as much as anywhere else. The difference is the magnitude of the need.

"It reminds me of an old saying about how to eat an elephant," Callaway says. "You pick a place to start and then do it one bite at a time."

Perhaps the most powerful relationship formed was with 80-year-old Ray Lynn, an Ocean Springs resident whose story is not only poignant but also a microcosm of the entire scene of loss and ruin.

Mr. Lynn told the students his story of water rising gradually to more than 10 feet in his home. After treading water for eight hours, his wife had a heart attack and died in his arms. He has no children or close relatives, just his dog, Tex, who also survived the flood.

The nursing students and faculty painted the interior of Mr. Lynn's home. But the key to the relationship, Storozuk says, was just "sitting on his porch and listening to his story." Dr. Novak and the students stay in touch with Mr. Lynn through cards, letters, and phone calls.

Pictured are (front left to right) Dr. Julie Novak, Ray Lynn, Lisa Storozuk. (Back left to right) Kay Walton, Allison Serse, Professor Lynn Davis, Justin Hayden, and Jeff Callaway.

(c)

Back Row:
Mark Varnau, Jerome Walker, John Gerhart, Gordon Schmitz, Linda (Hodson) Byers, Shirley (Money) Wilson, Max Williams, and Donald Cortrecht

Front Row:
Helen (Gallagher) Obuch, Richard Stevens, Alice (Pishan) Croner, Gwenivere (Hughes) Banker, and James Potts

Class of 1956 50th Reunion

"You get facts; not for the sake of knowing the facts, which you will soon forget anyway, but for the sake of learning how to go about getting facts.

You make experiments; not to teach you how to make the same experiments again, but to teach you how to make and test new experiments as you go through life.

You read books; not to memorize their contents, but to analyze their wisdom and discover the means of acquiring wisdom for yourself.

You listen to lectures; not to pass examinations on the information they contain, but to learn how the human mind tackles a human problem.

At every point and at every stage of a general, liberal education, you are active. You are not a trainee, you are not an audience, you are not a receptacle or a storehouse.

You are a learner. The product of liberal education is not learned men, but learning men."

— Quote posted on Dr. Cwalina's bulletin board in 1956

The School of Pharmacy and Pharmaceutical Science's Class of 1956 Reunion was held during October 19-21, 2006. Upon arriving on Thursday, guests attended at cocktail party at Hour Time Restaurant and Lounge in Lafayette. On Friday, alumni retraced their steps around campus and visited some of the many new sites on a highlight bus tour which included the Heine Pharmacy Building, The Chao Center for Industrial Pharmacy and Contract Manufacturing, the Dick and Sandy Dauch Alumni Center, Discovery Park, the Birck Nanotechnology Center, and a behind-the-scenes look at Mackey Arena. Later that evening, alumni gathered for their reunion dinner in the Purdue Memorial Union. Prior to the Purdue vs. Wisconsin football game on Saturday, guests also were invited to attend the President's Council brunch where President Jischke announced a gift made by Charlotte and Stanley (BS 1955) Beck that will be used to construct an outdoor plaza for the Pharmacy building. Special thanks goes to the reunion committee members for helping make this weekend possible: Mark Varnau, Chair, Alice (Pishan) Croner, Dick Kouns, Jim Potts, Gordon Schmitz, and Jerry Walker.

Darolyn (Davis) Quayle was unable to attend the reunion celebration [also see page 23], but was kind enough to send the School one of the lasting impressions during her time at Purdue. "A little blurb was posted on Dr. Cwalina's bulletin board on the right wall as we walked out of the Organic Lab," she recalls. "I remember being impressed with Dr. Cwalina's philosophy on education, so I stood at the bulletin board and wrote it off, and I've saved a typed rendition ever since." She shared that philosophy with her two daughters, and the School is pleased to share it with you now. [see column on the right]

the PURDUE pharmacist

The School of Pharmacy and Pharmaceutical Sciences
Purdue University
Heine Pharmacy Building, Room 104
575 Stadium Mall Drive
West Lafayette, IN 47904-2091

Non-Profit Org.
U.S. Postage
PAID
Purdue University

(d)

(e)

(f)

Fig. 2.22. (a)-(c) are three input raster images; (c)-(d) are the corresponding object maps generated by our algorithm, red for raster, green for vector, and blue for symbol.

Table 2.1.
Number of components used by the classic CCL algorithm and the proposed CCL algorithm

Test Page	Classic	Proposed	Reduction
1	330906	8845	97.33%
2	285499	10072	97.47%
3	376013	11137	97.04%
4	206120	6692	96.75%
5	577283	10469	98.19%
6	363944	9675	97.34%
7	773979	12426	98.39%
8	449335	9381	97.19%
9	717915	12722	98.23%
10	217363	7196	96.69 %
Average	429836	9862	97.46 %

2.5 Conclusion

In conclusion, we proposed a strip based object map generating algorithm, which reads in raster image pixel-streams and produces one strip of object map at a time. The generated object map can be used for object-oriented halftoning to achieve better print quality by rendering raster and symbol objects with the high-frequency screen, and vector objects with the low-frequency screen. The algorithm is very hardware friendly, which processes pixels in raster order. To achieve memory efficiency, we recycle the unneeded components and labels at each row. Compared to the classic CCL algorithm, our proposed algorithm reduces the number of components by an average of 97.56%.

Because of our limited resources, we are unable to print these raster pages by using the object maps generated by our proposed algorithm and evaluate the print quality. By visually comparing the object maps generated by our algorithm with the ones generated by the non-strip based algorithm [29], we do not see too much difference. Hence, it's difficult to judge which object map generating algorithm is better. This work is more focused on the performance side – how could we reduce the memory and make it more hardware friendly for an ASIC application. The notion of the strip based process and the novelty of the proposed data structure for the CCL can be used for variety of other applications as well.

3. ON-LINE PRINT QUALITY DIAGNOSTIC SYSTEM

Page quality (PQ) is most important in printing industry – it directly plays a role in users' satisfaction about their products. Page quality is degraded when PQ defects appear on the page, which could be caused by the EP process and associated print mechanism. To identify a PQ issue, customers have to consult a printer user manual or contact customer service to describe the problems, which can be very challenging [6]. Some web-based troubleshooting tools are also developed to allow customers solve the PQ issues by themselves [10] [11] [12] [13]. However, all these PQ diagnosis methods are too costly for customers. Instead, we want a system that can monitor the PQ automatically in the background even without customers knowing. Such system is proposed in Figure 3.1.

The proposed system requires a scanning device, like a scan bar which can be installed at the output end of a printer. As every page is printed from the printer, it will also automatically get scanned by the scanner. Print defects such as banding, streaking, etc. will be reflected on the scanned page and can be captured by comparing to its master image. The master image is the raster image generated from the RIP which can be extracted from printer firmware. The print defect detection algorithm monitors the PQ in the background. Once the print quality drops below a specified acceptance criteria level, the system notifies the user of the presence of print quality issues. Based on the types of print defect detected, the system can also predict the failure component in the printer that needs to be replaced. All these data will be pushed to the cloud, which can be obtained by customer service. Customer service will contact customers with these diagnostic data to provide the right and more specific help.

The system can be implemented on an ASIC embedded inside a printer along with the printer firmware control. As shown in Figure 3.2, the system is listening

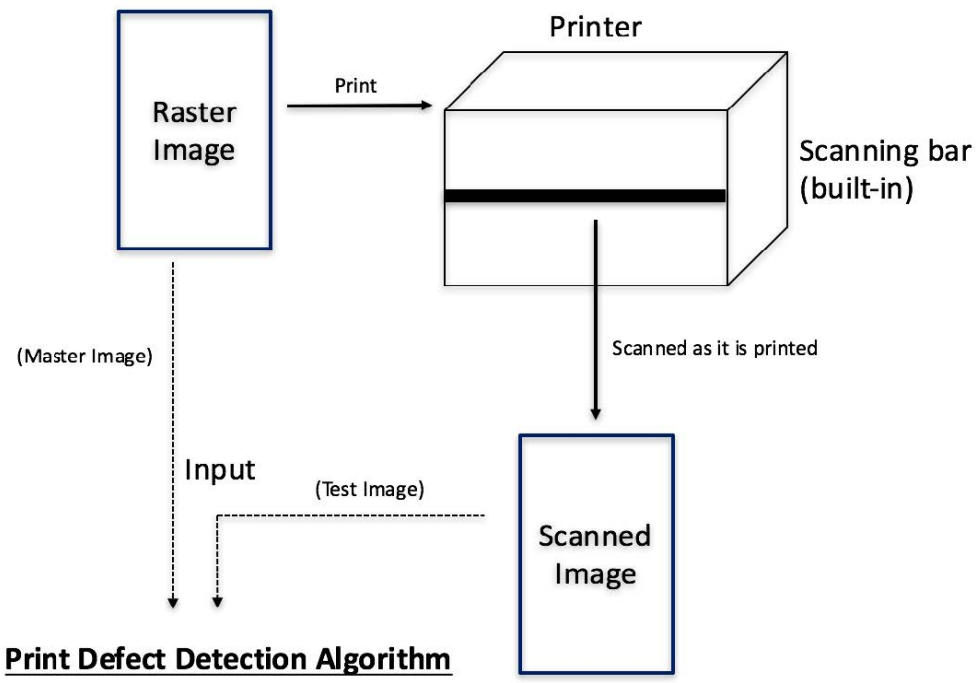


Fig. 3.1. Real time print quality diagnostics system.

to the printer firmware, and an event will be received by the system whenever a print job is issued. Once the event is received, the scanner is initialized to get ready to scan the printout coming out from the printer. The scanned page long with the calibrated master image retrieved from the printer firmware will be processed by the image processing block. It is possible that the image processing block takes much longer than the page throughput for multi-page jobs so that as the current page is being processed, a new page arrives before the system is ready. This can be solved by disabling the system from listening to the firmware until the current analysis is finished. This means that even though the customer's print quality is always being monitored, not every page needs to be scanned and analyzed.

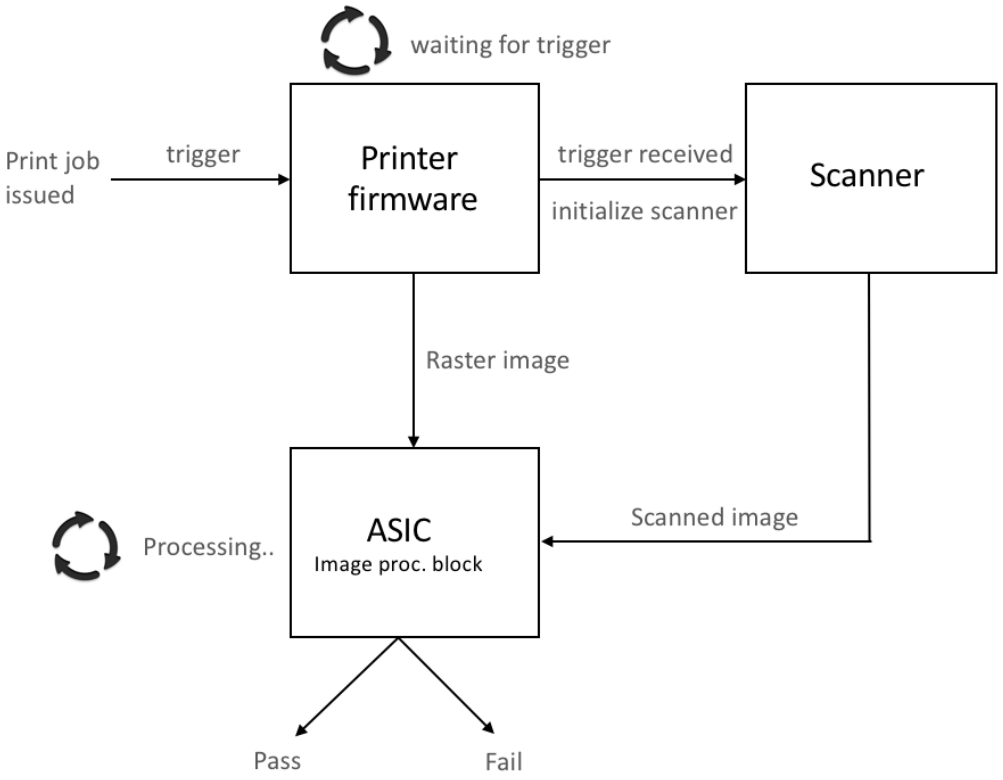


Fig. 3.2. Signal control flow of the PQ diagnostics system.

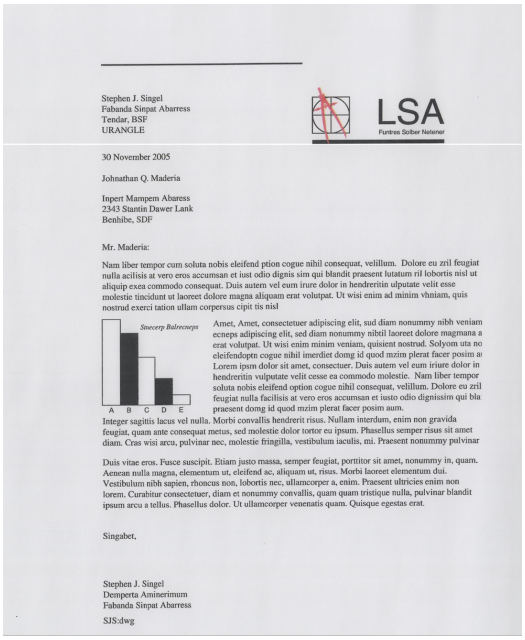
Even the print defects can be shown on the scanned image, before we compare it with the master image the scanned page may be subject to translation, scaling, skewing. Therefore, we have to spatially align these two pages first which is done by our image registration algorithm. In the next chapter, we will introduce a feature based image registration algorithm.

4. IMAGE REGISTRATION

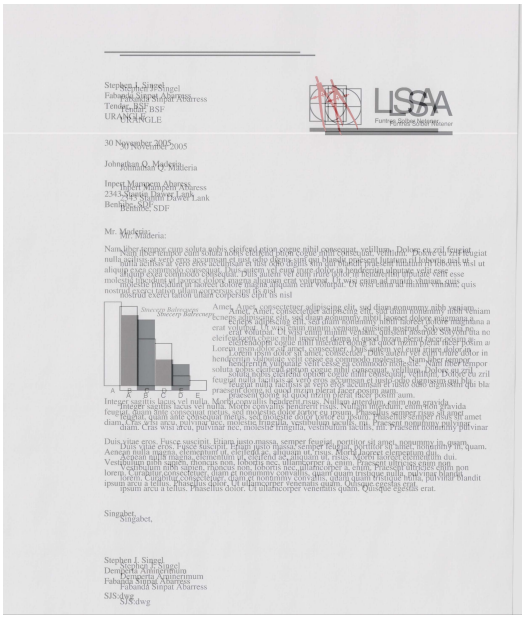
4.1 Introduction

When a page is printed from a printer and goes into a scan bar, it is very likely that the scanned image is misaligned with the master image. An example is shown in Figure 4.1, where the first two images are the master and the scanned images, and the last image is the overlaid image. This misalignment should be corrected by an image registration algorithm before we do any comparison. There are various approaches for image registration. The most common one is the frequency based approach, which explores the phase correlation of two images in Fourier domain [56]. However, due to the halftone pattern and the PQ defects on the scanned image, this method is not robust. A feature-based image registration algorithm is used instead.

(a) Master image.



(b) Scanned image.



(c) Overlaid image.

Fig. 4.1. The scanned image (b) is misaligned with the master image (a). (c) is the overlaid image that shows the misalignment.

4.2 Methodology

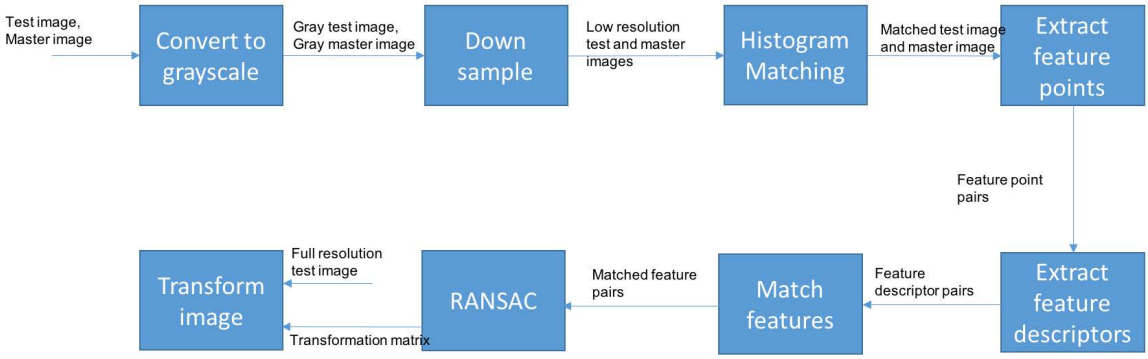


Fig. 4.2. Feature-based image registration algorithm pipeline.

The pipeline of our feature-based image registration algorithm is in Figure 4.2. The scanned image and the master image are converted to grayscale first, and then they are downsampled to a lower resolution for faster computation. Since the scanned page could have PQ defects such as fading, which may have intensity or colors different from the master image, a histogram matching is used to color balance the scanned page with the master page. It has been tested that the histogram matching can greatly improve the accuracy of the feature matching stage. Feature points are then extracted from these color balanced low-resolution grayscale images. There are various types of feature points can be used; the most simple and basic feature point detector is Harris corner [57]. The calculations of Harris corners are summarized as the expressions in Figure 4.3, where I is the input image; I_{x} and I_{y} are the partial derivatives with respect to x and y; G is the Gaussian filter; K is the sensitivity factor; CSF is the corner strength function. $CSF(x, y)$ will be large when the gradient along all directions is large, and it is close to zero at a smooth area. Points whose corner strength are larger than a threshold are selected as feature interest points. The locations of the Harris corners are found in sub-pixel accuracy [58]. The sub-pixel accuracy ensures that the Harris corners found are at the exact locations, instead of the rounded integer positions which have low precisions. The main drawback of Harris corners is that it fails to deal with scale changes [59]. Nevertheless, this is not a problem here; since the scanned image is printed from the master image, it should always be at the same scale as the master image.

After acquiring the feature points, feature descriptors can be extracted at each feature point. If we assume that the test image is skewed by only a small angle with respect to the master image, we could simply extract a block of pixel centered at each feature point as the feature descriptors. Notice this feature descriptors work poorly when the skew angle is large because two feature descriptors are compared based on calculating the pixel to pixel difference. The extracted feature descriptors can be matched with the sum of the squared differences (SSD), and SSD is preferred when there is a small variation in intensity and color between images [60]. However,

$$I_x = I * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}; I_y = I * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}';$$
$$I_x I_y = I_x \cdot I_y;$$
$$I_x^2 = I_x^2 * G;\ I_y^2 = I_y^2 * G;\ I_x I_y = I_x I_y * G;$$
$$CSF(x,y) = (I_x^2 \cdot I_y^2 - (I_x I_y)^2) - k \cdot (I_x^2 + I_y^2)^2$$

Fig. 4.3. Calculations of Harris corners. I is the input image. The output is the CSF (corner strength function), which indicate the aggregated gradients of all the pixels.

the test image could be a faded image, and this is the reason why we use histogram matching at the earlier stage to match the intensity and color of the scanned image with the master image. Another difficulty to achieve accurate matching happens when the scanned image and master image contain many text characters. A text character that is on top of a test image may be matched with same text character that appears in the middle or bottom of the master image. This can be solved by limiting the spatial distance between the feature pairs being matched. This is illustrated in Figure 4.4. For each feature point in the master image, instead of searching for the feature point with the minimum SSD over the whole scanned image, we only search within a local window to find the best-matched feature point.

The matched feature pairs are a set of 2D coordinates. A geometric matrix can be used to transform the feature points coordinates of the scanned image. If we only take skew angle and translations along x and y into account, such matrix with three degrees of freedom is showing in Equation 4.1. Ptm are the Pts feature points coordinates of the master image and the scanned image respectively. The skew angle parameter is θ; dx and dy are the translation offsets. With three pairs feature points, one unique solution can be solved for this matrix. For our problem which we have a large number of matched feature pairs, it becomes an overdetermined problem. This overdetermined problem can be solved by RANSAC (random sample consensus) [61], or the more robust MLESAC algorithm (maximum likelihood estimation sample consensus) [62],

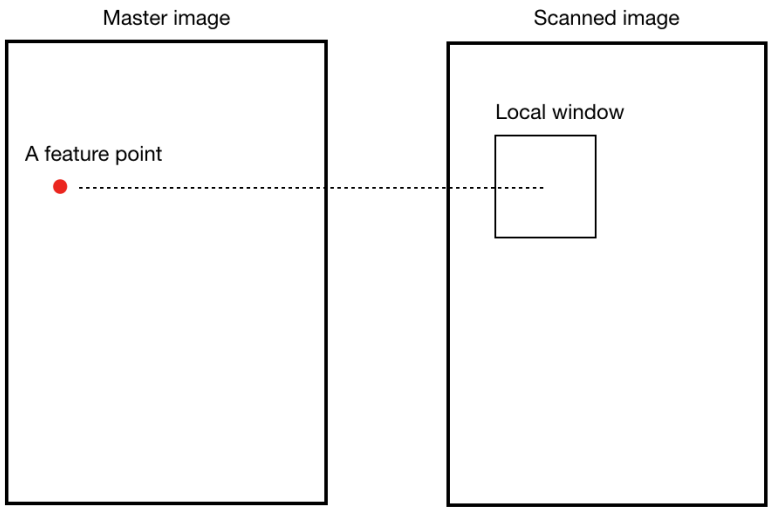


Fig. 4.4. Local matching of feature points. For each feature point in the master image, a best match is found locally from the scanned image.

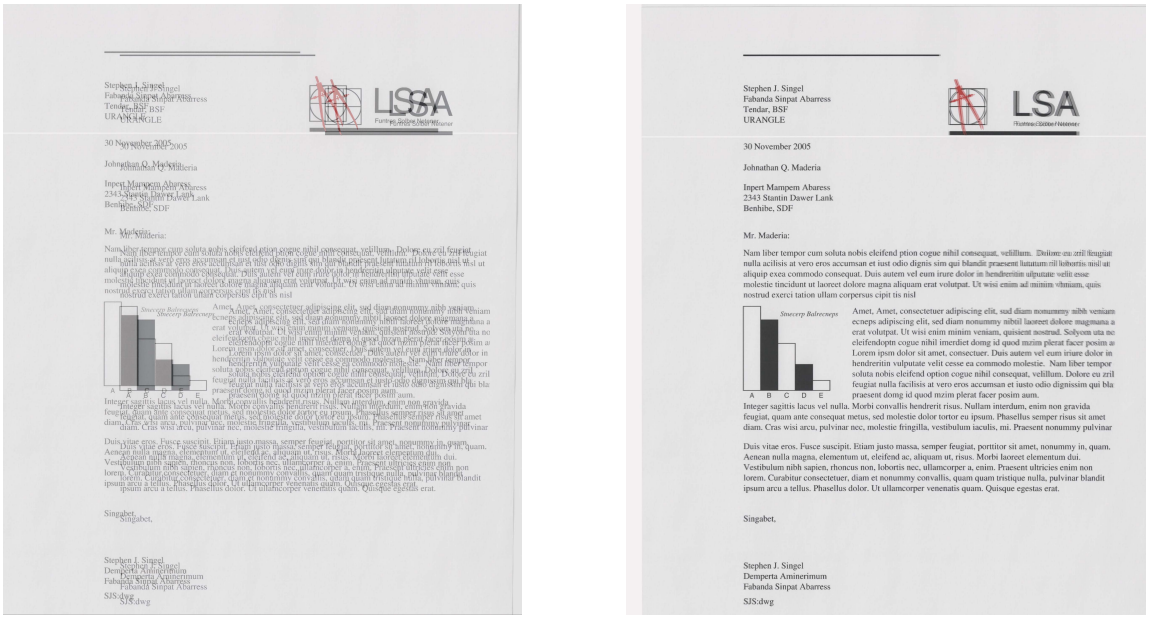
which seeks to minimize the distance between the predicted coordinates and the actual coordinates through an iterative optimization process. Note that the geometric matrix is estimated based on the downsampled images. Eventually, if we want to apply this matrix to the original full resolution scanned image, the translation offsets dx and dx have to be scaled up by a factor of the downsampled rate.

$$\begin{bmatrix} Ptm_x \\ Ptm_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & dx \\ \sin(\theta) & \cos(\theta) & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Pts_x \\ Pts_y \\ 1 \end{bmatrix} \tag{4.1}$$

4.3 Experiment Result

Figure 4.5(a) shows a scanned image and its master image overlapping with each other before image registration algorithm applied. They both have a resolution of about 6400×4928 at a spatial resolution of 600 dpi. It can be seen from Figure 4.5 that they are misaligned both vertically and horizontally, and the scanned is slightly

skewed. The test image and the master image are converted to grayscale and then down sampled by 4 in both directions. This results in two 800×600 low-resolution images as shown side by side in Figure 4.6. Feature points are extracted from both lower resolution images, and the feature descriptors are matched. The red dots and green circles on Figure 4.6 are feature points, and the feature pairs connected by yellow lines are the feature matched results. Even though some pairs are mismatched, these outliers can be rejected by the RANSAC or MLESAC algorithm. Figure 4.5(b) shows the transformed full resolution scanned image overlapped with the master image after the image registration. Besides this image, we have also run our algorithm on hundreds of other images which are not shown here.



(a) Before alignment.

(b) After alignment.

Fig. 4.5. Overlapped images before (a) and after (b) misalignment.

The image registration algorithm is served for text fading detection which will be discussed in the next chapter. So, to evaluate our image registration algorithm, we are most interested in how the text characters are aligned after we apply our image

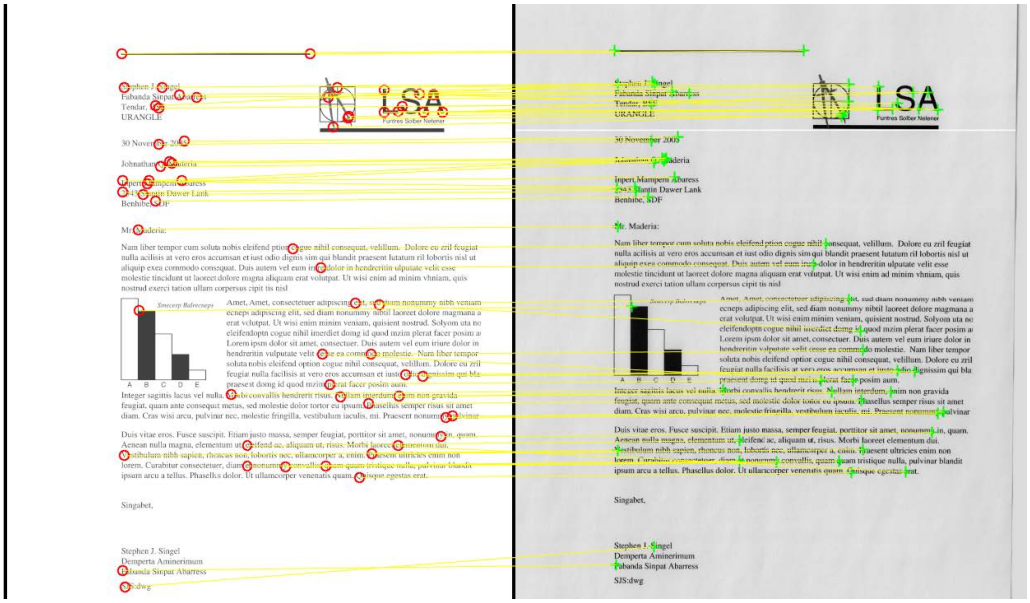


Fig. 4.6. Feature matching result. Red and green symbols are extracted feature points. Each yellow line connects each matched pair of feature points.

registration algorithm. We first extract all the text characters from the master image; the method of how to do this will be introduced in the next chapter. Then for each text character in the master image, we locally search the best match in the scanned image by using template matching method. The searching range is made as large as twice the template size. The spatial distance between the template and the best local match will represent the image registration errors. This process is carried out for all the text characters in the page, and finally, we calculate the mean, standard deviation, and 97% percentile. The 97% percentile represents the worst case for a page. The reason why do not use the maximum error is that the text character extraction method is not perfect. Most of the time, some contents other than text characters or noisy pixels are also extracted and mistreated as text characters, and template matching method could not give accurate results. The results for six test pages are tabulated in Table 4.1. All these pages have a resolution of 6400×4928 pixels, and there are different kinds of PQ defects on these pages. On average, we can

achieve an accuracy of 2 to 3 pixel difference for the mean and standard deviation, 7 to 9 pixel difference for the worst case. This evaluation method only accounts for the translation offset along x-axis and y-axis. Since the skew angle after image registration is very small, it is not taken into consideration.

Table 4.1.
The statistics of pixel difference [x,y] between all text character pairs for six test pages.

Test Page	# of text chars	Mean	Std	97% percentile
1	2917	[3.3, 3.8]	[2.4, 2.9]	[8, 9]
2	2439	[1.6, 2.9]	[2.4, 3.3]	[6, 11]
3	1723	[2.9, 4.5]	[2.1, 3.0]	[6, 9]
4	1585	[2.9, 3.7]	[2.0, 2.9]	[7, 10]
5	1876	[3.3, 3.7]	[2.6, 2.9]	[9, 10]
6	2062	[1.6, 2.5]	[1.8, 2.2]	[4, 6]
Average	2100	[2.6, 3.5]	[2.2, 2.9]	[7, 9]

4.4 Conclusion

The image registration algorithm is a global process, which is impossible to achieve pixel to pixel perfect alignment. One reason is that the halftone pattern, color difference, and intensity difference can increase the difficulty of feature matching. The second reason is that for each printed component, like a text character, the toners on the media may dilate due to, for example, toner overdevelopment [63], which makes each printed component slightly larger than the actual size. Consequently, the scanned page is never exactly the same as its master and perfect alignment is impossible. If we take a closer look at the top right corner of Figure 4.5(b), we can still see little misalignment. For text characters, they can be further aligned with

a local alignment method, after which, they are ready for comparison. This will be discussed in details in the next chapter.

5. TEXT FADING DETECTION[1]

5.1 Introduction

Fading can occur in customer pages with mixed contents, and the most common scenario is pure text documents. Therefore, this chapter mainly deals with the fading defect in the text regions only. For the fading occurred in the non-text regions, it will be addressed in the next chapter. With the proposed on-line PQ diagnostic system in hand, we can detect text fading by comparing the scanned image with the master image. However, as discussed previously, the image registration algorithm could not achieve perfect pixel to pixel alignment. For this reason, we have to extract all the text characters and locally align them first. The aligned text characters are then directly compared by calculating the color differences. We will see that the histogram of these color differences will appear diverged if some of the text characters are faded. If there is no fading, or all the characters are faded, the histogram will become more concentrated. We can then use the mean about the distribution as the indicator of how badly this page is faded.

5.2 Methodology

5.2.1 Local Alignment

The algorithm flow of the local alignment is illustrated in Figure 5.1. Images are binarized with an adaptive thresholding method, followed by a morphological operation to remove noisy pixels. Foreground pixels are connected by a connected component algorithm, and text characters are then extracted based on the criterion of, for example, bounding box size, number of pixels, etc. Note that only the text

[1]PATENT PENDING

characters from the master image are extracted, and for each extracted text character, it will be used as a template to find a match in the scanned binary image inside a localized range. This searching range entirely depends on how accurately these two images are aligned from the global image registration algorithm. Based on the worst cases of the test results in Table 4.1, the searching window is selected as 20×20.

Figure 5.2 shows the local alignment result of part of a binary image. The magenta is the master binary image. As we overlay the image with the scanned binary image which is represented by the cyan color, we can see that they are off by few pixels. After local alignment, the text characters of the master binary image are now completely overlapped with the text characters of the scanned binary image; blue color represents the overlapping area. Even though the local alignment is done on binary images, we extract the locally aligned text characters from the original RGB images for comparison. At the end of this process, we will have a list of components of text characters, and each component contains a pixel list that can be used for pixel-wise color comparison.

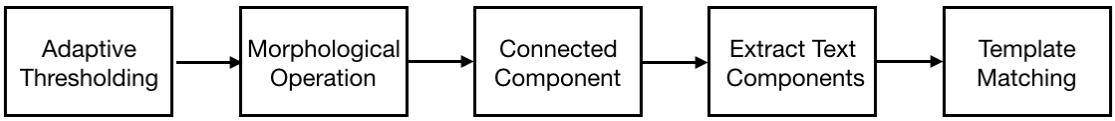


Fig. 5.1. Local alignment of text characters.

(a) Before local alignment. (b) After local alignment.

Fig. 5.2. Before (a) and after (b) local alignment of each text character. Cyan is the scanned binary image; magenta is the master binary image; blue is where they are overlapping.

5.2.2 Color Difference

To compare the difference between two colors, the most commonly used metric is ΔE, which is the Euclidean distance between two points in $CIEL^*a^*b^*$ color space. The pixel list extracted previously is in scanner calibrated RGB and will be converted to $CIEL^*a^*b^*$ first, and then ΔE is calculated for each pixel in the list. We take the mean of all the ΔE in this list to represent the average perceptual difference between two text characters. These calculations are formulated in Equation 5.1, where subscripts m and s represent master and scanned text character respectively, and N is the number of pixels inside a text character.

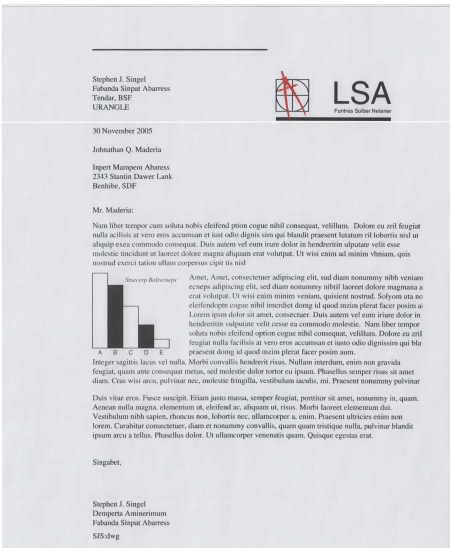
$$\Delta E_{char} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(L^*_{mi}-L^*_{si})^2+(a^*_{mi}-a^*_{si})^2+(b^*_{mi}-b^*_{si})^2} \tag{5.1}$$

5.2.3 Statistical Analysis

After calculating the color error of each text character, we can plot the histogram of all the color errors. An unfaded page, like Figure 5.3(a), will exhibit a very narrow spread histogram as shown in Figure 5.3(b). In contrast, when fading defect exists, like the sample in Figure 5.3(c), its histogram is pulled to the right by those faded text

characters and becomes a bimodal distribution as shown in Figure 5.3(d). However, as the fade get more and more server till the whole page is faded, we will expect the histogram becomes less dispersed again because all the text characters have large color errors. As the histogram spread to right, the mean of the histogram also increases. Therefore, we can just use the mean of the color errors as the measure of the fading level.

Consider the situation when we only have a very narrow strip of text characters that are faded, then the mean of the color errors in a large degree will be dominated by the larger population of those non-faded text characters. Fading defect can be more easily detected if we divide a page into many small regions, for example, strips, and then analyze the color errors locally for each strip region.



Stephen J. Singel
Fabanda Sinpat Abarress
Tendar, BSF
URANGLE

LSA
Furinss Sofler Nelenar

30 November 2005

Johnathan Q. Maderia

Inpert Mampem Abaress
2343 Stantin Dawer Lank
Benhibe, SDF

Mr. Maderia:

Nam liber tempor cum soluta nobis eleifend ption cogue nihil consequat, velillum. Dolore eu zril feugiat nulla acilisis at vero eros accumsan et iust odio dignis sim qui blandit praesent lutatum ril lobortis nisl ut aliquip exea commodo consequat. Duis autem vel eum iriure dolor in hendreritin ulputate velit esse molestie tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim vhnisim, quis nostrud exerci tation ullam corpersus cipit tis nisl

Stuscerp Bdrevnepu

A B C D E

Amet, Amet, consectetuer adipiscing elit, sad diam nonummy nibh veniam ecneps adipiscing elit, sed diam nonummy nibtil laoreet dolore magnana a erat volutpat. Ut wisi enim minim veniam, quisient nostrud. Solyom uta ne eleifendoptn cogue nibil imerdiet domg id quod mzim plerat facer posim at Lorem ipsum dolor sit amet, consectuer. Duis autem vel eum iriure dolor in hendreritin vulputate velit cesse ea commodo molestie. Nam liber tempor soluta nobis eleifend option cogue nihil consequat, velillum. Dolore eu zril feugiat nulla facilisis at vero eros accumsan et iusto odio dignissim qui bla praesent domg id quod mzim plerat facer posim aum.

Integer sagittis lacus vel nulla. Morbi convallis hendrerit risus. Nullam interdum, enim non gravida feugiat, quam ante consequat metus, sed molestie dolor tortor eu ipsum. Phasellus semper risus sit amet diam. Cras wisi arcu, pulvinar nec, molestie fringilla, vestibulum iaculis, mi. Praesent nonummy pulvinar

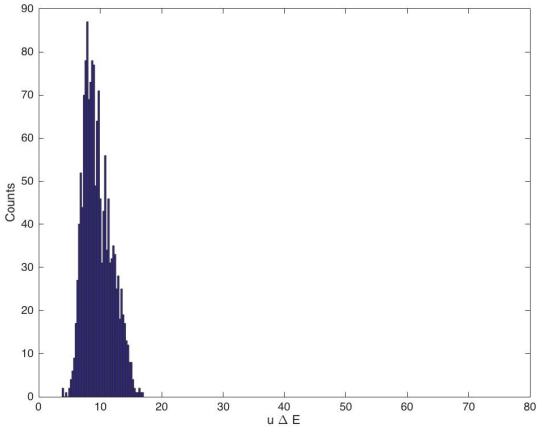
Duis vitae eros. Fusce suscipit. Etiam justo massa, semper feugiat, porttitor sit amet, nonummy in, quam. Aenean nulla magna, elementum ut, eleifend ac, aliquam ut, risus. Morbi laoreet elementum dui. Vestibulum nibh sapien, rhoncus non, lobortis nec, ullamcorper a, enim. Praesent ultricies enim non lorem. Curabitur consectetuer, diam et nonummy convallis, quam quam tristique nulla, pulvinar blandit ipsum arcu a tellus. Phasellus dolor. Ut ullamcorper venenatis quam. Quisque egestas erat.

Singabet,

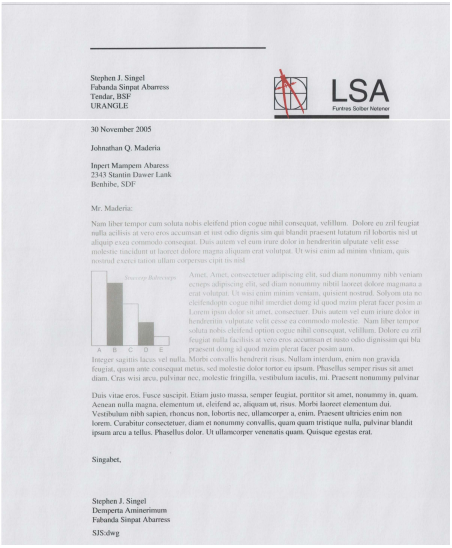
Stephen J. Singel
Demperta Aminerimum
Fabanda Sinpat Abarress

SJS:dwg

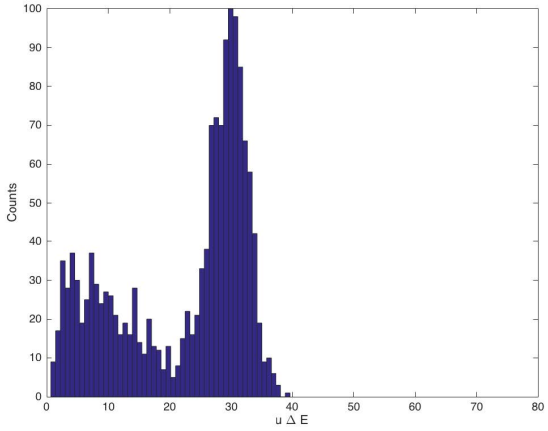
(a) A unfaded image.



(b) The histogram of the unfaded image.



(c) A faded image.



(d) The histogram of the faded image.

Fig. 5.3. Test images and the histograms of color errors.

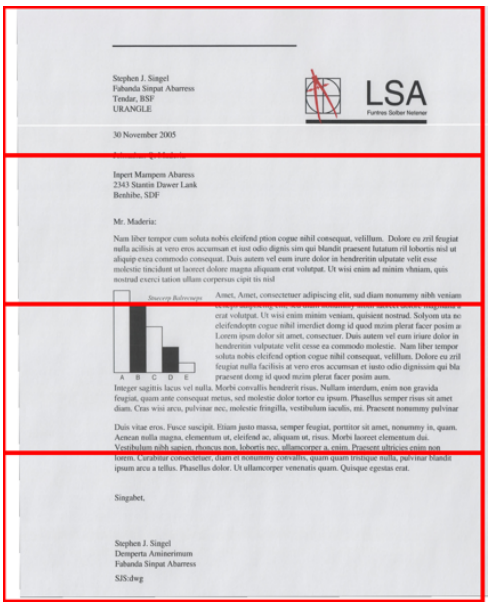
5.3 Experiment Result

A series of test pages with increasing level of text fading are shown in Figure 5.4(a)(c)(e)(g). For visualization purpose, all the test pages are divided into four horizontal strips. In practice, we split them into eight strips or more depending on the type of page being analyzed. The histogram plotted to the right of each test page is the distribution of the color errors of all the text characters within each strip compared to the master image in Figure 4.1(a). The first page, Figure 5.4(a) is clearly an unfaded page. As we can see from its histograms in Figure 5.4(b), they are all very narrow, and their peaks are located near their means, which are all less than about 11 ΔE. When fading arise in the middle two strips, shown in Figure 5.4(c), the middles two strips start to spread to right, and their means become larger. For the strips that there is no fading, their histograms remain almost unchanged. As the fading level increases, the histograms will further shift to the right. For the last test page, Figure 5.4(g), almost all the text characters in the middle two strips are faded.

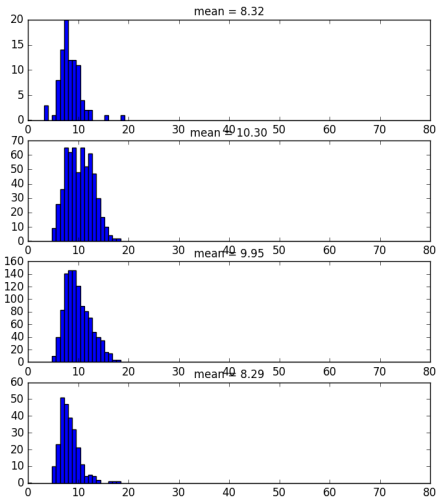
The histograms in Figure 5.4(h) are less dispersed compared with Figure 5.4(f), and the new peaks are located around 35 ΔE.

To determine the threshold of a fading point, we collected a set of images with gradually increasing degrees of fading. This can be done by keep printing the same document starting from a full cartridge until the cartridge is completely out. Several experts are then asked to sequentially examine each printed page until a page is noticeably faded. We then calculate the mean of the color errors of all the text characters for this page. The results are averaged among all the experts. This average will be used as the threshold, and any page above this threshold is considered as a faded page, otherwise, a non-faded page.

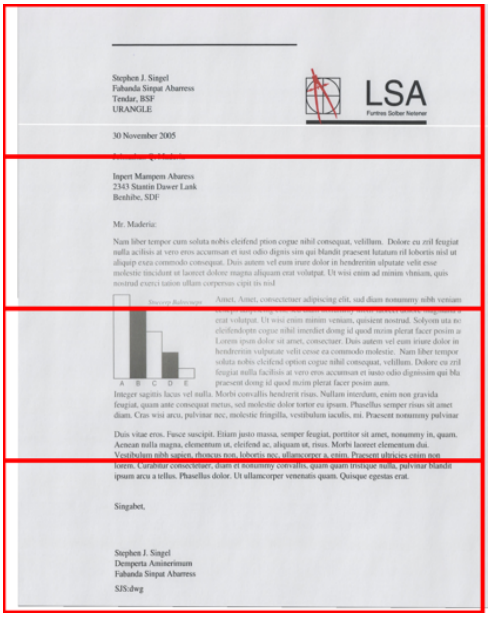
This system is implemented in Python on a Linux workstation along with a laser printer and a built-in scan bar of 600 ppi resolution. It can scan and process a full letter-sized text page in about 5 to 7 seconds. A GUI is also created, and it is shown in Figure 5.5. Whenever user prints a page, it automatically pulls out the master image from printer firmware and displays it in the left pane of the GUI. After the print gets scanned, it will be showing in the right pane of the GUI. Different levels fading are represented by a heat map to indicate the PQ level of each strip on the scanned image. When the fading becomes too severe, a text message is sent to a user's phone from the app to warn the user the presence of PQ issues of the printer.



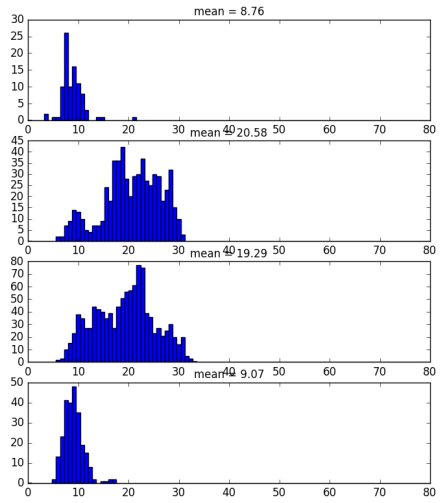
(a) Test page 1.



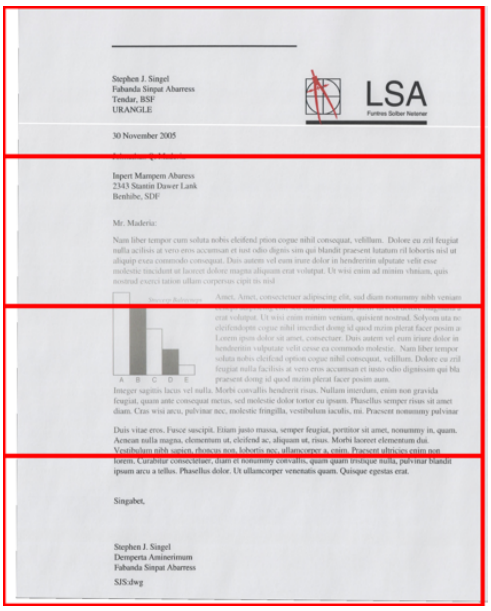
(b) Histogram 1.



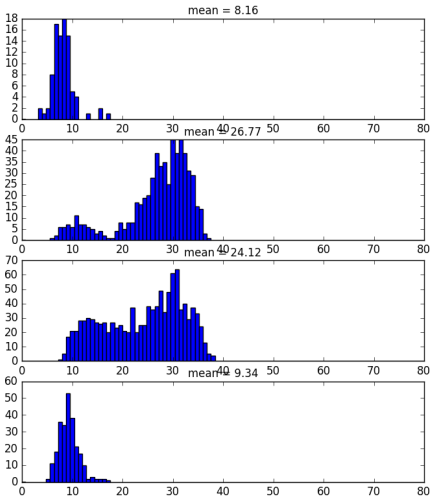
(c) Test page 2.



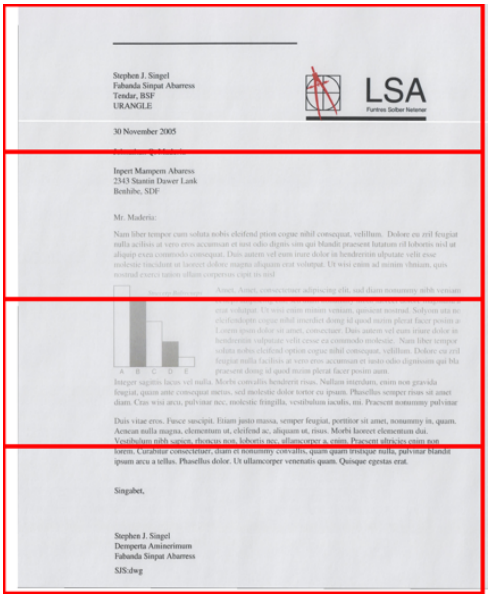
(d) Histogram 2



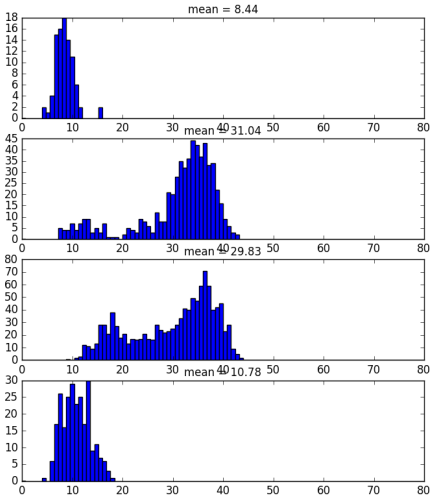
(e) Test page 3.



(f) Histogram 3.



(g) Test page 4.



(h) Histogram 4.

Fig. 5.4. Test images with increasing fading levels and the local histograms of color errors.

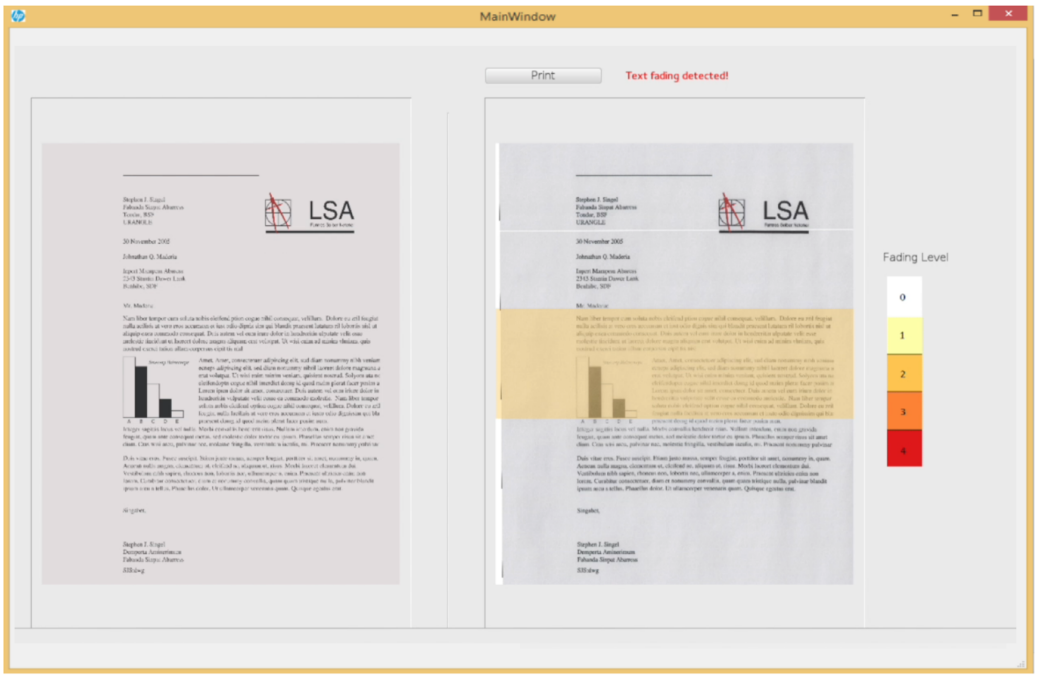


Fig. 5.5. GUI for text fading detection. The left panel displays the master image, and the right panel displays the scanned image. The fading level of a strip is indicated by a heat map.

5.4 Conclusion

In conclusion, this chapter proposed a text fading detection algorithm which compares the scanned image with a master image. After we first aligned the scanned image with the master image, we applied local alignment to all the text characters and calculated the color errors between them. We then used mean of all the errors to determine if a page is faded or not by comparing to a threshold, which is obtained through a psychophysical experiment.

This algorithm is on the basis of Ju's [17] work. In her work, in order to spatially align the scanned image with the master image, she put four fiducial marks on the four corners of the master image. By using these four fiducial marks, the two images can be easily aligned. However, in the real-life application, we can not just modify the mixed content pages that customers intend to print, and that's why we resort to

a feature based image registration algorithm. Furthermore, instead of comparing the text characters of the scanned image to the master image, she compared to paper white. For a faded text character, it will have a smaller color error compared to paper white and a larger color error for a non-faded text character. Unfortunately, a major problem with this approach is that it is based on the assumption that all the text characters on the page are uniform solid black colors. Imagine if the text characters on the master image have different shades of gray, then those light grays text characters will be treated as faded ones. Our proposed method will not suffer this problem since we are always comparing against a reference.

6. DETECTION OF COLOR FADING IN PRINTED CUSTOMER CONTENT [1]

6.1 Introduction

The fading detection algorithm proposed in the previous section is for text regions only, since text characters can be easily extracted with connected component analysis, and achieve almost perfect alignment with template matching method for direct pixel to pixel comparison. However, the calculated color errors cannot give us any information about the condition of a particular cartridge for a color print job. Color error increases when no matter whichever C, M, Y, K cartridge is depleted. It is of interest to know which cartridge it is so that it can be replaced right away. What's more, a customer page could contain mixed contents rather than text alone. Figure 6.1(a) shows an example of a mix contents page, which we do not only have text characters but also raster regions (detailed areas) and vector regions (smooth areas). Figure 6.1(b) shows a faded page caused by the low toner in both magenta and cyan cartridges. In any case, we want our algorithm to be able to detect color fading in the raster regions and vector regions as well, especially for the situation that a raster page does not contain any text character.

Due to the imperfection of the global image registration algorithm, there is still little misalignment for those raster and vector regions, and we could not adopt the same approach as we do for text characters. To deal with this problem, we first calculate the superpixel – the average of the pixels within a block – of both the master image and the scanned image to reduce the impact of this misalignment. Then we cluster all these superpixels based on their colors in a perceptual uniform color space. Each color cluster on the scanned image will be compared with a one on the master

[1]PATENT PENDING

(a) A raster page with mixed contents.

(b) A magenta&cyan faded page.

Fig. 6.1. A raster page (a) and a faded page (b) when magenta and cyan cartridges are running low on toner.

image by calculating the color difference between them. When fading occurs, we will see that some color clusters will significantly raise their color errors. By analyzing how these color clusters change their colors on the scanned image, we could predict which cartridge is depleted. For example, when skin tone looks very cool on Figure 6.2(b) compared with the master image Figure 6.2(a), we may infer that toner is low in the yellow cartridge. However, if we analyze the black color cluster, in which pixels are mainly from the hair of the lady, we can hardly see too much difference between them. A color fading detection algorithm is proposed in this section that will formulate all these analyses to predict which cartridge is low on toner based on each color cluster pair and then sum all the predictions result. By using the majority rule, we can confidently report or warn customer the low toner cartridge that needs to be replaced.

(a) A raster image with different tones.

(b) A yellow faded page.

Fig. 6.2. An example when yellow cartridge is running low on toner. The skin tone in (b) looks very cool compared to (a). However, the color of the hair remain almost the same.

The algorithm flow the color fading detection is depicted in Figure 6.3. Since our region of interest is non-text regions, we first remove all the text characters from the master image and the test image in the preprocessing step. Then we calculate the superpixels over both pages. These superpixels are later grouped into different clusters. Each cluster pair is compared by calculating the color difference between them. If the difference is large than a threshold, we predict the depleted cartridge based on this cluster pair. Next, we will explain each step in details.

Fig. 6.3. The algorithm flow of color fading detection.

6.2 Preprocessing

To compare the raster and vector regions of the scanned image with the master image, the global image registration algorithm is first applied. Since the regions of our interest are non-text regions, we could further locally align each text character with the techniques introduced in the previous section, and remove all these text characters from the scanned image. Figure 6.4(a) shows the scanned image in Figure 6.1(b) overlaid with the master image in Figure 6.1(a) before applying the image registration algorithm. After image registration algorithm, as shown in Figure 6.4(b), two images are roughly aligned. Each text character can be further aligned and then removed, which is represented by each black bounding box in Figure 6.4(c); the remaining areas will be our region of interest. Note even some pixels in the raster regions are masked out by mistake, we should still be able to detect any fading since fading is a large area defect.

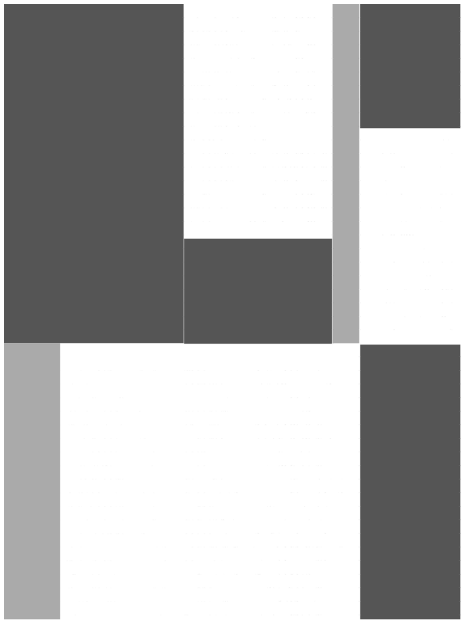
It's also worth to mention that when the master image is sent to the marking engine of a laser printer, an object map is generated and it tells the printer how this page should be rendered. A such object map of the master image Figure 6.1(a) is shown in Figure 6.4(d). Three types of objects are commonly seen in a customer image: symbol, raster, and vector. Symbol objects are mainly text characters and symbols, which are already removed from the region of interest, and from the object map. Raster objects are those rough regions that contain many details, which are represented by the dark gray in this object map. Light gray regions are vector objects, which are smooth areas. When a master image is rendered, it is first converted from $sRGB$ color space to a $CMYK_{printer}$ color space, which is a printer dependent color space. The conversion uses a look-up-table (LUT) which maps from a RGB value to a $CMYK$ value, and there is no way to convert it back. Since raster objects and vector objects are rendered with different frequency of screens, two different LUTs are also used for them. This implies that a color in the raster region can have completely different $CMYK$ compositions from a color in the vector region even they have the

same RGB value. For this consideration, we split all of our images into two parts – raster region and vector region – according to the object map. To avoid confusion, all the following processes will be done for these two kinds of regions separately without further clarification.

(a) Overlaid images before image registration.

(b) Overlaid images after image registration.

(c) Scanned images after removing text characters.



(d) Object map of the master image.

Fig. 6.4. Preprocessing; Scanned image is first aligned with the master image. Text characters are removed from the region of interest after local alignment. Images and then partitioned according to the object map.

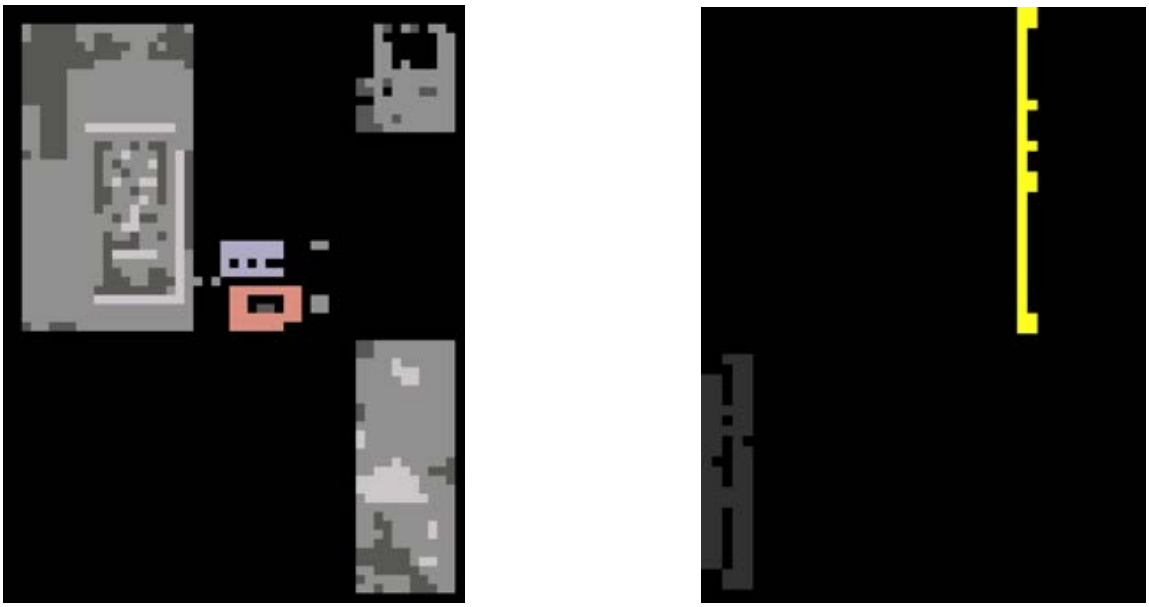
6.3 Extract Color Clusters

To detect color fading in the region of interest, we could not naively calculate the difference image: as mentioned earlier, the alignment is not perfect, and the misalignment can be ten-pixel difference in the worst case. To compensate for this alignment, we first calculate the superpixels by averaging all the pixels in the region of interest within each 100×100 size non-overlapping block. Because the images are in $sRGB$ which is a nonlinear color space, the images are converted to a linear and perceptually uniform color space, such as $CIEL^*a^*b^*$ color space; the calculations of the superpixels are then performed in this color space. Next, we group all the superpixels in the master image to different clusters by using mean shift clustering

technique [64] in the $CIEL^*a^*b^*$ color space. After all the superpixels on the master image are clustered, the cluster ID of a superpixel on the master image will be assigned to the superpixel on the scanned image at the same location; this is how all the superpixels in the scanned image are clustered. Now each color cluster contains a set of similar colors of superpixels in the $CIEL^*a^*b^*$ color space, and we only take the centroid of the cluster, or equivalently the mean of all the $L^*a^*b^*$ values within this cluster to represent this color cluster. When a particular cartridge runs out of toner, we expect that some color cluster centroids will yield large color errors compared to the master, some will not. Figure 6.5 shows the color clusters generated by the mean shift algorithm for raster region and vector region of a master image, and we are only showing the top 5 color clusters ranking according to their populations of superpixels. For vector region, there are only two color clusters. Superpixels in the same cluster are represented by the same color code in the image, which is their centroid. The scanned image has the similar color map which is not shown here. The only difference is that the calculated color cluster centroids are different.

There are two main benefits of adopting this approach. First, grouping all the superpixels to color clusters can reduce the impact of misalignment. A superpixel represents the average color of a block of pixels which is less affected by the minor populations of those misaligned pixels. Furthermore, after we have clustered all the superpixels, we do not care about where each color cluster is located on the page anymore, and we only care about how these color clusters change their colors when fading occurs. The second benefit is that the computation cost is tremendously lowered. The page samples we run for our algorithm are letter size pages printed and scanned at 600 ppi resolution; the resulting digital scanned pages have size around $6400\times4928\times3$ pixels. This approach avoids comparing pixel by pixel between the master and the scanned image over the whole page. Since there are so many pixels that have the same or similar colors, it is extremely inefficient to analyze them over and over again. By clustering them, we can just examine some dominant color clusters which have relatively large populations of superpixels. For those small population

clusters, we can also safely skip them, because they do not contribute too much to color fading, which is a large area defect.



(a) Raster color clusters.

(b) Vector color clusters.

Fig. 6.5. Generated color clusters by mean shift algorithm for raster and vector regions. Each superpixel is replaced with the centroid of the cluster that it belongs to.

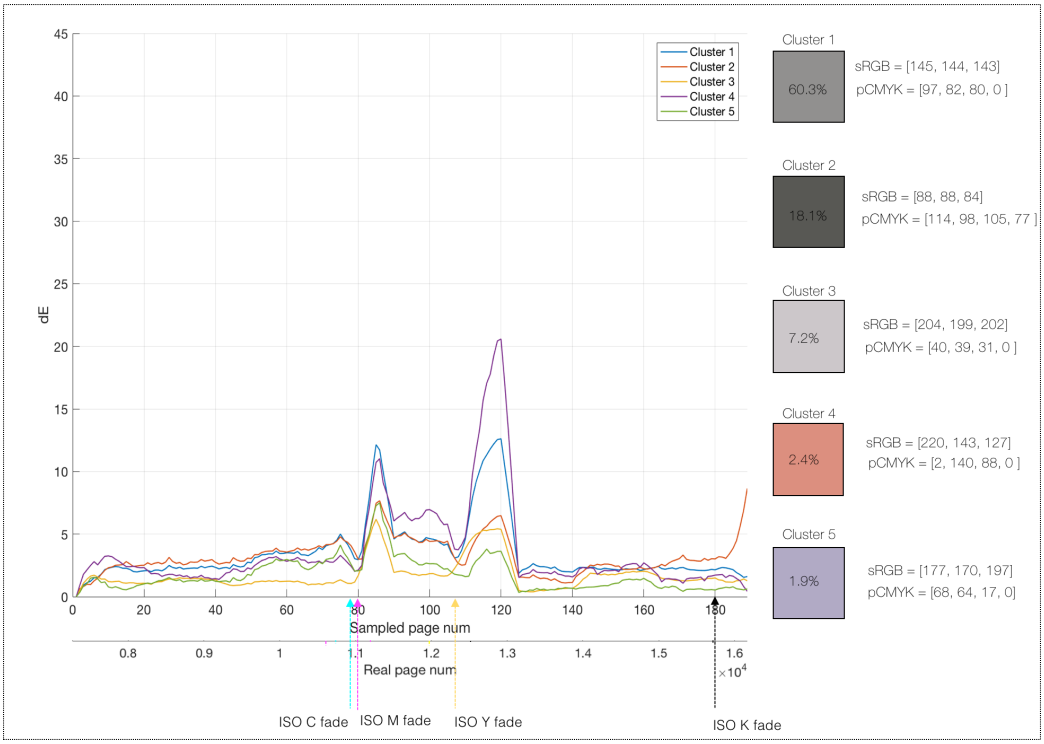
6.4 Color Difference

After we have extracted a list of color clusters from the master image and the scanned image, we can calculate the color difference between each pair of color clusters by again using ΔE as a metric. To test our algorithm, our HP partner selected a suite of test pages that are consists of several custom content pages and an ISO diagnostic page. Figure 6.1(a) is one of these customer content pages and the ISO diagnostic page the one in Figure 1.3. The pages in the suite were repetitively printed in an interleaving order. As there printed, they were also automatically scanned and organized in chronological order. During this process, we had one or more cartridge

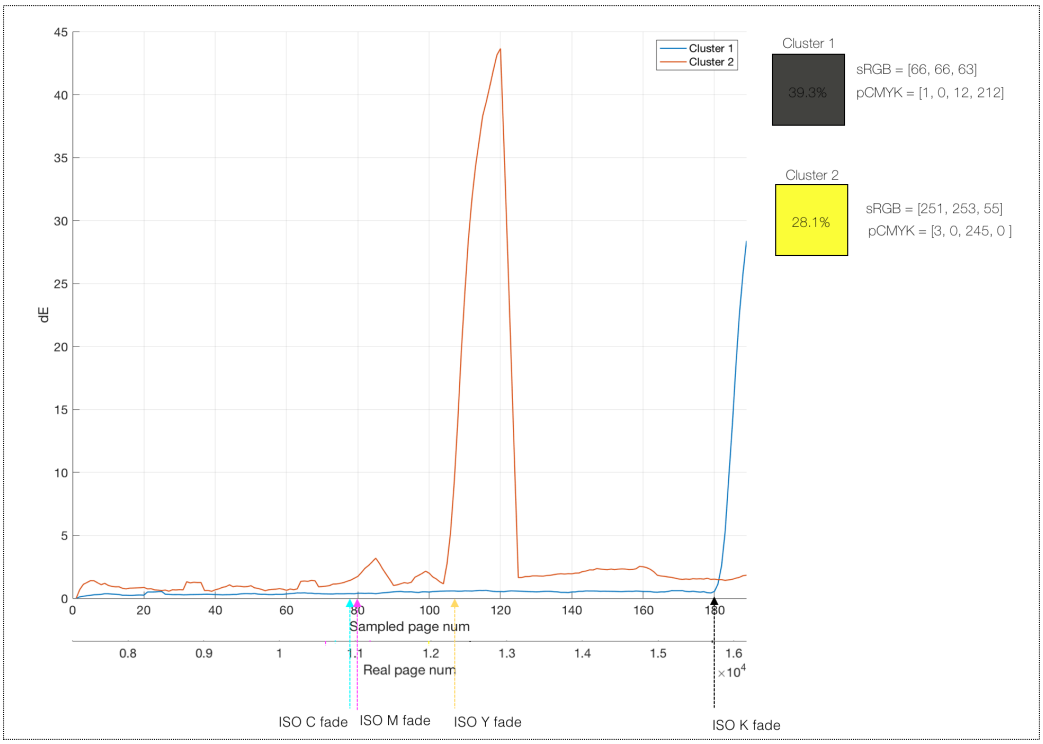
run out of toner at different pages. We did not change the depleted cartridge immediately until we obtained enough faded pages for testing our algorithm. There were around 16000 pages printed in total, and the master image Figure 6.1(a) was printed 190 times. We then collected all these 190 pages and applied our algorithm; the result is shown in Figure 6.6. In Figure 6.6, the x-axis shows both the sampled page number and the real page number.

Our ground truth is built on by asking an expert to examine the ISO diagnostic page in each printed suite. By following the standard of ISO/IEC 19798:2007, if a noticeably lighter area in the color bars of the diagnostics pages exceeds 3 mm, it is considered as fading. If the diagnostics page in a printed suite is faded, then all the customer content pages in this printed suite are also considered as faded. The ground truth for cyan fade, magenta fade, yellow fade and black fade are represented by the arrows on the x-axis in 6.6. The calculated color errors for raster color clusters and vector color clusters are shown separately in Figure 6.6(a) and Figure 6.6(b). In the plot, we have also shown the centroids of all the clusters in $sRGB$. These $sRGB$ values are consistent with the color codes used in Figure 6.5 as well. For each color cluster, we have besides shown the population, which is the percentage of superpixels who fall into this cluster, and the $pCMYK$ value, which is generated by our printer LUT. Referring to all the pages in chronological order, cyan and magenta cartridges run out of toner at about the same pages, one at 79^{th} page and one at 80^{th} page. We can see from Figure 6.6(a) that all the color cluster have increasing color errors at around 80^{th} page. For the clusters in Figure 6.6(b), since they have only a little or none magenta in their $pCMYK$ values, their color errors remain low. When yellow fading occurred at 107^{th} page, all the color errors are raised except cluster 1 in vector region because of the little Y value in its $pCMYK$ composition. It is interesting to note that when black fading occurred at 180^{th} page, even cluster 1 and cluster 3 in raster region looks neutral to us, they do not require black toner. These black colors are reproduced by the mixture of cyan, magenta and yellow toners of the printer, which sometimes are referred to as composite black. Whereas the reproduction of

some other neutral colors, especially in the vector region, like cluster 1 in Figure 6.6(b) does require a lot of black toners. All the color errors drop back down after we replaced the corresponding cartridges. Concluding what has been said above, when a cartridge is low on toner, it raises the color errors of the color clusters which require adequate toners from that particular cartridge. To predict if the fade of a black cluster is caused by low toners in the black cartridge or not, we have to look into the $pCMYK$ compositions of this black cluster by referring to the LUT.



(a) Raster color clusters.



(b) Vector color clusters.

Fig. 6.6. Color errors of all the color clusters for raster region (a) and vector region (b) by using ΔE as the metric.

6.5 Local Analysis

The main drawback of our current algorithm is that it fails to pick up the color error when fade only occurs in some local regions. To be more specific, When a small population of pixels is faded, and the remaining pixels in the same cluster are not faded, the average of all the pixels in this cluster will tend to be more like non-faded. A such example is shown in Figure 6.7(a). In Figure 6.7(a), the faded magenta pixels are on the left bottom of the page; however, there are much more magenta pixels that are not faded. The average of these pixels is expected to have very small color difference compared to the master image; this can be hardly picked up by our algorithm.

Fading detect can be better detected if the page is partitioned into many regions, and we only look at one region of the page at a time so that those faded pixels will not be averaged out. Figure 6.7(b) shows how the raster image in Figure 6.1(a) is partitioned. The page is divided into 12×8 blocks, and we apply the same algorithm for each block. In each block, we only look at the top three-color clusters ranking according to their populations. In total, $12 \times 8 \times 3 = 288$ color clusters are examined at most, since some block may have less than three clusters. For each color cluster, we can calculate the color difference. However, what we are more interested in is how could we predict the depleted cartridge based on these color clusters.

(a) A page with local fade.

(b) Partition of raster image.

Fig. 6.7. A page with local fade (a) and partition of a raster image to blocks (b).

6.6 Color Fading Prediction

Even the color errors increase when a page is faded, it does not give us any information about which cartridge is running low on toner. Here, we propose a color prediction algorithm based on the local color clusters.

6.6.1 Algorithm Flow

The color prediction algorithm flow is illustrated in Figure 6.8. For each local cluster pair, we first calculate the color difference. If the color difference is smaller than a threshold, we do not process them. If it is higher than the threshold, it implies that this color cluster in the scanned image is faded. To find out which cartridge is depleted, we calculate the four most faded colors – C, M, Y, K – based on the color cluster of the master image; the calculations will be explained shortly. The prediction of the depleted cartridge is based on which most faded color is closet to the color cluster of the scanned image. An array of four counters are created, and every cluster pair increments only one of these four counters depending on the prediction result. This process continues until all the cluster pairs have been processed. The final output, the array of four counters, also referred to as confidence scores are the total number of predicted faded color clusters for all CMYK channels. The most probable depleted cartridge can be selected with majority rule.

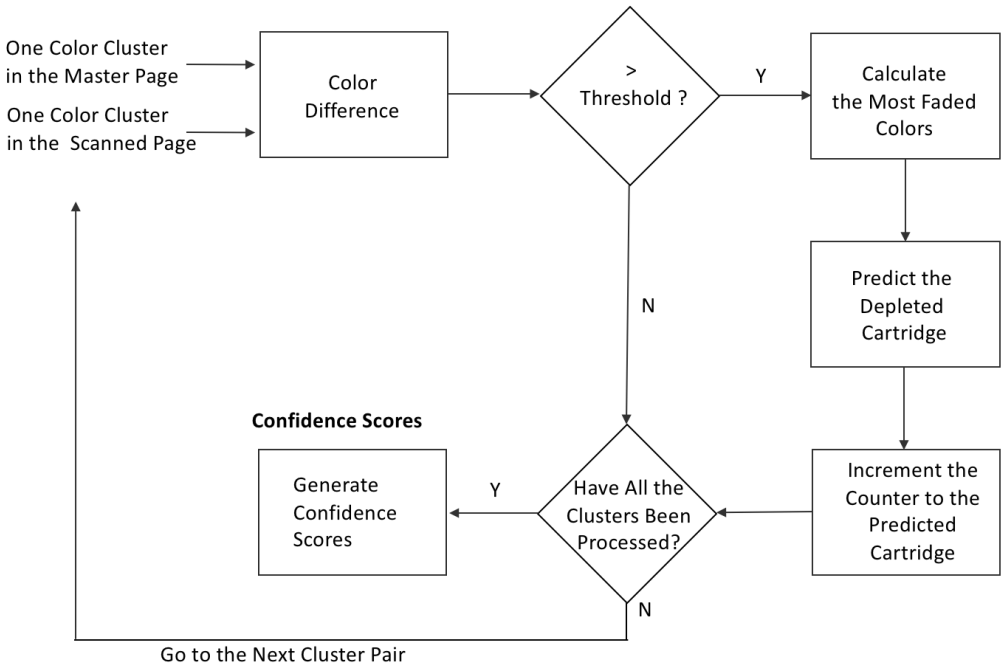


Fig. 6.8. Color fade prediction algorithm flow. The input is a set of cluster pairs, and the output is the count of faded clusters for each cartridge.

6.6.2 Calculate the Most Faded Colors

For a given color cluster from the master image, if all the cartridges are in excellent condition, we know that the printed and scanned color cluster should look the same, or at least similar if we take printer color gamut and calibration issues into account. When a scanned color cluster looks entirely different from the digital original, it must be caused by insufficient toner put on the medium – the paper. This means that a color cluster from the master image can be converted from $sRGB$ to $sCMY$, and if for example, cyan cartridge runs out of toner, only C values in the $sCMY$ should decrease. This analysis can be put into mathematics expressions shown as follows:

Convert master color cluster from sRGB to sCMY:

$$sCMY_o = [255, 255, 255] - sRGB_o$$

Calculate the most faded colors in sCMY color space:

$$sCMY_{Cfade} = [0, sM_o, sY_o]$$
$$sCMY_{Mfade} = [sC_O, 0, sY_O]$$
$$sCMY_{Yfade} = [sC_O, sM_o, 0]$$
$$sCMY_{Kfade} = sCMY_o - [min(sCMY_O), min(sCMY_O), min(sCMY_O)]$$

Convert them back to sRGB:

$$sRGB_{Cartfade} = [255, 255, 255] - sCMY_{Cartfade},$$

where $Cart \in \{C, M, Y, K\}$

The master color cluster is converted to $sCMY$ by simply subtracting its $sRGB$ value from 255; the subscript o denote master. The reason that we do not use the LUT for this conversion is that there is no way to convert back. The LUT takes a triple value and outputs a quadruple value. So, it is a map from a 3D space to a subspace of a 4D space. If we have a calculated $pCMYK$ value in the 4D space that does not fall into this subspace, we will be unable to find a corresponding mapping in the 3D $sRGB$ space. To simulate what a faded color caused by the low toner in one of the cartridges may look like, we can reduce that channel from the master $sCMY$. However, we do not know how much we should reduce – it completely depends on the condition of that particular cartridge. Here, we assume that the cartridge is empty. Then, we can just set the corresponding channel to zero. For black fading, since black can be composed of all the cyan, magenta and yellow toners together, we decrease the three channels at the same time, until one channel reaches zero. At last, we convert all these most faded colors back to $sRGB$.

An example of all these calculations is provided in Figure 6.9. In Figure 6.9, a master color cluster is given, and four most faded colors are calculated. As one cartridge has less and less toner, we expect that this master cluster gradually fades to one of these four colors. These simulated colors may be perceptually different from those actual printed colors since we did not use the LUT for color conversions. However, the eventual goal is not to calculate the color errors between them; as long as they are similar enough, we could use still them for our prediction problem, which is also a classification problem.

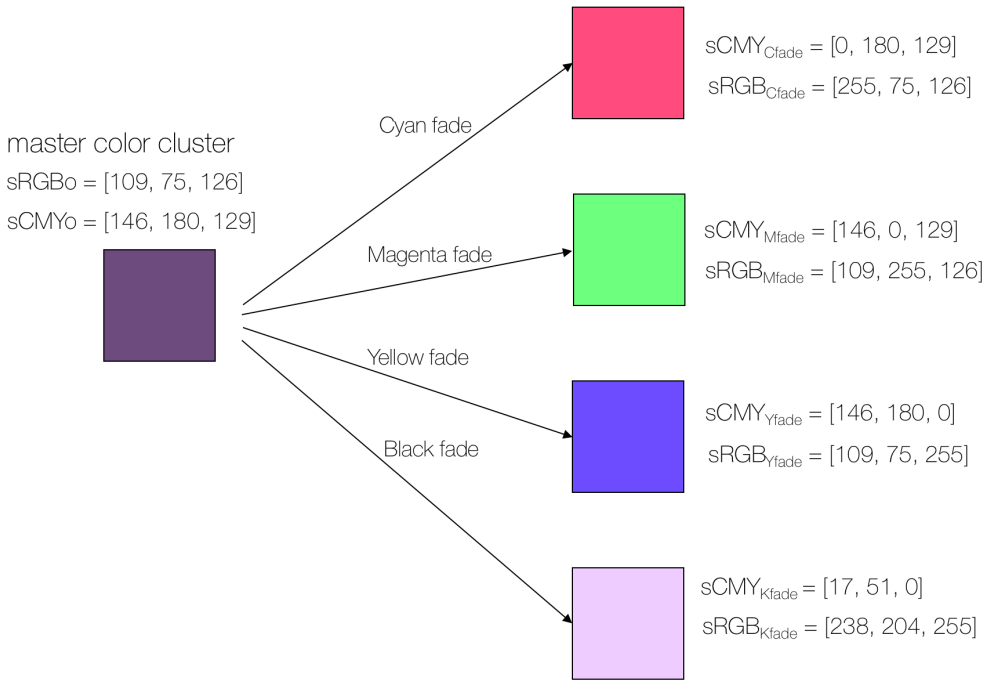


Fig. 6.9. An example of calculating the most faded color based on a master color cluster.

6.6.3 Predict the Depleted Cartridge

After we have the master color and the four most faded colors, we can further look at them in $CIEL^*a^*b^*$ color space. In Figure 6.10, the master color and four most faded colors are represented correspondingly by the color codes of their $sRGB$ values. We also use lines to connect all these four dots to the master color to represent the fade directions, which means, for example, when the magenta cartridge runs low on toner, the scanned color ideally is expected to gradually move away from the master color along the line towards the magenta most faded color. So, for a given scanned color, we can calculate the angles between the actual fade direction (the blue line) and all the other four fade directions. Whichever gives the smallest angle, we take that as our prediction result. For the example in Figure 6.10, we can intuitively see the actual scanned color is closer to the magenta fade direction, then it's reasonable to guess this cluster is magenta faded. The calculations are provided as follows:

Convert all colors to $CIEL^*a^*b^*$ space:

$$Lab_o = RGB2Lab(sRGB_o)$$
$$Lab_{scanned} = RGB2Lab(sRGB_{scanned})$$
$$Lab_{Cartfade} = RGB2Lab(sRGB_{Cartfade}),$$

Calculate the four fade directions and the actual fade direction:

$$Dir_{Cartfade} = norm(Lab_{Cartfade} - Lab_o)$$
$$Dir_{scanned} = norm(Lab_{scanned} - Lab_o)$$

Predict the depleted cartridge:

$$Cart_{predict} = \min_{Cart \in C,M,Y,K} Dir_{scanned} \cdot Dir_{Cartfade}$$

The above calculations give us the predicted cartridge which has the smallest angle with respect to the actual fade direction. To make the algorithm more robust, we have also imposed some criteria to our algorithm:

- The smallest angle needs to be less than a threshold.
- The absolute difference between the smallest angle and the second smallest angle needs to be greater than a threshold.
- The predicted faded channel in the pCMYK needs to have a value greater than a threshold; the pCMYK is generated by the LUT based on the master color cluster.

The first two criteria are quite straightforward. The third criterion can help us to validate our prediction result. For example, black fade is predicted by our algorithm. However, when we look at the $pCMYK$ value generated by the LUT and K value is zero, it means the reproduction of this color by the printer does not require black toner at all; our prediction of black fade must be wrong. It is possible that this black color is a composite black, and all the CMY cartridges are running low on toner. The previous calculations are carried out to all local color clusters, and we only count the ones who satisfy all these criteria.

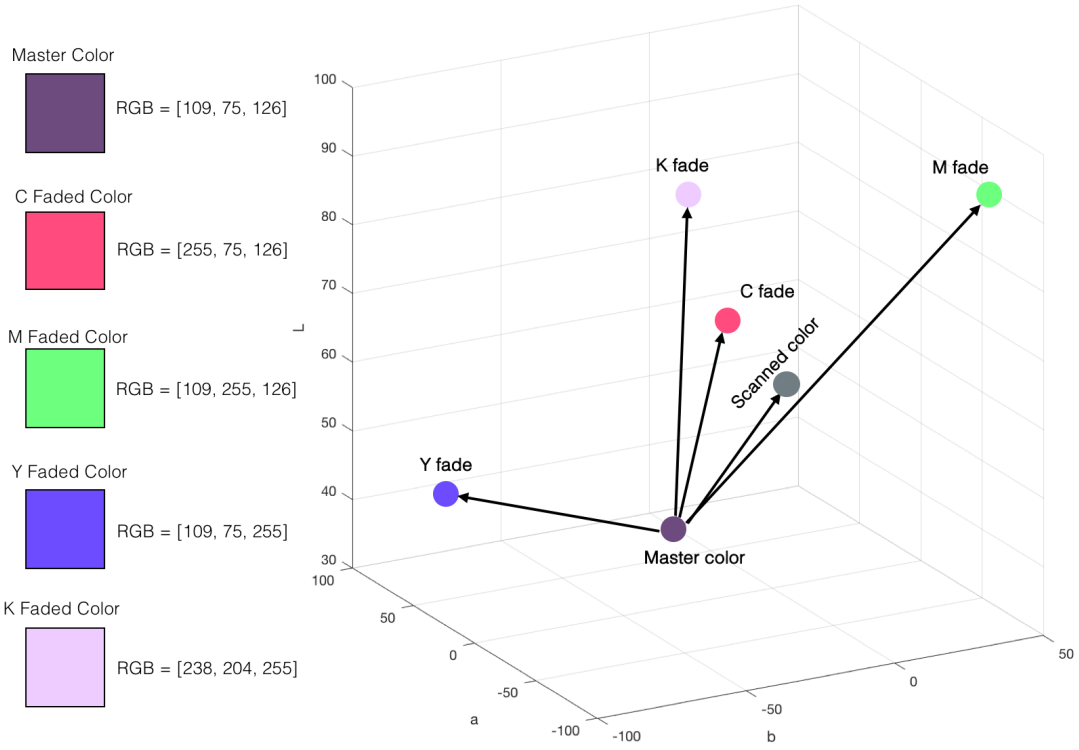


Fig. 6.10. The master color and the calculated most faded colors in $CIEL^*a^*b^*$ color space. The blue is the actual fade direction, and the other three lines are the fade directions of the most faded colors.

6.7 Experiment Results

Previously, we have shown the color difference result in Figure 6.6. Now, we apply our local analysis and color prediction algorithm to the same dataset; the result is shown in Figure 6.11. The x-axis is the same as before, which is the page number. The y-axis is the number of faded color clusters for each channel based on our prediction algorithm, represented by different colors. Cyan fade occurs one sampled page earlier than the magenta fade according to the ISO, which is around 79^{th} page. Our detection algorithm missed this page because only a small area of the page is faded. This can be seen from Figure 6.12(b) by comparing to the non-faded page in Figure 6.12(a) – a narrow band on the top left of Figure 6.12(b) appears reddish. At 80^{th} page shown in Figure 6.12(c), as the faded area increase, several color clusters are correctly predicted as cyan fade. Starting from 81^{th} page, not only we have cyan fade, but also magenta fade, which is shown in Figure 6.12(d). The fading become most severe at 87^{th} page, shown in Figure 6.12(e). The reason that all the color clusters are predicted as magenta is that magenta fade is more dominant. The most server case for yellow fade and black fade is shown separately in Figure 6.12(f) and Figure 6.12(g). From the plot in Figure 6.11 we can see that the prediction results align with the ISO fade very well. Figure 6.13 shows the result of another customer content page in the suite. The dataset contains the same number of pages, and the fading events occurred at the same pages. The master image of such dataset is in Figure 6.13(b). In Figure 6.13(a), the predictions again coincide with the ground truth in general. Although some color clusters are mispredicted, we can use the majority rule to select the most probable faded cartridge.

In the plot of Figure 6.11 and Figure 6.13(a), we have also shown the fading predicted results by a pixel counting algorithm on the top of each plot. This algorithm is currently used by many HP printer models in the market, and here we are unable to disclose any detail of the algorithm itself. As we can see, the pixel counting algorithm predicts fading thousand of pages earlier than the ground truth for all the cartridges.

The page difference for each cartridge is listed in Table 6.1. The table indicates that if we replace cartridges based on the pixel counting estimation of fade, the amount of toner that is wasted can be used for printing over two thousand more pages on average. Whereas for our algorithm, since it almost concurs with the ground truth, much less resource will be wasted.

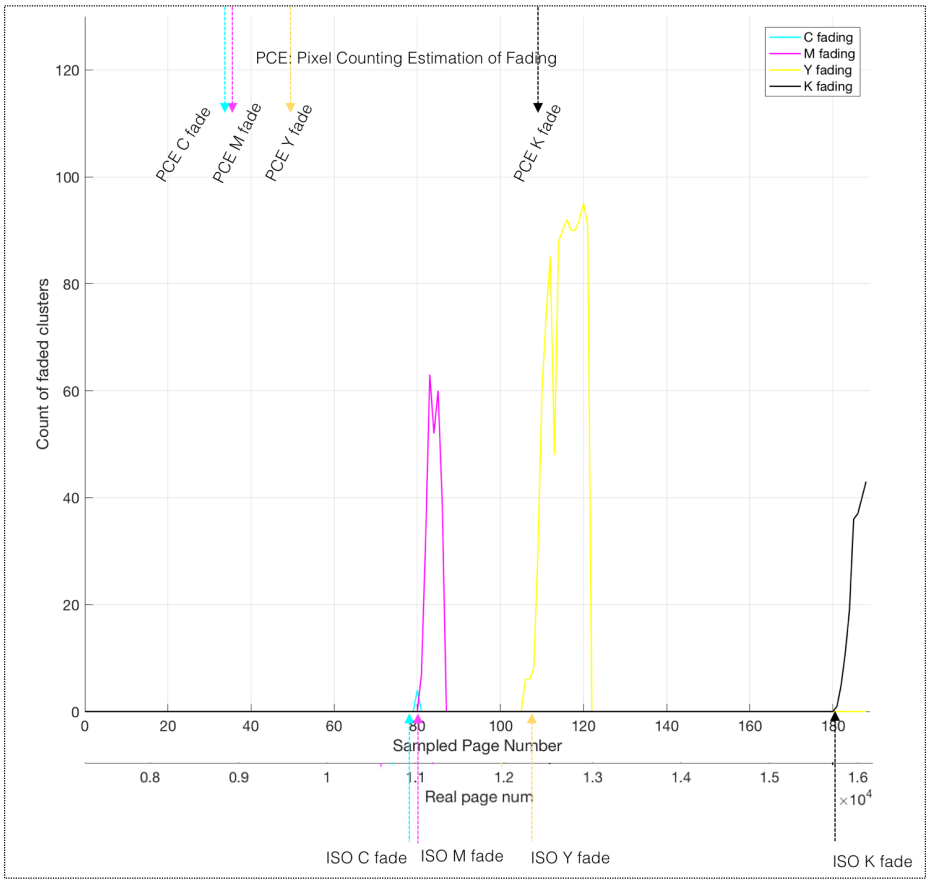


Fig. 6.11. Color fade prediction result of the dataset whose master image is in Figure 6.1(a). The y-axis is the number of faded color clusters for each cartridge; the x-axis contains the sampled page number on the top and the actual page number on the bottom. The arrows are the ground truth of the start of the faded events according to the ISO standard.

(a) Page 1, non faded.

(b) Page 79, cyan faded.

(c) Page 80, magenta and cyan faded.

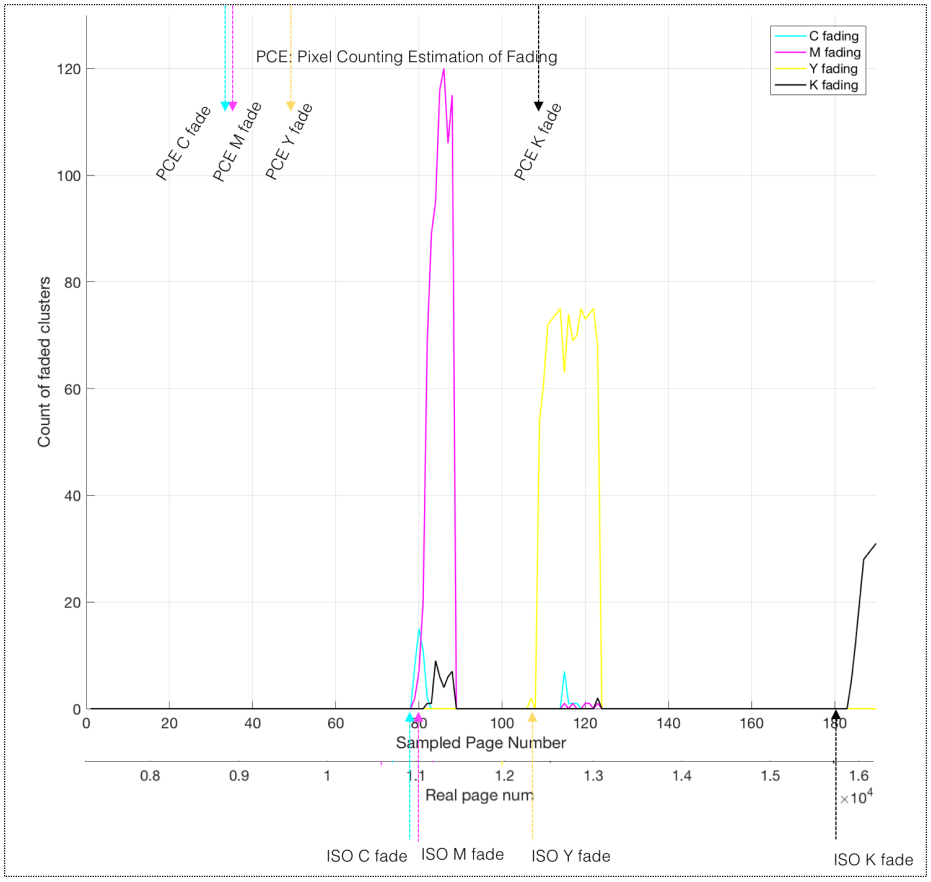
(d) Page 81, magenta and cyan faded.

(e) Page 87, magenta and cyan severely faded.

(f) Page 122, yellow severely faded.

(g) Page 189, black severely faded.

Fig. 6.12. Example of faded pages from a dataset caused by low toners in different cartridges. (a) is a non-faded page; (b) is a cyan faded page; (c)-(e) are magenta and cyan faded pages; (f) is a yellow faded page; (g) is a black faded page.



(a) Color fade prediction result.

(b) Master image.

Fig. 6.13. (a) Color fade prediction result of new customer content page; (b) the master image of the new customer content page.

Table 6.1.
The page difference between pixel counting estimation of fading (PCE) and the ground truth according to the ISO standard.

Cartridge	Page difference between PCE and ISO
Black	3156
Cyan	1880
Magenta	1867
Yellow	2519
Average	2356

6.8 Conclusion

To conclude this chapter, we proposed an algorithm to detect color fading based on custom content pages instead of diagnostics pages. The algorithm only examines the dominant color clusters over the page, which is computationally efficient. We have also provided a novel approach to predict the depleted cartridge based on those color clusters. The results have shown that, compared to the pixel counting estimation of fade that is currently widely used in many HP printers, our proposed algorithm is better aligned with the ISO standard, which results in less resource being wasted.

7. SUMMARY

In this thesis, we focused on dealing with print quality problems from two aspects. The first one tries to optimize the rendering process to avoid any potential print artifact. The second one aims to diagnose these print issues when the print defects are already on customer pages.

In the first chapter, We started by describing the printer mechanism and the major components that are involved in the EP process, and provided examples of how a particular print defect is associated with one of these components. We introduced some existing PQ diagnostic tools and methods. We highlighted the main drawback of these diagnostic approaches – the high cost of labors and time – and proposed the need for a more efficient methodology. Among so many print defects, we picked the fading defect for elaboration since it can degrade the PQ severely. We defined fading and introduced the ISO standard, which is later what our ground truth is built on. We then presented the general image processing pipeline of a printer and the notion of object-oriented halftoning. We pointed that the technical challenge of object-oriented halftoning is the need of correct object maps, and the current algorithm to generate object maps is too complicated for hardware implementation. With all these PQ problems brought up, the remainder of this thesis attempted to tackle each problem individually by proposing new image processing algorithms.

An object map generating algorithm was described in the second chapter. The algorithm was developed to generate a correct object map based on a raster image, instead of a PDL file. The reason is that the object maps extracted from PDL files most of the time are incorrect, and some print jobs do not even support PDL files. One most important part of the algorithm is the CCL process. We first offered the Classic CCL algorithm, and clearly showed how memory-hungry this algorithm is. Next, we attempted to adapt the algorithm to a strip-based process to achieve mem-

ory efficiency and put effort into dealing with the discontinuities between adjacent strips. In terms of implementation, we tailored a hardware-friendly union-find array data structure for this strip-based algorithm and added label recycling and path compression features to further push down the memory consumption. Our contributions to this part of work include:

- We developed an object map generating algorithm that only requires buffering a strip of an image at a time rather than the whole page.
- We proposed a novel union-find array data structure for the CCL process, along with memory-recycling and path compression features.

We also want to comment that our proposed strip-based CCL algorithm and its data structure can be used for many other applications as well. For example, for pedestrian detection in a traffic surveillance video, we can set the strip height a little larger than the height of a regular person in the video. This allows us to buffer only a tiny amount of memory for the CCL process, which is very beneficial for embedded applications.

In chapter three, we are mainly concerned with PQ troubleshooting methodologies. The traditional PQ diagnostic method requires experts examining the page to identify the PQ issues. Later on, more and more automatic print defect detection algorithms were developed and proposed, which freed those experts. However, almost all the existing algorithms rely on diagnostics pages to detect print defects. The diagnostics pages are stored in the printer memory when they were manufactured, and depending on the who the manufacturer is, the diagnostic page could be different. This posed a critical issue to all the existing algorithms, i.e., an algorithm developed for one kind of diagnostic page may not work for another. Another issue is that users still have to get involved. After they printed their documents, they first need to make a judgment if the PQ is acceptable to them or not. If not, then they will have to print the diagnostic page, scan them, and finally run the diagnostic program. For these considerations, we have made contributions of:

- Presented an on-line PQ diagnostic system to detect print defects on a scanned image by comparing to the raster image. The scanned image is obtained from an in-line built-in scanner on a customer's printer.

- Proposed a signal control flow including user, printer, and scanner to monitor the PQ in the background without any human interference.

With the proposed the on-line diagnostic system in hand, in chapter four, we focused on how could we spatially align the scanned image with the raster image, which is necessary before we are able to make any comparison between them. We came up with a feature based image registration pipeline. We have also done several novel optimizations to make the algorithm more robust and efficient, including:

- We downsampled both images for faster computation, but are able to return the aligned image of full resolution.

- We proposed a local feature matching approach to make the matching results more accurate.

- Used histogram matching to color balance the scanned image with the raster image to improve the feature matching. This is to account for the color discrepancy between them due to the print defect such as fading.

In chapter five, we targeted on text fading detection. We first locally aligned each text character between the scanned image and the raster image. Then we calculated the color difference between each text character pair. We analyzed the histogram of these color errors for those faded pages and non-faded pages. We correlated the mean of the color errors to the severity level of fading and set a threshold through a psychophysical experiment. To better capture local fade, we divided our pages into many strips and carried out our analysis to each strip. For this part of work, we contributed to the following:

- Developed a text fading detection algorithm by comparing each text character on the scanned image to the raster image.

- Implemented the text fading detection algorithm on a Linux station and created a GUI.

Finally, in chapter six, we extended our work to detect fading on non-text regions. To solve the misregistration problem, we calculated the superpixels and extracted the color clusters from the pages. This can also speed up the computations since we do not have to examine each pixel anymore. We have shown that the color error between a pair of the color cluster increases with the increasing level of fading. The difficulty is when a page is faded, how do we know which of the $CMYK$ cartridges is low on toner based on the RGB values without using an optical sensor to measure them directly. This problem was addressed in detail in the chapter with a cartridge depletion prediction algorithm proposed and described. Our proposed approach checks the color composition of each color cluster on the master image and simulates what a faded color may look like when the toner is low on each cartridge, then take the cartridge whose simulated color is closet to the scanned color cluster as the prediction result. Our results indicate that the proposed algorithm is better aligned with the ISO ground truth compared to the pixel counting estimation of fading algorithm that is used in many HP printer model in the current market. Hence, the cost can be greatly reduced as a result of less toner being wasted. Our contributions are:

- Developed an algorithm to determine if a page is faded or not based on the non-text regions of a scanned image and its raster image.

- For a faded scanned image, we proposed an algorithm to predict which cartridge is low on toner.

REFERENCES

REFERENCES

[1] L. B. Schein, in *Electrophotography and Development Physics*, 2nd ed. Morgan Hill, CA: Laplacian Press, 1996.

[2] A. K. Mikkilineni, N. Khanna, and E. J. Delp, "Forensic printer detection using intrinsic signatures," in *Proc. SPIE, Media Watermarking, Security, and Forensics*, vol. 7880, San Fransisco, California, 2011.

[3] W. Jang and J. P. Allebach, "Simulation of print quality defects," *Journal of Imaging Science and Technology*, vol. 49, no. 1, pp. 1–18, January 2005.

[4] R. P. Loce, W. L. Lama, and M. S. Maltz, "Modeling vibration-induced halftone banding in a xerographic laser printer," *Journal of Electronic Imaging*, vol. 4, no. 1, pp. 48–61, January 1995.

[5] G.-Y. Lin, J. M. Grice, J. P. Allebach, G. T.-C. Chiu, W. Bradburn, and J. Weaver, "Banding artifact reduction in electrophotographic printers by using pulse width modulation," *Journal of Imaging Science and Technology*, vol. 46, no. 4, pp. 326–337, July 2002.

[6] W. Jang, M.-C. Chen, J. P. Allebach, and G. T.-C. Chiu, "Print quality test page," *Journal of Imaging Science and Technology*, vol. 48, no. 5, pp. 432–446, 2004.

[7] J. C. Briggs, E. Hong, and D. Forrest, "Analysis of ghosting in electrophotography," in *IS&Ts NIP16 International Conference on Digital Printing Technologies*, Vancouver, British Columbia, Canada, October 2000.

[8] A. H. Eid, B. E. Cooper, and M. N. Ahmed, "Characterization of ghosting defects in electrophotographic printers," in *2007 IEEE International Conference on Image Processing*, San Antonio, Texas, October 2007.

[9] S. Leman and M. R. Lehto, "Interactive decision support system to predict print quality," *Ergonomics*, vol. 46, no. 1–3, pp. 52–67, January 2003.

[10] H. Santos-Villalobos, H. J. Park, R. Kumontoy, K. Low, M. Ortiz, J. P. Allebach, C. Kim, P. Choe, S. Leman, K. Oldenburger, M. R. Lehto, and X. Y. Lehto, "Web-based diagnosis tool for customers to self-solve print quality issues," *Journal of Imaging Science and Technology*, vol. 54, no. 4, pp. 1–13, July 2010.

[11] H. Santos-Villalobos, V. Loewen, M. Lehto, and J. P. Allebach, "A web-based troubleshooting tool to help customers self-solve color issues with a digital printing workflow," in *Proc. SPIE - The International Society for Optical Engineering*, vol. 7879, San Francisco Airport, CA, 2011, p. 787906.

[12] C. Kim, P. Choe, M. Letho, and J. P. Allebach, "Development of a web-based interactive self-help troubleshooting tool for print quality problems," in *International Conference on Human-Computer Interaction*, Las Vegas, NA, July 2005, pp. 22–27.

[13] P. Choe, C. Kim, M. R. Lehto, X. Lehto, and J. P. Allebach, "Evaluating and improving a self-help technical support web site: Use of focus group interviews," *International Journal of Human-Computer Interaction*, vol. 21, no. 3, pp. 333–354, 2006.

[14] J. Zhang, S. Astling, R. Jessome, E. Maggard, T. Nelson, M. Shaw, and J. P. Allebach, "Assessment of presence of isolated periodic and aperiodic bands in laser electrophotographic printer output," in *Proc. SPIE, Image Quality and System Performance X*, vol. 8653, Burlingame, CA, February 2013.

[15] A. Venosa, D. Burge, and D. Nishimura, "Effect of light on modern digital prints - photographs and documents," *Studies in Conservation*, vol. 56, no. 4, pp. 267–280, 2011.

[16] N. Yan, E. Maggard, R. Fothergill, R. Jessome, and J. P. Allebach, "Autonomous detection of iso fade point with color laser printers," in *Proc. SPIE, Image Quality and System Performance XII*, vol. 9396, San Francisco, CA, February 2015.

[17] Y. Ju, E. Maggard, R. Jessome, and J. P. Allebach, "Autonomous detection of text fade point with color laser printers," in *Proc. SPIE, Image Quality and System Performance XII*, vol. 9396, San Francisco, CA, February 2015.

[18] D. Boockholdt and H. G. Hooper, "Accurate toner level feedback via active artificial intelligence," US Patent 5 794 094A, August, 1998.

[19] T. A. Fischer, "Methods and apparatus for detecting toner fade in an imaging device," US Patent 6 636 705B2, October, 2003.

[20] M. Gao, Y. Ju, T. Nelson, T. Prenn, and J. P. Allebach, "Toner usage prediction for laser electrophotographic printers," in *Proc. IS&T, Color Imaging XXI: Displaying, Processing, Hardcopy, and Applications*, San Francisco, CA, February 2016.

[21] L. Wang, D. Abramsohn, T. Ives, M. Shaw, and J. P. Allebach, "Estimating toner usage with laser electrophotographic printers," in *Proc. SPIE, Color Imaging XVIII: Displaying, Processing, Hardcopy, and Applications*, vol. 8652, Burlingame, CA, February 2013.

[22] P. H. Notredame, E. H. Debaere, L. W. Depuydt, and J. J. Vlietinck, "Compressed merging of raster images for high speed digital printing," US Patent 6 049 390A, April, 2000.

[23] R. D. Christiansen, "Determining raster image processor cycle count to fully utilize a printer," US Patent 7 202 964B2, April, 2007.

[24] N. Marovac, "Page description language interpress in electronic publishing environment," *Computers & Graphics*, vol. 15, no. 3, pp. 423–434, 1991.

[25] J. A. Hewitt, "Load balancing for raster image processing across a printing system," US Patent 7 016 061B1, March, 2006.

[26] J. R. Sullivan, L. A. Ray, and R. Miller, "Design of minimum visual modulation halftone patterns," *IEEE Transactions on Systems, Man, & Cybernetics*, vol. 21, no. 1, pp. 33–38, 1991.

[27] S. J. Park, M. Q. Shaw, G. Kerby, T. Nelson, D.-Y. Tzeng, K. R. Bengtson, and J. P. Allebach, "Halftone blending between smooth and detail screens to improve print quality with electrophotographic printers," in *Proc. SPIE, Color Imaging XVIII: Displaying, Processing, Hardcopy, and Applications*, vol. 8292, Burlingame, California, 2012, pp. 829 210–1–12.

[28] Y.-T. Chen, D.-Y. Tzeng, T. Nelson, M. Q. Shaw, and J. P. Allebach, "Segmentation for better rendering of mixed-content pages," in *Proc. SPIE, Color Imaging XVIII: Displaying, Processing, Hardcopy, and Applications*, vol. 8652, Burlingame, California, 2013, pp. 865 209–1–15.

[29] L. Wang, "Estimating toner usage with laser electrophotographic printers, and object map generation from raster input," Ph.D. dissertation, Purdue University, December 2014.

[30] X. Jing, S. Astling, R. Jessome, E. Maggard, T. Nelson, M. Shaw, and J. P. Allebach, "A general approach for assessment of print quality," in *Proc. SPIE, Image Quality and System Performance X*, vol. 8653, Burlingame, CA, February 2013.

[31] X. Liu, G. Overall, T. Riggs, R. Silveston-Keith, J. Whitney, G. Chiu, and J. P. Chiu, "Wavelet-based figure of merit for macrouniformity," in *Proc. SPIE, Image Quality and System Performance X*, vol. 8653, Burlingame, CA, February 2013.

[32] W. Wang, G. Overall, T. Riggs, R. Silveston-Keith, J. Whitney, G. Chiu, and J. P. Allebach, "Figure of merit for macrouniformity based on image quality ruler evaluation and machine learning framework," in *Proc. SPIE, Image Quality and System Performance X*, vol. 8653, Burlingame, CA, February 2013.

[33] M. Q. Nguyen, R. Jessome, S. Astling, E. Maggard, T. Nelson, M. Shaw, and J. P. Allebach, "Perceptual metrics and visualization tools for evaluation of page uniformity," in *Proc. SPIE, Image Quality and System Performance X*, vol. 9016, San Fransisco, CA, February 2014.

[34] M. Q. Nguyen and J. P. Allebach, "Controlling misses and false alarms in a machine learning framework for predicting uniformity of printed pages," in *Proc. SPIE, Image Quality and System Performance XII*, vol. 9396, San Fransisco, CA, February 2015.

[35] W. Wang, Y. Guo, and J. P. Allebach, "Image quality evaluation using image quality ruler and graphical model," in *Proc. of ICIP-2015 IEEE International Conference on Image Processing*, Quebec City, QC, Canada, December 2015.

[36] S. Hu, H. Nachlieli, D. Shaked, S. Shiffman, and J. P. Allebach, "Color-dependent banding characterization and simulation on natural document images," in *Proc. SPIE, Color Imaging XVII: Displaying, Processing, Hardcopy, and Applications*, R. Eschbach, G. G. Marcu, and A. Rizzi, Eds., vol. 8292, Burlingame, CA, January 2012.

[37] X. Jing, H. Nachlieli, D. Shaked, S. Shiffman, and J. P. Allebach, "Masking mediated print defect visibility predictor," in *Proc. SPIE, Image Quality and System Performance IX*, vol. 8293, Burlingame, CA, January 2012.

[38] J. Zhang, H. Nachlieli, D. Shaked, S. Shiffman, and J. P. Allebach, "Psychophysical evaluation of banding visibility in the presence of print content," in *Proc. SPIE, Image Quality and System Performance IX*, vol. 8293, Burlingame, CA, January 2012.

[39] J. Zhang and J. P. Allebach, "Estimation of repetitive interval of periodic bands in laser electrophotographic printer output," in *Proc. SPIE, Image Quality and System Performance XII*, vol. 9396, San Francisco, CA, February 2015.

[40] X. Jing, S. Astling, R. Jessome, E. Maggard, T. Nelson, M. Shaw, and J. P. Allebach, "Electrophotographic ghosting detection and evaluation," in *Proc. IS&T, NIP & Digital Fabrication Conference*, no. 4, Portland, OR, January 2015, pp. 169–172.

[41] J. Wang, T. Nelson, R. Jessome, S. Astling, E. Maggard, M. Shaw, and J. P. Allebach, "Local defect detection and print quality assessment," in *International Congress of Imaging Science (ICIS)*, Tel Aviv, Israel, May 2014.

[42] ——, "Local defect detection and print quality assessment," in *Image Quality and System Performance XIII (Part of IS&T Electronic Imaging 2016)*, San Francisco, CA, February 2016.

[43] W. Wang, P. Bauer, J. Wagner, and J. P. Allebach, "MFP scanner diagnostics using a self-printed target to measure the modulation transfer function," in *Proc. SPIE, Image Quality and System Performance XI*, vol. 9016, San Francisco, CA, February 2014.

[44] M. Kim, P. Bauer, J. K. Wagner, and J. P. Allebach, "MFP scanner motion characterization using self-printed target," in *Proc. SPIE, Image Quality and System Performance XII*, vol. 9396, San Francisco, CA, February 2015.

[45] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM*, vol. 13, no. 4, pp. 471–494, October 1966.

[46] W. Kong, P. Chang, and Z. Bi, "Real-time sobel edge detector," in *Proc. 6th PSU-UNS International Conference on Engineering and Technology*, Novi Sad, Serbia, 2013.

[47] H. Samet and M. Tamminen, "Efficient component labeling of images of arbitrary dimension represented by linear bintrees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 579–586, July 1988.

[48] M. B. Dillencourt, H. Samet, and M. Tamminen, "A general approach to connected-component labeling for arbitrary image representations," *Journal of the ACM*, vol. 39, no. 2, pp. 253–280, April 1992.

[49] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641 – 647, June 1994.

[50] D. G. Bailey and C. T. Johnston, "Single pass connected components analysis," in *Proc. Image and Vision Computing and Vision Computing*, Hamilton, New Zealand, 2007, pp. 282–287.

[51] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM*, vol. 13, no. 4, pp. 471–494, October 1966.

[52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Chapter 21: Data structures for disjoint sets," in *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2015, pp. 562–585.

[53] R. Sedgewick and K. Wayne. (2015, September) Case study: Union-find. Princenton University. Accessed: 2018-02-02. [Online]. Available: https://algs4.cs.princeton.edu/15uf/

[54] Z. Galil and G. F. Italiano, "Data structures and algorithms for disjoint set union problems," *ACM Computing Surveys*, vol. 23, no. 3, pp. 319–344, September 1991.

[55] R. Walczyk, A. Armitage, and D. Binnie, "Comparative study on connected component labeling algorithms for embedded video processing systems," in *International Conference on Image Processing, Computer Vision, and Pattern Recognition*, Las Vegas, NV, 2010, pp. 853–859.

[56] B. S. Reddy and B. N. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE Transactions on Image Processing*, vol. 5, no. 8, pp. 1266 – 1271, August 1996.

[57] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of the Fourth Alvey Vision Conference*, University of Manchester, Manchester, August 1988, pp. 23.1–23.6.

[58] Y. Lv, Q. Feng, L. Qi, and Q. Chen, "Sub-pixel surface fitting algorithm in digital speckle correlation method," in *Proc. IEEE, 9th International Conference on Electronic Measurement & Instruments*, Beijing, China, August 2009.

[59] J. Li and N. M. Allinson, "A comprehensive review of current local features for computer vision," *Journal of Neurocomputing*, vol. 71, no. 10-12, pp. 1771–1787, June 2008.

[60] R. Hartley and A. Zisserman, "Chapter 4. estimation 2d projective transformations," in *Multiple view geometry in computer vision*, 2nd ed. Cambridge University Press, 2004, pp. 87–131.

[61] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[62] P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, April 2000.

[63] E. Bernal, J. Trask, and J. Allebach, "Model-based memory-efficient algorithm for compensation of toner overdevelopment in electrophotographic printers," *Journal of Imaging Science and Technology*, vol. 52, no. 6, pp. 60 504–1–60 504–15, 2008.

[64] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790 – 799, August 1995.

VITA

VITA

Zuguang Xiao received his Bachelor Science degree in Electrical Engineering from Purdue University in 2008. He then stayed at Purdue University for direct Ph.D. program in Electrical Engineering. In 2014, he joined Professor Jan P. Allebach's team and started to work on projects in the area of image processing and image quality. During his Ph.D., he had an eight-month internship in Apple to develop various image processing algorithms to improve the front-of-screen er experience for Apple's products, and a three-month internship in HP Inc. to develop and implement PQ diagnostics algorithm for HP laser printers. Besides image processing and image quality, his research interests also include computer vision and machine learning.