4-2016

Content-based image analysis with applications to the multifunction printer imaging pipeline and image databases

Cheng Lu
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Cheng Lu

Entitled: Content-Based Image Analysis with Applications to the Multifunction Printer Imaging Pipeline and Image Databases

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

JAN P. ALLEBACH

EDWARD J. DELP

MARY L. COMER

YUNG-HSIANG LU

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

JAN P. ALLEBACH

Approved by Major Professor(s): _____

_____

Approved by: V. Balakrishnan 04/27/2016

Head of the Department Graduate Program Date

CONTENT-BASED IMAGE ANALYSIS WITH APPLICATIONS TO THE MULTIFUNCTION PRINTER IMAGING PIPELINE AND IMAGE DATABASES

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Cheng Lu

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2016

Purdue University

West Lafayette, Indiana

To my parents.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest appreciation to my advisor, Prof. Allebach for his great support on my research and project. His enthusiasm about research and supportive attitude to my project has been a great inspiration to me. Particularly, he helped me realize the importance of communication, presentation and gave me a lot of guidance. He always supported my internship and conference trip which all turned out to be great experience for me. With this experience, I got the chance to build my skills and know peers in the area.

I would also thank to Hewlett-Packard Company for their continuous financial support during my entire Ph.D. period and three summer internship in 2013, 2014 and 2015. I would especially thank to Jerry Wagner, Mark Shaw, Brandi Pitta, David Lawson, Collin Day, Randy Guay, Lisa Li, Mike Shelton, Gorge Kerby and Peter Bauer at HP Boise for their great mentorship. Also, I'd like to say thanks to Jian Fan, Yang Lei, Jerry Liu and Steve Simske at HP 3D&printing lab for their support during my internship.

I deeply appreciate my family for their great support and shape who I am. My parents, grandpa, uncle and aunts, plus my sister and brother. They are always my unbreakable inspiration.

Thanks to anyone significant in my life.

Thanks to my home city.

Thanks to all my friends.

They are all my reasons.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Lu, Cheng PhD, Purdue University, May 2016. Content-Based Image Analysis with Applications to the Multifunction Printer Imaging Pipeline and Image Databases . Major Professor: Jan P. Allebach.

Image understanding is one of the most important topics for various applications. Most of image understanding studies focus on content-based approach while some others also rely on meta data of images. Image understanding includes several sub-topics such as classification, segmentation, retrieval and automatic annotation etc., which are heavily studied recently. This thesis proposes several new methods and algorithms for image classification, retrieval and automatic tag generation. The proposed algorithms have been tested and verified in multiple platforms. For image classification, our proposed method can complete classification in real-time under hardware constraints of all-in-one printer and adaptively improve itself by online learning. Another image understanding engine includes both classification and image quality analysis is designed to solve the optimal compression problem of printing system. Our proposed image retrieval algorithm can be applied to either PC or mobile device to improve the hybrid learning experience. We also develop a new matrix factorization algorithm to better recover the image meta data (tag). The proposed algorithm outperforms other existing matrix factorization methods.

1. INTRODUCTION

Content-based image analysis [1,2] has been intensively studied recently. Many applications built in different platforms like printer, PC or mobile devices, require content-based image analysis. These applications typically rely on techniques including image classification [3], image retrieval [4], image segmentation [5], object recognition [6] etc. Some content-based applications also utilize image meta-data [7] like tags or labels as supporting information to improve system performance. All these applications heavily rely image database to train and verify the algorithms. In this paper, we discuss several topics regarding the content-based image analysis and its applications in image databases with emphasis on multifunction printer.

In Chap. 2, Chap. 3, we introduce an image classification system which analyses the input images scanned by multifunction printer. The multifunction printer should choose the optimal processing pipeline for input image based on the classification result. The image content analysis in multifunction printer could significantly improve the copy image quality by applying certain enhancement algorithms. Reversely, incorrect analysis (misclassification) could cause severe image quality degradation. For example, if a text is scanned by picture pipeline, we will find blur edges of text strokes which are undesirable. In these two chapters, we also introduce new funcionalities on top of transitional image classification to speedup the analysis process and allow adaptive online learning. To verify our proposed algorithm, we build an image database which includes representative images of each type.

Different from Chap. 2, Chap. 3 where we target the entire raster image, we analyze each object from a vector image file in Chap. 4. The goal of this chapter is to optimally compress each object in a vector image file by applying the best compression algorithm. To achieve this goal, we need to analyse content of each object and process it properly. The processing includes 1) decide if the input image should be compressed

losslessly or lossily; 2) find the optimal Q factor if it is compressed by lossy algorithm; 3) decide which lossless algorithm is better for the given object if it is compressed by lossless algorithm.

Chap. 5 discusses a content-based analysis method for image retrieval. Similar to Chap. 2 and Chap. 3, it again takes raster image as input. However, the input image is captured by PC top-view camera or mobile devices. It compares the visual similarity between the query image and the images in the database, then finds the best match. The analysis includes two stages: 1)Bag-of-Word initial retrieval; 2) Hierarchial Weighted Spatial Ranking. The image database for this application is carefully designed for robustness. It includes challenging cases that are very visually similar.

Chap. 6 introduces a novel method for matrix factorization which can be used to improve the meta-data of image database. It is capable of correcting potential tagging errors in an image database. It serves as a supplementary technique to improve the content-based image analysis. That is because lots of content-based methods take tag information as ground truth for training and testing. With more correct tags as ground truth, we can develop reliable content-based analysis algorithms. Experimentally, we show that our proposed method outperforms other existing methods.

2. ONLINE LEARNING IMAGE CLASSIFICATION UNDER MONOTONIC DECISION BOUNDARY CONSTRAINT

2.1 Problem statement

All-in-one (AIO) devices are now widely used for various purposes. They typically provide scanning, copying and printing services. One major issue for an AIO device is its copy quality. In order to optimally process different types of input images, multiple processing pipelines [8] are included in an AIO device. Each of these processing pipelines is purposefully designed for one type of image. In our application, the AIO device includes three processing pipelines which target three types of images, respectively. Theses three types are: pure text, picture, and mix as is illustrated by the three examples in Fig. 2.1. An incorrect choice of processing pipeline (mode) when copying an image will lead to significantly worse output quality. As shown in Fig. 2.2, if a pure text image is chosen to be copied under the picture mode, the characters on the output image will have poor edge sharpness and low visual contrast both of which harm the reading experience [9]. Different types of misclassifications do not cause equally severe image quality degradations. This suggests that we need to apply a discriminative training strategy, which will be discussed later. Given this problem, an embeded-firmware-friendly automatic classification algorithm is required before sending the input image to its corresponding processing pipeline.

This classification algorithm is implemented in the firmware of low-end AIO device, so computational load is also one of our primary concern. We need to do the classification in real-time so that the customer will not experience noticeable wait after pushing the copy button.

(a) Mix (b) Picture (c) Text

Fig. 2.1.: Examples of three types of image

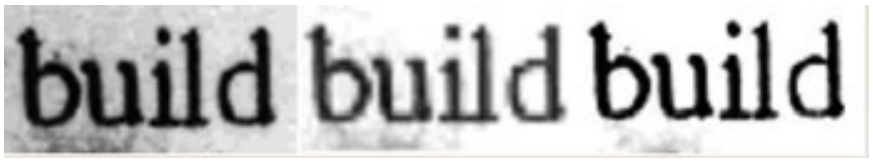


Fig. 2.2.: Image quality of original/ picture mode/ text mode

2.2 Related work

There has been a significant amount of research on the image classification topic and some methods have been proposed specifically to address this issue. A very important fact is that any algorithm applicable to our low-end copy pipeline must process image data one strip at a time and never revisit previously processed strips. This makes it impossible to apply some existing approaches [10] [11] [12] [13] [14]. Dong et al. [9] introduced several features which are statistically significant to classify different types of images. He used a simple threshold method to the make final classification decision. However, this method relies on having a 300 dpi high resolution input image, which is computationally expensive. This approach also lacked a

basis in a statistical machine learning method. Lu et al. [15] proposed a classification algorithm for digital copier based on Support Vector Machine [16]. This method allows limited quick decision capability, which may speed up classification for mix images. However, it applies only offline training. Therefore it excludes AIO users from training the AIO device in the future to expand the training set and thus customize the product's behavior. One of major challenges in this project is text recognition. There have been many approaches proposed to address this issue. Typically, these approaches can be summarized into three categories. The first category is based on the assumption that text characters have constant stroke width. Srivastav et al. [17] proposed to divide text detection into five stages: (1) preprocessing, (2) adaptive edge detection, (3) basic filtering, (4) classification based on nearest-neighbor constraints, and (5) classification based on stroke width and foreground color constraints. The second category assumes that the text string contains uniform color. Jain et al. [18] first does foreground extraction, and then applies connected components analysis to locate the text region. The third category [19] assumes that the text strokes have high contrast compared to the background. Since typical algorithms based on stroke-width constancy require horizontal or vertical alignment of the text strings, in our application we use both Categories 2 and 3 to guarantee accuracy and speed. Foreground and background extraction [20] [21] can also used to address this problem.Overall, image classification is extensively studied, and many publications discuss this issue from different perspectives [22] [23] [24] [25] [26] [27]. [22] mainly discusses how to use textual information in image to do classification. [23] approach this problem by utilizing global histogram information and support vector machines. Besides, some other works have also been done based on histogram as well [28] [29] [30].

2.3 Chapter organization

In this chapter, we present an algorithm which includes online SVM training [31] that allows the algorithm to be improved and customized by the users as the input

images accumulate. At the same time, it enables quick decision for mix images which is the most frequently copied image type, and significantly speeds up classification for picture documents. We will also introduce a method to make online SVM training and quick decision compatible with each other. The rest of the chapter is laid out as follows. Section 2.4 introduces overall structure and each feature for classification. Section 2.5 presents online SVM training, quick decision and how they combine. Section 2.6 describes experimental results. Finally, conclusions are provided in Sec. 2.7.

2.4 Classification flowchart and features

2.4.1 Overall Structure

The proposed classification strategy follows a hierarchical decision structure as shown in Fig. 2.3. The first SVM classifier is responsible to classify *text vs. nontext* images while the subsequent classifier takes charge of *picture vs. mix*. Both classifiers require a two dimensional feature vector. Given that the intended use of the page classification algorithm is as a real time application in a low-end digital copier, two pre-processing procedures are necessary for every original scanned document to reduce the computational load. We first transform the original scanned document into a gray scale image by averaging over the three RGB channels, which is easy for hardware implementation. Then, this gray level image is down-sampled to 75 dpi by block-averaging. So the input to the subsequent classifiers is a low-resolution gray-level image which makes the classification more challenging.

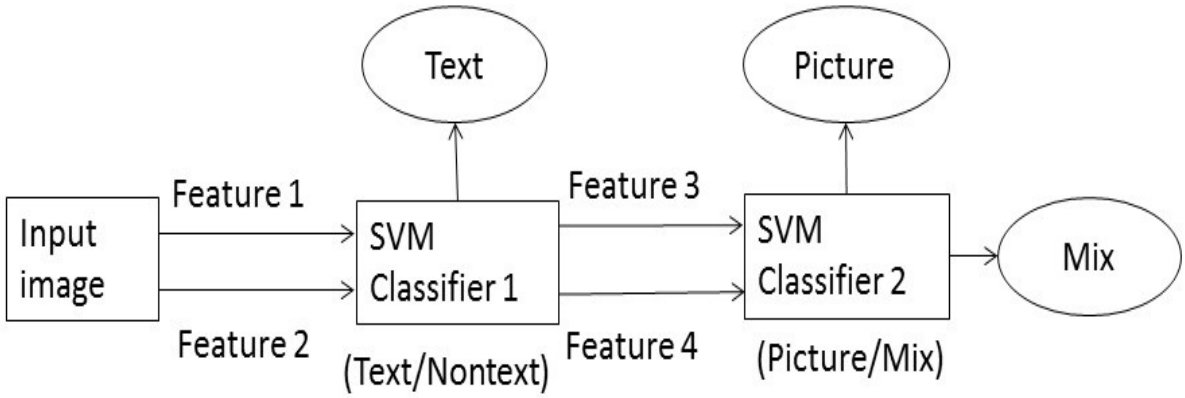


Fig. 2.3.: Classification structure

2.4.2 Pre-processing

As mentioned in Sec.2.1, we want to reduce the computational load in the firmware for real-time processing purpose. So we first transform the input RGB space image to NIQ space image. The NIQ color space is defined as followed

$$\begin{bmatrix} N \\ I \\ Q \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & -\frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2.1}$$

In equation 2.1, N is the luminance channel, I is the red-green color channel, and Q is the yellow-blue color channel. Compared with other existing opponent color space, NIQ color space is computationally efficient for the firmware. Unlike typical opponent color space, it simply averages over R, G and B channels to get luminance information, which can be easily achieved by printer firmware. Besides, we can obtain both I and Q channels by applying bit shift which is significantly faster than float point operation that is required in other opponent color space calculation.

By transforming the input RGB space to this opponent NIQ color space, we can obtain the luminance information from N channel which is the only channel we need for doing classification to reduce computational load. After getting the luminance channel information, we down-sample the input 300 dpi luminance image to 75 dpi

by block-averaging. The averaged 75 dpi image O is obtained from original 300 dpi luminance image N through

$$O(m,n) = \sum_{i=0}^{3}\sum_{j=0}^{3} \frac{N(4m+i, 4m+j)}{16} \tag{2.2}$$

2.4.3 Features

The first two features are inputs for the text/nontext classifier as shown in Fig. 2.3. The latter two features are the input features for the subsequent picture/mix classifier.

2.4.4 Histogram flatness score

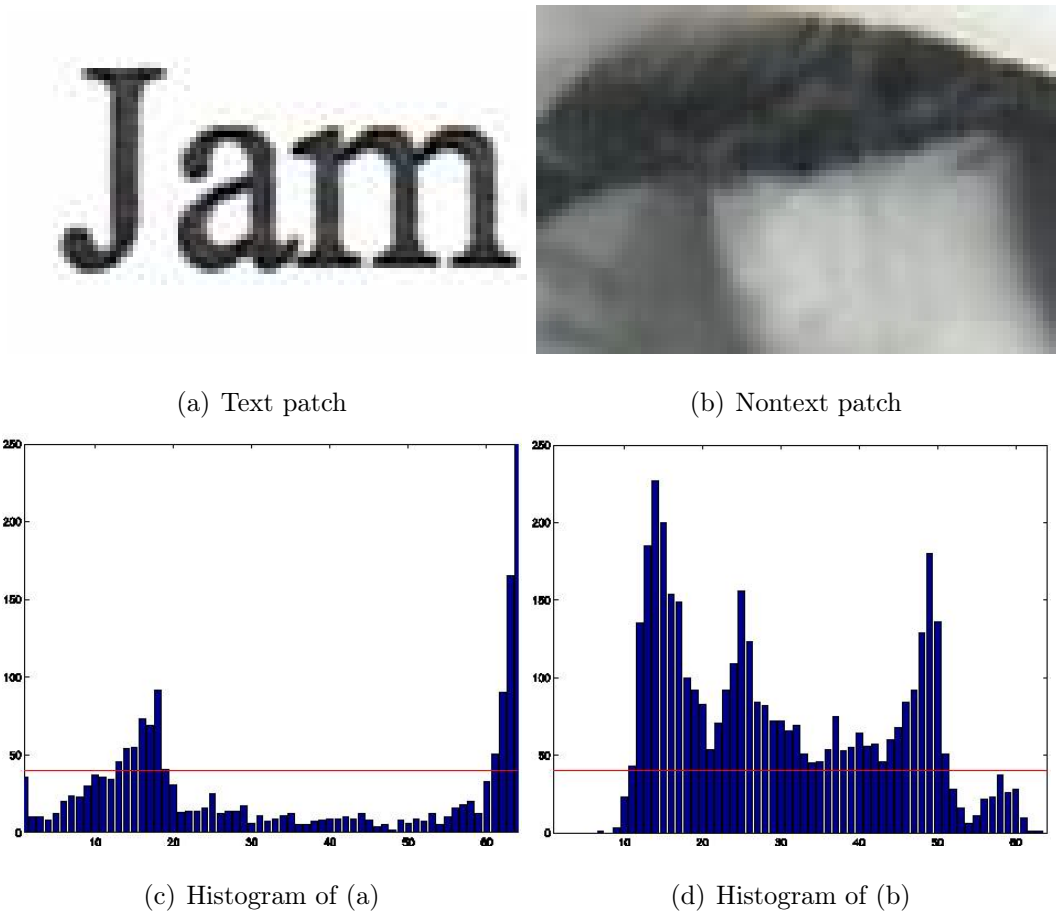
This feature is proven effective in [9]. Typically, the histogram for a text image tends to have sharper peaks while the histogram for a non-text image is more uniform. An example to illustrate this is given in Fig. 2.4

We divide the input image into blocks of 8×64 pixels. For every block, we build a 64 bin histogram. The span of a histogram for every block is defined as the maximum number of consecutive bins whose values are greater than a threshold value K. The span of an image is defined as the maximal span over all blocks. The feature vector X of one image consists of the spans of the image for different threshold values. In our experiments, $K = 3, 6, ..., 30$. (X is a 10×1 vector). The feature vector X is assumed to be a Gaussian Mixture [32] random variable with mean m_i, covariance matrix Λ_i, and prior probability $\pi_i = 0.5$ ($i = text$, $nontext$). m_i and Λ_i are estimated from the feature vectors of all text and nontext images in the training set respectively. For a given feature vector X of a target image, the log-likelihood ratio $L(X)$ is defined as:

$$L(X) = \ln(\pi_{nontext} P(X|m_{nontext}, \Lambda_{nontext})) - \ln(\pi_{text} P(X|m_{text}, \Lambda_{text})) \tag{2.3}$$

From 2.3 we can have

$$L(X) \propto -(X - m_{nontext})^T \Lambda_{nontext}^{-1} (X - m_{nontext}) + (X - m_{text})^T \Lambda_{text}^{-1} (X - m_{text}) \tag{2.4}$$



(a) Text patch

(b) Nontext patch

(c) Histogram of (a)

(d) Histogram of (b)

Fig. 2.4.: Example of histogram flatness for text and picture documents

If we assume the two class have the same covariance matrix $\Lambda = \Lambda_{text} = \Lambda_{nontext}$, we now define the histogram flatness score based on 2.4

$$HF = (m_{nontext} - m_{text})\Lambda^{-1}X. \tag{2.5}$$

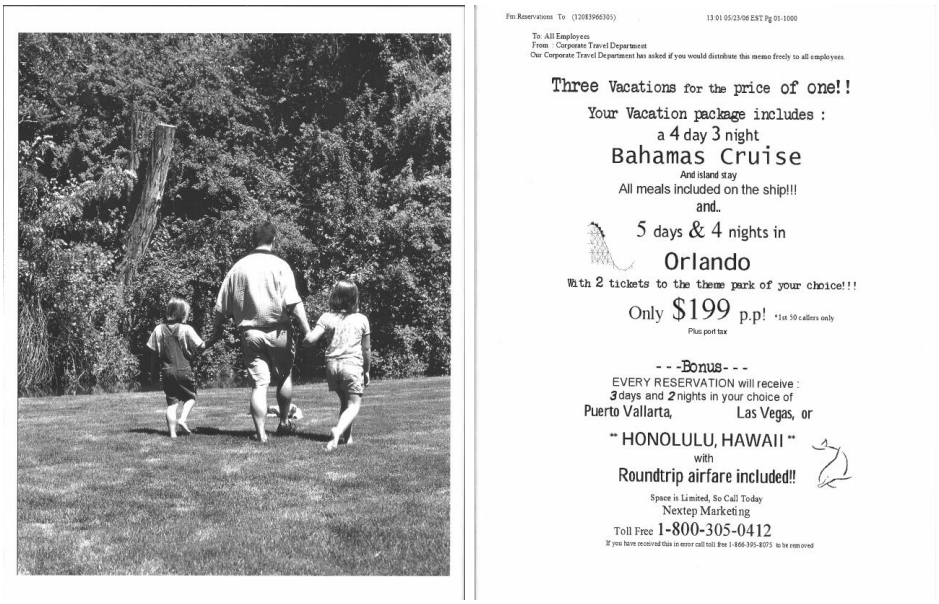
Here, $m_{nontext}$ and m_{text} are the average span vectors we obtain from the training sets for text and non-text documents, respectively; and Λ is covariance matrix between the text and non-text documents in the two training sets. Because the histogram flatness score for the image is defined as the maximum HF over all the blocks, we can tell that it is always monotonically increasing.

2.4.5 Histogram variability score

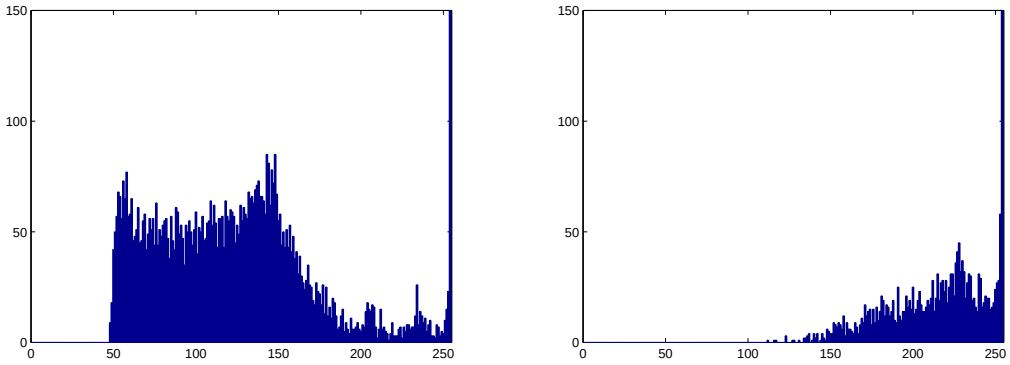
It is reasonable to assume that the non-text region of a text document contains only a few gray level values. So we build a block-mean histogram for the image to calculate a histogram variability score. We first cut the image into blocks of 8×8 pixels and calculate the mean pixel value of each block that contains no text edge. *Text edges* are defined as three neighboring pixels in either the horizontal or vertical direction satisfying the following two criteria: (1) their pixel values are monotonically increasing or decreasing; and (2) the difference between the first and third values is larger than a threshold T ($T = 100$ in our application). So we build a 256-bin histogram of mean values of all blocks that do not contain a text edge. Finally, the histogram variability score of an image is defined as the largest number of non-zero bins of the block mean histogram. Figure. 2.5 gives an example to show typical difference between text and nontext documents in terms of block-mean histogram. We can see that in the example given in 2.5, the text document has more zero-value bins in 8×8 block-mean histogram compared to the nontext document, especially in the low-luminance region. It is also a feature that increases monotonically as the algorithm processes the image in raster order.

2.4.6 Text edge count

The *text edge count* utilizes the fact that mix images typically have more edges compared to picture images [9]. Several publications have discussed text edge in an image [33] [34] [35] [36]. For our application, text edges are defined as three neighboring pixels in either the horizontal or vertical direction satisfying the following two criteria: (1) their pixel values are monotonically increasing or decreasing; and (2) the difference between the first and third values is larger than a threshold T. The text edge count is the maximum number of edges over all the 64×64 blocks. Thus it is also a monotonically increasing feature.



(a) Nontext document (b) Text document



(c) 8×8 block-mean histogram of (a) (d) 8×8 block-mean histogram of (b)

Fig. 2.5.: Example of block mean-histogram of text/nontext documents

2.4.7 Stroke color variance

The second feature stroke color variance uses the fact that there is high color consistency inside a character stroke. We cut the image into blocks of 64×64 pixels. For every block, we first distinguish background pixels, edge pixels, and potential character stroke pixels. The K-means clustering algorithm is used in this process to

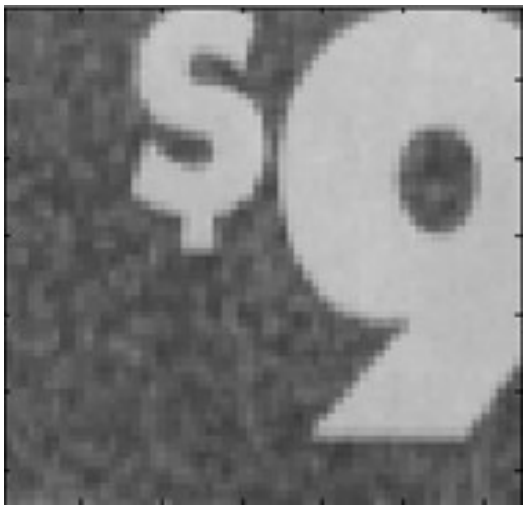
distinguish among these three types of pixels. The feature vector of every pixel for K-means clustering includes two elements: *pixel value* and *pixel local variance*. Pixel local variance is defined as the sum of the absolute differences between a given pixel $p(m, n)$ and its neighbours

$$D(m, n) = \sum_{(i,j)\in\{\pm1,\pm1\}} |p(m, n) - p(m + i, n + j)|. \tag{2.6}$$

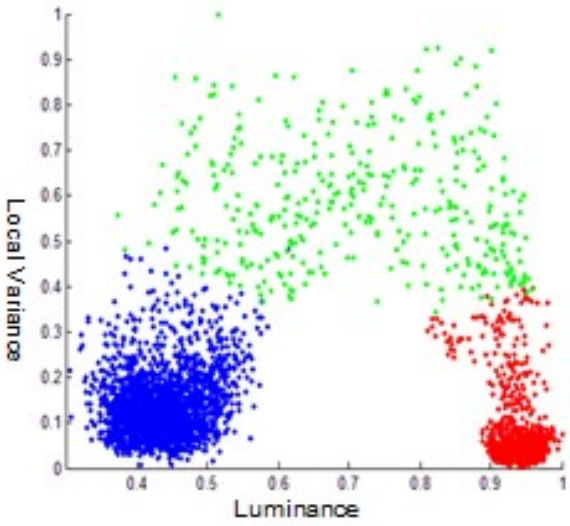
We normalize the feature vector for every pixel and classify it into one of three clusters. These three clusters represent: (1) pixels with low gray value and low local variance (background or stroke), (2) pixels with high gray value and low local variance (background or stroke), and (3) pixels with high local variance (edges). The smaller cluster with low local variance is identified as the possible stroke region. An example for this clustering process is given in Fig. 2.6. We calculate the *pixel value variance* β in this cluster to get the stroke color variance of this block. The stroke color variance β^* of an image is defined as

$$\beta^* = min(\sqrt{\beta}) \tag{2.7}$$

Since the stroke color variance of an image is the minimum of β overall the blocks, we can conclude that it is a feature that decreases monotonically as the algorithm processes the image in raster order.



(a) Original 64×64 block



(b) Pixel clustering

(c) Block with candidate region

Fig. 2.6.: An example of candidate text region

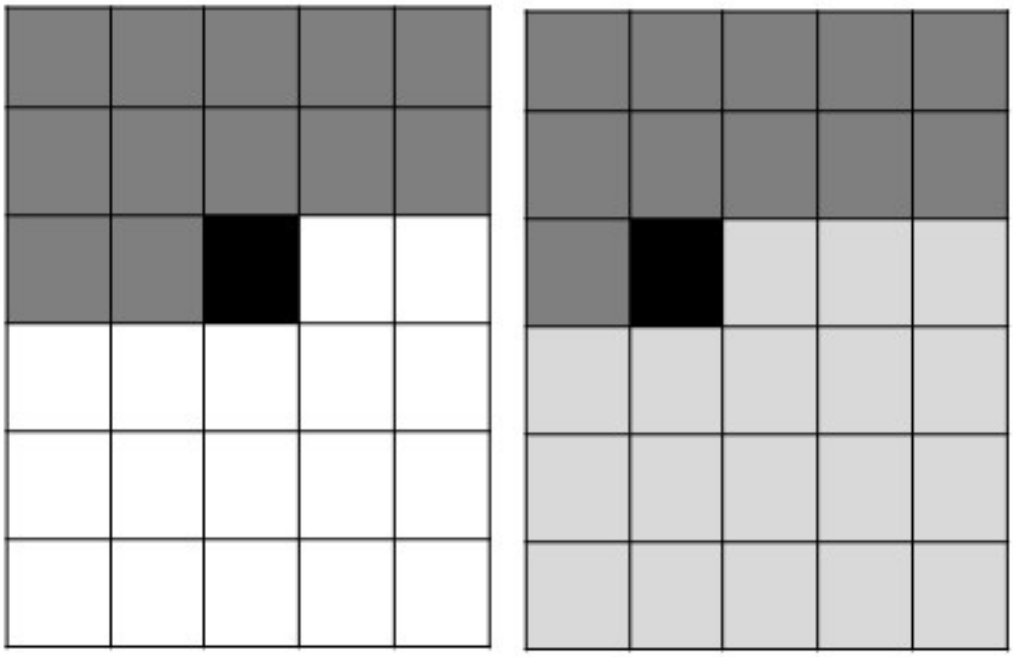
2.5 Quick decision and online SVM

2.5.1 Quick decision

Among the three types of image, the mix type is most frequently copied by AIO devices. This fact motivates us to come up with a strategy to speed up the mix decision. In fact, for many mix images, it is not necessary to wait until the very last block to make a final decision. Existence of both text and picture components may be already detected before reaching a certain block, which allows an early decision for the mix type. Figure 2.7(a) illustrates the mix quick decision. The program initially calculates all four features during processing of the gray blocks. When the program reaches the black block, it has obtained sufficient evidence of both textual and pictorial components. So the remaining white blocks are ignored and the image is classified as mix.

A relatively weaker version of quick decision for picture type images is *feature discard*. The hierarchical decision structure shown in Fig. 2.1 suggests that if we find pictorial components in certain block, we may skip the text/nontext classifier for the remaining blocks. This is based on the fact that the existence of pictorial components eliminates the possibility of pure text; and thus only text detection is necessary to distinguish between mix and picture. An example of the feature discard for the picture type is given in Fig. 2.7(b). The program initially calculates all four features when processing dark-gray blocks. When the program reaches the black block, the pictorial components are detected and the text/nontext classifier will be skipped for the remaining blocks. Then only the features for the mix/picture classifier need to be calculated in the light-gray blocks, which leads to a speedup.

It is also possible that the program decides to apply the feature discard followed by a quick decision. This happens when the program first detects a pictorial component, and start skipping the text/nontext classifier. Then it confirms the textual parts which leads termination.



(a) Quick decision (b) Feature discard

Fig. 2.7.: Examples of quick decision and feature discard

A very important foundation of the quick decision and feature discard is that the SVM classification requires much less computation compared to feature extraction. Based on that, we can afford to apply SVM classification in every block.

2.5.2 Online SVM training

Offline algorithm provides little flexibility for users to customize and improve the AIO devices [15]. However, one major issue for the online SVM [31] is that it is not naturally compatible with the quick decision capability. In our application, the two SVM decision boundaries of text/nontext and mix/picture classifiers are:

$$y_1 = f_t(x_1), \tag{2.8}$$

$$y_2 = f_m(x_2), \tag{2.9}$$

where x_1 is the histogram flatness score, y_1 is the color variability score, x_2 is stroke color variance, and y_2 is the text edge count. We require that f_t be monotonically decreasing and f_m monotonically decreasing to guarantee the quick decision capability. Examples to illustrate this are given in Fig. 2.8. Assume that i blocks of an input image have been processed and the feature vectors at i are $S_t^i = (x_1^i, y_1^i)^T$, $S_m^i = (x_2^i, y_2^i)^T$. Based on the design of features in Sec. 2.4.3, we have

$$x_1^j \geq x_1^i, \;\; y_1^j \geq y_1^i, \;\; x_2^j \leq x_2^i, \;\; y_2^j \geq y_2^i; \;\; \forall j > i, \tag{2.10}$$

This means the *final* feature vectors S_t should be located in the first quadrant with respect to (x_1^i, y_1^i), as indicated by the dashed lines in Fig. 2.8(a). Similarly, the *final* feature vector S_m should be located in the second quadrant with respect to $(x_2^i, y_2^i)^T$ as indicated by the dashed lines in Fig. 2.8(b). Assume there is a SVM decision boundary as shown Fig. 2.8(a), which is convex. At the ith block, the feature vector $(x_1^i, y_1^i)^T$ is already in the nontext area. According to our design of the quick decision in Sec. 2.5.1, the text/nontext classifier should be skipped from now on, because we can conclude that this image belongs to the nontext type. However, because the decision boundary starts to increase and crosses the dashed line, it generates a shadowed area that the early decision will be incorrect. So f_t needs to be monotonically decreasing; and similarly, f_m should be monotonically increasing.

In order to guarantee the validity of this early decision, we need to choose the appropriate kernels and parameters in the initial offline SVM training. More importantly, we need to preserve the monotonicity of the decision boundaries during the online SVM training process.

In SVM training, separating function is defined as

$$f(x) = \sum_j \alpha_j y_j K(x_j, x) + b, \tag{2.11}$$

where x_j is every training vector, and $y_j = \pm 1$ represents its label. In order to obtain α_j, we need to minimize a convex quadratic function under constraints

$$\min_{0 \leq \alpha_i \leq C} : W = \frac{1}{2} \sum_{i,j} \alpha_i Q_{ij} \alpha_j - \sum_i \alpha_i + b \sum_i y_i \alpha_i, \tag{2.12}$$

where $Q_{ij} = y_i y_j K(x_i, x_j)$ and with *Kuhn-Tucker* conditions:

$$g_i = \frac{\partial W}{\partial \alpha_i} = y_i f(x_i) - 1 = \begin{cases} \geq 0; & \alpha_i = 0 \\ = 0; & 0 \leq \alpha_i \leq C_i \\ \leq 0; & \alpha_i = C_i \end{cases} \tag{2.13}$$

$$\frac{\partial W}{\partial b} = \sum_j y_j \alpha_j = 0, \tag{2.14}$$

where C_i is the penalty constant for each training image. Since different kinds of misclassifications are not equally severe in our application, we assign different C values for each image type, which means $C_i \in \{C^t_{text}, C^t_{nontext}, C^m_{mix}, C^m_{picture}\}$. Three types of vectors are: the set S of *margin support vectors* strictly on the margin ($y_i f(x_i) = 1$); the set E of *error support vector* exceeding the margin; and the remaining set R are *ignored vectors* within the margin. The entire training data set is denoted as D. If a new vector with coefficient α_c is added, the margin vector coefficients change accordingly to keep the KT conditions satisfied. Since $g_i \equiv 0$, we have

$$\Delta b = \beta \Delta \alpha_c, \tag{2.15}$$

$$\Delta \alpha_j = \beta_j \Delta \alpha_c; \quad \forall j \in D. \tag{2.16}$$

The coefficient sensitivities are given by

$$\begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_{l_s} c} \end{bmatrix} = -\Theta \begin{bmatrix} y_c \\ Q_{s_1 c} \\ \vdots \\ Q_{s_{l_S} c} \end{bmatrix}, \tag{2.17}$$

where Q is the extended kernel Jacobian, $\Theta = Q^{-1}$, and $\beta_j \equiv 0$ for all j outside S. Combined with the differential expression of g_i, we have:

$$\Delta g_i = \gamma_i \Delta \alpha_c; \quad \forall i \in D \cup \{c\}, \tag{2.18}$$

where γ_i is the sensitivity defined as

$$\gamma_i = Q_{ic} + \sum_{j \in S} Q_{ij} + y_i \beta; \quad \forall i \notin S. \tag{2.19}$$

Given the equations 2.15 and 2.17, we can implement the *bookkeeping* step which determines the largest possible increment $\Delta\alpha_c$ according to:

1. $g_c \leq 0$, with equality when c joins S;
2. $\alpha_c \leq C_c$, with equality when c joins E;
3. $0 \leq \alpha_j \leq C_j, \forall j \in S$, with equality 0 when j transfer from S to R, and equality C_j when j transfer from S to E;
4. $g_i \leq 0, \forall i \in E$, with equality when i transfer from E to S;
5. $g_i \geq, \forall i \in R$, with equality when i transfer from R to S.

Lastly, we need to recursively update Θ for all c to S according to the equation 2.19 in [31].

$$\Theta \leftarrow \begin{bmatrix} & & 0 \\ & \Theta & \vdots \\ 0 & \cdots & 0 \end{bmatrix} + \frac{1}{\gamma_c} \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_{l_s}} \\ 1 \end{bmatrix} \tag{2.20}$$

At this point, we can obtain the unconstrained SVM decision boundary by setting $f(X) = 0$ in 2.11, where X is a two dimensional feature vector $X = (x, y)^T$ as given in 2.8 and 2.9. Both classifiers apply polynomial kernels with order of 2 because a low-order polynomial kernel is more likely to generate smooth and monotonic decision boundary in the feature domain. More importantly, the monotonicity of a decision boundary which is generated by the second-order polynomial kernel can be easily decided by solving two quadratic equations. So the kernel function $K(X_j, X)$ in equation 2.11 can be expressed as:

$$K(X_j, X) = (X_j^T X + c)^2. \tag{2.21}$$

Based on equation 2.16, we can conclude that the decision boundary which is generated by setting $f(X) = 0$ in equation 2.11 has the form:

$$P(x, y) = Ax^2 + Bxy + Cy^2 + b = 0. \tag{2.22}$$

Subsequently, equation 2.16 is expressed differentially as:

$$\frac{\partial y}{\partial x} = -\frac{2Ax + By}{Bx + 2Cy}. \tag{2.23}$$

To guarantee the monotonicity of $f(x)$ in the feasible feature domain is equivalent to satisfying the following conditions:

1. There is no more than one solution $x_0 \in (x^{min}, x^{max})$ to the quadratic equation $P(x, y_0) = 0$, $\forall y_0 \in (y^{min}, y^{max})$;

2. There is no more than one solution $y_0 \in (y^{min}, y^{max})$ to the quadratic equation $P(x_0, y) = 0$, $\forall x_0 \in (x^{min}, x^{max})$;

3. $\frac{\partial y}{\partial x} \geq 0$ for all solutions (x_0, y_0).

Note that we choose polynomial kernel with order of 2 only because of its simplicity. Actually, any kernel which is partially differentiable with respect to X can be used to check the monotonicity since we use only two dimensional feature vector.

Finally, the overall incremental procedure can be described as algorithm 1 . After the initial offline training, assume that a new training point c is added into the training set D. So $l \rightarrow l + 1$ and $D^{l+1} = D^l \cup \{c\}$. The solution $\{\alpha_i^{l+1}, b^l + 1\}$, $i = 1, \cdots, l + 1$ should be calculated based on $\{\alpha_i^l, b^l\}$, the Jacobian inverse Θ and the new training data (x_c, y_c).

Algorithm 1

1. Initialize α_c to 0;

2. If $g_c > 0$, terminate (c does not belong to either margin or error vector);

3. If $g_c \leq 0$,apply the largest possible increment α_c so that (the first) one of the following conditions occurs:

   (a) $g_c = 0$: Add c to margin set S, update Θ accordingly, and terminate;

   (b) $\alpha_c = C$: Add c to error set E, and terminate;

   (c) Element of D^l move across S, E, and R (ignored set) as described in *bookkeeping*: Update membership of elements and, if S changes, update R accordingly;

   (d) Check the monotonicity of the updated decision boundary $y = f^{l+1}(x)$ in the feasible feature domain. Reject the update if the monotonicity of f^{l+1} is inconsistent with f^l;

4. For the text image defined by the user, apply steps 1-3 for the text/nontext classifier. For the mix and picture images defined by the user, apply steps 1-3 for both text/nontext and mix/picture classifiers.

2.6 Experimental result

2.6.1 Dataset description

For our experiment, we have a library of 888 images. These images were carefully selected and labelled by HP engineers based on their contents. This library covers a variety of images which are representative of frequently scanned or copied pages. Details of this library are listed in Table 2.1 below.

2.6.2 Offline training

Since we require the user to choose color/mono mode, we can train separate pairs of classifiers for text/non-text and mix/picture for the two cases of mono and color to optimize overall classification performance. The two training results for the text/nontext

Table 2.1.: Number of documents of every class in the image library

	color mix	color text	color picture	color total
number	132	110	225	467
	mono mix	mono text	mono picture	mono total
number	93	105	223	421

classifier are provided in Fig. 2.9. In Fig. 2.9, the green dots represent text images in the library set while the red dots are nontext images in the library. The two training results for the subsequent mix/picture classifier are given in Fig. 2.10. In Fig. 2.10, the green dots represent mix images while the red dots represent picture images. In the training process, we need to consider a balance between accuracy and overfitting by choosing appropriate kernel functions and their corresponding parameters. Thus, we apply nonlinear kernels in the SVM for the text/nontext classifier. For both mono and color text/non-text classifiers, we apply polynomial kernels. For mono documents. the degree of the polynomial is $d = 2$. For color documents, the degree is $d = 3$. The polynomial kernel is defined as

$$K(x, y) = (x^T y + c)^d \tag{2.24}$$

Similarly, we apply kernels in the SVM for the mix/picture classifier. For both mono and color mix/picture classifiers, we apply Gaussian radial basis function kernels. For mono documents, we apply a Gaussian radial basis function kernel with free parameter $\sigma = 1.1$. For color documents, we use $\sigma = 0.8$. The Gaussian radial basis function kernel is defined as

$$K(x, y) = exp(-\frac{||x - y||^2}{2\sigma^2}). \tag{2.25}$$

To evaluate the performance of our algorithm, we perform a full-fold cross validation and compare it with the algorithm in [9] The overall classification accuracy of both algorithms is provided below in Table 3. For our application, misclassifications

are not equally weighted. For example, misclassifying mix as text is a more severe error than misclassifying picture as mix, due to the viewer's sensitivity to text distortion. So in the training process, our goal is to minimize the cost function

$$F = \sum_i P_i W_i \tag{2.26}$$

where i is type of misclassification and P_i represents the percentage of misclassification for misclassification type i. Weights for all types of misclassifications are given in Table 4.

Table 2.2.: Overall classification accuracy based on our proposed method

	classification results		
ground truth	mix	text	picture
mix	83.9%	6.3%	9.7%
text	16.2%	83.8%	0.0%
picture	17.5%	0.0%	82.5%

(a) Mono document accuracy

	classification results		
ground truth	mix	text	picture
mix	94.1%	0.7%	5.2%
text	22.7%	75.5%	1.8%
picture	20.9%	0.0%	79.1%

(b) Color document accuracy

By comparing Tab. 2.2 and Tab. 2.3, we plug them it to 2.26 and we find that for color image, the weighted error improvement is 39.1%. For mono image, the weighted error improvement is 9.4%

Table 2.3.: Overall classification accuracy based method in [9]

	classification results		
ground truth	mix	text	picture
mix	83.9%	10.7%	5.4%
text	9.4%	89.6%	1.0%
picture	42.3%	0.0%	57.7%

(a) Mono document accuracy

	classification results		
ground truth	mix	text	picture
mix	96.9%	2.3%	0.8%
text	22.7%	77.3%	0.0%
picture	81.0%	0.4%	18.6%

(b) Color document accuracy

Table 2.4.: Weights of misclassifications for both color and mono.

	classification results		
ground truth	mix	text	picture
mix	0	5	5
text	3	0	5
picture	1	5	0

2.6.3 Quick decision

We compare the average processing time for every image before and after applying quick decision and feature discard. The timing analysis is based on the personal computer as shown in Tab. 2.5

Table 2.5.: Speedup for quick decision and feature discard.

Image type	Quick decision	Original	Speedup
Color mix	24.9 ms	77.0 ms	3.09
Color text	79.9 ms	79.3 ms	0.99
Color picture	35.7 ms	44.1 ms	1.24
Mono mix	21.1 ms	92.4 ms	4.38
Mono text	84.2 ms	83.2 ms	0.99
Mono picture	44.1 ms	50.2 ms	1.14

Table 2.6.: Average classification time for each type of document on Beagle Board

	Color mix	Color text	Color picture	Mono mix	Mono text	Mono picture
Time	302 ms	910 ms	413 ms	291 ms	954 ms	527 ms

According to Tab. 2.5, we can see that our algorithm significantly speeds up the classification for mix document because of quick decision strategy. As expected, it also speeds up the classification for picture document which is cause by feature discard strategy. However, we can see that it slightly slows down classification for text document. That is because we need to do extra SVM classification after each block. However, the computation for SVM is much smaller than feature extraction for each block, so quick decision and feature discard which save extra feature extraction could significantly speeds up the whole process even we apply extra SVM classification.

In order to test the real-time capability of our proposed algorithm, we implement it on the Beagle Board which is close to the AIO device firmware environment. We record the average processing time for each type of document, and the results are given in Tab. 2.6. We can see that the average processing time for every type of document is no more than 1 second.

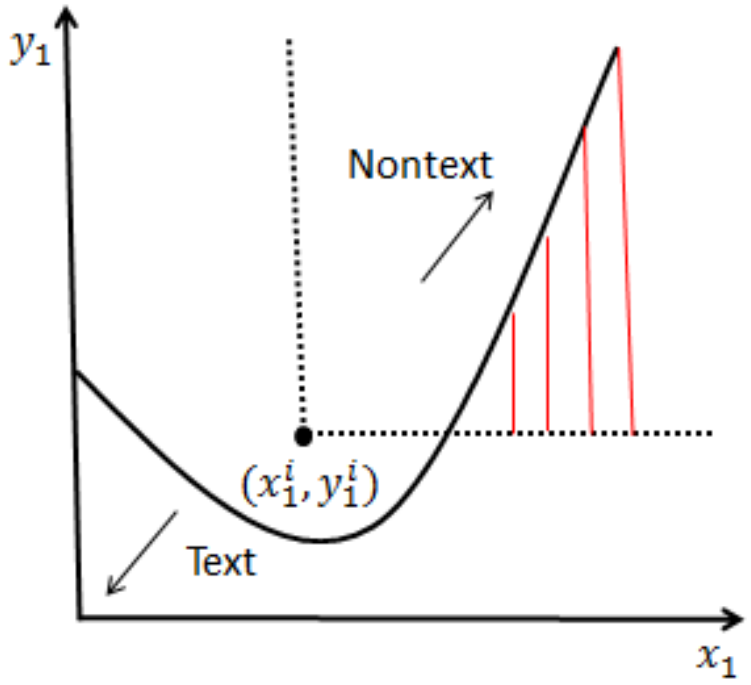
2.6.4 Online training

In our online training experiment, we use $C^t_{text} = 1$, $C^t_{nontext} = 2$, $C^m_{picture} = 1$, $C^m_{mix} = 2$. Figure. 2.11 illustrates our experimental online training process. The dashed line represents the initial decision boundary based on the offline training (31 nontext, 41 text). The solid line is the decision boundary after processing 20 more online training images (10 nontext, 10 text), which simulates 20 user inputs. We can see that the initial monotonicity is preserved. In real situation, we also need to store the initial decision boundary before any online learning. So that customer can always reset classification to the default settings.

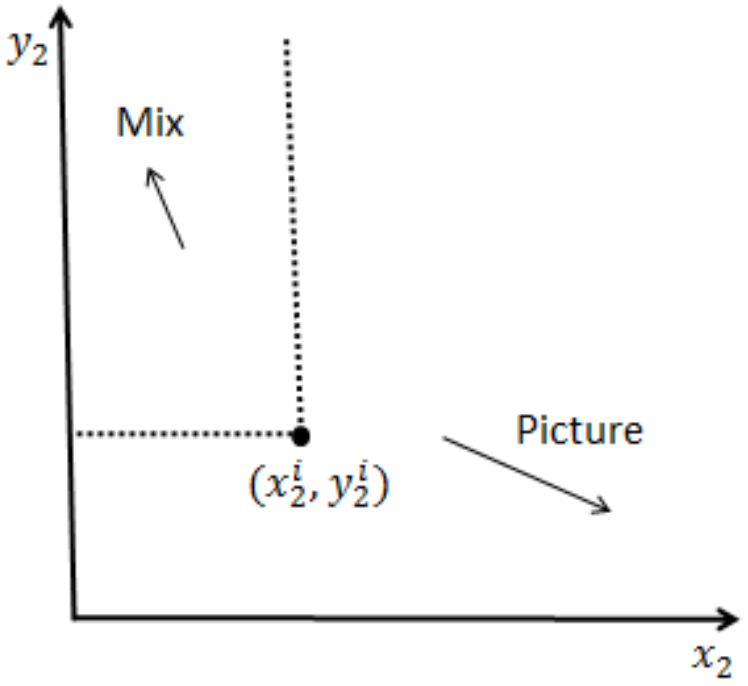
2.7 Summary

In this chapter, we have presented an SVM-based algorithm to classify the input image for a digital copier. The proposed classification algorithm follows a binary-tree, two-stage decision structure. Each individual classifier takes two input features and makes a decision based on them. To control the error rate and model complexity, we apply polynomial and Gaussian radial basis kernel functions. Based on the classification result, digital copier chooses corresponding processing pipeline to produce relatively high quality output. The classification is based on the low resolution image which is 75 dpi. We utilize only local features for both classifier input, and that enable the quick decision and feature discard strategies. With quick decision strategy, we significantly speeds up the classification process. Besides local monotonic features, we also require monotonicity of SVM decision boundary to enable quick decision. We also introduced an online image classification algorithm for the AIO device. Unlike the traditional incremental and decremental SVM, our algorithm requires that the derived decision boundaries preserve the initial monotonicity, so that quick decision conditions will not be destroyed. With this constraint, user can add new training images to improve the performance of the AIO device.

Because this algorithm is used in the low-end AIO product, we can not afford extensive computation for classification. Given this fact, we only extract two features for every classifier. For the same reason, we choose polynomial kernel with order of 2 when applying online SVM update. However, same idea can be applied to feature vectors with higher dimension and other kernels which are differentiable.

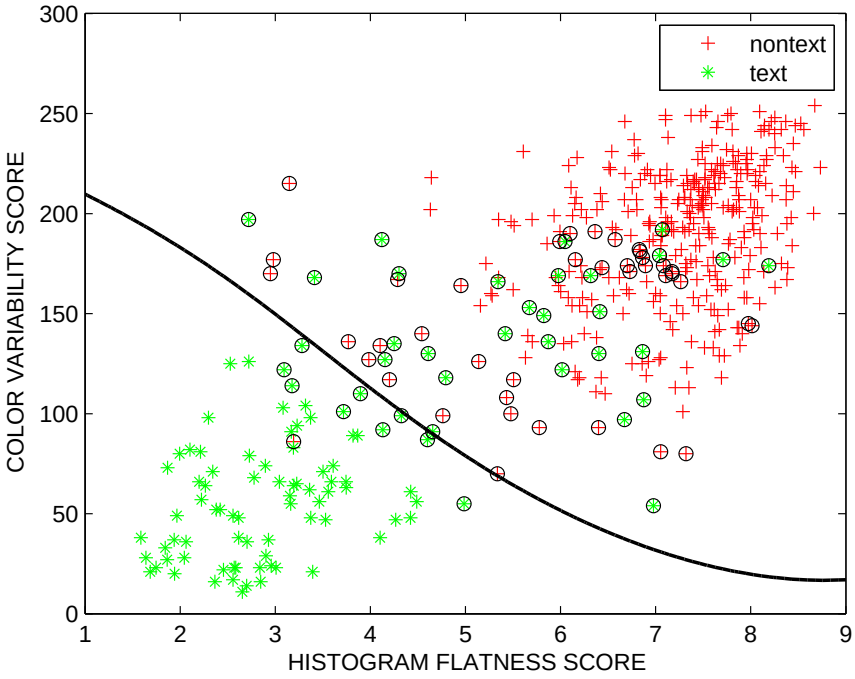


(a) Text/nontext feature space

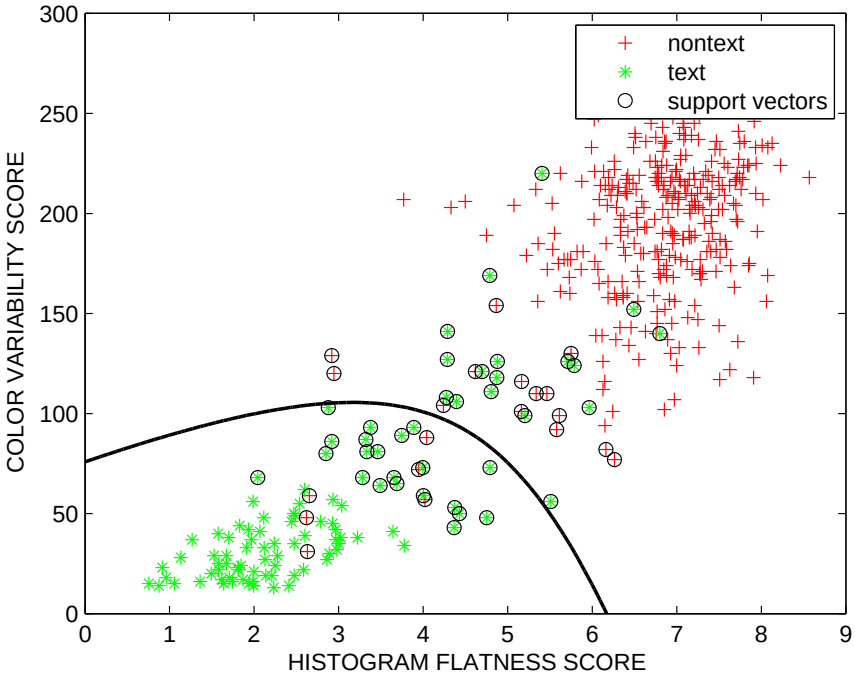


(b) Mix/picture feature space

Fig. 2.8.: Examples illustrating the role of decision boundary monotonicity

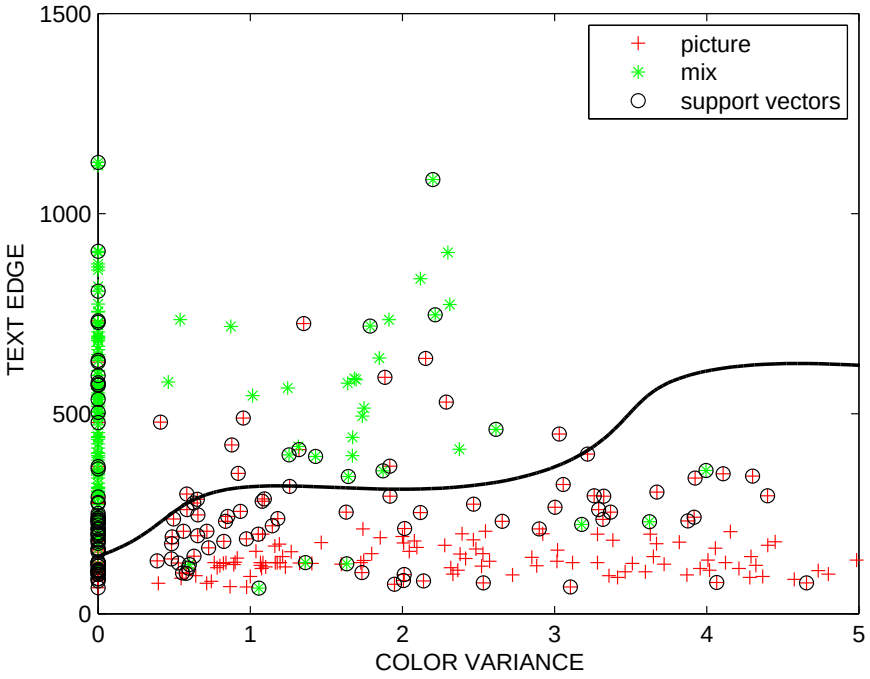


(a) Color image classification

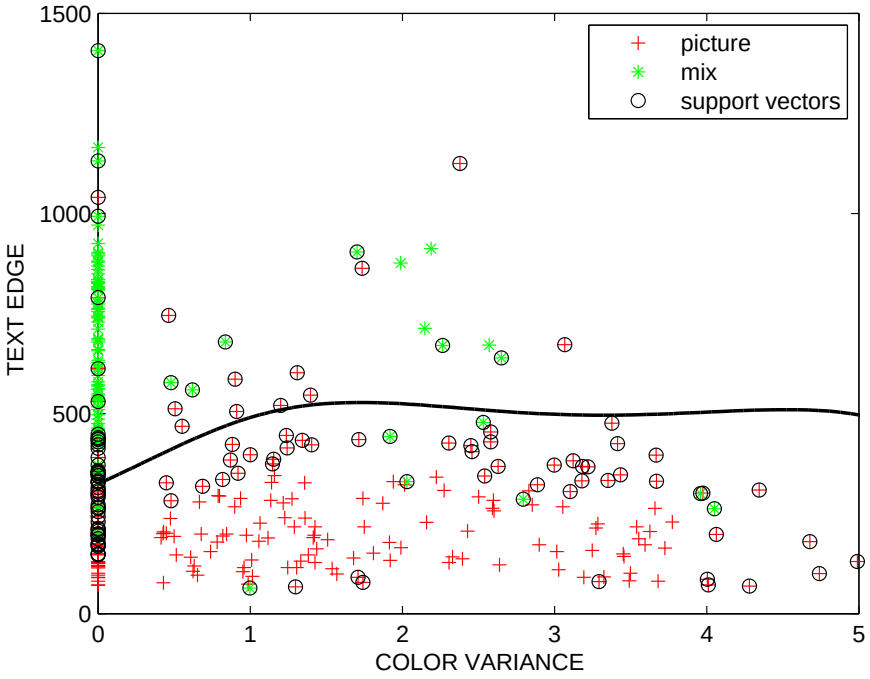


(b) Mono image calssification

Fig. 2.9.: Text/nontext SVM classification boundary



(a) Color image classification



(b) Mono image calssification

Fig. 2.10.: Mix/picture SVM classification boundary

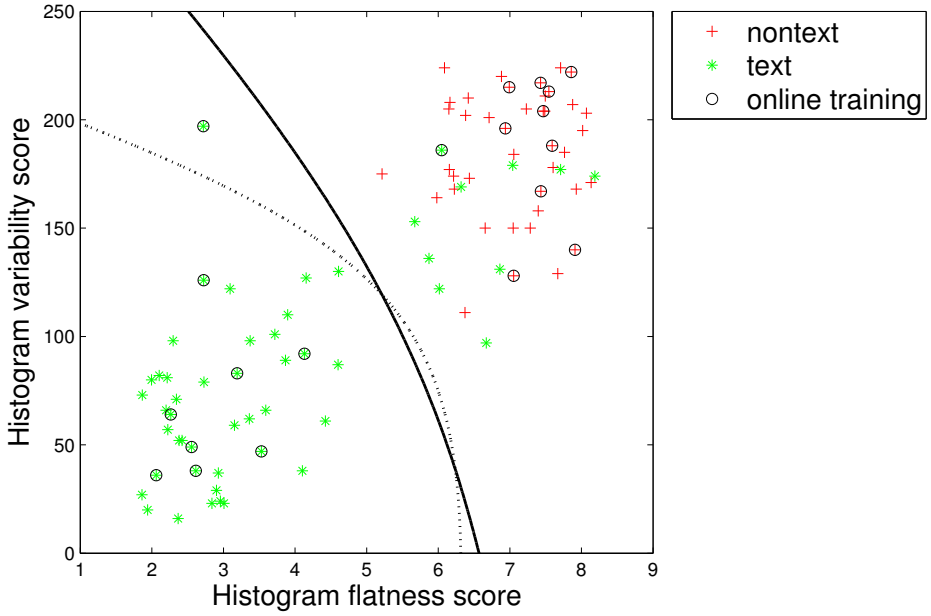


Fig. 2.11.: Decision boundary evolution

3. EXTENDED SCANNED IMAGE CLASSIFICATION FOR ALL-IN-ONE PRINTER

3.1 Problem statement

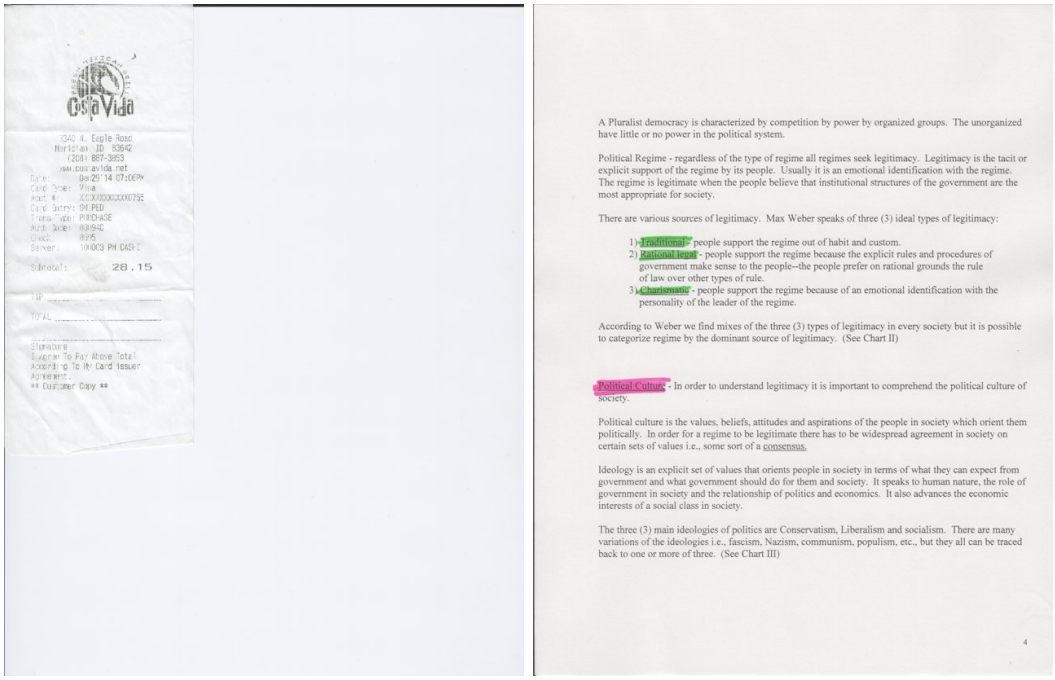
In Chapter 2, we discuss the scanned image classification problem for AIO printer. The purpose for the classification is sending image to its optimal processing pipeline to reach the best output image quality. However, we discussion is limited to classify the input image into one of three classes: mix, text or picture. In some other applications, we need to distinguish more than 3 classes. Some printers are equipped with other processing pipelines which can better handle more than mix, text and picture. At the same time, high-end product with more processing pipelines typically have greater computational power, and thus quick decision is not necessary.

3.2 Introduction

For some AIO printers, there are several processing pipelines which are designed specifically for a type of image. These processing pipelines can perform content-based optimization for the input images and thus reach the optimal copy quality for each input. In Chapter 2, we discuss this problem when we have three target types and quick decision for mix type is required. However, we'd like to extend this topic in this chapter to make out system more comprehensive. First, we now consider a high-end AIO device which is equipped with 5 processing pipelines instead of 3. Second, given the greater computational power of the device, we do not require quick decision for certain type any more. Our AIO device can safely calculate all features and make classification in real-time so we do not need to bias to certain type like in Chapter

2. These two differences lead to changes in both feature design and classification structure compared with Chapter. 2.

Other than the three types we discuss in Chapter 2, we now introduce two more types: *receipt* and *highlighted-text*. Examples of these two types are given in Fig. 3.1.



(a) Receipt (b) Highlighted-text

Fig. 3.1.: Examples of receipt and hightlighted-text

Receipt represents a type of document images that have very low contrast and faded text. Its corresponding processing pipeline needs very strong contrast enhancement and text recovery [37, 38] to guarantee readability of the copy. Compared with example given in Fig. 2.2, receipt type needs even stronger enhancement for better quality. Highlighted type typically contains colors that are more saturated compared with color in the natural images. An example is people using fluorecent pen to highlight text line in a document image. These highlighter usually produces very saturated color in order to attract attention of readers. However, these saturated colors [39, 40] are sometimes difficult to captured by the scanning devices, or it is sensed as less

saturated color by the scanner. Figure 3.2 shows an captured image with highlight colors, which illustrate the vision of our device when sensing those colors. WE can tell the colors in upper half are very weak in eyes of scanners, but colors in lower half are much more visible. However, if all these colors are very visible and clear on the physical page on which we put highlight marker. So if we can tell the input image is a highlighted text document, we can produce more saturated color when generating output to match the appearance of actual highlighter ink. This can improve user experience when copying highlighted document page for better readability and preserving important area in the page.

But still, our device can only process the input image strip by strip without knowing the global information. And input color space is either RGB or LCH from hardware [41].

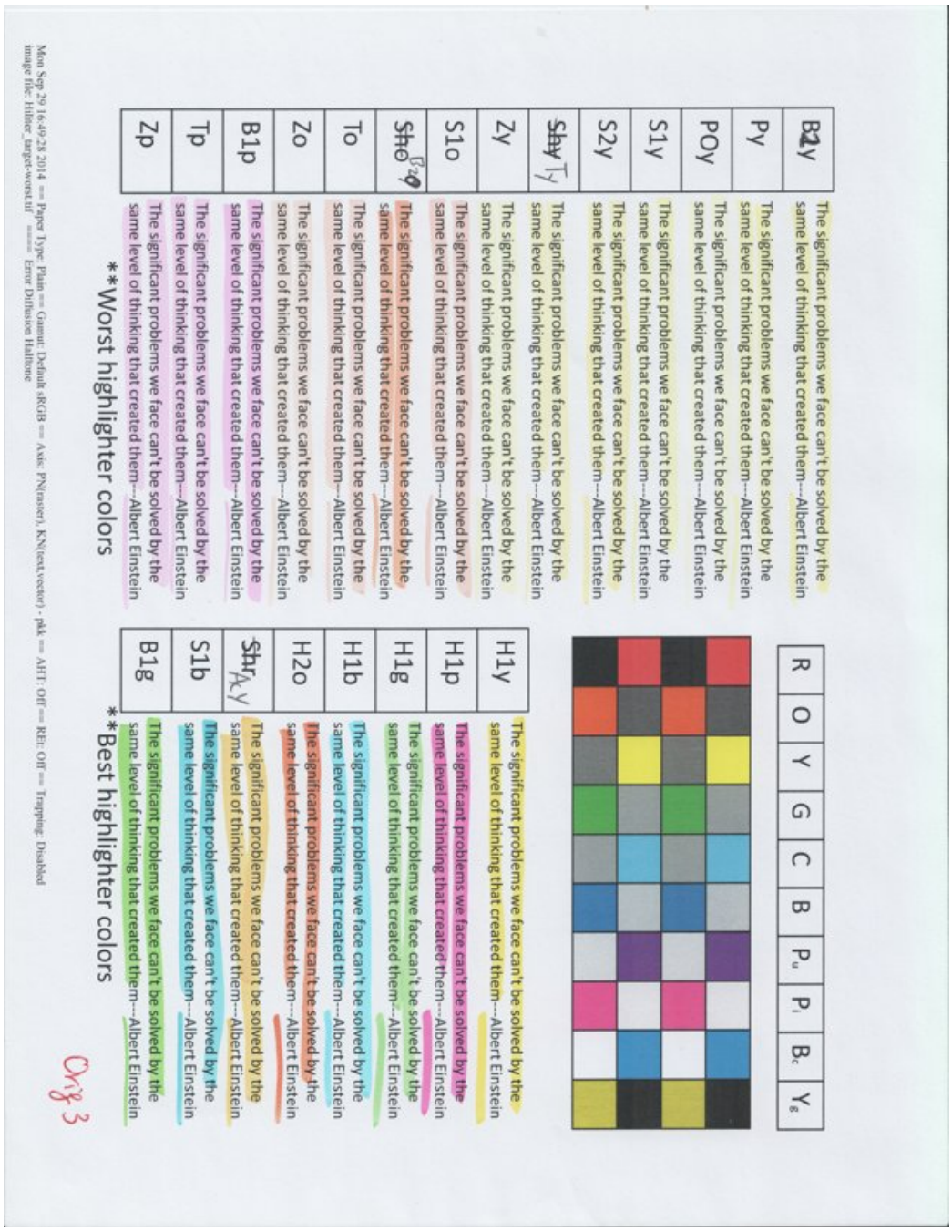


Fig. 3.2.: Highlighted colors that are difficult to capture

Subsequently, we need to develop new features which can distinguish these two types on top of features we used in Sec. 2.4.3. These features will be introduced in the latter section. Also, since quick decision in not necessary in this case any more, we do

not need to terminate the classification process before scanning the entire page. This gives us more flexibility when designing the classifiers and their structures. So we do not need to follow the hierarchical structure in Fig. 2.3 which is designed specifically for early termination of mix type.

3.3 Features

3.3.1 Chroma histogram flatness

Although both natural images and highlighted text have chroma information, a very significant difference between natural color image and highlight text is that highlighted text tends to include only one or few color while natural images have richer color information. This can be illustrated in Fig. 3.4 where we can find only pink in highlighted text image while chroma in natural image is very rich. Then we need a numerical value to represents this difference.

In Sec. 2.4.4, we introduce the Histogram Flatness Score in the luminance channel and it assumes that text image has more peaky histogram. Similarly, if we build a histogram for a highlighted text in chroma space, we can expect that there is a single or few peaks, while histogram for the natural image is more flat. Different from luminance histogram, chroma histogram is two-dimensional. Since our input image can be either RGB or LCH color space, we need to build histogram correspondingly. For RGB input, we transform it to YUV space and build histogram on UV plane. For YUV input, we can build histogram on CH space directly. As shown in Fig. 3.3(a), UV space is in Cartesian coordinate system so we can uniformly partition it into $8 * 8$ areas and build histogram based on that. However, CH space is described in polar coordinate system so we need to partition it differently. We uniformly divide the hue and chroma into 8 segments respectively, and thus give us 64 areas which is not uniform in terms of space. LCH partition for building histogram is shown in Fig. 3.3(b) For every input image, we cut it into 32×32 pixel block and build a histogram

for pixels in the block i which is denoted as h_i. The Chroma Histogram Flatness for the block i is denoted as f_i

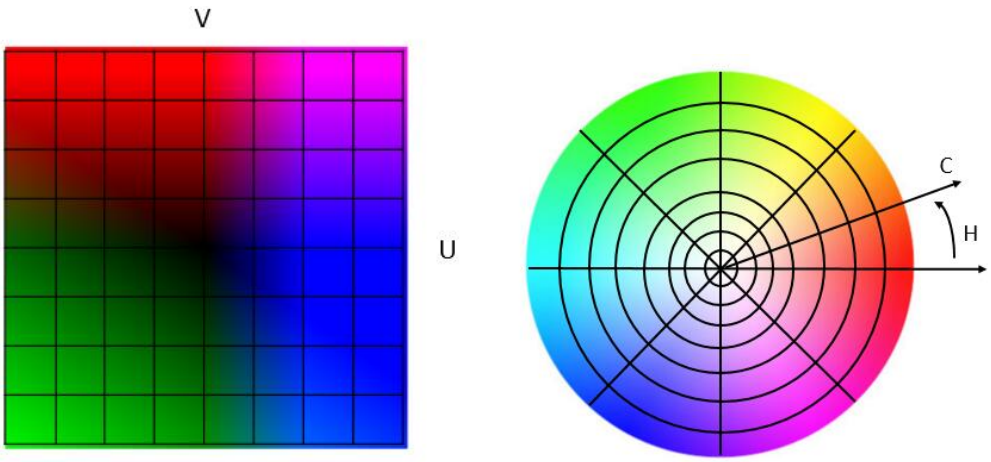
$$f_i = \frac{\sqrt[N]{\prod_{n=0}^{N-1} h_i(n)}}{\frac{\sum_{n=0}^{N-1} h_i(n)}{N}} \tag{3.1}$$

where $N = 60$ in our case. It is worth noticing that we do not use all the bins of histogram to calculate f_i. That is because we should only focus on the chroma flatness and exclude those areas which are close to gray. Highlight-text will also have very low flatness if we consider those gray pixels which correspond to black text and white background. For YUV space, we ignore the central four bins and for LCH space we ignore the central 8 bins that are inside the smallest circle.

the Chroma Histogram Flatness F for the entire image is define as maximum f_i in of all blocks in the image.

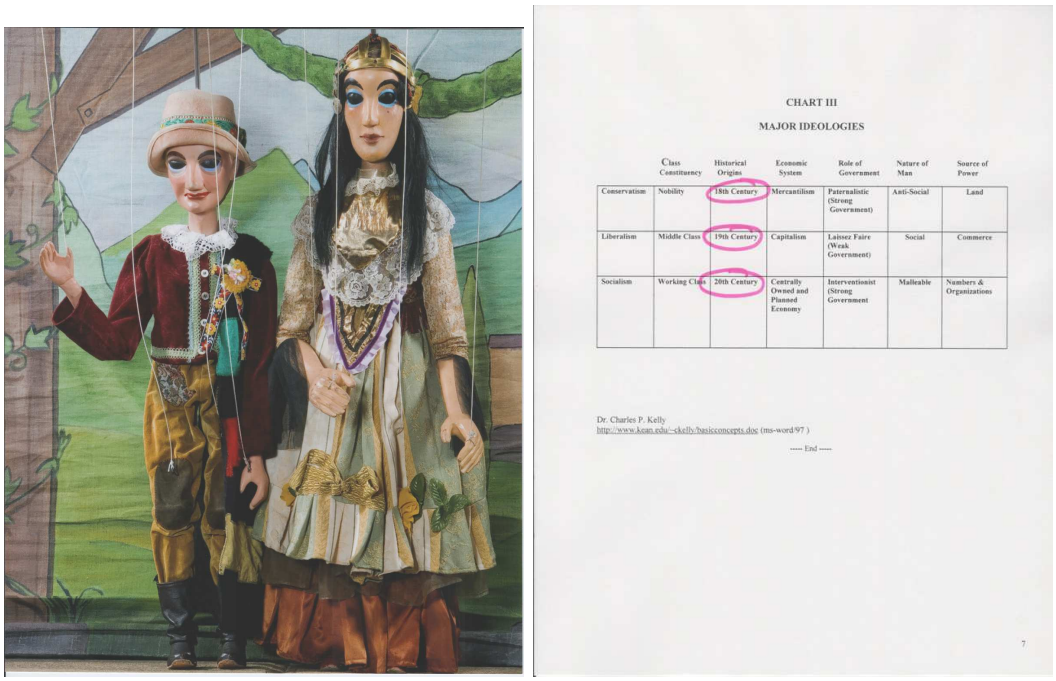
$$F = max(f_i) \tag{3.2}$$

Because we do not expect seeing flat histogram of any block in the highlighted text image.



(a) LUV space partition (b) LCH space partition

Fig. 3.3.: LUV and LCH color space histogram



(a) Natural image (b) Highlighted-text

Fig. 3.4.: natural image v.s. highlighted text

3.3.2 Chroma around text

On top of Chroma Histogram Flatness, we design another feature to detect chroma information of highlighted text image. Typically, we can expect that people put use highlighter to emphasize text information, which means text strokes are usually covered and surrounded by highlight colors as shown in Fig. 3.5. However for natural images, chroma information does not necessarily exists around edges. Given the fact, we can try to detect if there is chroma existence along the text edges in the image.

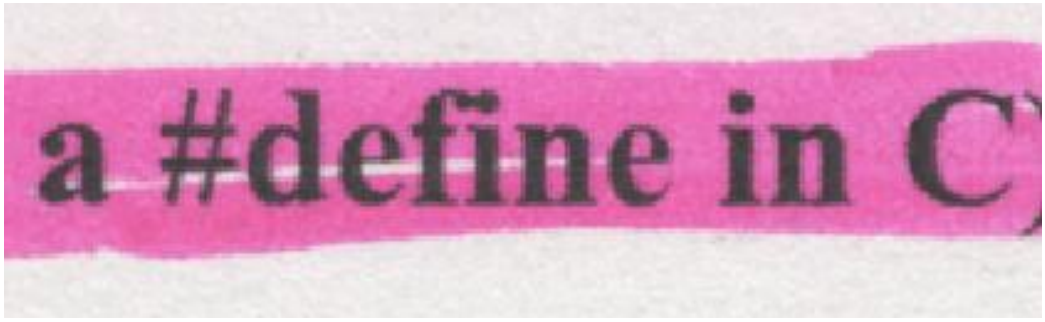


Fig. 3.5.: Highlighted colors around text

To find chroma around the text strokes, we first need to find text edges in the image block. We follow the method introduced in Sec. 2.4.6 to find text edges. However, we need to note that reverse contrast text should not be considered any more. That is because it is rare that people mark light text with darker highlight which will cause poor readability. After finding a text edge, we search along its luminance increase direction to find if there is any chroma existence. Then we calculate chroma strength (c) for two pixels along the luminance increase direction outside of text edge. In YUV space, chroma strength of a target pixel at position of (m, n) is defined as

$$c(m, n) = u(m, n) + v(m, n) \tag{3.3}$$

Note that we use simple summation of u and v here as approximation for faster computation. In case of LCH space, chroma strength is equivalent to $c(m, n)$. The process can be illustrated in Fig. 3.6 after finding a text edge.

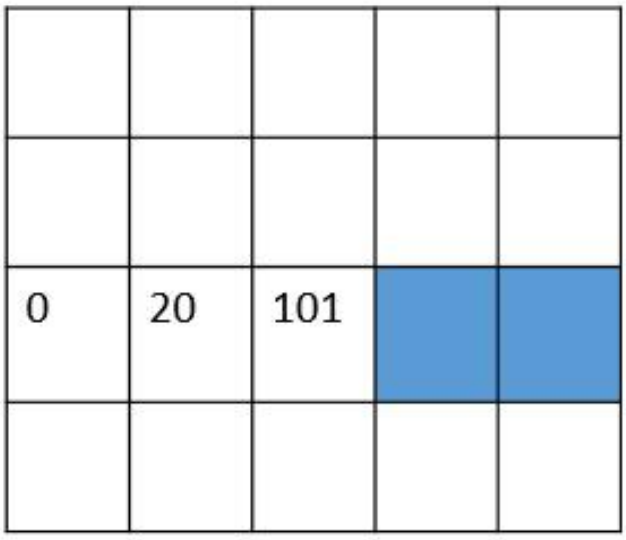


Fig. 3.6.: Calculate $c(m, n)$ of two pixels which are cover by blue

Similarly, we cut the input image into blocks with 32×32 pixels and then find $c(m, n)$ for all pixels outside text edges. Chroma Around Text (c_i) of a block i is defined as

$$c_i = \frac{mean(c(m, n))}{std(c(m, n))} \tag{3.4}$$

We also consider standard deviation of $c(m, n)$ because highlight should be consistent in a single block. Small $std(c(m, n))$ indicates that this block is more likely to contain highlight color. Similarly, chroma around text (C) of a image defined as maximum of c_i of all blocks.

$$C = max(c_i) \tag{3.5}$$

3.3.3 Color block ratio

Chroma Flatness and Chroma Around Text together can provide good discriminative power for detecting highlight image. However, they is not capable of handling text with color background. One such example is given in Fig. 3.7. According to our feature design in Sec. 3.3.1 and Sec.3.3.2, both features would strongly indicates that

Fig.3.7 should be classified as highlighted text. That is because both features only focus on local chroma and ignore global information.

Fig. 3.7.: Text with yellow background

To address this problem, we introduce Color Block Ratio. We calculate number of *color pixels* in every 32×32 pixel block. A pixel is defined as color if its chroma strength is greater than a threshold value $T_c = 10$. The chroma strength is defined in Eqa. 3.3 for YUV space. In LCH space is simply C channel. A block i is considered color if 10% of its pixels are color. We set $m_i = 1$ if the block is color otherwise $m_i = 0$. Color block ratio (R_c) of an image is defined as

$$R_c = \frac{\sum m_i}{\sum i} \tag{3.6}$$

3.3.4 White block ratio

Compared with text image, receipt typically only occupies small part of area on flat-bed scanner. An example is given in Fig. 3.1(a) which shows that scanned receipt only occupies upper-left corner of the plane. To utilize this difference, we design White Block Ratio. Again, we cut the input image into 32×32 pixel blocks. For each block i, we check if 95% of pixels have luminance value larger than 230. If this is true, we consider this block to be a white block $w_i = 1$, otherwise $w_i = 0$. White block ratio (R_w) of an image is defined as

$$R_w = \frac{\sum w_i}{\sum i} \tag{3.7}$$

3.4 Classification structure

As stated in Sec. 3.2, we do not need quick decision now thus structure in Fig.2.3 is not necessary. In order to make a more balance multi-class classification and for easier tuning, we apply Directed Acyclic Graph-Support Vector Machine (DAG-SVM) [42]. DAG-SVM is a tree-structured classifiers that capable of making multi-class classification. Instead of making one v.s. rest decision, it tentatively make one v.s. one decision which allows more judgement from lower nodes. DAG-SVM classifiers that we use are shown in Fig. 3.8

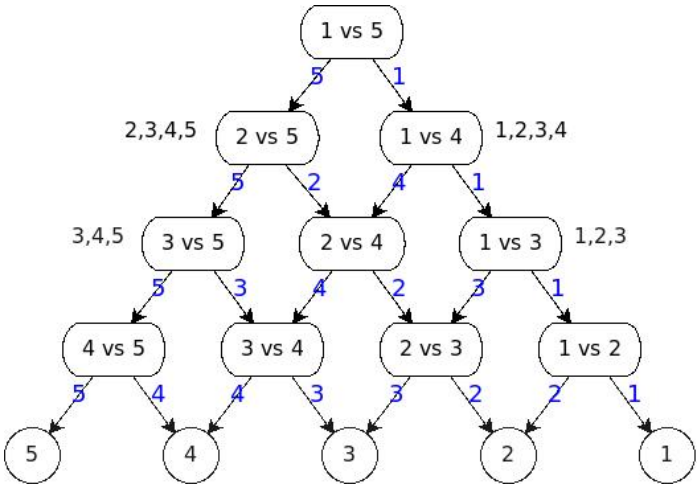


Fig. 3.8.: DAG-SVM

It first decide if the input is non-mix or non-highlight. If the image is classified as non-mix, then it is sent to the right node, other wise it is sent to the left node. As we can see, this tentative solution can make more careful decision compared with one v.s. rest structure as shown in Fig. 2.3.

3.5 Feature selection

In Sec. 3.3, we introduce four new features to detect receipt and highlight. Together with features we discussed in Sec. 2.4.3, we can represent each image k with a feature vector $f_k \in R^8$. However, we need to evaluate the contribution of each feature to the classification. To test the impact of each feature on the overall classification accuracy and time consumption, we adopt the leave-one-out feature selection.

Because misclassifications are not equally weighted, we need to first consider the metric for feature selection. We define the *weighted misclassification rate* (W_m) as below

$$W_m = \frac{\sum_{i,j} w(i,j)n(i,j)}{\sum_j n(i,j)} \tag{3.8}$$

where $w(i,j)$ is the weight of misclassification given in Table 3.2 and $n(i,j)$ is number of images in this corresponding entry. To avoid biased measure due to different size

of image set, the weighted sum needs to be normalized by the size of image set of each type. In our case, we have $\sum_j n(i,j) = 100$ for all the five types. But for completeness, we still keep the normalization term in Equation 3.8.

We first evaluate M_m^8 when we used all 8 features. Then we drop each feature d at a time and evaluate $M_m^{\bar{d}}$. Then the *feature impact factor* for d is defined as

$$I_d = \frac{M_m^{\bar{d}} - M_m^8}{M_m^8} \tag{3.9}$$

According to the Equation 3.9, the feature with larger I_d is more important to our classification. Also, we measure time consumption for each feature and take average over all the images. The results are shown in Table 3.1.

Table 3.1.: Feature impact factor and time consumption

feature	histogram flatness	color variability	text edge count	text color variance	chroma around text	chroma histogram flatness	white block ratio	color block ratio
time (ms)	12.01	12.51	14.97	73.92	36.12	11.45	0.67	0.81
I_d	24.61%	13.84%	11.02%	1.9%	10.2%	3.4%	48.43%	13.65%

According to Table 3.1, we find text color variability has the smallest impact on overall classification accuracy and it consumes the most time. Subsequently, we exclude this feature from our application and the final feature vector of an image k is a 7-dimensional vector $f_k \in R^7$. This f_k serves as the input to the DAG-SVM classifiers discussed in Sec. 3.4.

3.6 Experimental results

To test the performance of our designed system, we build an image set which includes 500 images scanned by flat-bed scanner. Each type of images have 100 images.

Except for text color variance in Sec. 2.4.7, we utilize all feature in Sec. 2.4.3 plus the features we discuss in Sec. 3.3. We exclude Stroke Color Variance because this feature is computationally expensive.

Similar to Sec. 2.6.2, different misclassification is weighted differently due to its impact to image quality. Some misclassifications lead to larger image quality degradation and they should be assigned larger weight. Some other misclassification cause small impact on image quality and should be assign lower weight. The weight table of misclassification is given in Table. 3.2.

Table 3.2.: Weights of misclassifications $w(i, j)$

	classification results				
ground truth	mix	text	picture	receipt	highlight
mix	0	3	5	6	4
text	3	0	10	6	2
picture	3	10	0	10	15
receipt	6	8	3	0	8
highlight	10	10	10	10	0

To test the performance of our proposed algorithm, we conduct leave-one-out cross validation. We use rbf kernel with σ for every node in Fig. 3.8. Then we do exhaustive search for σ and box constraint C to find the best combination that produce the best cross validation result. The goodness of cross-validation result is measured by weight misclassification rate (W_m) which is defined in Equation 3.8. When $W_m = W_m^*$ reaches minimum, the corresponding confusion matrix $n(i, j)$ in different color spaces are given in Table. 3.3 and Table. 3.4.

Table 3.3.: Confusion matrix $n(i, j)$ in YUV space

	classification results				
ground truth	mix	text	picture	receipt	highlight
mix	78	5	11	2	4
text	3	68	0	4	25
picture	5	0	94	1	0
receipt	0	3	3	88	6
highlight	4	8	1	5	82

Table 3.4.: Confusion matrix $n(i, j)$ in LCH space

	classification results				
ground truth	mix	text	picture	receipt	highlight
mix	76	6	12	1	5
text	4	72	0	9	15
picture	7	0	92	0	1
receipt	0	6	0	89	5
highlight	3	14	1	6	76

3.7 Conclusion

In this chapter, we introduce some novel features to handle multi-class classification for AIO printer with scanning functionality. It extends the scope of the topic discussed in Chapter. 2. Our proposed algorithm utilize the chroma information of input image for better classification. In this case, quick decision is not necessary so DAG-SVM can be applied.

4. DYNAMIC PRINT STREAM CLASSIFICATION AND OPTIMAL JPEG COMPRESSION

4.1 Problem statement

Image compression is the prerequisite for many applications. In some applications, different types of images may favor different compressions. For PC printing purpose, the system needs to choose optimal compression algorithm and parameters in order to obtain the best balance between image quality and compressed file size. For example, pure text with simple background image is suitable for lossless compression like Run Length Encoding, because it preserves the image quality while having small compressed file size. However, complex natural image may favor lossy compression like JPEG since it reaches good compression ratio at the price of image quality. In this case, we need to find an optimal compression level so that it reaches the best balance between the image quality and the compression ratio. In this chapter, we propose a system that finds an optimal compression algorithm given the input image. Also, if the input image is decided to be compressed by the lossy compression (JPEG), the system will find the optimal compression level.

4.2 Introduction

Image compression is becoming increasing crucial for various purpose in today's information age [43–47]. One of these applications is image compression for printing [48, 49]. People nowadays print a variety of images or documents from either mobile devices or PC. When printing images or documents, there are two major factors which impact user experience: image quality [50, 51] and printer process time. Ideally, we hope to reach image quality as high as possible while process time as quick as possible.

However, the two factors here can not be improved simultaneously. Improving image quality is typically at the expense of longer processing time because we need to deal with larger file size which is caused by less lossy compression [52]. Heavily lossy compressed images can be processed very fast by printer due to small file size, however it generates poor image quality outputs. So we propose a new approach which could jointly optimize both of these factors under the constrains given by current printer system.

The current printing system supports three compression modes: *Delta Row Compression* (DRC) [53], *Run Lenth Encoding(RLE)* [54] and *JPEG* [55]. In these three modes, DRC and RLE are lossless compression while JPEG is a lossy compression algorithm. Processing natural images using DRC or RLE is very inefficient. First, Human Vision System (HVS) is not sensitive to high frequency information in the image [56, 57], so it can not perceive some information loss in the lossy compression process. Secondly, using RLE and DRC will yield to very large file size for natural images, because typical natural images do not have much spacial redundancy [58, 59]. These two reasons drive us to use lossy compression algorithm such as JPEG to process natural images. On the other hand, simple structure image is more suitable for DRC or RLE lossless compressions. Simple structure image such as small logo or pure text have very high spatial redundancy which can be compressed heavily by RLE or DRC with no cost of image quality. Plus, JPEG compression will blur edges of text [60], which is undesirable in case of simple structure image.

Given these facts and constraints, it is clear that the proposed system need to first distinguish if the input belongs to natural image or simple structure image. This requires a classifier which extracts certain features from the input images, then makes a binary decision. If the decision is a natural image, we feed the image to the JPEG compressor. Otherwise, if it is decided to be a simple structured image, we send it to one of lossless, RLE or DRC, compressors. For natural images, we need to control the image quality by choosing optimal Q factor [61]in JPEG compressor which reaches

a good balance between output image quality and compressed file size. A larger Q factor yields to better image quality at the expense of larger file size.

Image quality is not our concern if the input is classified as simple structure image because both RLE and DRC are lossless compressions, which means they are equivalent in terms of image quality. However, they may differ in terms of decompression time at printer firmware. So we need to make choice between RLE and DRC based on only one metric: decompressed time at printer. Unfortunately, decompression algorithms are only available in printer firmware but not PC, so we need to develop an classification algorithm in PC or mobile devices which can predict the decompression time consumed by printer, if the input image needs to be losslessly compressed.

Content-based image classification [3, 25, 26], compression and image quality have been intensively studied separately. In this chapter, we combine these three factors together to optimize the printing performance. This system can also apply to other platforms that require balance between image quality and compression ratio.

We applied the Support Vector Machine [62] for classification purpose at the first stage which distinguishes lossy and lossless compressions. With more than 7000 training images, we managed to build a robust SVM classifier. JPEG compressed image quality is a long-studied topic [63–65]. Most of these studies focus on finding correlation between human perceived image quality and certain features [63]. In this chapter, we propose a Dynamic Print Stream Compression (DPSC) engine to finding the optimal compression level (Q factor) in JPEG compression, which could reach a good balance between image quality and compression ratio. This would largely enhance the PC printing performance. With DPSC engine, we could significantly improve the efficiency of printing. Proper compression for different types of input images could reduce the decompression time in the printer firmware while preserving good print quality. It makes sure that we get good print quality at the lowest price of decompression load.

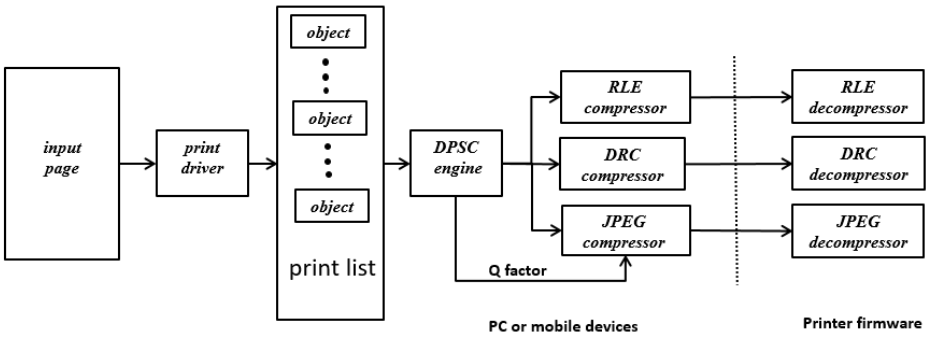


Fig. 4.1.: Printing system structure

4.3 Printing system description

As described in introduction, we have a printing system as shown in Fig. 4.1. Our input page is a document file such as PDF, Word or HTML Web, and it is first sent to print driver. The print driver generates a list of all the objects in the page and describes the page in high-level page description language [66]. Each object in the print list is then sent to our proposed DPSC engine to choose its optimal compression algorithm.

Our proposed DPSC engine and three compressors are implemented in PC or mobile devices. Computationally, PC or mobile devices are much more powerful compared with printer firmware. So we can safely ignore processing time at PC end, which is shown to the left of dash line in Fig. 4.1. Then we assume that overall processing time is fully determined by decompression time consumed by printer firmware.

4.4 DPSC engine

4.4.1 structure

Our proposed DPSC engine follows a hierarchical decision structure as shown in Fig. 4.2. Since we will only only use luminance channel information in the DPSC

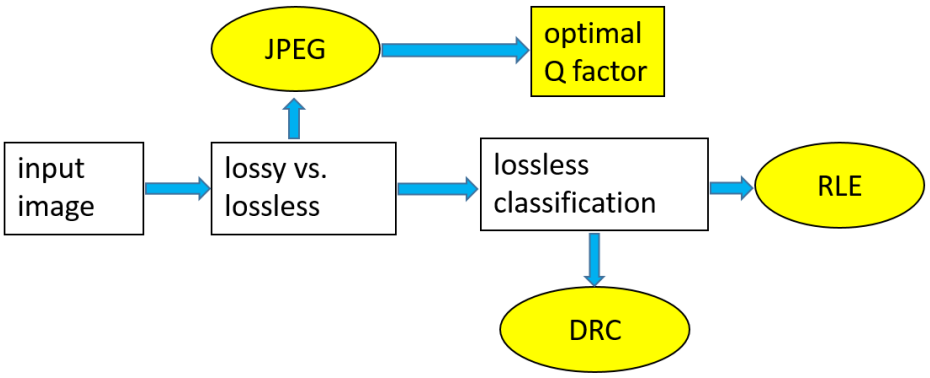


Fig. 4.2.: DPSC engine structure

engine, we first transform the original image into LUV color space and keep only luminance channel. Using only luminance channel information will significantly speed up subsequent image classification and image quality analysis. At the first classification stage, DPSC engine decides if the input image should be compressed by lossy or lossless compressors. If the input is a natural image, we will use JPEG to compress it and find its optimal Q factor. If it is a simple structure image such as pure text or small logo, we apply the second stage classification between RLE and DRC depending on the prediction of the decompression time consumed by printer firmware.

4.4.2 Lossy vs. lossless classification

As described in introduction 4.2, we first need to decide if the input image is a natural image or simple structure image, then apply compressor correspondingly. With more than 7000 training images, we develop a SVM classifier for the task. The input of this SVM classifier is a 3-dimensional feature vector. The three elements in the feature vectors are *Histogram Flatness, Histogram Span and Luminance Variablity Score*. We use these three features to train our first lossy vs. lossless SVM classifier.

Histogram flatness

If we build a histogram for a simple structure image such as pure text or logo, we can expect that this histogram is very peaky. There are roughly two peaks in this kind of histogram, one for text pixels and the other one for background pixels. However, if build the same histogram for a natural image, it should be more flat and widespread. A typical example of the histogram difference is given in Fig. 4.3. In order quantify this difference, we define the histogram flatness as its geometric average over arithmetic average

$$Flatness = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}} \tag{4.1}$$

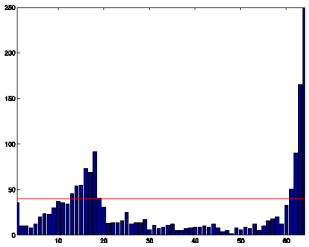
where $x(n)$ is the number in bin n.

Histogram span

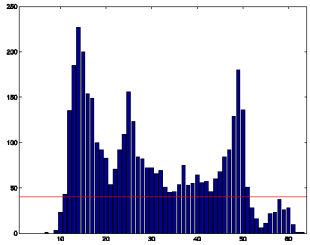
The first feature may not work very well when histogram is relatively sparse. For example, if we have a histogram which satisfies $X(2n) = k$ and $X(2n+1) = 0$, its corresponding image should be closer to natural but rather simple structure. However, the first feature would decide the image is very peaky. To solve this issue, we develop the second feature $HistogramSpan$. It is defined as width of the smallest interval that includes 75% pixels.

Luminance variability score

This feature is developed in [9]. This feature is based on the fact that the nontext region of a text image typically contains only a few gray level value. We cut the input image into 8×8 pixel blocks and calculate the mean value of the block. Then we build a 16-bin histogram for these block-mean values in the entire image. The Luminace Variability Score is defined as the number of non-zero bins in this block-mean histogram.



(b) Simple structure image histogram



(c) Natural image histogram

Fig. 4.3.: Natural and simple structure image histogram comparison

4.5 Lossless classification

Run Length Encoding and Delta Row Compression are two widely used lossless compression algorithm. As described in Sec. 4.2, we only need to consider the decompression time in the printer firmware.

Here we describe how we generate ground truth to train this classifier. For all the images which are labeled as simple structure in the training set, we compressed and decompressed it by both DRC and RLE. Then we measure its decompression time T^d_{DRC} and T^d_{RLE} respectively. If $T^d_{DRC} < T^d_{RLE}$, we label this image as DRC, otherwise we label it as RLE. This gives a two-class training set to train the classifier.

The most intuitive solution to finding faster decompression in printer firmware is doing both RLE and DRC decompression at PC or mobile end, then choose the faster one. However, due to our system constraint, we are not allowed to implement decompression algorithm on PC or mobile devices. So we need to resort to other features to do this prediction on PC or mobile end.

Although we can not apply decompression on PC or mobile, we can still compress the input image in both ways and find good predictors in the compression process. Two feature we find useful are *compression time ratio (CTR)* and *compression size ratio (CSR)*

For every input image, we compress it by both RLE and DRC. We measure compression time T^c_{RLE} and T^c_{DRC} respectively. Also, we measure the compressed file size F_{RLE} and F_{DRC}. Two features are defined as

$$CTR = \frac{T^c_{RLE}}{T^c_{DRC}} \tag{4.2}$$

$$CSR = \frac{F_{RLE}}{F_{DRC}} \tag{4.3}$$

With training set and features above, we can train a SVM classifier which is able to classify simple structure image into RLE or DRC.

4.6 Optimal JPEG compression

If the input image is classified as a natural image, we will use JPEG to compress it. In this case, we want to compress it as heavily as possible to reach smaller file size. However, if we compress it too hard, we will have very poor image quality output. In order to reach a balance between the compressed file size and output image quality, we need to find an optimal Q factor which is a variable in JPEG that controls the compression ratio and image quality.

However, Q factor does not change linearly with either compression ratio or image quality. [63] introduces three features that can quantify the JPEG image quality. All of these features are calculated horizontally and then vertically. The three features are *average differences across block boundaries* (B), *in-block absolute difference* (A) and *zero-crossing rate* (C).

If we have a image signal as $x(m, n)$ for $m \in [1, M]$ and $n \in [1, N]$, and calculate a difference signal along each horizontal lines:

$$d_h(m, n) = x(m, n + 1) - x(m, n), \quad x \in [1, N - 1] \tag{4.4}$$

Average differences across block boundaries shows blockiness effect caused by JPEG compression, and it is defined as

$$B_h = \frac{1}{M(\lfloor N/8 \rfloor - 1)} \sum_{i=1}^{M} \sum_{j=1}^{\lfloor N/8 \rfloor - 1} |d_h(i, 8j)| \tag{4.5}$$

The other two features are related to the activity of the image signal. The activity is measured by two factors. The first is in-block absolute difference which is defined as

$$A_h = \frac{1}{7}[\frac{8}{M(N-1)} \sum_{i=1}^{M} \sum_{j=1}^{N-1} |d_h(i,j) - B_h|] \tag{4.6}$$

The second activity measure is the zero-crossing rate. We first define

$$z_h(m,n) = \begin{cases} 1, & \text{horizontal zero-crossing at } d_h(m,n) \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

The horizontal zero-crossing rate then can be estimated as:

$$Z_h = \frac{1}{M(N-2)} \sum_{i=1}^{M} \sum_{j=1}^{N-2} Z_h(m,n) \tag{4.8}$$

Similarly, we can get vertical features B_v, A_v and Z_v. We average over horizontal and vertical features to get the overall features;

$$B = \frac{B_h + B_v}{2}, \ A = \frac{A_h + A_v}{2}, \ Z = \frac{Z_h + Z_v}{2}. \tag{4.9}$$

The final prediction of Mean Opinion Score (MOS) can be calculated using the above three features as

$$MOS = \alpha + \beta B^{\gamma_1} A^{\gamma_2} C^{\gamma_3}, \tag{4.10}$$

Figure.4.4 shows the result of MOS prediction trained by two groups of images.

We set a threshold Mean Opinion Score MOS_T depending on how good image quality we want to reach.

For any input natural image, we compress it starting from $Q_i = 10, 20, 30...., 100$. At each Q_i, we extract 3 features of compressed image as described in [63]. Then we

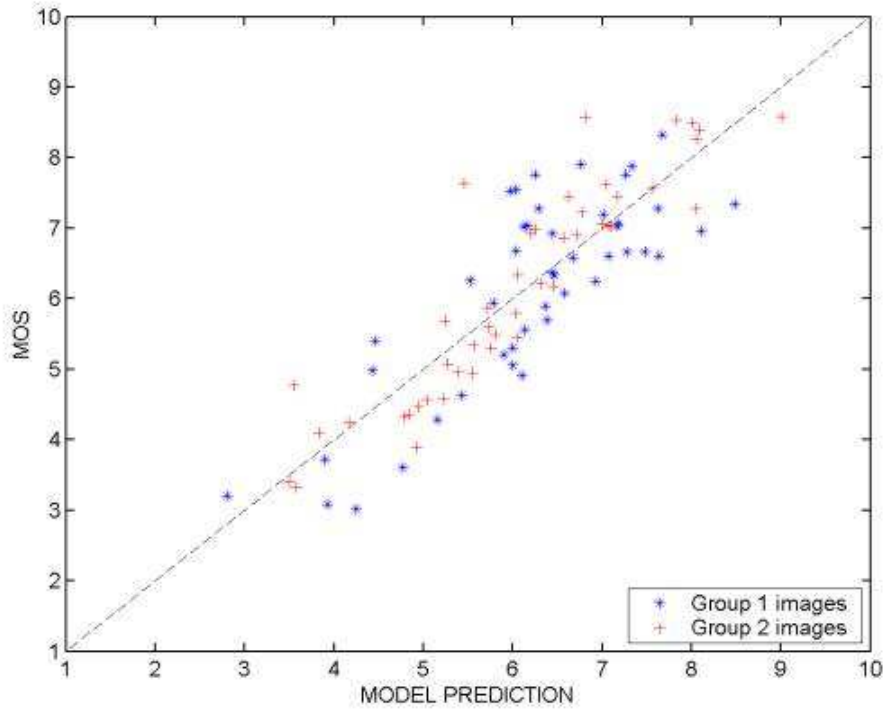


Fig. 4.4.: MOS prediction trained by two groups of images

calculate the predicted MOS_i at Q_i based on these 3 features. We find the first Q_i which has $MOS_i > MOS_T$, and set $i^* = i$. Thus, the optimal Q factor is defined as

$$Q^* = Q_{i^*} \tag{4.11}$$

Then we send the JPEG compressed image at the Q^* to printer.

4.7 Experimental results

4.7.1 lossy vs. lossless classification

To train the lossy vs. lossless classifier, we generate 8565 images from print drive engine. Among these 8565 images,4535 of them are labeled as natural images which ideally should be sent to lossy compressor (JPEG), while the other 4030 images are simple structure images which should be sent to lossless compressors (RLE or DRC).

We use standard F_1 metric [67] to evaluate the performance of this binary classification problem. 4-fold cross validation is conducted in our experiment. The SVM classifier utilize the RBF kernl. Subsequently, the best F_1 score, precision, recall are shown in Table. 4.1, and its corresponding confusion matrix is given in Table. 4.2. In order to visualize this classification, we show the feature distribution of the image set in Fig. 4.5.

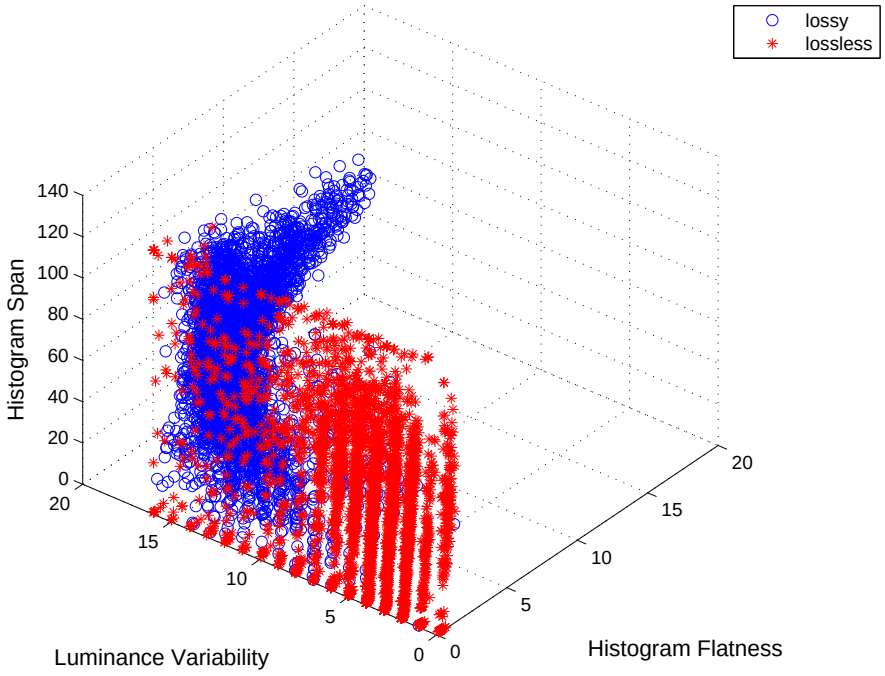


Fig. 4.5.: lossy and lossless classification in feature space

Table 4.1.: Best F_1 score in cross validation for lossy vs. lossless

metric	precision	recall	F_1^*
data	0.946	0.898	0.924

Table 4.2.: Confusion matrix at F_1^* for lossy vs. lossless

outcome / groundtruth	lossy	lossless
lossy	4291	244
lossless	488	3532

4.7.2 Lossless classification

To train the RLE vs. DRC classifier, we have a training set with 4027 simple structure images. They are either pure text documents or simple logo image patches. As described in Sec. 4.5, we labeled these simple structure images based on their decompression time. Each image is labeled as the algorithm which is faster to decompress. Similar to lossy vs. lossless classification evaluation , 4-fold cross validation is conducted. And we also use RBF kernel in the SVM classifier. The best F_1 score, precision, recall are shown in Table. 4.3, and its corresponding confusion matrix is given in Table. 4.4. In order to visualize this classification, we show the feature distribution and decision boundary of the training set in Fig. 4.6.

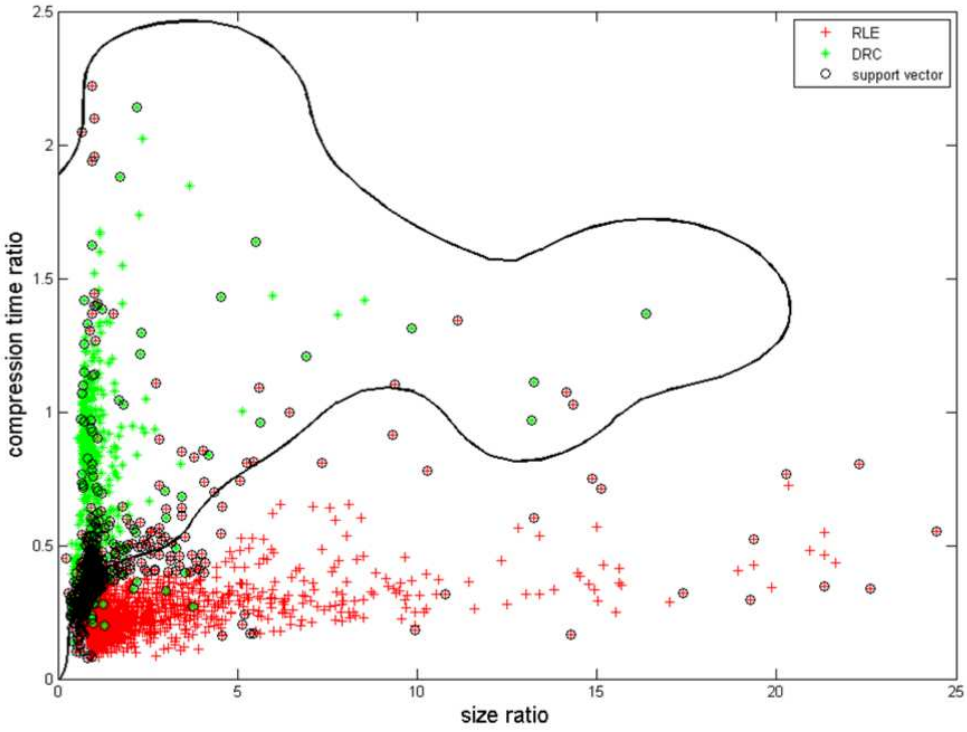


Fig. 4.6.: RLE and DRC classification in feature space and its decision boundary

Table 4.3.: Best F_1 score in cross validation for RLE vs. DRC

metric	precision	recall	F_1^*
data	0.899	0.967	0.932

Table 4.4.: Confusion matrix at F_1^* for RLE vs. DRC

outcome / groundtruth	lossy	lossless
lossy	3075	344
lossless	104	504

4.8 Conclusion

In this chapter, we propose a dynamic printing stream compression engine which is able to choose best strategy to compress the input image. The DPSC engine follows a hierarchical decision structure. It first decides if the input is a natural image or simple structure image. The natural image will be sent to JPEG compressor while the simple structure image will be compressed losslessly. The second classifier will decide if the simple structure image should be compressed by RLE or DRC based on prediction of decompression time in each way. If the image is decided to be compressed by JPEG, DPSC engine will choose the optimal Q factor to reach good balance between image quality and compressed file size.

5. HIERARCHICAL CONTENT-BASED IMAGE RETRIEVAL FOR HYBRID LEARNING

5.1 Problem statement

Image retrieval (IR) has been intensively studied recently. Primarily, there are two search methods based on different query input: image meta search and content-based image retrieval (CBIR). Image meta search takes keywords or text as input and then makes query in the database to search for the corresponding image. However, CBIR relies on a query image as input and then search for the similar or same image in the database. In this chapter, we introduce a IR engine which requires CBIR for educational purpose. Different from typical image retrieval engines, our engine should find the exactly same image in the database but not very similar one, which poses more difficulties when dealing with images with very minor difference. Compared with traditional Bag-of-Word(BoW) approach, we introduce a new search metric when ranking the images in the database so it can better distinguish the similar results to reduce confusion. Besides, we utilize spatial information of query image hierarchically to refine the results from BoW stage, which is ignored by BoW.

5.2 Introduction

Significant amount of research work has been done in the field of image retrieval [68–71] recently. Based on different query inputs to the image retrieval system, most of these research focus on two approaches: image meta search [72] and content-based image retrieval(CBIR) [4]. Image meta search requires descriptive keywords or text as input and then search for the images which most accurately display the semantic meaning of keywords or text. The difficulty of image meta search is how to do

indexing and build database efficiently, especially when database is huge and real-time is required [72], so that retrieval can be done fast and accurate. In contrast, CBIR takes an query image as input and search for images in the database based on visual similarity. CBIR servers as the prerequisite for many applications such as annotation extraction, reader evaluation and supplementary content embedding etc. We developed an blended-reality image retrieval engine which bridges hardcopy chapter to its digital version in the database. The retrieval engine utilizes both feature-space and spatial information in a hierarchical structure. The IR engine we propose can be applied to different platforms like PC or mobile devices.

The CBIR engine we introduce in this chapter is primarily used for educational purpose. It is integrated into a larger educational system called HP METIS. The METIS system improves teaching and learning experience for teachers and students. The METIS system provides a powerful tool for teachers to publish notes or material for students through blended-reality. It can project notes, URLs or supplementary content which help student understanding on the hardcopy textbook. It also includes other functionalities such as student reading behavior evaluation, automatic annotation generation, learning concept map generation and student peer review etc. All these functionalities of the system requires CBIR as a prerequisite. For example, if the teacher needs to know where is the most difficult part for students to understand, a very important indicator is how long the students stay on each page of text book. Longer stay on certain page indicates that it is more difficult for student to digest content on this page compared with others. However, student may prefer hardcopy text book rather than digital version since it is more readable and easier to make notes. Fortunately, we can easily capture the current page that the student is reading using a top-view camera or mobile device. Besides, our IR system store a digital copy of the textbook in the database. By matching the captured page with the digital version in the database, we can know which page the student is reading and evaluate how long the student stay on this page. This evaluation based on reading time will be sent to teacher so that teaching plan can be improved accordingly. Similarly, if

the teach makes any note on the textbook and need to project these notes on the hardcopy textbook of student, a retrieval process is also necessary to find the corresponding page. The flowchart how our IR engine is integrated into the METIS system is illustrated in Fig.5.1.

CBIR has been discussed in many literatures. Most of these techniques requires to represent the query image using a feature vector, then compare this feature vector with other feature vectors generated in the database to obtain similarity. Traditionally, people consider three types of features for image retrieval: color feature, texture feature or shape feature [73]. However, all of these approaches suffer some limitation. Color feature can not make distinguish between document image which carry different semantic meaning. texture feature typically requires transformation into a frequency domain for similarity comparison which ignores spatial information. Shape feature usually needs existence of certain object in the image which is not always guaranteed in application [74]. In [75], author propose a similarity ranking method based on nearest neighbor distance which outperforms SVM-based image retrieval method.

In this chapter, we propose a new Hierarchical Weighted Spatial Ranking which better utilize spatial information for CBIR on top of BoW approach. Our testing image set includes 5523 images and lots of them have high visual similarity, such as printed form with minor different hand-writing text.

5.3 Image Retrieval Engine Description

5.3.1 Overall Structure

Our image retrieval system follows a hierarchical structure which applies Bag-of-Word [76] first followed by spatial verification. BoW stage utilize SIFT [6] feature and apply K-means clustering [77] to generate centroid in feature space. If the BoW finds a highly significant matched image in our database, our IR engine will terminate the retrieval process and output this significant result. Otherwise, the BoW stage will output 5 most likely images in the database according to feature-space similarity

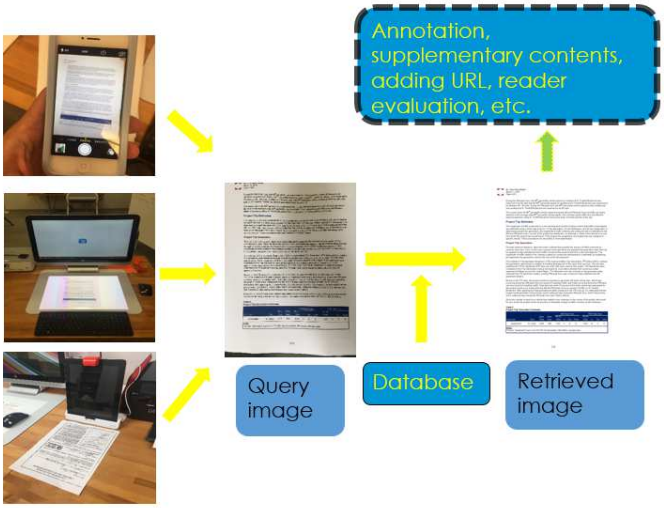


Fig. 5.1.: METIS system flowchart

score and feed them to subsequent spatial verification stage. We put BoW stage first because spatial verification requires point-wise matching between query image and stored images, and this process is computationally expensive. It will become computationally impossible if we apply spatial verification first. Instead, we apply BoW first to narrow down the possible matching candidates and then conduct point-wise spatial verification. The overall structure of our proposed IR engine is shown in Fig. 5.2

5.3.2 Bag-of-Word Training

Instead of matching each individual points in two images which is computationally intensive, BoW feature reduction can significantly improve the retrieval speed. Similar to traditional BoW approach, we first need to train a code book in the feature descriptor space which are N centroids in the feature descriptor space. There are some studies about how to pick optimal N for different applications. In our case, we choose $N = 400$ empirically. And we choose SIFT descriptor to build the BoW code book. The training process of BoW is shown in Fig. 5.3

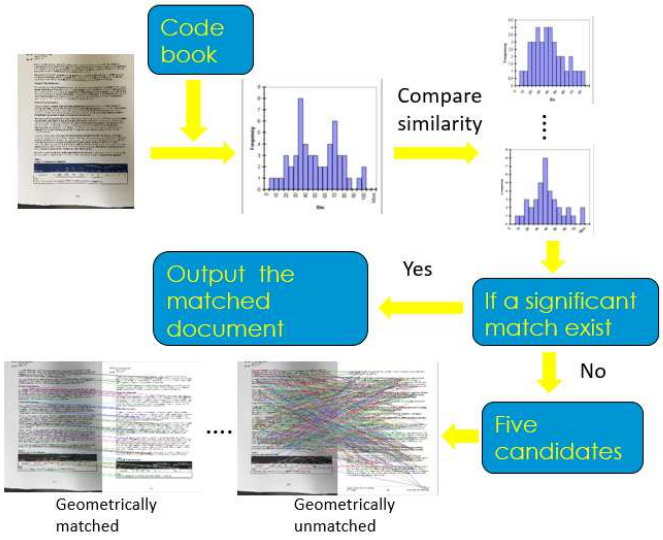


Fig. 5.2.: Proposed IR engine flowchart

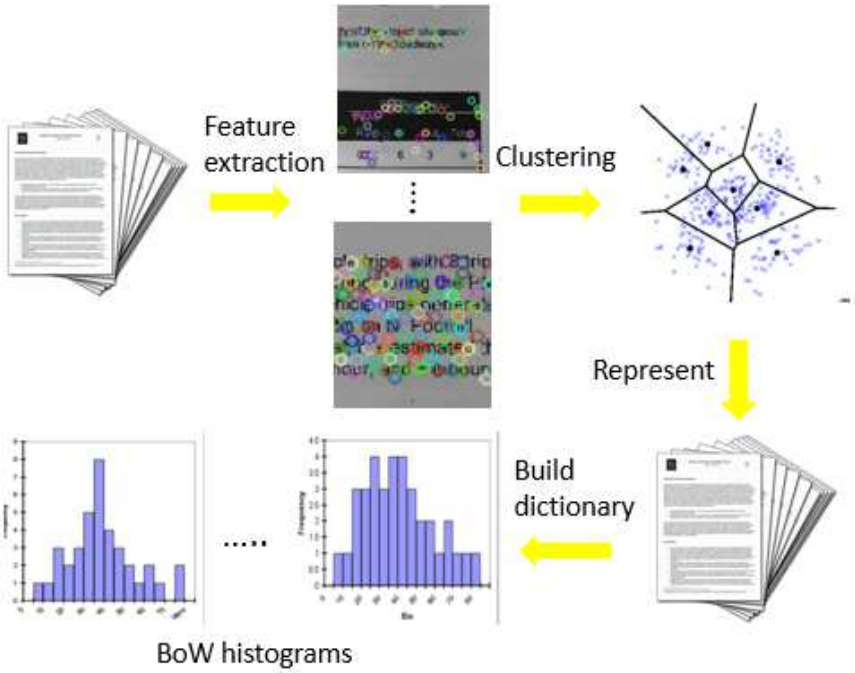


Fig. 5.3.: BoW training

In order to build the code book, we extract points of interest of all images in the database and then find the descriptors associated with all these points of interest. In our case, these descriptors are 128-dimensional vectors. Then we can apply K-means clustering for all the descriptors. However, given the scale of our image set, such clustering process turns out to be extremely difficult in terms of computational load. That is because complexity of this unsupervised learning process grows exponentially with the number of descriptors. Fortunately, we only need these code book centroids to describe distribution of descriptors in each image, so we do not necessarily requires the exact clustering results. Instead of clustering all descriptor collectively, we can apply sequential K-means which has the same descriptive power. Compared with standard K-means clustering, it only requires small number of descriptors to initialize the clustering process and then feed the rest descriptors sequentially. This reduces complexity of building code book centroids from exponential to linear. With the code book centroids, we can build a feature vector for each image in the image set. In each image in the image set, we assign every point of interest to its closest code book centroid. Then we can build a histogram for each image, where every bin in this histogram represents number of points assigned to this centroid. So in our case, image i in the database is represented by a 400-bin histogram which is equivalent to a 400-dimensional vector V_i.

5.3.3 BoW Retrieval

In the retrieval process, some preprocessing is necessary under certain capture situations. Two situations we consider there are: perspective distortion and noisy background. One example which suffers both situations is given in Fig. 5.4. With complex background like in Fig.5.4, IR engine will get many noisy points of interest and descriptors that break the BoW description. In such case, we need to calibrate our capture device and make perspective correction accordingly. In our case, we create a bounding box on our captured plane and make sure the image is in the bounding box.

We can easily get homography matrix by calibrating this bounding box in capture image. Therefore it solves both problems.

Fig. 5.4.: Capture image with perspective distortion and noisy background

After preprocessing, we follow the similar training process in Sec. 5.3.2. We extract all points of interest and associated descriptors in the captured image. Then assign each descriptor to the closet centroid we obtain in Sec. 5.3.2. Again, we build a 400-bin histogram where each bin represents how many descriptors are assigned to each centroid. By doing this, we can again represent the capture image using a 400-dimensional vector V_c.

Traditionally, BoW approach will just find a V_i which has the largest cosine similarity with V_c and declare i^* as the final result like in Eqa. 5.1.

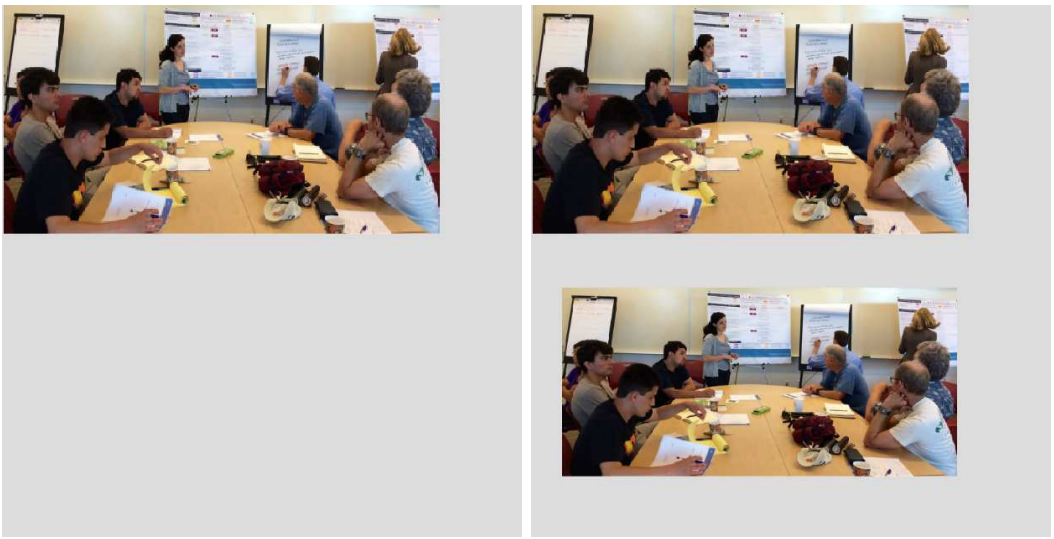
$$i^* = \arg\max\,(S_i = \frac{V_c V_i}{|V_c||V_i|}) \tag{5.1}$$

However, this solution has two potential problems. First, as discussed earlier it ignores the spatial information of the image. For example, two images may have same content but are organized spatially different may have very high similarity score but they are not a good match. Secondly, cosine similarity only evaluates the angle of two vectors and ignores their magnitudes. For example, Fig. 5.5(a) and Fig. 5.5(b) ideally should have feature vector of the almost identical direction, but feature vector of fig.5.5(b) should roughly double the magnitude of fig. 5.5(a). When doing retrieval, BoW can not tell the difference of these two images because their feature vectors are pointing to the identical direction however they are visually very different.

But still for some images in database, we can expect that BoW itself has enough discriminative power for retrieval. In real case, one S_{i^*} can be significantly higher than other result. This fact suggests that if we can obtain a significant result using only BoW, it does not worth more effort to make any refinement of the result. So if have an $S_{i^*} - S_{i^{**}} > T_1$ where $S_{i^{**}}$ is the second largest similarity score in the database, our IR engine will terminate the retrieval process and output i^*. For most other query images, we will get several top results which have close similarity scores. And we need more processing to refine the results. In our case, we send the top 10 candidates to the subsequent stage.

5.3.4 Hierarchical Weighted Spatial Ranking

To address above problems, we propose the *Hierarchical Weight Similarity Ranking*(HWSR). Instead of searching for the feature vector V_i in the database which has largest cosine similarity, we also consider the spatial consistency which is weighted by the number of matched points used for generating homography.



(a) image 1 (b) image 2

Fig. 5.5.: Example of difficult case for naive BoW

For each candidate image, we conduct point-wise matching with the query image in the feature descriptor space and find a group of points that have smallest distances. These points are considered possible matches in feature descriptor space. However, we need to verify them spatially. With the group of matched points and their coordinates, we can build a homography which represents the possible perspective relation between these two images. We can expect that a real matched candidate in the database should have very good homography. In contrast, false candidate may have good similarity in BoW, but will show a poor homography with query image. For each candidate j, we have a homography $(H_j)_{3\times3}$ associated with query image. In order to evaluate the goodness of H_j we need to apply SVD to the $(H_j)_{3\times3}$. Then we compute the eigen value ratio R. Smaller R indicates that we get a good homography. However, small R itself can not guarantee that we find a good match. Some candidates may have very significant homography because it relies on only small number of matched points. However, some good candidates may have relatively larger R because small

number of points are not matched correctly. So instead of using R alone, we introduce Weighted Spatial Ranking Score (WR)

$$WR_j = \frac{R_j}{N_j} \tag{5.2}$$

where N_j is number of points we used for generating $(H_j)_{3\times3}$. Now, we can re-rank the similarity of all candidates using $WR_j(j = 1, 2...10)$ and note top result as j^*. Similarly, if we can find a significant WR_j^* so that $WR_{j^*} - WR_{j^{**}} > T_2$ where j^{**} is the second best result, we terminate the process and output j^*

At this point, it is still possible that we get more than one good results which all have good homography. A good homography indicates that there exists a solid perspective transformation. However, if we find more than one good perspective transformation, we can compare if any candidate has even stronger spatial relationship: affine transformation. We can obtain the strongness of affinity A_j from $(H_j)_{3\times3}$

$$A_j = |H_j(3,1)| + |H_j(3,2)| \tag{5.3}$$

The we can find the answer by affine strongness

$$j^* = \arg\min(A_j) \tag{5.4}$$

By imposing this more strict verification, we can refine and improve the results if there are multiple close candidates.

5.4 Experimental Results

5.4.1 Dataset description

We collect 5523 images to test performance of our system. It covers various types of images which includes document image, natural photo, hand-written forms etc. Some images in the data set are very visually similar and challenging to retrieve. Especially some form with hand-written entries, only minor visual difference exists in small part of images.

5.4.2 Accuracy

To test the performance of our system, we randomly select 336 images in our data set and print them out using HP all-in-one printer. These images are then captured by PC(HP Sprout) top-view camera as query images. The captured images may suffer from perspective distortion, noisy background and various illuminance condition. The testing results are shown in Table. 5.1.

As we mentioned in Sec. 5.2, the retrieval needs to be real-time to guarantee the user experience. As the image set (search pool) expands, computational load can grow very significantly and thus violate the real-time requirement. To verify the computational performance, we conduct retrieval on the same PC which is used for image capture and time each query. Since we apply two-stage hierarchical structure, some images can stand out right after the first stage while others require HWSR. Subsequently, we time these two cases separately and average results are shown in Table. 5.2.

Table 5.1.: Retrieval accuracy

success	fail	total
294	42	336
87.5%	12.5%	100%

Table 5.2.: Time of retrieval

with HWSR	without HWSR
1.50s	2.01s

5.5 Conclusions

In this chapter, we propose a image retrieval engine to improve hybrid learning experience for students and teachers. It utilizes the BoW approach to find the candidate pool first. Then the candidate images will be re-ranked by our proposed Hierarchical Weighted Spatial Ranking to find the best match in the data set.

6. TAG RECOMMENDATION VIA ROBUST PROBABILISTIC DISCRIMINATIVE MATRIX FACTORIZATION

6.1 Problem statement

Low-rank matrix factorization serves as a key technique in learning latent factor models for many applications in machine learning. However, in many applications, observed data often exhibits different levels of noise. To address this issue, we propose a Robust Probabilistic Discriminative Matrix Factorization (RPDMF) method for binary matrix factorization on noise polluted data. We illustrate the benefits of our approach in real examples, and show how our method significantly outperforms Probabilistic Discriminative Matrix Factorization (PDMF) and classical method Weighted Nonnegative Matrix Factorization (WNMF) in the application of image tag completion.

6.2 Introduction

Low-rank matrix factorization serves as a key technique in learning latent factor models for many applications. Most matrix factorization methods seek to represent the original matrix as the product of two low-rank matrices. Typical applications of matrix factorization include image annotation [78], image tag completion [79–81], collaborative prediction [82] and clustering [83]. For any specific real application [84,85], the corresponding optimality criterion is defined so that the difference between the original matrix and its factorized form is expected to be minimized. Matrix factorization serves as a very effective method to address problems of missing data recovery and prediction. Because matrix factorization can recover missing data without help

of extra features, it can be applied to different fields without designing new features which may require domain-specific knowledge.

Among all collaborative filtering problems [86–88], image tag completion is a very typical application in matrix factorization. Different from movie ratings and recommendations [89, 90], the matrix representation of image tag contains only binary elements. A positive sample in the matrix means that a certain tag data is associated with a given image, while a negative sample means that the information described by this tag has not been assigned to the given image.

Many methods [91–93] have been proposed for matrix factorization. However, these methods assume that the observed matrix is noise-free. Besides, the data in typical matrix factorization with missing elements can take on any arbitrary real value. The factorization is performed so that an objective function is minimized. The objective function is usually composed of the observed data in the original matrix and a regularizer that controls the model complexity. Some factorization methods are performed under certain restrictions. For example, Nonnegative Matrix Factorization (NMF) [91, 94], as suggested by its name, requires that all elements of the original matrix be nonnegative.

In this chapter, motivated by the image tag completion task, we discuss a specific setting of matrix factorization with entries restricted to binaries, representing positive and negative samples respectively. Also in this setting, some observed elements are polluted by noise. In order to perform the data recovery, we mask some of the elements in the original matrix as unknown. An example of observed matrix is

$$\begin{pmatrix} ? & n_s & p_s & p_s \\ p_s & ? & p_s & n_s \\ n_s & p_s & ? & n_s \end{pmatrix}$$

,

where p_s represents a positive sample, n_s is a negative sample, and the '?' represents our masked missing data. We should notice that some of the elements with value p_s or n_s are mislabelled (flipped) in the observed matrix due to the noise. As

we discussed previously, image tag completion falls exactly into this category. Given the original matrix representation X of an image set $X_{ij} = p_s$ means that the ith image has the tag indexed as j, while $X_{ij} = n_s$ means that the ith image does not include the information of tag j. We may lose some tags because the user input is not trusted, or because the information conveyed by a certain tag is difficult to distinguish in a given image. Those tags are represented by '?' in the matrix. And the noise (mislabelling) may be introduced during transmission or by human error.

We propose Robust Probabilistic Discriminative Matrix Factorization (RPDMF) in order to recover those missing elements which are labeled as '?' in the noisy binary matrix, i.e., to predict whether any missing element is either p_s or n_s. Another method proposed specifically to deal with missing data in a matrix is Weighted Nonnegative Matrix Factorization (WNMF) [95], which excludes missing data in the cost function by introducing a masking matrix as part of the optimization.

6.3 Robust probability discriminative matrix factorization

Given a matrix $X \in S^{m \times n}$, $S = \{p_s, n_s\}$ with missing values, let G denote the set of observed elements in the matrix X. All these observed entries are either 1 or -1, i.e., $p_s = 1$, $n_s = -1$. All the missing elements are denoted as 0. Given the observed set G, our goal is to predict weather $X_{ij} = 1$ or -1 for all $X_{ij} \notin G$.

We need to find a low-rank matrix $\hat{X} = \hat{W} \times \hat{H}^T$ to approximate the target matrix X. We obtain $\hat{X}$ by minimizing a linear combination of the norms of W and H, which are two regularizers intended, respectively, to avoid overfitting, and to control its logistic loss:

$$\begin{aligned} \min_{W \in R^{m \times p},\, H \in R^{n \times p}} C \sum_{(i,j) \in G} \log(1 + e^{-X_{i,j} \langle W_{i.}, H_{j.} \rangle}) \\ + \alpha \|W\|_F^2 + \beta \|H\|_F^2. \end{aligned} \tag{6.1}$$

Here α and β are parameters controlling the strength of regularizers for W and H. In our case, $\alpha = \beta$ since they are of equal importance. Since the logistic loss

has probabilistic interpretation [96], we call this Probabilistic Discriminative Matrix Factorization (PDMF).

In real-world applications, the training data X may be polluted by noise. To handle the noise, we propose a robust version of PDMF that is called Robust Probabilistic Discriminative Matrix Factorization (RPDMF). For any $X_{i,j}$, we introduce a random variable $I_{i,j}$ where $I_{i,j} = 1$, if $X_{i,j}$ is not polluted; and $I_{i,j} = 0$, otherwise.

$$\begin{aligned} \min_{W \in R^{m \times p},\, H \in R^{n \times p}} & C \sum_{(i,j) \in G} I_{i,j} \log(1 + e^{-X_{i,j} \langle W_{i.}, H_{j.} \rangle}) \\ & + \alpha \|W\|_F^2 + \beta \|H\|_F^2 - q \sum_{(i,j) \in G} I_{i,j}. \end{aligned} \tag{6.2}$$

Here, q is parameter that controls the noise level; and C is the box constraint.

6.3.1 Optimization

We now discuss the optimization shown in (6.2) with respect to $W \in R^{m \times p}$, $H \in R^{p \times n}$ and $\{I_{i,j} | (i,j) \in G\}$. Joint optimization [97] with respect to W, H, and $\{I_{i,j} | (i,j) \in G\}$ is very difficult due to nonconvexity of the cost function. However, if we optimize only one of W, H, or $\{I_{i,j} | (i,j) \in G\}$ at a time, the problem becomes easy to solve. So we propose an alternate optimization method by repeating following three steps until convergence is reached.

Note that PDMF is a special case RPDMF, when all $I_{i,j}$ are set to 1 for $(i,j) \in G$.

Step 1: Fix H and $\{I_{i,j} | (i,j) \in G\}$, optimize W. Optimize (6.2) with respect to W.

$$\begin{aligned} W^* = & \arg \min_{W \in R^{m \times p},\, H \in R^{n \times p}} C \sum_{(i,j) \in G} I_{i,j} \log(1 + e^{-X_{i,j} \langle W_{i.}, H_{j.} \rangle}) \\ & + \alpha \|W\|_F^2 + \beta \|H\|_F^2 - q \sum_{(i,j) \in G} I_{i,j} \\ = & \arg \min_{W \in R^{m \times p},\, H \in R^{n \times p}} \sum_{(i,j) \in G} I_{i,j} \log(1 + e^{-X_{i,j} \langle W_{i.}, H_{j.} \rangle}) \\ & + \alpha \|W\|_F^2. \end{aligned} \tag{6.3}$$

We can further decompose the problem in (6.3) into a set of independent subproblems, where each subproblem is optimization over a row of W. If we assume that we need to optimize with respect to i-th row of W denoted as $W_{i.}$

$$W_{i.}^{*} = \arg\min_{W_{i.} \in \mathcal{R}^p} \frac{1}{2}\|W_{i.}\|_F^2 + C \sum_{(i,j)\in G} I_{i,j} \log(1 + e^{-X_{i,j}\langle W_{i.}, H_{j.}\rangle}). \tag{6.4}$$

Then the problem above become a standard convex optimization problem which can be easily solved by any gradient method.

Step2: Fix W and $\{I_{i,j}|(i,j) \in G\}$, optimize H. Symmetrically, we optimize respect to each row of H in the same manner.

Step 3: Fix W and H, optimize $\{I_{i,j}|(i,j) \in G\}$.

This problem can be decomposed into a set of independent problems, each of which responds to $I_{i,j}$:

$$\begin{aligned} I_{i,j}^{*} &= \arg\min_{I_{i,j}} C I_{i,j} \log(1 + e^{-X_{i,j}\langle W_{i.}, H_{j.}\rangle}) \\ &\quad + \alpha\|W\|_F^2 + \beta\|H\|_F^2 - qI_{i,j} \\ &= \arg\min_{I_{i,j}} C I_{i,j} \log(1 + e^{-X_{i,j}\langle W_{i.}, H_{j.}\rangle}) - qI_{i,j}. \end{aligned} \tag{6.5}$$

Then the optimization can be easily done by setting $I_{i,j}$ to 1 if $\log(1+e^{-X_{i,j}\langle W_{i.}, H_{j.}\rangle}) < q/C$; and setting $I_{i,j}$ to 0 otherwise.

We summarize these three steps in the Algorithm 1. Overall, in every iteration, we first fix H, I and update all rows of W. Then we fix W, I and update all rows of H. Finally, we fix W, H and update I. In each step, a convex optimization problem is solved by a gradient-descent method. Since different rows of W or H can be updated independently given fixed H or W, respectively, the optimization methods can be easily run in parallel to speed up computation.

Our proposed algorithm is guaranteed to converge, since the objective function is lower bounded by zero, and each of the updating steps can only decrease the objective function, or leave it unchanged.

Algorithm 1 One-Class Maximum Margin Matrix Factorization

Require: $X \in \{1, -1\}^{m \times n}$ with G, the set of observed entries; p, the dimension of the latent space.

Initialize $W \in R^{m \times p}$, $H \in R^{n \times p}$
for t = 1,...,max_iter **do**
  for i = 1,..., m **do**
    Update $W_{i.}$.
  end for
  for i = 1,..., n **do**
    Update $H_{i.}$.
  end for
  for i, j $\in$ G **do**
    Update $I_{i,j}$.
  end for
end for
return W, H

6.4 Experimental results

In order to evaluate our proposed RPDMF method, we first conduct an experiment on a **synthetic dataset**. Then we apply PDMF to the task of image tag completion. The two public datasets used for performing image tag completion are **NUS-WIDE TAGGED** [98] and **MIRFLICKR-25K** [99].

To create the **synthetic dataset**, we first generate two base matrices $W' = (w'_{ij})_{m \times p}$ and $H' = (h'_{ij})_{n \times p}$, where the elements of W' and H' are uniformly distributed $w'_{ij} \sim U[0, 1]$, $h'_{ij} \sim U[0, 1]$. Then we threshold the matrix $X' = W' \times (H')^T$ to obtain the binary matrix X so that approximately 50% of elements in X are positive samples while the rest are negative samples.

The **NUS-WIDE TAGGED** dataset includes $269,648$ images and 81 associated tags (*e.g* airport, animal, beach, bear, *etc.*). So the original X' is a $269,648 \times 81$ matrix in which each row represents a tagged image, while each column represents a possible tag. If the ith image has a specific tag j, then the $X'(i,j)$ should be a positive sample. However, many images in this dataset have a small number of tags. Thus they provide little information about statistical correlations among different tags. So we apply preprocessing to this dataset to exclude those images that have fewer than 10 tags. Because of this preprocessing, the matrix $X = (x_{ij})_{m\times n}$ to be factorized has much fewer rows than the original matrix X'.

Similarly, **MIRFLICKR-25K** contains $25,000$ tagged images with 38 different tags. We generate a matrix representation to this dataset as described above. Again, we apply preprocessing to this datasets to exclude the images which provide insufficient tag information. On **MIRFLICKR-25K**, we only use images that have more than 12 tags. The dimensionality of matrix representation (after preprocessing) for these three datasets is give in Table 6.1.

Table 6.1.: Dimensionality of the three datasets

Dataset	m	n	p
Synthetic	100	100	40
NUS_WIDE TAGGED	89	81	40
MIRFLICKR-25K	104	38	20

We compare RPDMF with the competing methods PDMF and weighted nonnegative matrix factorization (WNMF).

Now we discuss the parameters and performance measures for these three methods. On all three datasets, we label the positive samples in X as 1. However, due to the different nature of WNMF, PDMF, and RPDMF, we need to label negative samples in X differently depending on which method is applied. For WNMF, we label negative

samples as 0, since X_{ij} should be nonnegative value. For PDMF and RPDMF, we should label negative samples as -1. After that, we randomly mask 20% of elements in X, and use them as the testing set while the rest are used as the training set. For PDMF and RPDMF, the masked elements are labeled as 0. These are expected to be recovered. For WNMF, we can achieve masking by specifying the weight matrix M in the objective cost function:

$$O_{wnmf} = ||M \odot (X - WH)||_2^F, \tag{6.6}$$

where $m_{ij} = 0$ if this element is masked for testing, while $m_{ij} = 1$ if it is in the training set. In order to test the robustness of these three methods, we randomly flip a portion of σ elements in the training set. This process introduces noise into our training set as discussed in Sec. 6.3. A larger value of σ means that the training data is more heavily polluted.

For faster convergence of RPDMF, we initialize $W^0 = (w_{ij}^0)_{m\times p}$, $H^0 = (h_{ij}^0)_{n\times p}$ as $w_{ij}^0 \sim N(0,1)$, $h_{ij}^0 \sim N(0,1)$, and $I_{ij} = 1$. On all the datasets, our experiments suggest that 30 iterations are sufficient for RPDMF to reach convergence. In every iteration k, we sequentially optimize all rows of W^k followed by all rows of H^k, and finally I. To update each row $W_{i.}^k$, all the rows of H^k which correspond to non-zero entries in X are viewed as samples. And we update each row $H_{j.}^k$ symmetrically. We use *BFGS* Quasi-Newton approach [100] in every iteration to update from W^k, H^k to W^{k+1}, H^{k+1}. Our experiment show that BFGS has the fastest convergence compared to other gradient methods such as *DFP* or *Conjugate Gradient* [100]. After 30 iterations, we calculate $\hat{X}' = W^{30} \times (H^{30})^T$. For the WNMF method, we simply update W^k and H^k in very iteration as in [95]. Our experiments suggest that 40 iterations are sufficient for WNMF to reach convergence. So $\hat{X}' = W^{40} \times (H^{40})^T$ for WNMF. For all the three methods, $\hat{X}'$ is binarized at threshold value T to obtain the recovered binary matrix $\hat{X}$. Finally, we evaluate the performance of factorization in the testing set, which are those masked elements that we choose at the beginning.

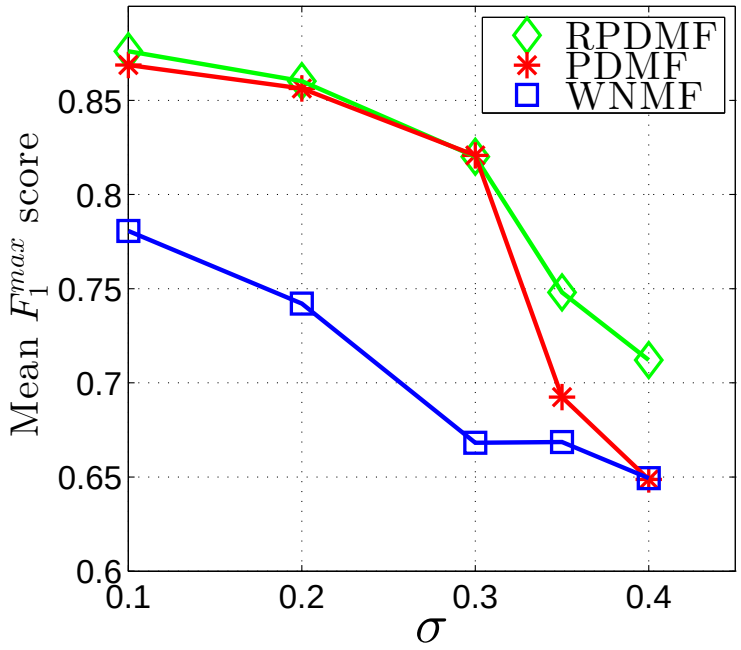


Fig. 6.1.: Synthetic

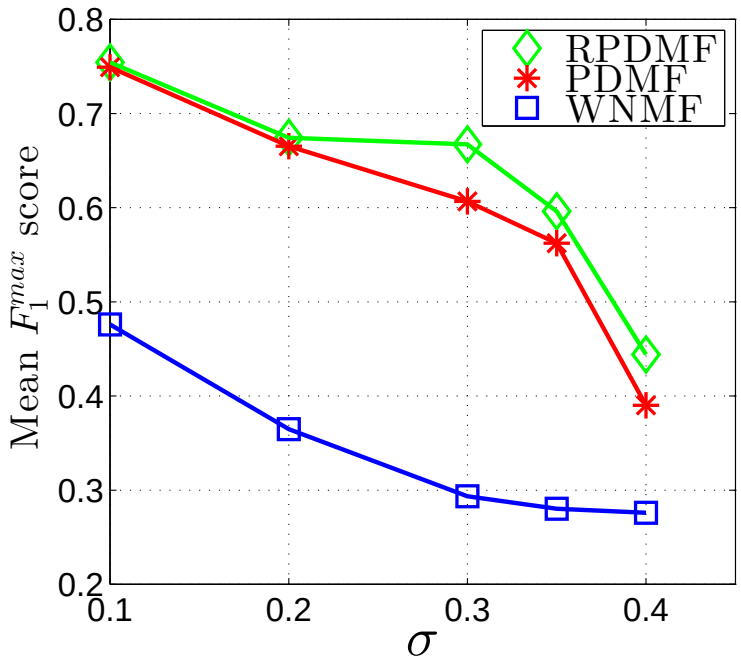


Fig. 6.2.: NUS-WIDE TAGGED

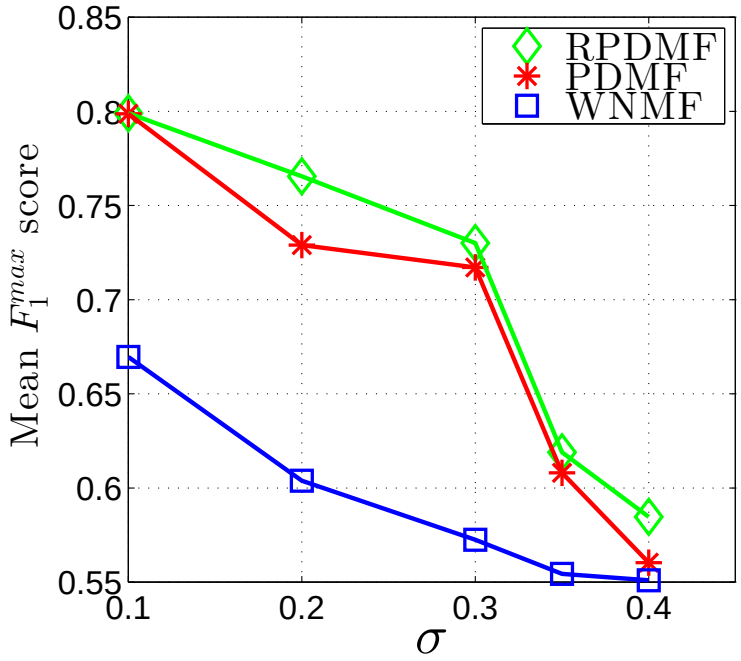


Fig. 6.3.: MIRFLICKR-25K

For both PDMF and RPDMF, different values for box constraint C and threshold T will generate different estimates $\hat{X}$. We perform an exhaustive search on C and T to find the best combination (C^*, T^*) that can maximize the F_1 score in the testing set. For WNMF, we only need to search for the optimal T^* that leads to the maximum F_1 score in the testing set. Because of the randomness in initialization and optimization, we repeat the whole factorization process five times for each method, at every level of σ, to obtain five maximum F_1 scores. For every σ, we calculate its corresponding mean maximum F_1 score mF_1^{max} for all three methods. The testing results on three datasets at different noise levels $\sigma = 0.1, 0.2, 0.3, 0.35, 0.4$ are given in Table 6.2. As a visualization of Table. 6.2, we also present our testing results in Fig. 6.1, Fig. 6.2 and Fig. 6.3.

According to Table. 6.2, we can conclude that RPDMF achieves the highest mF_1^{max} score when compared to WNMF and PDMF on all three datasets in very case but one. Even for that particular case, its score is very close to that of the

Table 6.2.: mF_1^{max} scores on three datasets for 3 methods as noise level σ increases

Dataset	Synthetic			NUS-WIDE TAGGED			MIRFLICKR-25K		
σ	WNMF	PDMF	RPDMF	WNMF	PDMF	RPDMF	WNMF	PDMF	RPDMF
0.1	0.7807	0.8688	0.8761	0.4760	0.7493	0.7544	0.6696	0.7987	0.7990
0.2	0.7421	0.8563	0.8602	0.3649	0.6654	0.6743	0.6039	0.7290	0.7655
0.3	0.6682	0.8208	0.8201	0.2936	0.6067	0.6672	0.5725	0.7171	0.7301
0.4	0.6686	0.6925	0.7481	0.2804	0.5623	0.5962	0.5544	0.6081	0.6190
0.5	0.6494	0.6487	0.7121	0.2761	0.3902	0.4442	0.5510	0.5604	0.5847

best result. Our method significantly outperforms WNMF in terms of mF_1^{max} score on all three datasets. We can also see that the performance of PDMF degrades more rapidly compared to RPDMF, as we increase the number of polluted (flipped) elements. This points to the robustness of our algorithm. In the three plots, we see that the mF_1^{max} scores of RPDMF drop faster in the tails compared to WNMF as σ increases, primarily because the performance of WNMF is already at a very low level. Since our original matrix is binary, it should be noted that any result close to $P(X_{ij} = p_s)$ means that the factorization provides little practical value. That is because the probability of a random guess for any element in this matrix should be $P(X_{ij} = p_s)$.

However, RPDMF is more computationally expensive compared to WNMF and PDMF, according to our experiment. Even though we avoid joint optimization and apply the Quasi-Newton approach BFGS, we still need to do intensive convex optimization in every iteration when updating $W_{i\cdot}^k$ or $H_{j\cdot}^k$ based on 'samples' in the other matrix. So the computational complexity of RPDMF grows exponentially with the size of the original matrix X. In contrast, WNMF utilizes simple gradient-based method to reach convergence for both W^k and H^k. In every iteration of WNMF, W^k and H^k update once respectively, and the updates involve only simple matrix multiplication. In terms of computational complexity, RPDMF is almost equivalent to multiple (≈ 10) iterations of PDMF, because we need to tune one more parameter q in (6.2).

6.5 Conclusion

In this chapter, we presented a new method RPDMF, which can be used to recover missing data in a noisy binary matrix. This method is motivated by the real-word application: image tag completion. In RPDMF, we introduce the logistic loss into the cost function and optimize two base matrices W and H alternately to reach convergence. We evaluate RPDMF on three datasets, and compare it with the competing methods WNMF and PDMF. According to our experiment, we see that RPDMF has a significant advantage over WNMF and PDMF in terms of the F_1 measure when dealing with noisy matrices.

7. CONCLUSION

In this thesis, we propose several new algorithms targeting different image understanding tasks and these algorithms have been tested and verified on multiple platforms.

For the image classification discussed in Chap. 2, we designed new features which can help distinguish text, mix and picture images. These features are designed to meet the monotonicity requirement so that quick decision can be achieved. Also for the reason of quick decision, we re-designed the SVM classifiers carefully to meet the monotonicity requirement. Compared with the traditional image classification methods, the proposed algorithm do not require the global information of image. Instead, it only requires a strip at a time for feature calculation, which is computationally affordable to low-end AIO printer. Besides, the proposed algorithm is capable of making quick decision for certain type of images by our design of features and classifiers. On top of that, we creatively incorporate the adaptive learning into our system while preserving the quick decision functionality. We formulate the problem of combining the online SVM training with quick classification decision. This problem is solved by controlling the decision boundary when using different kernels and selectively adding new training data.

For the extend scanned image classification problem discussed in Chap. 3, we designed several new features to distinguish highlighted text and receipt plus the three types in Chap. 2. Unlike the algorithm introduced in Chap. 2 which only uses luminance information, we also utilize the chroma information to find indicators to these 5 types. These new features are Chroma Histogram Flatness, Chroma Around Text, Color Block Ratio and White Block Ratio. Experimentally, these features show discriminative powers for highlighted text and receipt.

In Dynamic Print Stream Compression discussed in Chap. 4, we propose a new engine which is capable of choosing the optimal compression algorithm for input im-

age given the system constraint. we use similar features discussed in early chapters for first stage classification. However, these features are re-design to be more computationally efficient. We design new features to make better choice between RLE and DRC compression, where the goodness of the choice is determined by their decompression time at the firmware. We also introduce a tunable system for optimal JPEG compression. With image quality requirement from the users, the system can make the optimal JPEG compression by balancing between image quality and compression ratio.

For image retrieval task discussed in Chap.5, we follow the BoW approach when doing the initial retrieval. At the second stage, we propose Hierarchical Weighted Spatial Ranking to find the best match in the database. Compared with traditional homography goodness test through SVD, the proposed method can handle some challenging cases by introducing a new term when doing similarity ranking. Also, we hierarchically search for the best match if more than one good homographies are found. To find the best result from several top candidates, we apply affinity goodness check for better accuracy.

For the image tag recommendation discussed in Chap. 6, we propose a new method for matrix factorization. This method can better handle binary matrix with errors by introducing a robust term in the cost function. It strengthens the probabilistic approach of matrix factorization with more tolerance to noisy input. Experimentally, it outperforms other existing matrix factorization methods in the task of image tag recommendation.

To conclude, the primary contribution of the work is listed as below

- Scanned Image Classification

  $\Rightarrow$ novel features for image classification

  $\Rightarrow$ quick decision functionality which is based on feature design and classification decision boundary control

⇒ online SVM adaptive learning of system which is compatible with quick decision functionality

- Extended Scanned Image classification

  ⇒ novel features to recognize receipt and hightlight text images

  ⇒ chroma information analysis

- Dynamic Print Stream Compression

  ⇒ design features which are more computationally efficient

  ⇒ new features to choose between RLE and DRC

  ⇒ introduce a tunable system which is used for optimal JPEG compression

- Image Retrieval for Hybrid Learning

  ⇒ propose hierarchical weighted spatial ranking to achieve better point matching

- Tag Recommendation via Robust Probabilistic Discriminative Matrix Factorization

  ⇒ a new method can better handle binary matrix with errors

  ⇒ introduce the robust term in the cost function

  ⇒ outperform existing methods experimentally

REFERENCES

REFERENCES

[1] J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei, "Content-based image indexing and searching using daubechies' wavelets," *International Journal on Digital Libraries*, vol. 1, no. 4, pp. 311–328, 1998.

[2] S. Sclaroff, L. Taycher, and M. La Cascia, "Imagerover: A content-based image browser for the world wide web," in *Content-Based Access of Image and Video Libraries, 1997. Proceedings. IEEE Workshop on*. IEEE, 1997, pp. 2–9.

[3] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, no. 6, pp. 610–621, Nov 1973, doi: 10.1109/TSMC.1973.4309314.

[4] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, Dec 2000, doi: 10.1109/34.895972.

[5] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *The proceedings of the seventh IEEE international conference on Computer vision, 1999.*, vol. 2. Kerkyra: IEEE, Sep 1999, pp. 1150–1157, doi: 10.1109/ICCV.1999.790410.

[7] B. S. Manjunath and W.-Y. Ma, "Texture features for browsing and retrieval of image data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 8, pp. 837–842, 1996.

[8] G. S. Yovanof, "Compression in a printer pipeline," in *1995 Conference Record of the Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, A. Singh, Ed., vol. 1. Pacific Grove: IEEE, 1995, pp. 219–223, doi: 10.1109/ACSSC.1995.540544.

[9] X. Dong, K.-L. Hua, P. Majewicz, G. McNutt, C. A. Bouman, J. P. Allebach, and I. Pollak, "Document page classification algorithms in low-end copy pipeline," *Journal of Electronic Imaging: Reasoning*, vol. 17, no. 4, Oct 2008, doi: 10.1117/1.3010879.

[10] H. Cheng and C. A. Bouman, "Document compression using rate-distortion optimized segmentation," *Journal of Electronic Imaging*, vol. 10, no. 2, pp. 460–474, 2001, doi: 10.1117/1.1344590.

[11] R. L. de Queiroz, "Compression of compound documents," in *1999 International Conference on Image Processing, 1999. ICIP 99. Proceedings.*, vol. 1. Kobe: IEEE, 1999, pp. 209–213, doi: 10.1109/ICIP.1999.821599.

[12] S. V. Revankar and Z. Fan, "Picture, graphics, and text classification of document image regions," in *Photonics West 2001-Electronic Imaging*, R. Eschbach and G. G. Marcu, Eds. San Francisco: International Society for Optics and Photonics, 2000, pp. 224–228, doi: 10.1117/12.410794.

[13] S. J. Simske and S. C. Baggs, "Digital capture for automated scanner workflows," in *Proceedings of the 2004 ACM symposium on Document engineering*, E. Munson, Ed. Milwaukee: ACM, 2004, pp. 171–177, doi: 10.1145/1030397.1030431.

[14] W. Wang, I. Pollak, T.-S. Wong, C. A. Bouman, M. P. Harper, and J. M. Siskind, "Hierarchical stochastic image grammars for classification and segmentation," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 3033–3052, 2006, doi: 10.1109/TIP.2006.877496.

[15] C. Lu, J. Wagner, B. Pitta, D. Larson, and J. Allebach, "SVM-based automatic scanned image classification with quick decision capability," in *Proc. SPIE 9015, Color Imaging XIX: Displaying, Processing, Hardcopy, and Applications*, R. Eschbach, G. G. Marcu, and A. Rizzi, Eds. San Francisco: SPIE, Feb 2014, doi: 10.1117/12.2047335.

[16] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995, doi: 10.1007/BF00994018.

[17] A. Srivastav and J. Kumar, "Text detection in scene images using stroke width and nearest-neighbor constraints," in *TENCON 2008-2008 IEEE Region 10 Conference*. Hyderabad: IEEE, 2008, pp. 1–5, doi: 10.1109/TENCON.2008.4766826.

[18] A. K. Jain and B. Yu, "Automatic text location in images and video frames," in *Fourth Internatioal Conference on Pattern recognition*, vol. 2, no. 4. IEEE, 1998, pp. 1497–1499, doi: 10.1109/ICPR.1998.711990.

[19] R. Lienhart and A. Wernicke, "Localizing and segmenting text in images and videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 4, pp. 256–268, 2002, doi: 10.1109/76.999203.

[20] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM Transactions on Graphics (TOG)*, J. Marks, Ed., vol. 23, no. 3. ACM, 2004, pp. 309–314, doi: 10.1145/1186562.1015720.

[21] B. Lei and L.-Q. Xu, "Real-time outdoor video surveillance with robust foreground extraction and object tracking via multi-state transition management," *Pattern Recognition Letters*, vol. 27, no. 15, pp. 1816–1825, 2006, doi: 10.1016/j.patrec.2006.02.017.

[22] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, Nov 1973, doi: 10.1109/TSMC.1973.4309314.

[23] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999, doi: 10.1109/72.788646.

[24] J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 4, pp. 779–785, 1994, doi: 10.1109/36.298007.

[25] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, I. Essa, S. B. Kang, and M. Pollefeys, Eds. Miami: IEEE, 2009, pp. 1794–1801, doi: 10.1109/CVPR.2009.5206757.

[26] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *2010 IEEE Conference on Computer Vision and Pattern Recognition*. San Francisco: IEEE, June 2010, pp. 3360–3367, doi: 10.1109/CVPR.2010.5540018.

[27] M. Szummer and R. W. Picard, "Indoor-outdoor image classification," in *1998 IEEE International Workshop on Content-Based Access of Image and Video Database, 1998. Proceedings.* IEEE, 1998, pp. 42–51, doi: 10.1109/CAIVD.1998.646032.

[28] M. I. Sezan, "A peak detection algorithm and its application to histogram-based image data reduction," *Computer vision, graphics, and image processing*, vol. 49, no. 1, pp. 36–51, 1990, doi: 10.1016/0734-189X(90)90161-N.

[29] O. J. Tobias and R. Seara, "Image segmentation by histogram thresholding using fuzzy sets," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1457–1465, 2002, doi: 10.1109/TIP.2002.806231.

[30] K. Sim, C. Tso, and Y. Tan, "Recursive sub-image histogram equalization applied to gray scale images," *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1209–1221, 2007, doi: 10.1016/j.patrec.2007.02.003.

[31] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Denver: MIT Press, 2001, pp. 409–415.

[32] J. A. Bilmes *et al.*, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[33] X. Zhang and F. Sun, "Pulse coupled neural network edge-based algorithm for image text locating," *Tsinghua Science & Technology*, vol. 16, no. 1, pp. 22–30, 2011, doi: 10.1016/S1007-0214(11)70004-9.

[34] C. Liu, C. Wang, and R. Dai, "Text detection in images based on unsupervised classification of edge-based features," in *2005. Proceedings Eighth International Conference on Document Analysis and Recognition*, vol. 2. Seoul: IEEE, 2005, pp. 610–614, doi: 10.1109/ICDAR.2005.228.

[35] X. Liu and J. Samarabandu, "Multiscale edge-based text extraction from complex images," in *2006 IEEE International Conference on Multimedia and Expo*. Toronto: IEEE, 2006, pp. 1721–1724, doi: 10.1109/ICME.2006.262882.

[36] J. Zhang and R. Kasturi, "Text detection using edge gradient and graph spectrum," in *2010 20th International Conference on Pattern Recognition (ICPR)*. Istanbul: IEEE, 2010, pp. 3979–3982, doi: 10.1109/ICPR.2010.968.

[37] D. Chen, K. Shearer, and H. Bourlard, "Text enhancement with asymmetric filter for video OCR," in *2001. Proceedings. 11th International Conference on Image Analysis and Processing*. Palermo: IEEE, Sep 2001, pp. 192–197, doi: 10.1109/ICIAP.2001.957007.

[38] H. Li and D. Doermann, "Text enhancement in digital video using multiple frame integration," in *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, J. Buford and S. Stevens, Eds. Mason, OH: ACM, Oct 1999, pp. 19–22, doi: 10.1145/319463.319466.

[39] X.-C. Huang and B. D. Nystrom, "Multilevel ink mixing device and method using diluted and saturated color inks for inkjet printers," Jan. 9 2001, US Patent 6,172,692.

[40] V. Bulović, A. Shoustikov, M. Baldo, E. Bose, V. Kozlov, M. Thompson, and S. Forrest, "Bright, saturated, red-to-yellow organic light-emitting devices based on polarization-induced spectral shifts," *Chemical Physics Letters*, vol. 287, no. 3, pp. 455–460, 1998.

[41] K. Okada, Y. Ueda, J. Oyabu, N. Ogasawana, A. Hirayama, and K. Kodama, "Plaque color analysis by the conventional yellow-color grading system and quantitative measurement using LCH color space," *Journal of interventional cardiology*, vol. 20, no. 5, pp. 324–334, 2007.

[42] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, Aug 2002.

[43] D. Taubman and M. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice: Image Compression Fundamentals, Standards and Practice*. Springer Science & Business Media, 2012, vol. 642, dOI: 10.1007/978-1-4615-0799-4.

[44] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, Aug 2000, doi: 10.1109/83.855427.

[45] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *Signal Processing Magazine, IEEE*, vol. 18, no. 5, pp. 36–58, 2001, doi: 10.1109/79.952804.

[46] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 244–250, 1992, doi: 10.1109/83.136601.

[47] E. J. Delp and O. R. Mitchell, "Image compression using block truncation coding," *IEEE Transactions on Communications*, vol. 27, no. 9, pp. 1335–1342, Sep 1979, doi: 10.1109/TCOM.1979.1094560.

[48] A. Said and W. A. Pearlman, "Reversible image compression via multiresolution representation and predictive coding," in *Visual Communications' 93*, B. G. Haskell and H.-M. Hang, Eds., vol. 2094. International Society for Optics and Photonics, 1993, pp. 664–674, doi: 10.1117/12.157984.

[49] P. Haffner, P. G. Howard, P. Simard, Y. Bengio, Y. Lecun *et al.*, "High quality document image compression with DjVu," *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 410–425, July 1998, doi: 10.1117/1.482609.

[50] Z. Wang and A. C. Bovik, "A universal image quality index," *Signal Processing Letters, IEEE*, vol. 9, no. 3, pp. 81–84, March 2002, doi: 10.1109/97.995823.

[51] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004, doi: 10.1109/TIP.2003.819861.

[52] V. K. Goyal, A. K. Fletcher, and S. Rangan, "Compressive sampling and lossy compression," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 48–56, March 2008, doi: 10.1109/MSP.2007.915001.

[53] G. L. Vondran Jr, "Method and apparatus for delta row decompression," Sep. 19 1995, US Patent 5,452,405.

[54] S. C. Hinds, J. L. Fisher, and D. P. D'Amato, "A document skew detection method using run-length encoding and the hough transform," in *1990. Proceedings., 10th International Conference on Pattern Recognition*, vol. 1. Atlantic City: IEEE, June 1990, pp. 464–468, doi: 10.1109/ICPR.1990.118147.

[55] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, Jan 1991, doi: 10.1145/103085.103089.

[56] J. Lubin, "A human vision system model for objective picture quality measurements," in *Broadcasting Convention, 1997. International.* Amsterdam: IET, Sep 1997, pp. 498–503, doi: 10.1049/cp:19971319.

[57] G. E. Legge and J. M. Foley, "Contrast masking in human vision," *JOSA*, vol. 70, no. 12, pp. 1458–1471, Dec 1980, doi: 0030-3941/80/121458-1.

[58] M. Tagliasacchi, A. Trapanese, S. Tubaro, J. Ascenso, C. Brites, and F. Pereira, "Exploiting spatial redundancy in pixel domain wyner-ziv video coding," in *2006 IEEE International Conference on Image Processing.* Atlanta: IEEE, Oct 2006, pp. 253–256, doi: 10.1109/ICIP.2006.313173.

[59] F. H. Fitzek and M. Reisslein, "MPEG-4 and H. 263 video traces for network performance evaluation," *Network, IEEE*, vol. 15, no. 6, pp. 40–54, Aug 2001, doi: 10.1109/65.967596.

[60] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi, "A no-reference perceptual blur metric," in *2002. Proceedings. 2002 International Conference on Image Processing.*, vol. 3. Rochester: IEEE, 2002, pp. III–57, doi: 10.1109/ICIP.2002.1038902.

[61] C. Chen, Y. Q. Shi, W. Chen, and G. Xuan, "Statistical moments based universal steganalysis using jpeg 2-d array and 2-d characteristic function," in *2006 IEEE International Conference on Image Processing*. Atlanta: IEEE, Oct 2006, pp. 105–108, doi: 10.1109/ICIP.2006.312383.

[62] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[63] Z. Wang, H. R. Sheikh, and A. C. Bovik, "No-reference perceptual quality assessment of JPEG compressed images," in *2002. Proceedings. 2002 International Conference on Image Processing*, vol. 1. Rochester: IEEE, 2002, pp. I–477, doi: 10.1109/ICIP.2002.1038064.

[64] R. V. Babu, S. Suresh, and A. Perkis, "No-reference JPEG-image quality assessment using gap-rbf," *Signal Processing*, vol. 87, no. 6, pp. 1493–1503, Dec 2007, doi:10.1016/2006.12.014.

[65] A. M. Eskicioglu and P. S. Fisher, "Image quality measures and their performance," *IEEE Transactions on Communications*, vol. 43, no. 12, pp. 2959–2965, Aug 1995, doi: 10.1109/26.477498.

[66] J. Brown, C. Hoang, C. Knapp, and J. Wellman, "Image printing solution for a printing device," Sep. 19 1999, US Patent 5,960,166.

[67] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation," in *AI 2006: Advances in Artificial Intelligence*. Springer, 2006, pp. 1015–1021.

[68] Y. Rui, T. S. Huang, and S.-F. Chang, "Image retrieval: Current techniques, promising directions, and open issues," *Journal of visual communication and image representation*, vol. 10, no. 1, pp. 39–62, July 1999, doi: 10.1145/1282280.1282347.

[69] D. L. Swets and J. J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 8, pp. 831–836, Aug 1996, doi: 10.1109/34.531802.

[70] V. N. Gudivada and V. V. Raghavan, "Content based image retrieval systems," *Computer*, vol. 28, no. 9, pp. 18–22, 1995.

[71] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–534, 1997, doi: 10.1109/34.589215.

[72] S. R. Lawrence and C. L. Giles, "Meta search engine," Feb. 14 2006, US Patent 6,999,959.

[73] M. Yang, K. Kpalma, and J. Ronsin, "A survey of shape feature extraction techniques," *Pattern recognition*, pp. 43–90, Nov 2008.

[74] S. Deb and Y. Zhang, "An overview of content-based image retrieval techniques," in *2004. AINA 2004. 18th International Conference on Advanced Information Networking and Applications*, vol. 1. Vancouver, Canada: IEEE, 2004, pp. 59–64, doi: 10.1109/AINA.2004.1283888.

[75] G. Giacinto, "A nearest-neighbor approach to relevance feedback in content based image retrieval," in *Proceedings of the 6th ACM international conference on Image and video retrieval*, N. Sebe and M. Worring, Eds. Amsterdam: ACM, July 2007, pp. 456–463, doi: 10.1145/1282280.1282347.

[76] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[77] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14, Oakland, CA, USA., Oct 1967, pp. 281–297.

[78] Z. Li, J. Liu, X. Zhu, T. Liu, and H. Lu, "Image annotation using multi-correlation probabilistic matrix factorization," in *International Conference on Multimedia*, A. Bimbo and S.-F. Chang, Eds. Firenze, Italy: ACM, Oct 2010, pp. 1187–1190, doi: 10.1145/1873951.1874183.

[79] L. Wu, R. Jin, and A. K. Jain, "Tag completion for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 716–727, May 2013, doi: 10.1109/TPAMI.2012.124.

[80] Q. Wang, B. Shen, S. Wang, L. Li, and L. Si, "Binary codes embedding for fast image tagging with incomplete labels," in *Computer Vision–ECCV 2014*. Springer, Sep 2014, pp. 425–439.

[81] Z. Lin, G. Ding, M. Hu, J. Wang, and X. Ye, "Image tag completion via image-specific and tag-specific linear sparse reconstructions," in *IEEE Conference on Computer Vision and Pattern Recognition*. Portland: IEEE, June 2013, pp. 1618–1625, doi: 10.1109/CVPR.2013.212.

[82] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[83] C. H. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering." in *SIAM International Conference on Data Mining*, vol. 5. Newport Beach, CA: SIAM, April 2005, pp. 606–610, doi: 1.9781611972757.70.

[84] D. Agarwal and B.-C. Chen, "fLDA: matrix factorization through latent dirichlet allocation," in *Proceedings of the third ACM international conference on Web search and data mining*, B. Davison and T. Suel, Eds. New York City: ACM, Feb 2010, pp. 91–100, doi: 10.1145/1718487.1718499.

[85] C. Liu, H.-c. Yang, J. Fan, L.-W. He, and Y.-M. Wang, "Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce," in *Proceedings of the 19th international conference on World wide web*, M. Rappa and P. Jones, Eds. Raleigh, NC: ACM, April 2010, pp. 681–690, doi: 10.1145/1772690.1772760.

[86] W.-Y. Chen, J.-C. Chu, J. Luan, H. Bai, Y. Wang, and E. Y. Chang, "Collaborative filtering for orkut communities: discovery of user latent behavior," in *Proceedings of the 18th international conference on World wide web*, J. Quemadad and G. Leon, Eds. Madrid: ACM, April 2009, pp. 681–690, doi: 10.1145/1526709.1526801.

[87] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, V. Shen and N. Saito, Eds. HongKong: ACM, April 2001, pp. 285–295, doi: 10.1145/371920.372071.

[88] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, G. Copper and S. Moral, Eds. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., July 1998, pp. 43–52.

[89] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Proceeding Eighteenth national conference on Artificial intelligenceAAAI/IAAI*, R. Dechter and M. Keams, Eds., Edmonton, Alberta, July 2002, pp. 187–192.

[90] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM conference on Information and knowledge management*, J. Shanahan, Ed. Napa Valley,CA: ACM, Oct 2008, pp. 931–940, doi: 10.1145/1458082.1458205.

[91] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*, T. Leen and T. Dietterich, Eds. Denver: MIT Press, 2000, pp. 556–562, doi: 10.1.1.31.7566.

[92] N. Srebro, J. D. M. Rennie, and T. S. Jaakola, "Maximum-margin matrix factorization," in *Advances in Neural Information Processing Systems 17*, L. Saul and Y. Weiss, Eds. Vancouver, Canada: MIT Press, Dec 2005, pp. 1329–1336.

[93] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, B. Scholkopf, J. Platt, and T. Hofman, Eds. Vancouver, B.C.: MIT Press, Dec 2007, pp. 1257–1264.

[94] D. D. Lee and H. S. Seung, "Learning the parts of objects by nonnegative matrix factorization," *Nature*, vol. 401, pp. 788–791, Oct 1999, doi: 10.1038/44565.

[95] Q. Gu, J. Zhou, and C. H. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs." in *SDM*. SIAM, 2010, pp. 199–210, doi: 10.1137/1.9781611972801.18.

[96] M. Nickel and V. Tresp, "Logistic tensor factorization for multi-relational data," *arXiv preprint arXiv:1306.2084*, 2013.

[97] C. Tang, A. Veis, R. Ulichney, and J. Allebach, "Irregular clustered-dot periodic halftone screen design," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2014, pp. 90 150S–90 150S.

[98] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: A real-world web image database from national university of singapore," in *International Conference on Image and Video Retrieval*, Y. Kompatsiaris and S. Marchand-Maillet, Eds. New York, NY, USA: ACM, July 2009, pp. 48:1–48:9, doi: 10.1145/1646396.16464525.

[99] G. Zhu, S. Yan, and Y. Ma, "Image tag refinement towards low-rank, content-tag prior and error sparsity," in *Proceedings of the international conference on Multimedia*, A. Bimbo and S.-F. Chang, Eds. Firenze, Italy: ACM, Oct 2010, pp. 461–470, doi: 10.1145/1873951.1874028.

[100] E. K. Chong and S. H. Zak, *An introduction to optimization*. John Wiley & Sons, 2013, vol. 76.

VITA

VITA

Cheng Lu received his B.S. degree in Electrical Engineering from Purdue University in 2012. Since 2012, he has been working on his Ph.D. in Electrical Engineering ar Purdue University. He had two summer internship with Hewlett-Packard, Boise, in 2013 and 2014 respectively. He had another summer internship with HP 3D&Printing Lab at Palo Alto, CA, in 2015. His research focuses on image processing, computer vision and machine learning. He has been supervised by Prof. Allebach since 2012.