TEXT DETECTION IN NATURAL SCENES THROUGH WEIGHTED MAJORITY VOTING OF DCT HIGH PASS FILTERS, LINE REMOVAL, AND COLOR CONSISTENCY FILTERING

by

Dave Snyder

B.S., Rochester Institute of Technology, 2009

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in the Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology

May, 2011

Signature of the Author ________________________________

Accepted by ________________________________

Coordinator, M.S. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Dave Snyder
has been examined and approved by the
thesis committee as satisfactory for the
thesis required for the
M.S. degree in Imaging Science

Dr. Richard Zanibbi, Thesis Advisor

Dr. Carl Salvaggio

Dr. Jeff Pelz

Date

THESIS RELEASE PERMISSION
ROCHESTER INSTITUTE OF TECHNOLOGY
CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE

Title of Thesis:

TEXT DETECTION IN NATURAL SCENES THROUGH WEIGHTED MAJORITY VOTING OF DCT HIGH PASS FILTERS, LINE REMOVAL, AND COLOR CONSISTENCY FILTERING

I, Dave Snyder, hereby grant permission to Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature ____________________________
Date

Acknowledgments

I wish to thank Paul Romanczyk for spending more time on my thesis than his own; my committee and advisor Dr. Carl Salvaggio, Dr. Jeff Pelz, and Dr. Richard Zanibbi, for their direction and insight into this research; Sue Chan and Bethany Choate for administrative help; Eugenie Song for helping develop the color quantization code in undergraduate DIP; the DPRL lab members for letting me use far too many server resources; and my friends and family for getting me through the past two years.

TEXT DETECTION IN NATURAL SCENES THROUGH WEIGHTED MAJORITY VOTING OF DCT HIGH PASS FILTERS, LINE REMOVAL, AND COLOR CONSISTENCY FILTERING

Publication No. ___________________

Dave Snyder, M.S.
Rochester Institute of Technology, 2011

Supervisor: Richard Zanibbi

Abstract

Detecting text in images presents the unique challenge of finding both in-scene and superimposed text of various sizes, fonts, colors, and textures in complex backgrounds. The goal of this system is not to recognize specific letters or words but only to determine if a pixel is text or not. This pixel level decision is made by applying a set of weighted classifiers created using a set of high pass filters, and a series of image processing techniques. It is our assertion that the learned weighted combination of frequency filters in conjunction with image processing techniques may show better pixel level text detection performance in terms of precision, recall, and f-metric, than any of the components do individually. Qualitatively, our algorithm performs well and shows promising results. Quantitative numbers are not as high as is desired, but not unreasonable. For the complete ensemble, the f-metric was found to be 0.36.

Table of Contents

Acknowledgments **iv**

Abstract **v**

List of Tables **ix**

List of Figures **x**

Chapter 1. Introduction **1**

- 1.1 Motivation . . . 1
- 1.2 Overview . . . 2

Chapter 2. Background **7**

- 2.1 Transform Functions . . . 11
  - 2.1.1 Discrete Cosine Transform . . . 12
  - 2.1.2 Discrete Wavelet Transform . . . 14
- 2.2 Segmentation . . . 16
  - 2.2.1 Edge Detection . . . 16
  - 2.2.2 Color and Contrast Segmentation . . . 18
  - 2.2.3 Discrete Cosine Transform Coefficients . . . 19
  - 2.2.4 Wavelets for Segmentation . . . 20
- 2.3 Features . . . 20
  - 2.3.1 Wavelet Features . . . 20
  - 2.3.2 Other Features . . . 21
- 2.4 Classification of Regions . . . 22
  - 2.4.1 Support Vector Machine . . . 22
  - 2.4.2 Neural Network . . . 23
- 2.5 Post Processing . . . 23

    - 2.5.1 Binarization 23
    - 2.5.2 Temporal Heuristics 24
    - 2.5.3 Text Tracking 25
  - 2.6 Published Results 25
  - 2.7 Summary 28

Chapter 3. Methodology **30**
  - 3.1 Dataset 30
    - 3.1.1 Ground Truth 32
  - 3.2 Algorithm Overview 35
    - 3.2.1 Feature Selection 35
    - 3.2.2 Creation of Classifier Ensembles 42
    - 3.2.3 Weighted Majority 48
    - 3.2.4 Line Removal 50
    - 3.2.5 Color Filtering 55
    - 3.2.6 Edge Detection 60

Chapter 4. Experimental Results **68**

Chapter 5. Discussion **81**
  - 5.1 Features and Classifiers 81
  - 5.2 Weighted Majority Algorithm 83
  - 5.3 Post Processing 87
  - 5.4 Analysis of Results 89
  - 5.5 Comparison With Other Algorithms 90

Chapter 6. Conclusions and Future Work **108**

Bibliography **110**

Vita **119**

List of Tables

2.1 Overview of reviewed algorithms 10

4.1 Classifier comparison; Individual DCT-based Classifiers 70

List of Figures

1.1 Video CAPTCHA 2

2.1 In-scene vs. superimposed text 8
2.2 JPEG compression illustration 13
2.3 Block diagram of the DWT 15
2.4 Block diagram of two scales of the DWT 16

3.1 Example image from the training data in its original form. . . 31
3.2 Pixel level vs. word level ground truth 33
3.3 Pixel level ground truth example 34
3.4 Training algorithm – overview 36
3.5 Training algorithm – detailed 37
3.6 Classification algorithm 38
3.7 Y channel of our example image converted from RGB to YCC. 39
3.8 Highpass Gaussian filters 41
3.9 Highpass example image 43
3.10 f-metric mean scores for filters 44
3.11 Example thresholded filtered image 45
3.12 Example thresholded filtered image 46
3.13 Example thresholded filtered image 47
3.14 Weighted Majority algorithm 49
3.15 Histogram of aspect ratios computed for ground truth connected components. 51
3.16 Eigenvalue line filter applied to the classifier shown in figure 3.11. 52
3.17 Eigenvalue line filter applied to the classifier shown in figure 3.12. 53
3.18 Eigenvalue line filter applied to the classifier shown in figure 3.13. 54
3.19 Histogram of average color differences computed for ground truth connected components. 56

3.20 Color consistency filter example 57
3.21 Color consistency filter example 58
3.22 Color consistency filter example 59
3.23 Example Canny edge detector result 61
3.24 Morphological filling of connected components 62
3.25 Result of running the linearity filter on the Canny edge detection image. 63
3.26 Morphological filling example 64
3.27 The intersection of the edge detection results and a classifier. . 66
3.28 Final result 67

4.1 Precision for individual components 71
4.2 Recall for individual components 72
4.3 f-metric for individual components 73
4.4 Histogram of precision metric on testing data for the complete ensemble. 74
4.5 Histogram of precision metric on testing data for post processing only. 74
4.6 Histogram of recall metric on testing data for the complete ensemble. 75
4.7 Histogram of recall metric on testing data for post processing only. 75
4.8 Histogram of f-metric on testing data for the complete ensemble. 76
4.9 Histogram of f-metric on testing data for post processing only. 76
4.10 Example mostly correct image 78
4.11 Example partially correct image 79
4.12 Example incorrect image 80

5.1 Alternate linearity threshold example 84
5.2 Alternate color consistency threshold example 85
5.3 Change in classifier weight during training. 86
5.4 Example discussed result 91
5.5 Example discussed result 92

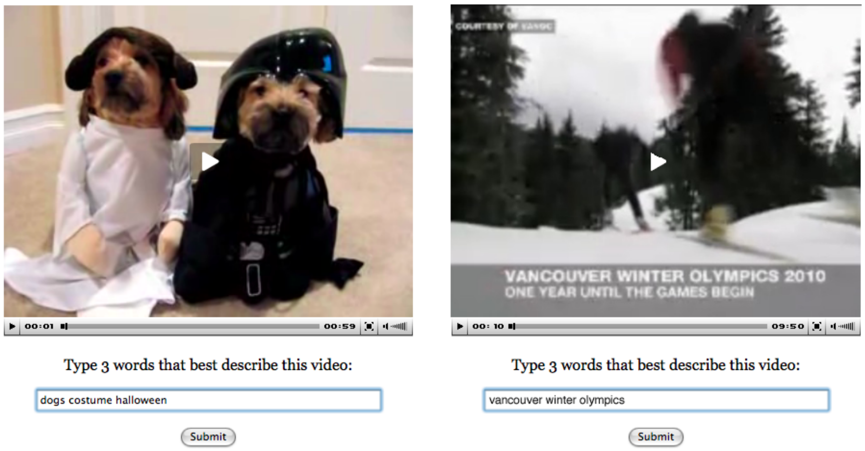
5.6 Example discussed result 93
5.7 Example discussed result 94
5.8 Example discussed result 95
5.9 Example discussed result 96
5.10 Example discussed result 97
5.11 Example discussed result 98
5.12 Example discussed result 99
5.13 Example discussed result 100
5.14 Example discussed result 101
5.15 Example discussed result 102
5.16 Example discussed result 103
5.17 Example discussed result 104
5.18 Example discussed result 105
5.19 Example discussed result 106
5.20 Example discussed result 107

Chapter 1

Introduction

1.1 Motivation

A Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is a challenge-response test used to distinguish humans from computers. This test is very commonly presented as a warped image containing text that is easy for a human to understand but difficult for a machine to understand. Much research has been done, however, in using machines to break CAPTCHA challenges and emulate a human response. And so, this growing ability for machines to produce the same response as humans has necessitated the evolution of CAPTCHA, and the advancement of the challenge that is presented. In an effort to balance usability with security, Kurt Kluever explored using Video CAPTCHA [20]. Currently, random videos taken from YouTube are presented to the user, unaltered. The user then types three words describing the video, as is shown in Figure 1.1. These words are compared with the video's tags and with the tags of some related videos. The challenge is passed successfully if one of the words typed by the user is in the video tag word list. Many of these videos contain text, and it seems likely that this text may be used to pass the challenge. To prevent this potential


00:01
00:59
COURTESY OF VANOC
VANCOUVER WINTER OLYMPICS 2010
ONE YEAR UNTIL THE GAMES BEGIN
00:10
09:50
Submit
Submit

Type 3 words that best describe this video:

dogs costume halloween

Type 3 words that best describe this video:

vancouver winter olympics

Figure 1.1: Left: Video CAPTCHA without text[20]. Right: Video CAPTCHA with text.

vulnerability it may be possible to preprocess videos, detect the text, and mask it before displaying it as a CAPTCHA.

Two distinct problems are encountered, detection and segmentation of text in video, and removal of this text in an elegant, unobtrusive manner. We are concerned with the first problem: detection of text in video.

1.2 Overview

Detecting text in images presents the unique challenge of finding both in-scene and superimposed text of various sizes, fonts, colors, and textures in complex backgrounds. In-scene text is text in an image which has been captured from the natural scene by the imaging system. Superimposed text is text which has been added to an image during post processing. Ideally a system can be constructed which is able to find all text in images, regardless of the specific properties of any given text sample.

The goal of this system is not to recognize specific letters or words but only to determine if a pixel is text or not. This can then be used in conjunction with a standard OCR algorithm to recognize words, word level bounding boxes can be constructed, or it can be used for other applications. Our motivation for creating a text detector is to enhance the security of video CAPTCHA. Finding text in the video and removing it with inpainting or a similar technique before presenting it to the user greatly reduces this security risk.

Seeking to accomplish this task, we focus on still images and single frames of video. Unlike all of the other methods surveyed in Chapter 2, our method attempts to perform text detection at the pixel level to recover individual characters. All other methods researched work instead at the word level; that is a bounding box is constructed around all characters which have been determined to be in a single word. Because of this, both our ground truth and our scoring metrics differ slightly from that of others.

Starting with a public dataset with word level bounding box ground truth, pixel level ground truth was manually created using an interactive technique. Images were color quantized and background colors were manually selected for removal until only foreground text pixels remained, creating a pixel level ground truth mask.

Images are converted to grayscale by converting to the YCC color space and extracting the luminance component. A series of high pass filters are applied in the DCT frequency domain. Our method makes use of the Hedge(β) algorithm [13], a variant of the Weighted Majority algorithm [30]. Weights are assigned to a set of classifiers constructed by thresholding these filters using the Weighted Majority algorithm. Classifiers which maximize the f-metric of precision and recall, based on pixel level ground truth, are assigned a high weight. The final binary classification mask is constructed for any given image by applying each classifier, assigning each pixel selected by the classifier the value of the weight of that classifier, summing the classification results, and thresholding at 0.5; that is pixels receiving a majority vote from the weighted classifiers are used as the final classification mask for that image.

False positives are reduced in a multi-step post processing procedure. Starting with the classifier output, the ratio of the Eigenvalues of the coordinate covariance matrix of each connected component is used to remove objects which are too linear and therefore more likely to be high frequency artifacts instead of being text characters. Connected components are simply groups of pixels which are neighboring each other. They may be text characters or false alarms. Components are morphologically filled in and color consistency within each filled connected component is used to remove objects which are not consistent in color, and therefore not likely to be text characters. By filling in connected components we are also reducing false negatives, since interior pixels which may have been labeled background may now be labeled as text in the final output. Threshold values for linearity and color consistency are extracted from the training data.

Working with the grayscale image, the Canny edge detector is used to perform segmentation. The same connected component linearity and color consistency rules applied to the classifier output are applied here as well. Additional constraints are applied to connected components which are nested within other components. These constraints help avoid incorrect filling of background and the interior of letters.

The intersection of the processed classifier results with the Canny segmented results determines the final output. Filled connected components from the edge detection process remain in the final result only if they are present in the classifier output. During training, post processing is run within the Weighted Majority algorithm to determine which classifiers maximize the f-metric after the entire chain has been run on a given image.

Text in images can be extracted to different levels of success by using frequency domain filtering and image processing techniques which exploit specific constraints, including color consistency and the linearity of text characters, among other possible methods. It is our assertion that the learned weighted combination of frequency filters in conjunction with image processing techniques may show better pixel level text detection performance in terms of precision, recall, and f-metric, than these components do individually.

The remainder of this document is organized as follows. Chapter 2 covers background information, including related work and features used by our algorithm. Chapter 3 describes our methodology, detailing the specifics of our algorithm. In Chapter 4, results are presented. A discussion is given in Chapter 5, followed by a conclusion and future work, in Chapter 6.

Chapter 2

Background

Although the end goal of this technique is application to video, development has focused exclusively on still images due to the availability of data and time constraints. Similarly, much related work in the area of text detection in video focuses on still frames and still images; with only a handful of algorithms taking full advantage of video. Table 2.1 indicates which algorithms explicitly use video and which do not.

Text that is displayed in a video often provides a useful context regarding the content of the video, which can then be used for understanding, retrieval, and search. The nature of readable text may make it more easily segmented and correctly identifiable than other objects, perhaps even correctly identifiable as specific characters or words. Temporal information provided by video is also useful and can be exploited for this task, assuming text is present in more than one frame of video.

Two types of text may be present in a frame of video, such as that shown in Fig 2.1. Artificial text, or superimposed text, is regarded by most researchers of video text detection techniques as the most common. This type is superimposed on the video during the editing process and is usually used

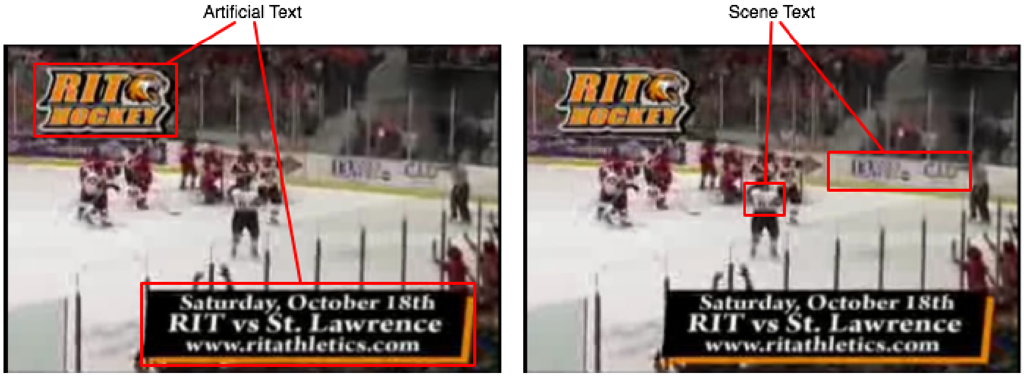


Figure 2.1: Video clip of a hockey game taken from YouTube which contains both in-scene and superimposed text.

for captions or titles. In-scene text, on the other hand, is text which is present during the filming of the video, such as sponsor banners placed around the rink at a hockey game.

Much work has been done in the past decade on text detection in natural images. Research has focused on both still imagery and video. Two comprehensive surveys of text detection [17, 18] reveal three approaches which can be taken to solve the text detection problem. These are region-based methods, including the use of color, connected components, and edges; texture-based methods, including Gabor filters, wavelets, the discrete cosine transform (DCT), and the Fourier transform; as well as other approaches, including adaptive thresholding, model based approaches, and motion based approaches when video is used. Some algorithms are able to pick out in-scene text to an extent, but the majority focus on artificial text. With the exception of Epshtein et al. [11], in the reviewed algorithms, testing was not done on artificial text and scene text independently so the extent to which any given algorithm performed on one task in comparison to the other is unknown quantitatively. Epshtein et al. focus on in-scene text only.

It is common to segment text using edge detection [6, 29] for simplicity and speed. Other more exotic methods have also been proposed which use multiscale wavelet features [52] or discrete cosine transform coefficients [36]. To avoid as many false positives as possible, classification of detected regions can also be performed. Ye et al. use only a support vector machine (SVM) [52], Chen et al. use a neural network and an SVM [6], while Lienhart and Wernicke use a neural network for their approach [29]. Each technique suggests different features to use for classification, some more intuitive than others. In contrast, Qian et al. and Lienhart and Effelsberg do not perform a classification to improve segmentation, instead focusing on approaches to segmentation which produce results that standard OCR software can use to perform the final text recognition task [28, 36]. Likewise, Crandall et al. also rely only on image processing information and do not perform any classification techniques[8].

In a more recent paper, Ephstein et al. introduce the Stroke Width Transform for detecting in-scene text [11]. The researchers present their work as a region-based text detection approach similar to connected components. The dataset used for training and testing our proposed algorithm, [1], was used by Ephstein et al., allowing for direct comparison.

Table 2.1: Overview of reviewed algorithms

Ref	**Segmentation**	**Features**	**Classifier**	**Post processing**	**Video**
[6]	Edge, Morphological	Gradients, DCT Coefficients	SVM, MLP	OCR	No
[29]	Edge, Projection Profiles	Raw Data	MLP	Temporal Heuristics	Yes
[52]	Wavelet, Morphological, Projection Profiles	Wavelet, Crossing Count Histogram	SVM	Multiscale Merging	No
[36]	DCT, Contrast, Morphological, Projection Profiles	–	–	Text Tracking	Yes
[28]	Color, Contrast, Motion, Heuristics	–	–	OCR	Yes
[8]	DCT, Morphological, Heuristics	–	–	Text Tracking, Temporal Heuristics	Yes
[11]	Canny Edge, Stroke Width Transform	–	–	Filtering, Heuristics	No

Table 2.1 provides an overview of several techniques reviewed. While it is difficult to generalize an algorithm into a few words, the table provides a guide for the segmentation, features chosen, classifier used with the chosen features, post processing run after classification, and if the algorithm is designed specifically for motion video.

Segmentation refers to the means by which each frame of video was divided into regions of interest which are likely to contain text. Edge detection, wavelet coefficients, discrete cosine transform coefficients, projection profiles, and other information is often combined with morphological image processing and some type of heuristics for this task. Morphological processing allows for the growing or shrinking of regions, many times with the goal of closing selected pixels into a solid shape, expanding the selected region.

Features used are specific to each paper and include gradient maps, DCT coefficients, wavelet coefficients, using the raw data itself, and using other customized features. Note that [8, 28, 36] did not explicitly use a feature set other than the binary result of segmentation. For their methods if a region of interest was segmented, it was classified as text. Also note that these papers did not explicitly train classifiers for the same reason, and the region was labeled text if it was segmented. Other authors used more conventional classifiers including neural networks and support vector machines to decide if a segmented region of interest was text or not based on the features provided.

In the post processing phase, one of three options were chosen for rejecting any false alarms that may have made it past classification. A third party OCR was used, the redundancy of temporal information was exploited, or in the special case of [52], multiple scales produced by the wavelet transform were merged.

2.1 Transform Functions

The discrete cosine transform, the wavelet transform, and the MPEG encoding process are often leveraged for text detection in video. To better understand these topics, brief background information is provided. This material is referenced from [14].

2.1.1 Discrete Cosine Transform

The discrete cosine transform (DCT) expresses a function as the sum of cosine functions of varying frequency. Many variations exist, however DCT-II is the most commonly used for image processing applications. DCT-II is shown in Equation 2.1 using the notation of [47] where $\alpha(k)$ is $\sqrt{1/2}$ for $k = 0$ and 1 otherwise.

$$X_{II}^{(N)}(k) = \sqrt{\frac{2}{N}}\alpha(k)\sum_{n=0}^{N-1} x(n)\cos\left(\frac{(2n+1)\,\pi k}{2N}\right), 0 \leq k \leq N-1 \tag{2.1}$$

In the equation, N is the total number of pixels in the sample window being processed, n is the index of a pixel in the spatial domain, and k is the index of a pixel in the DCT domain. This is the 1-D form of the equation.

A common use of the DCT is in lossy JPEG and MPEG compression. Typically, the DCT is computed over a finite size window. JPEG and some types of MPEG use an 8×8 window size, however any $N \times N$ window is valid. The goal of using a window is to find regions in the image which are nearly uniform and represent those with less data than regions contain more information. The DCT of a window with little variation contains many low coefficient values which can be set to zero and thus thrown away, while the DCT of a window with a significant amount of variation would not contain many low coefficient values and very little information is likely to be removed.

Fig 2.2 illustrates the effect of throwing away higher and higher values after taking the DCT with an 8×8 window. At first there is very little difference, however as more values are removed each of the entire 8×8 region becomes represented by their largest value, introducing strong blocking artifacts.

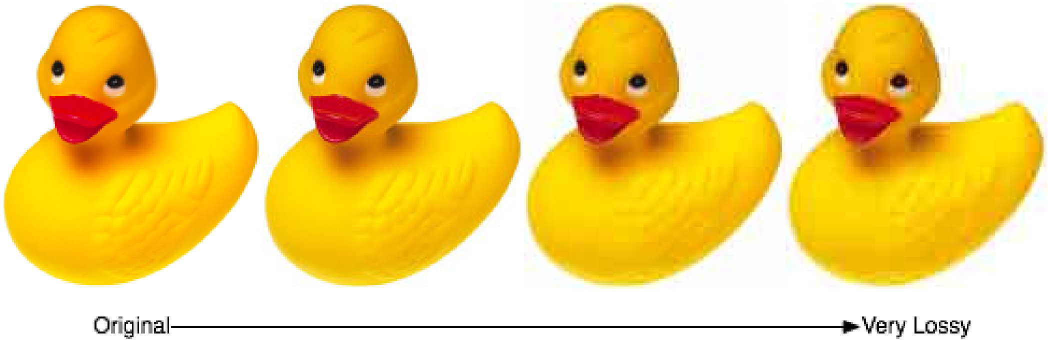


Figure 2.2: Visibility of the underlying 8×8 DCT increases as more coefficients are removed from each block.

Assuming compression has not been too severely imposed on an image or video, these DCT coefficients provide valuable information regarding frequency information in an image. DCT coefficients have been used directly by many researchers for text detection, however our approach uses the DCT to accomplish a different goal. As discussed in Chapter 3, we have chosen to use the DCT for frequency filtering instead.

2.1.2 Discrete Wavelet Transform

Similar to the Fourier transform which, leverages sinusoids as basis functions, the wavelet transform makes use of an alternate set of basis functions with which to represent information which may be more suited for a given problem. Both the continuous and the discrete wavelet transform (DWT) exist, however only the discrete transform will be discussed here due to its direct application to digital image processing.

The basis functions used are not specific to the transform, but rather are selected based on the problem. Common basis functions include the Haar basis functions, the Daubechies basis functions, the CDF 5/3, and the CDF 9/7 basis functions. As an example, the JPEG 2000 image compression standard uses the CDF 5/3 based wavelet for lossless compression and the CDF 9/7 based wavelet for lossy compression. Regardless of the specific set of basis functions used, two properties must be true. First, there are only two basis functions for any given set, and second, one acts as a low pass filter while the other acts as a high pass filter. The Haar and Daubechies basis functions are orthogonal, however this is not necessarily a requirement.

Given an input function and a set of basis functions, the discrete wavelet transform can be computed. The basis functions are applied as high and low pass filters to the original signal resulting in twice as much data out as data in. These outputs are downsampled by two, resulting in the same amount of data in as data out. The output are the coefficients which map the input function into wavelet space. This process is shown in Fig 2.3.

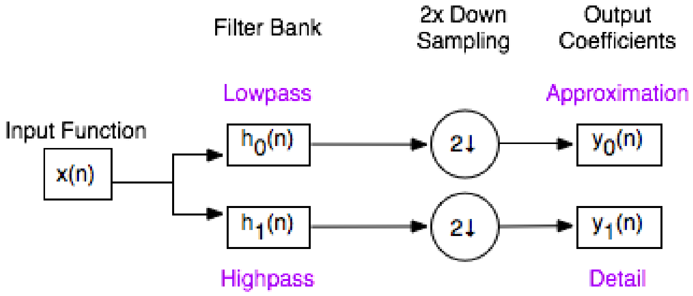


Figure 2.3: Block diagram of the DWT

While a single filtering pass is interesting and useful, wavelets become far more useful through iteration. Given a lowpass, downsampled image, the same highpass and lowpass filters can again be applied and the results downsampled by two. Since in the practical case, the input data is finite, this iteration can occur until only a single element of data remains or until a threshold is reached. Thus by recursively applying the discrete wavelet transform, we produce multiresolution imagery, or imagery which contains several different scales. This is useful when incorporating scale invariance into an algorithm, or when an application may require several different scales of data. The recursive discrete wavelet transform is diagramed in Fig. 2.4.

Regarding computational complexity, the discrete wavelet transform can be implemented as the fast wavelet transform, which has linear complexity. Regarding the inverse transform, assuming the coefficients have not been altered, the original function can be perfectly reconstructed.

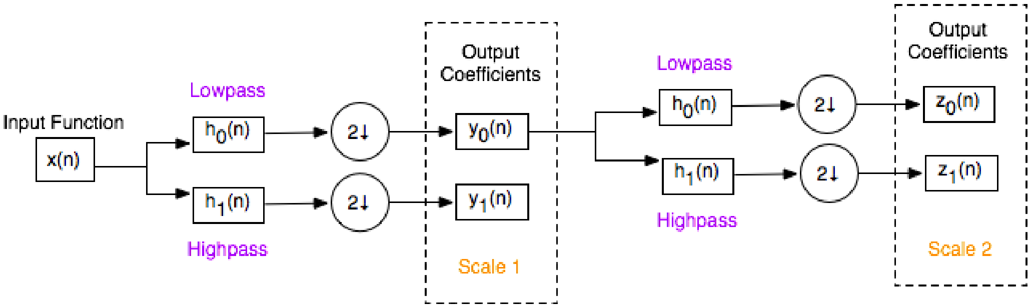


Figure 2.4: Block diagram of two scales of the DWT

2.2 Segmentation

Given a sequence of video frames, potential regions of interest must be segmented. Methods of segmentation include edge detection, color and contrast segmentation, thresholding discrete cosine transform coefficients, and thresholding wavelet transform coefficients.

2.2.1 Edge Detection

The Canny edge detector is the most widely used edge detection algorithm among the text detection algorithms reviewed here, and is used in our algorithm. This standard image processing algorithm is described in full detail in [45]. The algorithm is optimal according to the following three criterion: detection, localization, and one response. That is the detector seeks to not miss important edges, it seeks to minimize the distance between the resulting line and the actual edge, and it seeks to produce a single line per edge. This is accomplished by first slightly blurring the image with a Gaussian smoothing filter. Next local normal directions are calculated and used to locate edges using non-maximal suppression. The magnitude or strength of the edges are computed and thresholded using a high and low threshold. If desired, the process can be repeated with different Gaussian filters and combined using feature synthesis.

For their technique, Chen et al. in [6] extract regions of interest from each frame of video based on vertical and horizontal edges produced by a Canny edge filter. Dilation of the extracted edges is performed using empirically derived rectangle kernels of size 1×5 for the vertical direction and 6×3 for the horizontal direction. When comparing the regions selected to ground truth text regions, assuming a region is correctly located if there is an 80% overlap between the two, this method extracted 9,369 text lines but produced 7,537 false alarms from the test data. This results in a precision of 55.4%. Regions of interest that were selected can be fed into a classifier for further refinement.

Chen et al. further refine their text selection by using a baseline detection algorithm previously developed by Chen, Bourlard, and Thiran[5]. Additionally, final selections are constrained to be between 75 and 9,000 pixels in area, with a horizontal-vertical aspect ratio of 1.2 and a height between 8 and 35 pixels. No reason is given for these criteria, however the authors do note that text can vary greatly in size and a scaled image pyramid, or simply a series of images down sampled by a factor of two, could allow this algorithm to work on larger text regions.

2.2.2 Color and Contrast Segmentation

Digital images are typically captured and stored in the RGB color space. In this space, quantitative color differences have little meaning in terms of human color perception. Instead, it is advantageous to make use of the CIELAB 1976 color space where we are able to compute such differences meaningfully. It is not possible to directly convert RGB to CIELAB, however by making assumptions about illumination, the observer, and the calibration of the imaging system, we are able to make a good approximation. Given these assumptions, the just noticeable difference, that is the difference between two colors required for the standard observer to notice the difference, is between 2 and 3. Further information on CIELAB 1976 and other color spaces is given in [2].

Lienhart and Wernicke use region growing to create bounding boxes for the selected text regions. Profile projections in both horizontal and vertical directions were used in an iterative process to further refine bounding boxes by merging overlapping regions or deleting some selections which were under threshold intensity values. The number of iterations as well as the thresholds used for bounding boxes were determined empirically. Color information was used to group text and remove as many background pixels as possible.

To segment individual characters and remove the background, text selections are first scaled to have a height of 100 pixels by using a bilinear interpolation. Next pixels of the same color as those just past the bounding box are removed from within the bounding box, as they are assumed to be background. For video, temporal information is used to assist in the background removal process as it is assumed the text changes very little from frame to frame, while the background can change significantly, assuming text is superimposed and not in-scene.

Similar to the approach of Lienhart and Wernicke, our technique makes use of color, however, rather than actively grouping regions, we discount regions which contain too large a color difference. To accomplish this, we make use of the CIELAB color space. Specific transformations and assumptions used for our method are detailed in Chapter 3.

2.2.3 Discrete Cosine Transform Coefficients

Crandall et al. in [8] perform connected component analysis on detected text blocks followed by a procedure similar to morphological dilation. Blocks which are larger than 8 pixels in width and length are kept as localized regions of text.

Crandall et al. perform text detection by analyzing texture features provided by the Discrete Cosine Transform (DCT). An 8×8 block-wise DCT is performed on each video frame, producing a set of DCT coefficients. A specific, empirically determined subset of these coefficients, are used to characterize "text energy" in the horizontal and vertical directions. A dynamically chosen threshold, driven by an empirically derived formula based on the contrast of the entire video, is used to determine whether or not each block contains text. Since an 8×8 block size is used, subsampling is performed to detect text of different sizes. The author cites [10] for an efficient way of subsampling in the DCT domain.

2.2.4 Wavelets for Segmentation

Ye et al. compute the wavelet transform of each video frame using the Daubichie basis functions. Wavelet coefficients are used with an energy function as a feature to locate potential text regions in the coarse detection step of this algorithm. Text is assumed to produce large wavelet coefficients and therefore have a higher energy than non-text regions. A dynamic threshold based on the energy histogram of each image is used to select which energy value is sufficient for text detection. The morphological close operation is used to combine nearby pixels of text into clusters. Projected profiles are then used to separate individual lines. Finally, any clusters smaller than eight pixels in height, or with a width-to-height ratio of less than one, are discarded.

2.3 Features

2.3.1 Wavelet Features

For Ye et al. fine detection is accomplished using four collections of wavelet features and a support vector machine classifier. Features used include wavelet moment; histogram; co-occurrence features including energy, entropy, inertia, local homogeneity, and correlation; and crossing count histogram. A total of 225 features from each line of text are generated using this information. Since a large number of features may hinder the overall performance of a classifier, a forward search algorithm was used to reduce the number of features used to just 32. Of the original features, wavelet moment and the crossing count histogram appear to be the most important based on the results of the forward search.

Large text produces a relatively low frequency signal, while small text produces a relatively high frequency signal. In the wavelet domain, low frequency information produces a stronger response in a deep scale, after only a few iterations of the wavelet transform, while high frequency information produces a stronger response in a shallow scale, after many iterations. These two responses will be similar, thus multiscale text detection is possible by choosing different scales from which to extract features.

2.3.2 Other Features

Other researchers who used the DCT coefficients for classification or the raw image intensity values did no further processing of this information before training a classifier. Half of the works reviewed did not explicitly use a classifier, instead focusing on the segmentation task, assuming that good segmentation eliminates the need for further classification.

2.4 Classification of Regions

2.4.1 Support Vector Machine

Chen et al. chose to verify text regions by making use of a support vector machine (SVM) trained on four feature types. Grayscale spatial derivatives, a distance map, constant gradient variance, and discrete cosine transform coefficients are the features used. Constant gradient variance was developed by the authors but is fairly straightforward. Given a nine by nine region of pixels, a local mean and variance are computed. This local information is then compared with a global gradient variance for the whole image, and the gradient magnitude for a given pixel. In this way, changes in contrast can be exploited as a feature.

A confidence threshold was set for determining what was text and what was not based upon the four features used. On the training data, the SVM using the constant gradient feature had an error rate of 1.07%. For the grayscale spatial derivatives the error was 3.99%, for the distance map, 2.56%, and for the discrete cosine transform, 2.00%. Using the same test data previously used by Chen et al. in Section 2.2.1, 7255 of the 7537 false alarms were removed by classification, and 23 true text lines were removed. Overall the authors report that this produces a 97% precision rate and a 0.24% rejection rate.

Ye et al. also trained an SVM, using their selected wavelet features with boosting, to improve performance. Specific details about the kernel used are not provided by the authors.

2.4.2 Neural Network

Lienhart and Wernicke employ gradient maps as features which are fed into a neural network. Image gradients, or gradient maps are the two-dimensional derivative of the image. They a complex-valued feed-forward neural network containing 200 total neurons arranged in one input layer and two hidden layers. Training was conducted on a set of data containing 6,000 text patterns and 5,000 non-text patterns. A validation set was used to further tune the network before testing.

2.5 Post Processing

2.5.1 Binarization

The binarization performed by Crandall et al. is composed of three steps: preprocessing, logical level thresholding, and character candidate filtering. Preprocessing rotates any non-horizontal localized text regions into the horizontal orientation, and doubles the resolution via a linear interpolation. Contrast is enhanced by histogram equalization, and ten frames are used with a rigid text tracker for temporal averaging. Logical level thresholding is applied to the image after a conversion to CIELAB color space, using only the luminance channel, with an empirically determined threshold value. This color space consists of light-dark, red-green, and yellow-blue opponent channels and seeks to be perceptually accurate to the average human observer. Standards have been published which allow for the conversion from the more familiar

RGB (red, green, blue) color space into CIELAB space. This step is applied to the original image, and its inverse, to account for the text being lighter or darker than the background. Once again, connected component analysis is performed and any characters that do not meet certain criteria set by the authors are discarded. A voting strategy is used to determine if the original or inverse image should be kept. Taken into account for each character are height similarity, width similarity, horizontal alignment, aspect ratio, clean spacing, and periodicity of vertical projection.

Lienhart and Wernicke perform binarization by using a simple threshold which is halfway between the average background and foreground color.

2.5.2 Temporal Heuristics

For video, Lienhart and Wernicke exploit temporal redundancy to help remove false alarms, reduce noise, and improve text segmentation accuracy. In order to reduce complexity, video was sampled every second, and if text was found each frame available from one second before and one second after was then analyzed for text content. Profile projections were used as signatures for text, allowing text to be tracked from frame to frame assuming very little variation in the text from frame to frame. Due to noise, compression artifacts, and other issues it is difficult to track text from frame to frame perfectly. Thus a dropout threshold is used to allow tracking to continue even if a few frames are skipped. Text which occurs for less than a second or is present in less than

25% of the frames in a sequence is ignored.

2.5.3 Text Tracking

Crandall et al assume text tracking is rigid; that is, text moves with a constant velocity and in a linear path. Motion vectors provided by the MPEG compression process are used to predict text motion [34, 35]. By using only macroblocks with more than four edge pixels, as computed by the Sobel edge detector, motion vectors can be more useful over short video sequences. The authors note that MPEG motion vectors are computationally efficient as the work has already been done, however they are often too noisy to use directly as a tracker, depending on the quality of MPEG encoding used. A comparative technique is used to augment motion vectors for the text tracking task. Frame-by-frame comparisons of connected components are used with a threshold to determine if a localized text region exists in multiple frames.

2.6 Published Results

Two standard datasets for word level text detection, including standard metrics to use have been published [18, 46], however the number of researchers using these and publishing results is limited. No known dataset has been published which focuses on pixel level text detection, as we are working with. Whether these data or self-created data are used, it is important to be aware of limitations and potential issues. If the task of a researcher was to draw a bounding box around the text instances in several thousand frames of video, it would not be unreasonable to expect variation in where the box is placed around the text and other similar errors. Likewise the algorithm may have successfully found the text but the box it draws is too large or shifted or skewed in some way compared with the ground truth. To account for these issues, a threshold for percent overlap can be set to decide when two regions are "close enough" to be called the same. What this value depends on is a researcher's personal preference. Alternately, boxes could only be counted when they exactly overlap, as done by [8], however, it seems that exact overlap will push scores lower than they need to be.

Other issues in comparing results include a lack of common data, and different sizes of the sets of data used. Some groups chose thousands of frames of video over which to score their techniques, while others chose only a small number of frames. To properly compare these methods, each would need to run on the same data using the same percent overlap of bounding box when reporting correct results. Nevertheless, since we do not have the ideal conditions, results are reported here as they have been reported by their respective researchers.

Three metrics are commonly used to report results: recall, precision, and false alarm rate. Recall, shown in equation 2.2, is simply the number of correct detects over the total number of targets in the ground truth. A perfect score for Recall is 100%. Precision, shown in equation 2.3, is the number of correct detects over the total number of detects reported by the algorithm. Ideally this would also be 100%. Finally false alarm rate is the number of false alarms over the number of correct detections, shown in equation 2.5. Ideally this would be zero. It is also common to combine recall and precision into a single metric called the f-metric or f-measure, shown in equation 2.4.

$$\text{Recall} = \frac{\text{correct detects}}{\text{correct detects} + \text{missed detects}} \tag{2.2}$$

$$\text{Precision} = \frac{\text{correct detects}}{\text{correct detects} + \text{false alarms}} \tag{2.3}$$

$$f\text{-metric} = \frac{1}{\frac{0.5}{\text{precision}} + \frac{0.5}{\text{recall}}} \tag{2.4}$$

Epshtein et al. using the same ICDAR dataset which we use reported precision of 73%, recall of 60%, and an f-metric of 0.66%.

Crandall et al. tested their algorithm on two datasets of MPEG videos totaling over 11,000 frames at a 320×240 pixel resolution. Precision and recall were computed. On the first dataset, the proposed algorithm achieved a recall of 46% and a precision of 48% for detection and localization of caption text. On the second dataset, for the same task, a precision of 74% and a recall of 74% were achieved.

Ye et al. performed experimental testing on a dataset of 221 images, each of size 400×328 pixels. Ground truth was marked by hand, and a detection was marked successful if more than 95% of the ground truth box was contained in more than 75% of the detected rectangle. Based on this definition of a correct detection, recall (Equation 2.2) and false alarm rate (Equation 2.5) were computed.

$$\text{False Alarm Rate} = \frac{\text{False Alarms}}{\text{Correct Detections}} \tag{2.5}$$

Based on these metrics, the authors report a recall rate of 94.2% and a false alarm rate of 2.4% for their testing dataset.

Lienhart and Wernicke use a relatively small testing set of only 23 videos. Videos ranged in size from 352×240 to 1920×1280. If the detected bounding box overlapped the ground truth by 80% a detection was labeled correct. For individual images, a 69.5% hit rate, 76.5% false hit rate, and a 30.5% miss rate was found. For video, significant improvements were seen with a 94.7% hit rate, 18.0% false hit rate, and a 5.3% miss rate.

2.7 Summary

Surveyed work focused on edge information, frequency information, and specific properties about text. Edge information may be extracted using the Sobel operator, Canny edge detector, or other methods. Frequency information is captured using the DCT, Fourier transform, or Wavelet transform. Commonly color, aspect ratio, and other properties of text were also exploited. Researchers used simple segmentation only techniques, complex neural networks, and combinations of these and other approaches to classify text in images.

Similar to previous work, our algorithm makes use of frequency based features for classification and edge and color information for post processing. We learn a weighted ensemble of classifiers constructed from the raw feature data. A small number of statistically determined parameters are applied in post-processing. The post-processing step makes use of the Canny edge detector, color consistency, and aspect ratio. Unlike other definitions of aspect ratio, we use the ratio of eigenvalues of the covariance matrix of the coordinates of each connected component. Our approach is explained in detail in Chapter 3.

Chapter 3

Methodology

Using themes from related work, we selected to use frequency based features captured using the discrete cosine transform, edge information captured using the Canny edge detector, an aspect ratio threshold, and a color consistency threshold.

3.1 Dataset

For the original task of text detection in video, we were unable to locate a dataset of video with ground truth. However, since we have simplified the problem and are only dealing with still images, two datasets were acquired. The first, published by the Linguistic Data Consortium and promoted by Kasturi et al. [18], featured keyframes of videos, mainly of news broadcasts. These were useful, however most of the text was not in-scene text, but rather superimposed text. Further, this dataset is not as diverse as is desired, potentially causing problems for the learning algorithm.

The second dataset we were able to obtain is available free of charge from ICDAR [39, 46]. This set features images taken by many different authors, using different cameras in a variety of illumination conditions, and in different resolutions. In total there are 500 images, 250 set aside as testing and 250 as training. This data has been used by other researchers, including Epshtein et al. in [11], allowing for easy comparison for the word level task.

Throughout this chapter an image from the training dataset which we have titled “Osborne Garages” will serve as an example with which each step of the algorithm is illustrated. The original image is shown in Figure 3.1.

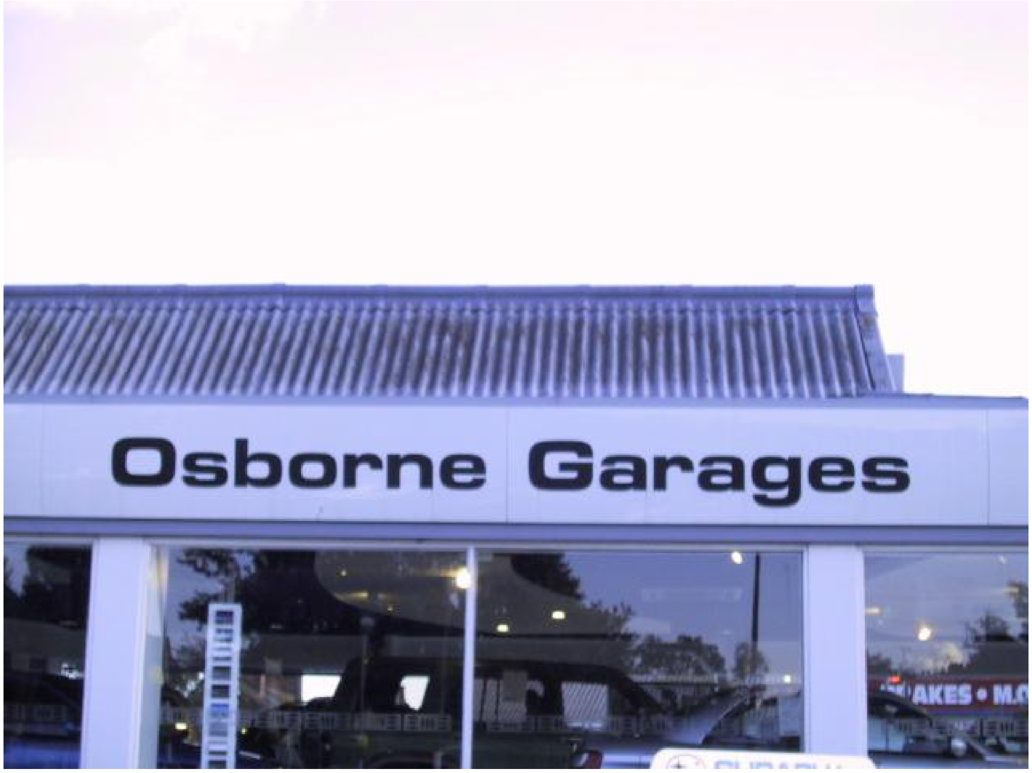


Figure 3.1: Example image from the training data in its original form.

3.1.1 Ground Truth

Since word level ground truth regions often contain a large amount of background with the desired foreground text regions, it was determined that more accurate ground truth was needed to enhance the performance of the learning algorithm. Originally, working with word level ground truth, we found the presence of background introduced a significant amount of error in learning which was reduced by opting to use pixel level ground truth instead. Additionally, creation of pixel level ground truth allows us to score the performance of our algorithm on the pixel level. This fine-grained pixel level of detail is a unique contribution to this area of research and has not been previously seen in the literature on the subject.

Creating true pixel level ground truth for the data is not practical due to time constraints. Instead a semi-automated approach was taken which reduced the amount of manual labor needed for ground truth creation. Color quantization was applied to training images, reducing them from millions of colors to only 16 colors. Each image was converted to the CIELAB 76 color space. A histogram of colors was created, sorting colors from most popular to least popular. Starting with the most popular color, the next most popular color was selected which has a color difference greater than the just noticeable difference of 2.5. Once 16 colors have been selected which are both popular and noticeably different, these colors are assigned to existing colors in the image such that the color difference between the original color and the color in the palette is minimized.

These color quantized training images are masked using the word level ground truth provided. The resulting boxed regions are shown one at a time to the person creating pixel level ground truth. The user has the ability to turn on and off each of the colors and attempts to select colors which contain foreground, while removing colors containing background. In this way pixel level ground truth is created, without manually selecting each pixel in the text of each image. Original bounding box level ground truth is compared with our pixel level ground truth in Fig. 3.2. Our ground truth will be made available on our lab website: `http://www.cs.rit.edu/~dprl`.

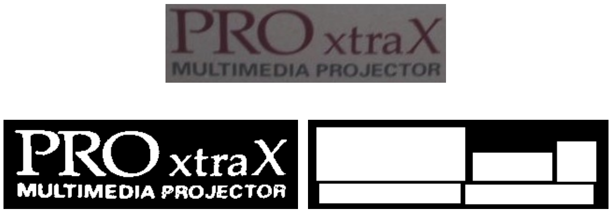


Figure 3.2: Top: Original image. Bottom Left: Pixel level ground truth. Bottom Right: Word level ground truth.

For the “Osborne Garages” example, pixel level ground truth is shown in figure 3.3. Note that the text in the lower right corner of the image is not in the ground truth. This was not in the word level ground truth, so it has not been propagated into the pixel level ground truth either.

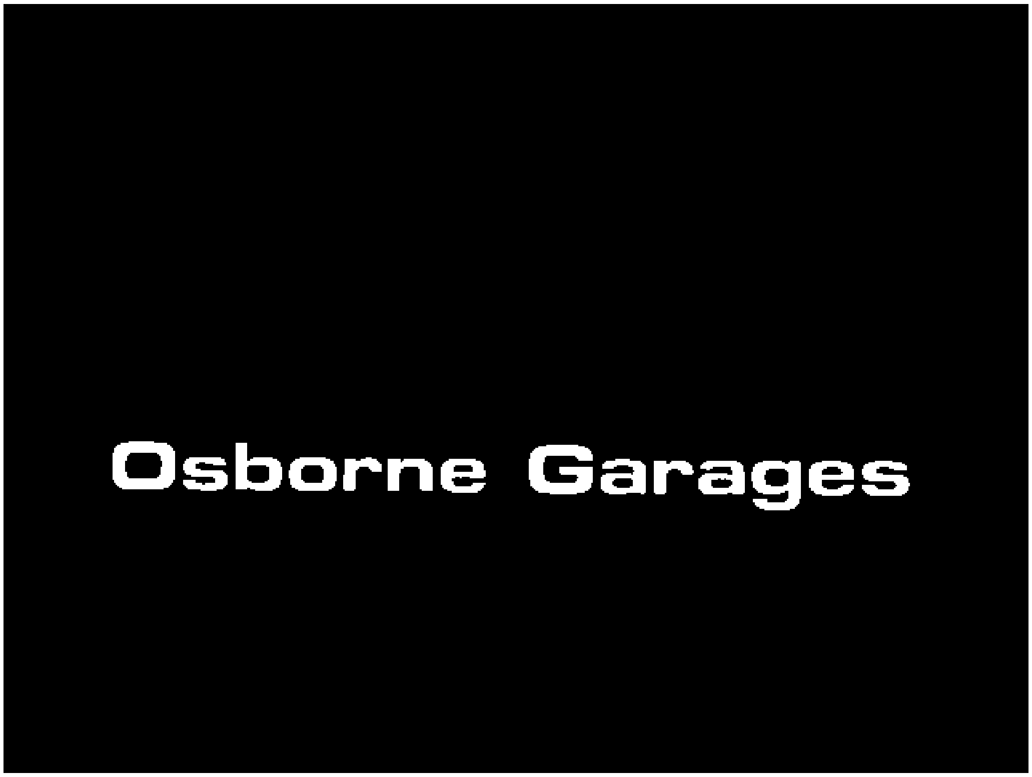


Figure 3.3: Pixel level ground truth for our example image.

3.2 Algorithm Overview

The approach taken makes use DCT frequency filters, edge detection, a linearity filter, and a color consistency filter. Figure 3.4 illustrates the process. Each step is described in detail below. After training has been completed, the algorithm is slightly modified to produce binary classification masks, as shown in figure 3.6.

3.2.1 Feature Selection

Consistent with previous work, we have chosen four features with which to perform text detection. The discrete cosine transform (DCT) is used in conjunction with frequency filters and the Weighted Majority (WM) learning algorithm to classify text on the pixel level. The Canny edge detector is used independently of the DCT to perform segmentation, followed by morphological filling of connected components. Our linearity and color features serve as constraints to the previous two; preventing anything which is too linear or not color consistent from being selected.

Spatial frequency information captured by the DCT is used as our primary feature; and it is the only feature on which the WM learning algorithm is used. Given an image, it is first converted to grayscale by converting from RGB to the YCC color space, and retaining only the intensity (Y) channel. The result of converting our example image into grayscale using this method is shown in figure 3.7. The image is then segmented into 8×8 blocks on

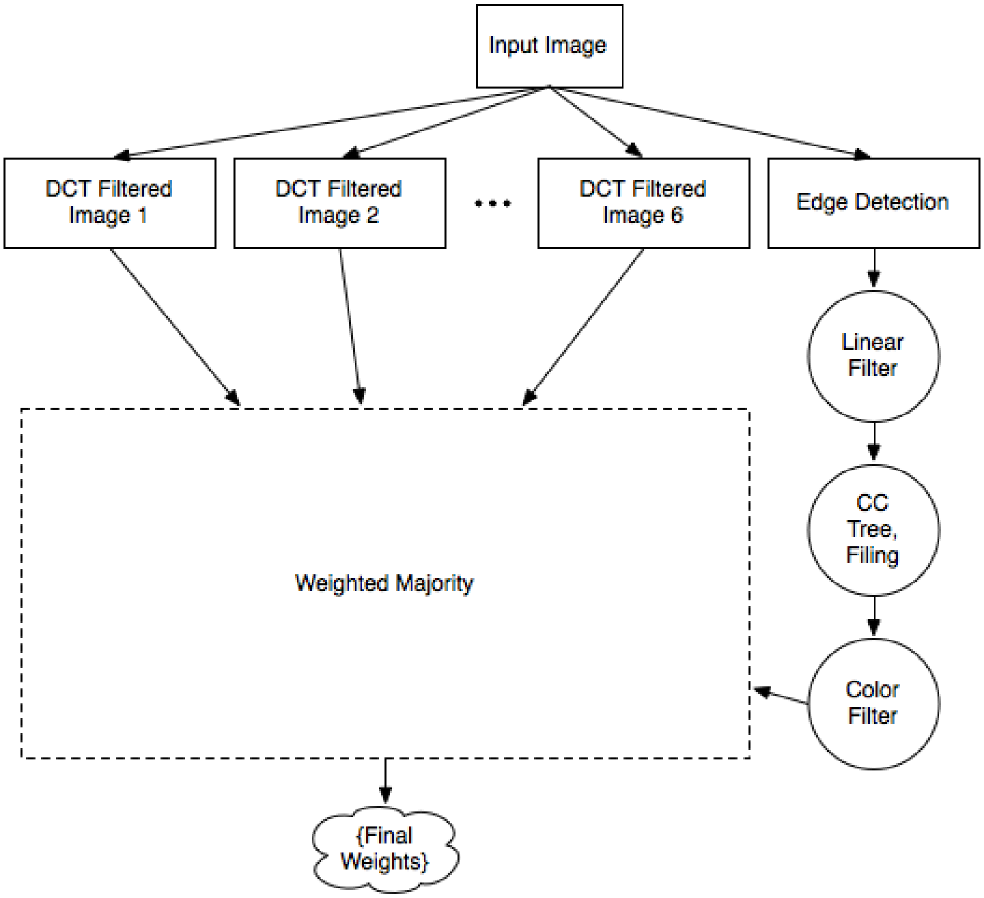


Figure 3.4: Training algorithm. High pass filtered images are created from a grayscale input image. These are thresholded to create 18 total classifiers which are weighted by WM. Post processing is carried out in parallel and used within the WM step. WM is detailed in figure 3.5.

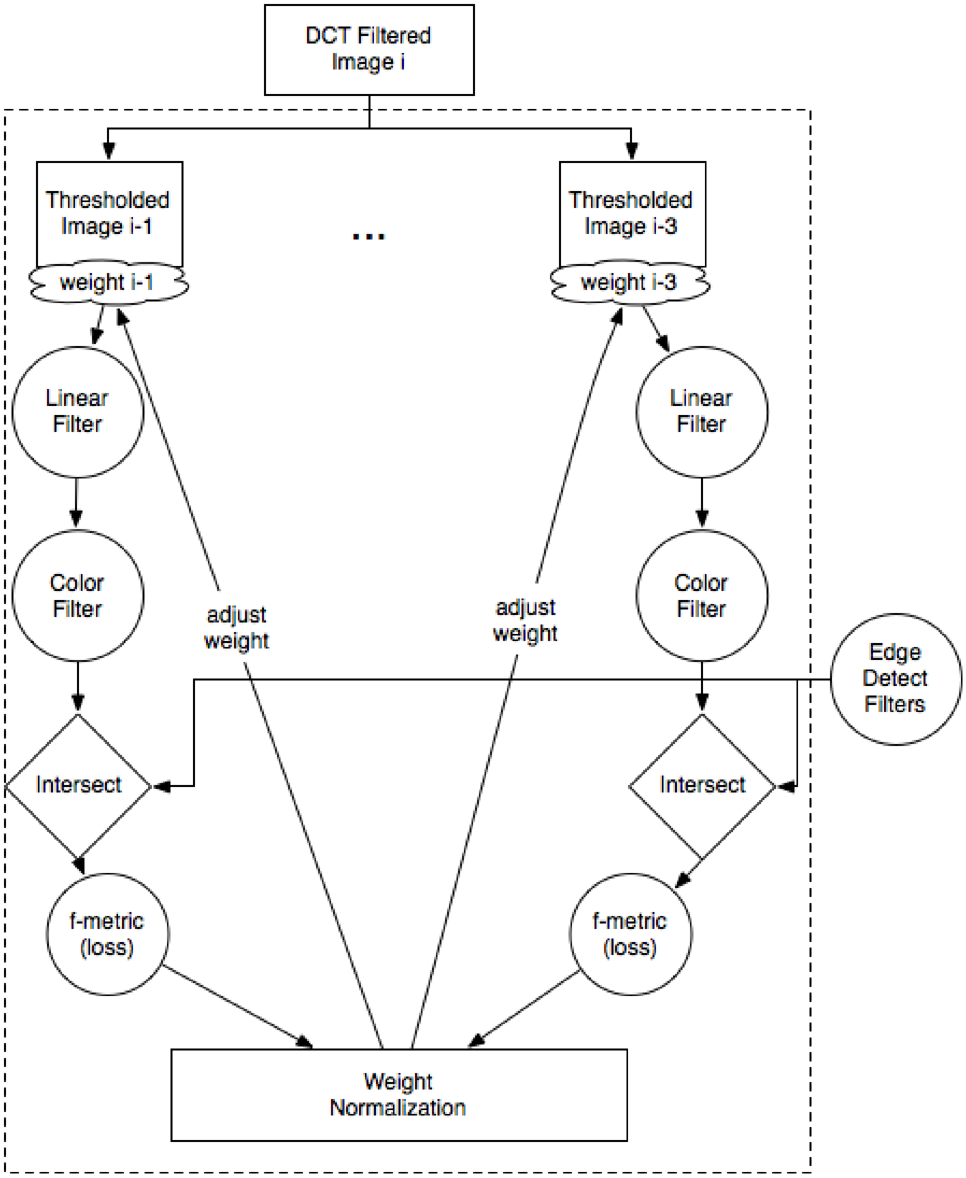


Figure 3.5: Training algorithm. Each highpass filtered image is thresholded three times to create a total of 18 classifiers (6 filters, 3 thresholds each). Weights for these classifiers are initially equal. Loss is computed and weights are normalized. Successive runs start with the previously calculated weight value for each classifier and continue updating weights until there are no additional input images. We have selected to randomly shuffle the input images 10 times and continue WM to allow for additional training.

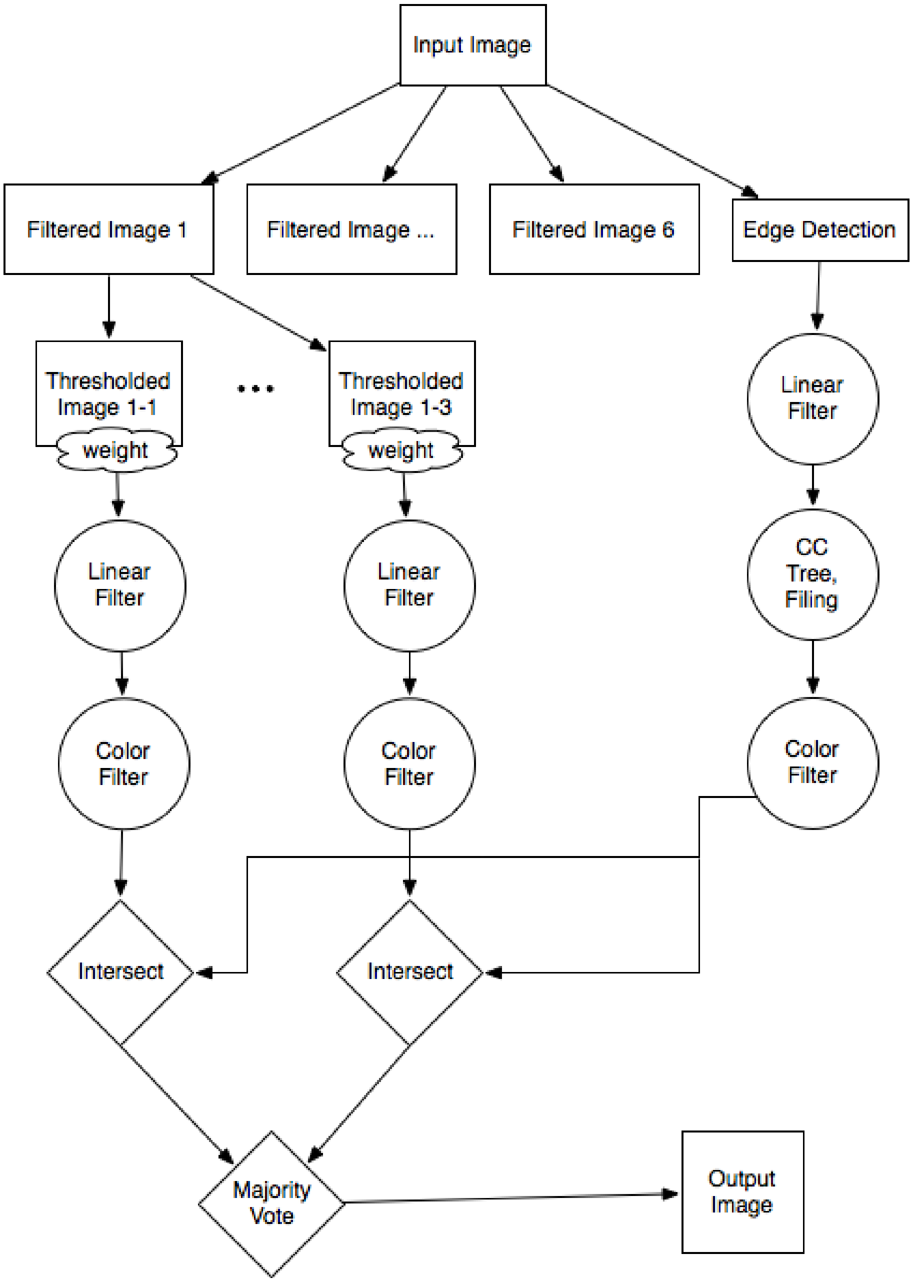


Figure 3.6: Classification algorithm. Weights computed using weighted majority are applied to each classifier. Weighted classifiers are summed and thresholded at 0.5; retaining pixels which have received a majority vote.

which DCT-II is computed. This block size was selected as it is the standard block size for JPEG encoding. Since not all images are divisible by 8 in both directions, edges are zero padded when needed.

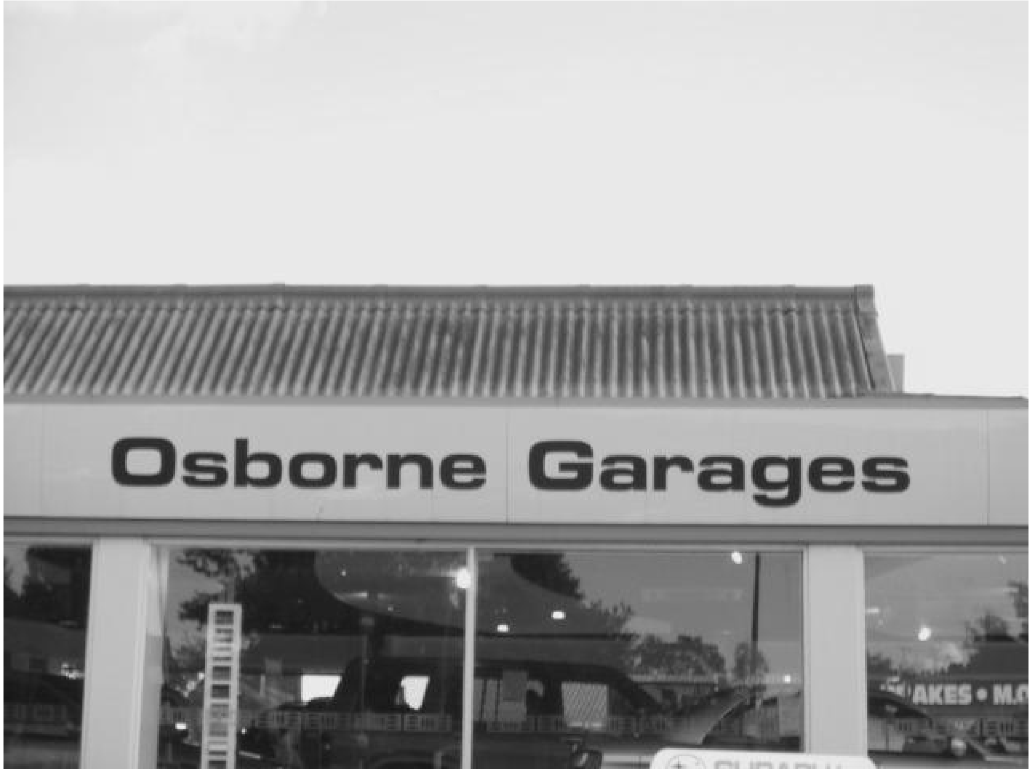


Figure 3.7: Y channel of our example image converted from RGB to YCC.

Six 8×8 Gaussian highpass filters with $\sigma = 0.5, 1.0, 1.5, 2.0, 2.5, 3.0$ have been selected for frequency filtering. Since we are working with 8×8 windows, these filters have been selected as they are each able to provide useful information without being more redundant than is desired. This is determined by observing the discrete filter values exhibit very little change when increasing σ past $\sigma = 3.0$. These 2D filters are shown in figure 3.8.

These filters are applied independent of each other, resulting in 6 feature images produced for every one input image. Frequency filtering is done directly in the DCT domain using the technique of Viswanath et al. described in [47]. For consistency with this work, we use Viswanath's notation here. As the DCT is 2D separable, we are able to work with 1D equations. First, 8×8 windows of the original image are computed using the type-II DCT as described previously, and as is expressed in Equation 3.1. In the Equation, $\alpha(k) = \sqrt{1/2}$ for $k = 0$ and 1 otherwise.

$$X_{II}^{(N)}(k) = \sqrt{\frac{2}{N}}\alpha(k)\sum_{n=0}^{N-1} x(n)\cos\left(\frac{(2n+1)\,\pi k}{2N}\right), 0 \le k \le N-1 \tag{3.1}$$

Next, the type-I DCT is used to transform each filter into DCT space. This version of the DCT is expressed in Equation 3.2, where $\beta(k) = 1/2$ for $k = 0$ and $k = N$ and is 1 otherwise.

$$H_{I}^{(N)}(k) = \sqrt{\frac{2}{N}}\beta(k)\sum_{n=0}^{N} h(n)\cos\left(\frac{n\pi k}{N}\right), 0 \le k \le N \tag{3.2}$$

These coefficients are then rearranged into a matrix as shown in Equation 3.3.

$$D^{(N,N)} = \begin{bmatrix} H_I^{(N)}(0) & 0 & 0 & \cdots & 0 \\ 0 & H_I^{(N)}(1) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & H_I^{(N)}(N-1) \end{bmatrix} \tag{3.3}$$

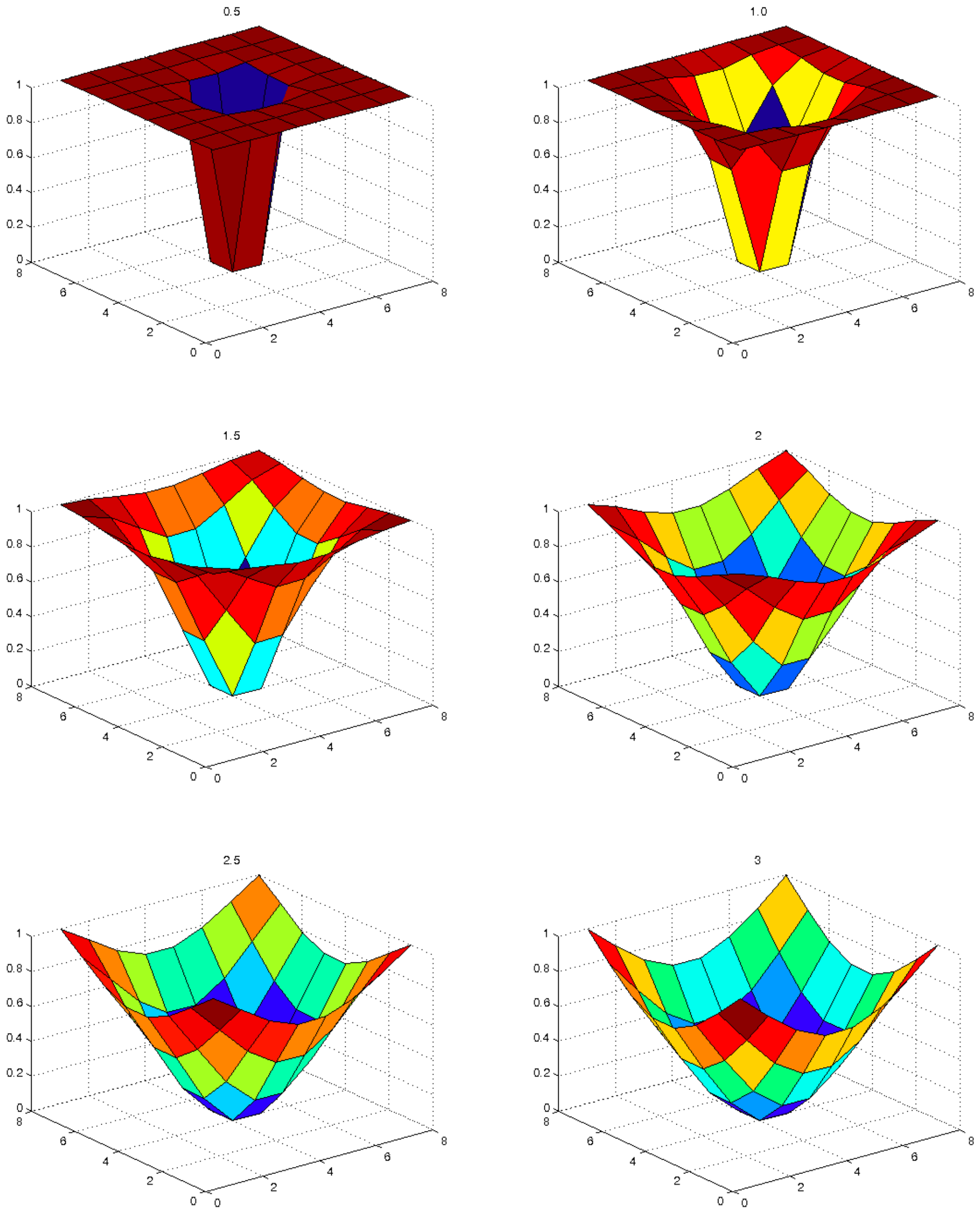


Figure 3.8: Highpass Gaussian filters. Top left to lower right, σ increases from 0.5 to 3.0 in increments of 0.5.

Filtering in the DCT domain can then be expressed as:

$$Y_i^{(N)} = D^{(N,N)} X_i^{(N)} \tag{3.4}$$

Where in equation 3.4, $X_i^{(N)}$ and $Y_i^{(N)}$ are the input and output of type-II DCT blocks.

Continuing our example, the result of filtering the grayscale version of the "Osborne Garages" image with a kernel where $\sigma = 0.5$ is shown in figure 3.9.

3.2.2 Creation of Classifier Ensembles

Highpass frequency filtering of grayscale images produces real-valued outputs which is normalized to be in the domain [0,1]. To function as a classifier, these filters need to be binarized, creating masks such that the pixel value 1 indicates text and pixel value 0 indicates background. To find the threshold, training set images were thresholded from in the range [0,1] at increments of 0.1. The mean and standard deviation of the f-metric score was calculated across the training set data and plotted as shown in figure 3.10. Note that the threshold value 0 indicates all pixels in the image have been selected. From the figure it is apparent that the first three threshold values produce a better result than including all pixels in the image, but that additional thresholds beyond the first three perform worse than the zero threshold. For this reason, the first three thresholds have been selected from

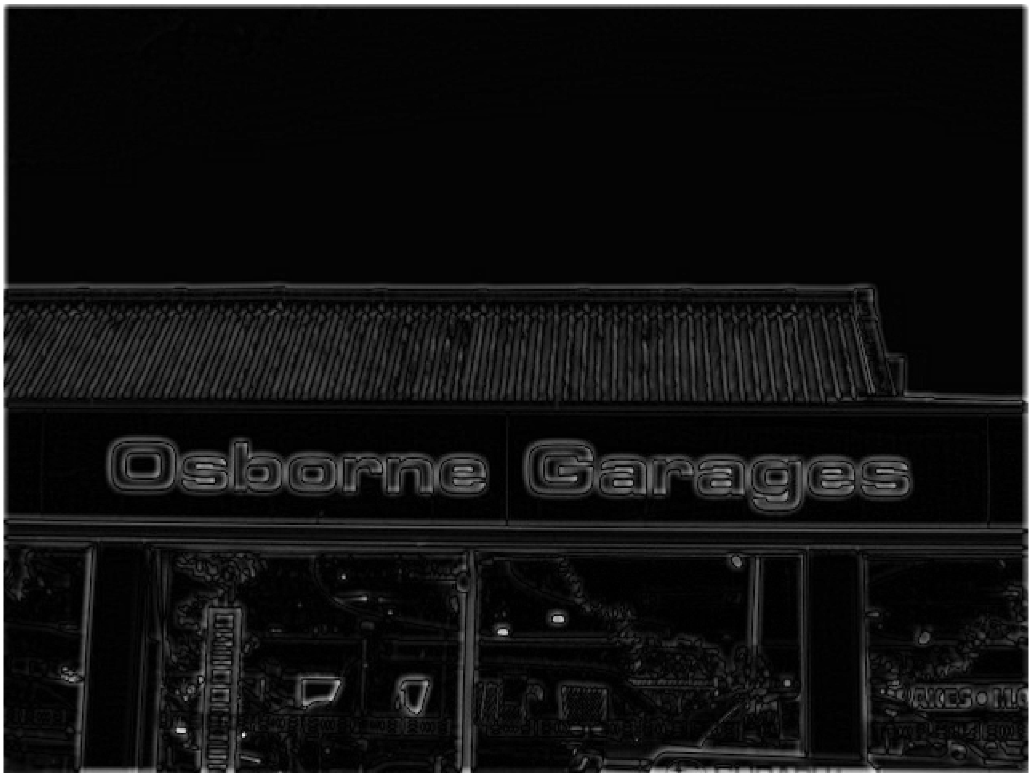


Figure 3.9: Gaussian highpass filter with $\sigma = 0.5$ applied to our example image.

which to create binary classification masks from the DCT filters, resulting in a total of 18 classifiers being created. That is for each of the 6 DCT filters, 3 threshold are applied to each filter, for a total of 18 classifiers.

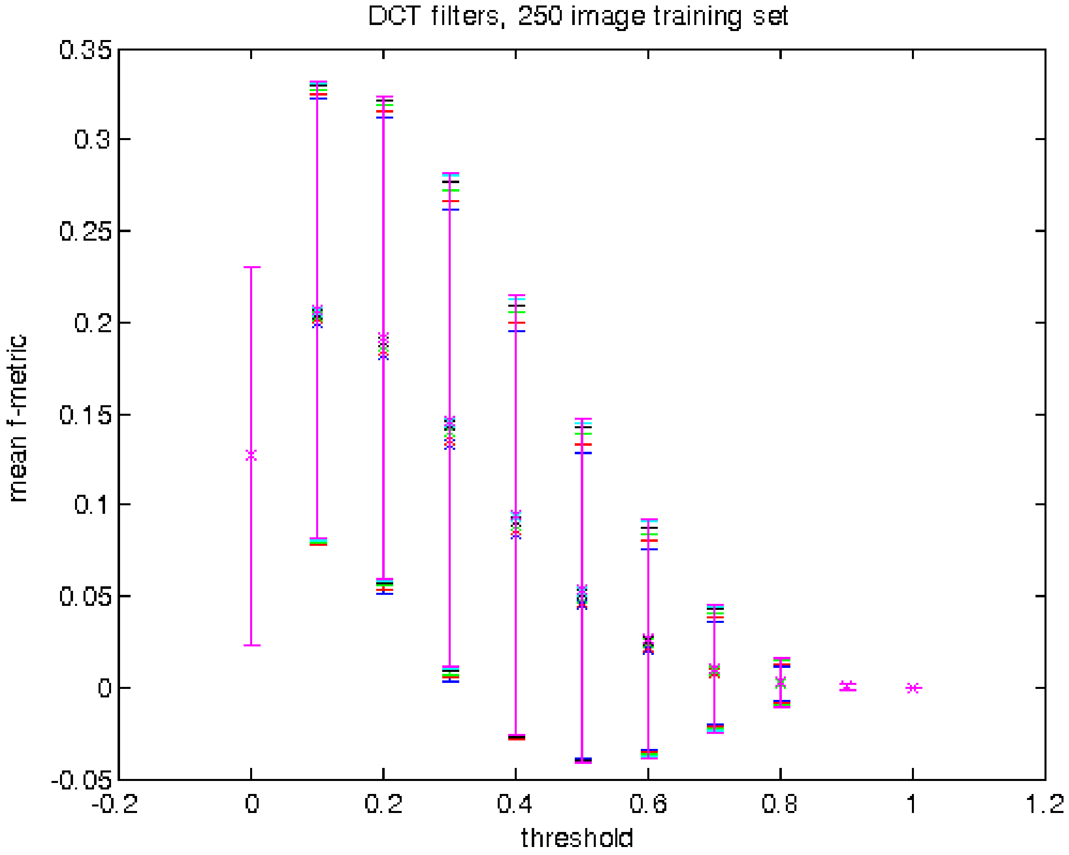


Figure 3.10: f-metric mean scores for all filters across the training set. Each of the 6 filters are represented by a different color.

Using the filtered image from figure 3.9 and applying three thresholds results in the three classifiers shown in figures 3.11-3.13.

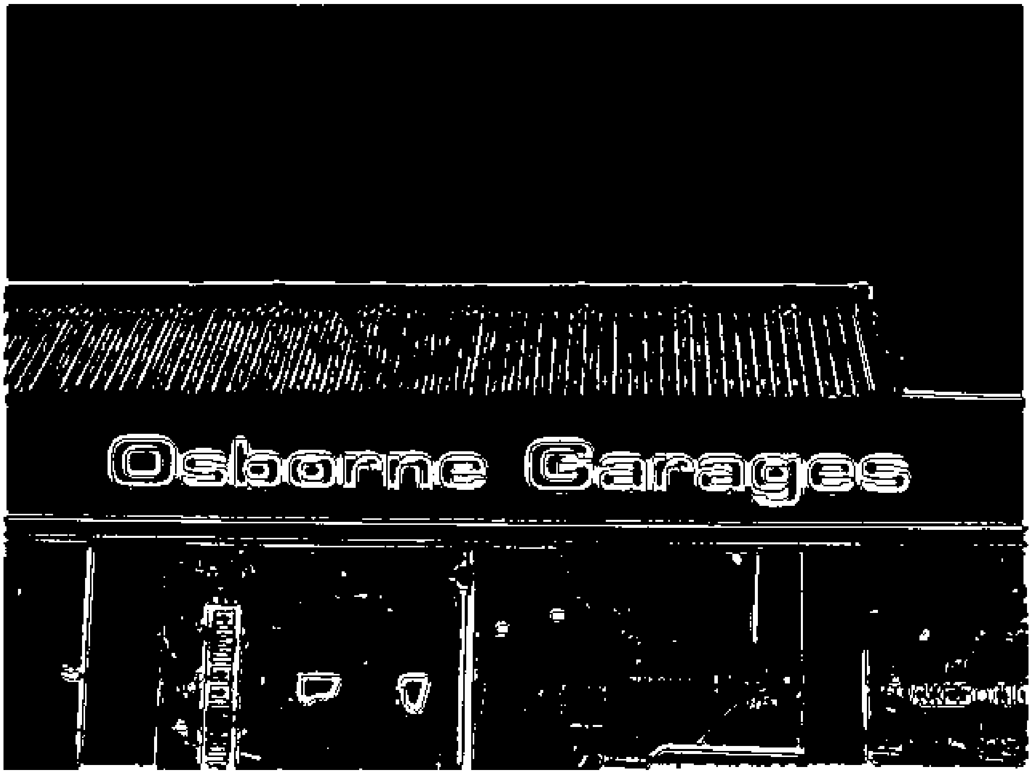


Figure 3.11: Applying three thresholds, we create three classifiers from one filtered image. This is the first of three thresholds.

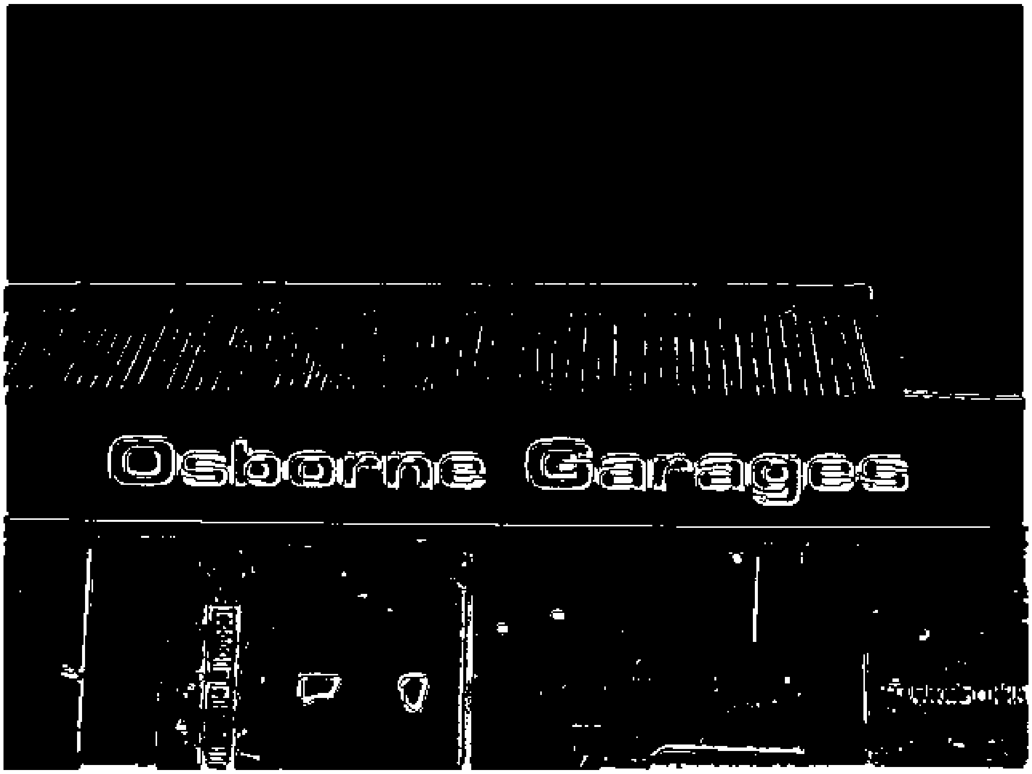


Figure 3.12: Applying three thresholds, we create three classifiers from one filtered image. This is the second of three thresholds.

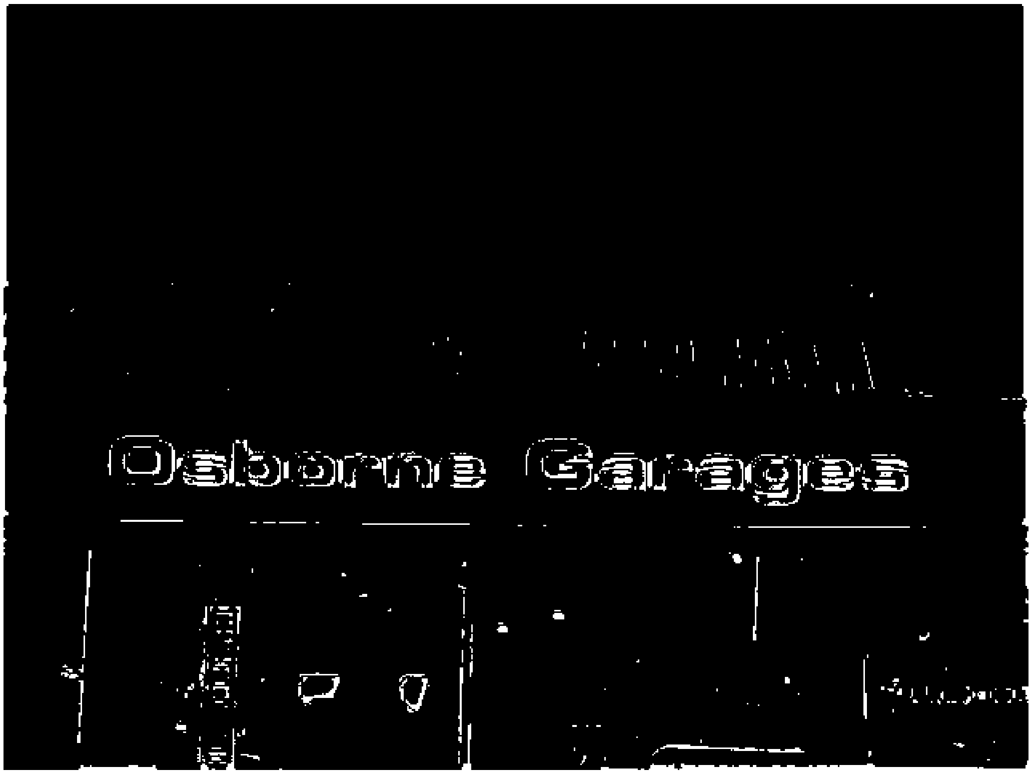


Figure 3.13: Applying three thresholds, we create three classifiers from one filtered image. This is the third of three thresholds.

3.2.3 Weighted Majority

Working with the classifiers produced by applying our threshold selection process to our features, the Weighted Majority algorithm (WM) [22] is applied. Starting with uniform weights, loss is calculated and the weight of a classifier is updated accordingly. Classifiers which perform poorly carry less weight than those which perform better.

We modified the original WM algorithm described in [22] such that individual loss, l_i^j is no longer simply 1 for misclassification and 0 for correct classification. In our modification, l_i^j ranges in $[0, 1]$ according to the pixel level f-metric, such that if precision and recall are high, loss is low, and if precision and recall are low, loss is high.

Additionally, the post processing step which includes edge detection, a linearity filter, and a color consistency filter are applied prior to the loss calculation to allow WM to select the optimal combination of classifiers to use in conjunction with these steps.

The output of this weighting process is normalized and a threshold of 0.5 is applied such that values which have received a majority (greater than 0.5) vote by the ensemble remain, while those which have received less than a majority vote are removed.

Weighted Majority is run 10 times, randomly shuffling the input data for each run. The total cumulative loss for each run is computed and used to

```
Given:
  • Classifier ensemble, $\mathcal{D} = \{D_1, \dots, D_L\}$
  • Labeled dataset, $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$
1. Initialize the parameters
    • Pick $\beta \in [0, 1]$ to be 0.9
    • Set weights
      $\mathbf{w}^1 = [w_1, \dots, w_L], w_i^1 \in [0, 1], \sum_{i=1}^{N} w_i^1 = 1$ (Usually $w_i^1 = \frac{1}{L}$)
    • Set cumulative loss $\Lambda = 0$
    • Set individual loss $\lambda_i = 0$, $i = 1, \dots, L$
2. For all $\mathbf{z}_j \in Z$, $j = 1, \dots, N$
    • Calculate the distribution of classifier likelihoods by
        $$p_i^j = \frac{w_i^j}{\sum_{k=1}^{L} w_k^j}, i = 1, \dots, L$$
    • Find the individual losses:
        – compute pixel level recall,
          $r = |\{true\ positive\}|/|\{true\ positive + false\ negative\}|$
        – compute pixel level precision,
          $p = |\{true\ positive\}|/\{true\ positive + false\ negative\}|$
        – compute $f$-metric, $f = 2pr/(r + p)$
        – compute individual loss, $l_i^j = 1 - f$
    • Update the cumulative loss, $\Lambda$, and individual losses, $\lambda$
        $$\left(\Lambda \leftarrow \Lambda + \sum_{i=1}^{L} p_i^j l_i^j\right) \qquad \lambda_i \leftarrow \lambda_i + l_i^j$$
    • Update the weights
        $$w_i^{j+1} = w_i^j \beta^{l_i^j}$$
3. Calculate and return $\Lambda$, $\lambda_i$, and $p_i^{N+1}$, $i = 1, \dots, L$.
```

Figure 3.14: Weighted Majority algorithm

compare the results of each run. The final weights selected correspond to the run with the lowest total cumulative loss.

3.2.4 Line Removal

Given a binary mask, each connected component is analyzed for linearity. This applies to both the result of the DCT classification and to the edge detection step discussed later. The spatial coordinates in the image of each pixel in a given connected component are extracted and the covariance matrix of this is computed. Next we calculate the eigenvalues of the covariance matrix and take the ratio of the largest and second largest elements. These eigenvalues represent the spread of the data in each of its two principal directions. The ratio is bounded by [0,1] such that zero indicates a straight line and one indicates a perfect circle. To ensure the ratio is bounded by [0,1], the eigenvalue which corresponds to the direction of greatest variation must be used as the denominator. Since we know this value is the largest, as the shape becomes more linear the denominator becomes larger, and the ratio goes to zero. On the other hand, if the variance in both directions is the same, the ratio becomes one. This accounts for the aspect ratio of a character, allowing us to filter out excessive lines produced by highpass and edge filtering.

The threshold value was determined by computing this ratio on each character in the training data and finding the mean value. A histogram of the ratio of connected components in the ground truth is shown in Figure

3.15. The mean is 0.39 and the median is 0.36. The shape of this histogram indicates that the threshold value should perhaps be lower. The histogram peaks at 0.04. Sample results of using 0.04 as the threshold value, as well as additional insight into threshold selection are discussed in Chapter 5. A sample image is shown in Figure 5.1.

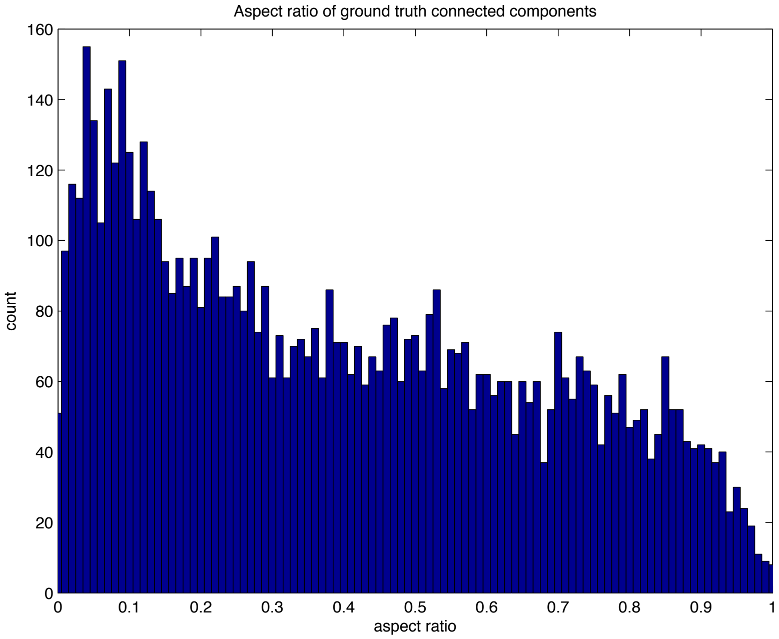


Figure 3.15: Histogram of aspect ratios computed for ground truth connected components.

Applying line removal to each of the three classifiers shown in figures 3.11-3.13 above, we see the results shown in figures 3.16-3.18, respectively.

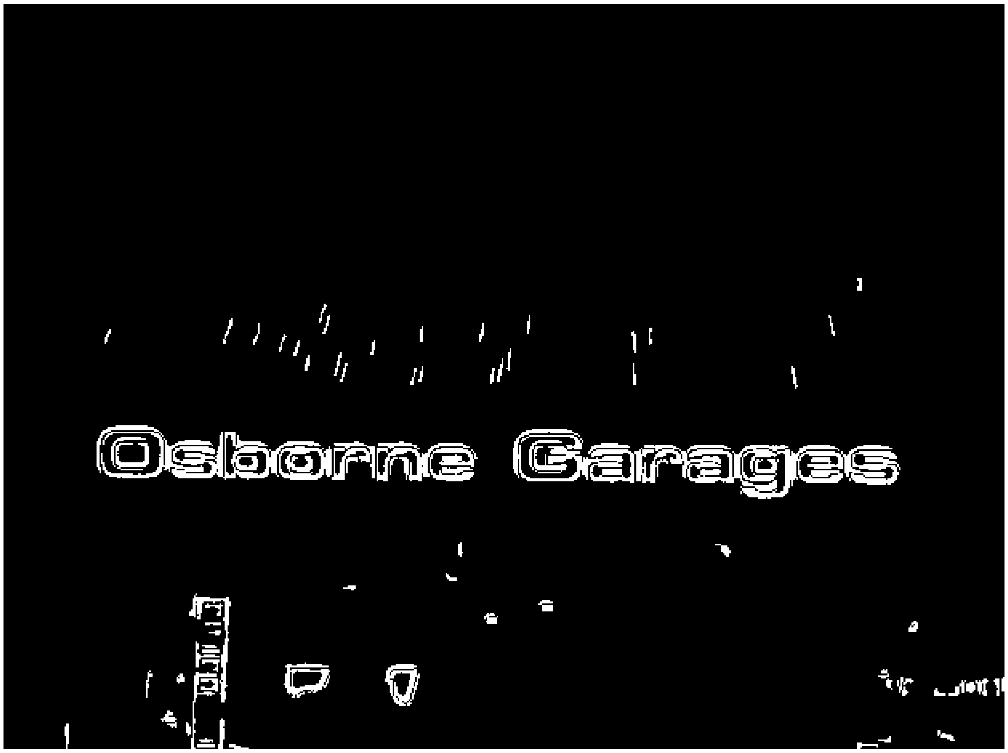


Figure 3.16: Eigenvalue line filter applied to the classifier shown in figure 3.11.

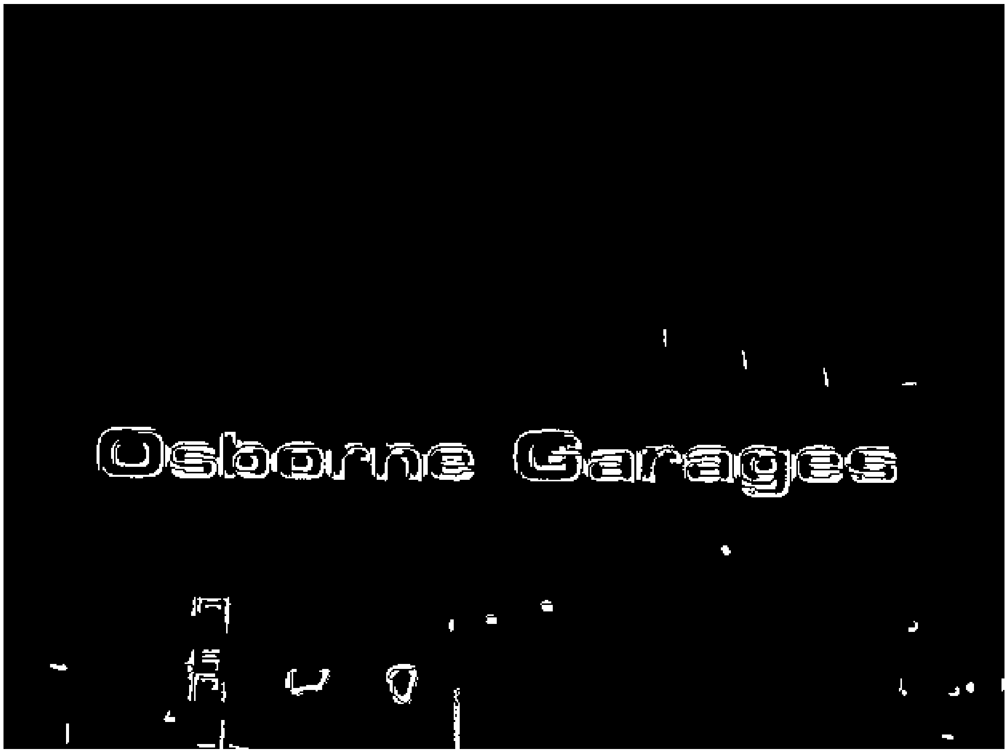


Figure 3.17: Eigenvalue line filter applied to the classifier shown in figure 3.12.

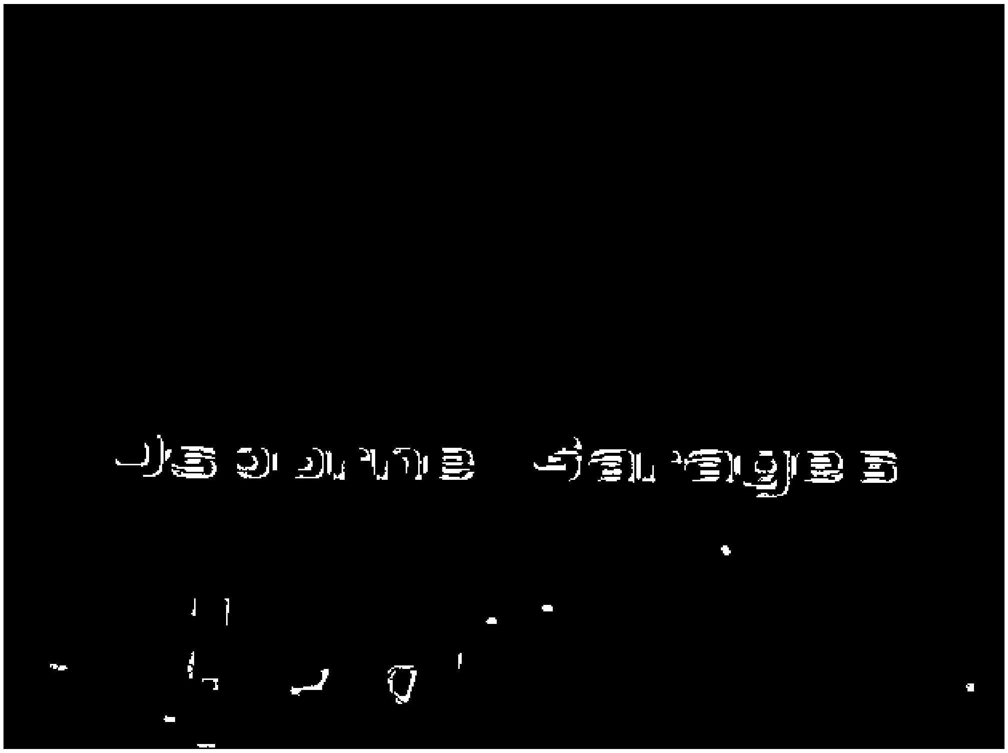


Figure 3.18: Eigenvalue line filter applied to the classifier shown in figure 3.13.

3.2.5 Color Filtering

Given a filled connected component, we first quantize colors down to 16 using the same method described previously for semi-automated ground truth creation. Colors are sorted according to their popularity in the component, colors which have a color difference greater than 2.5 are kept until 16 colors are accumulated. These colors are assigned to pixels in the connected component based on minimum color difference from available colors with the original color. Next we compute the average color difference between each color in the component with all other colors in the component. If that average difference is below a threshold, the component is considered to be consistent in color and therefore more likely to be a character. If, on the other hand, color across the component is not consistent, it is removed.

The threshold value was determined in much the same way as the line removal threshold. Working with the ground truth data, the average color difference for each individual connected components was determined. A histogram of the ratio of connected components in the ground truth is shown in Figure 3.19. The mean is 13.3 and the median is 13.5. The shape of this histogram indicates that the threshold value should perhaps be lower. The histogram peaks at 2. Sample results of using 2 as the threshold value, as well as additional insight into threshold selection are discussed in Chapter 5. An example of using 2 as a threshold is shown in Figure 5.2.

Similar to line removal, color filtering is used as a constraint for both the

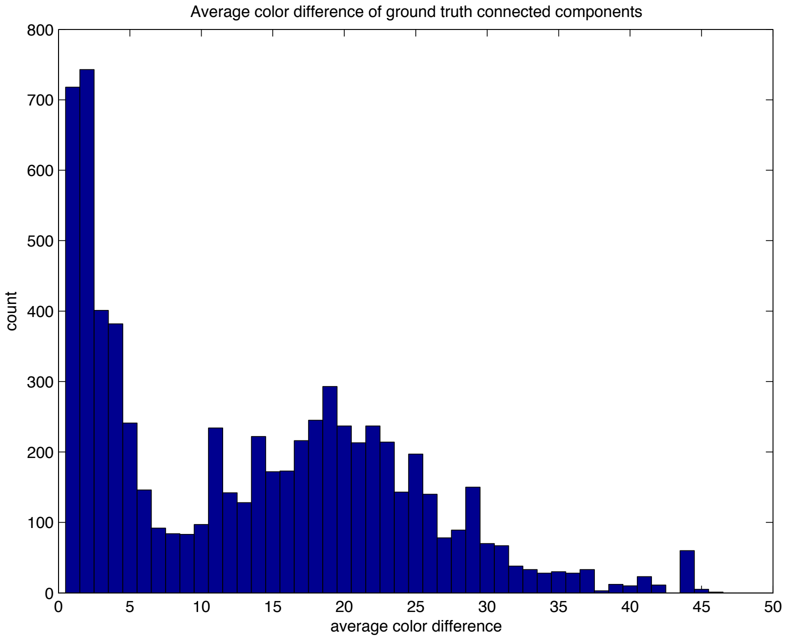


Figure 3.19: Histogram of average color differences computed for ground truth connected components.

DCT classifier and the edge detection procedure. Applying color filtering to our continuing example classifiers, we get the results shown in figures 3.20-3.22. For this particular example, the results are subtle, however some connected components have been removed when compared with the results of line filtering shown in figures 3.16-3.18.

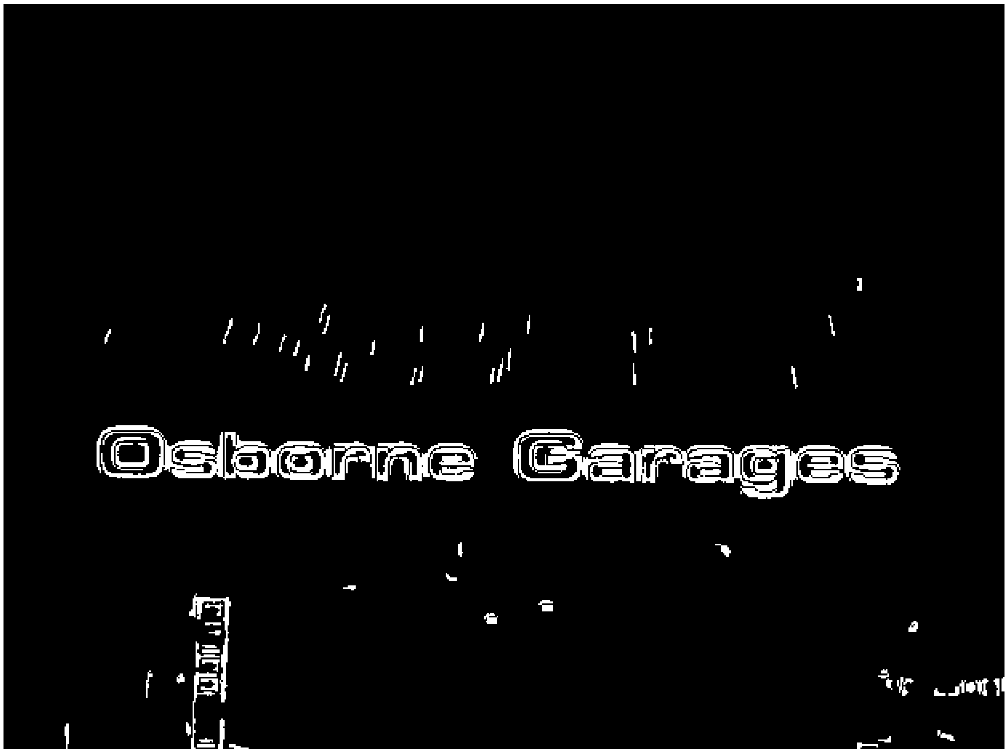


Figure 3.20: Results of the color consistency filter being applied to the previous step in the post processing chain for this classifier, shown in figure 3.16.

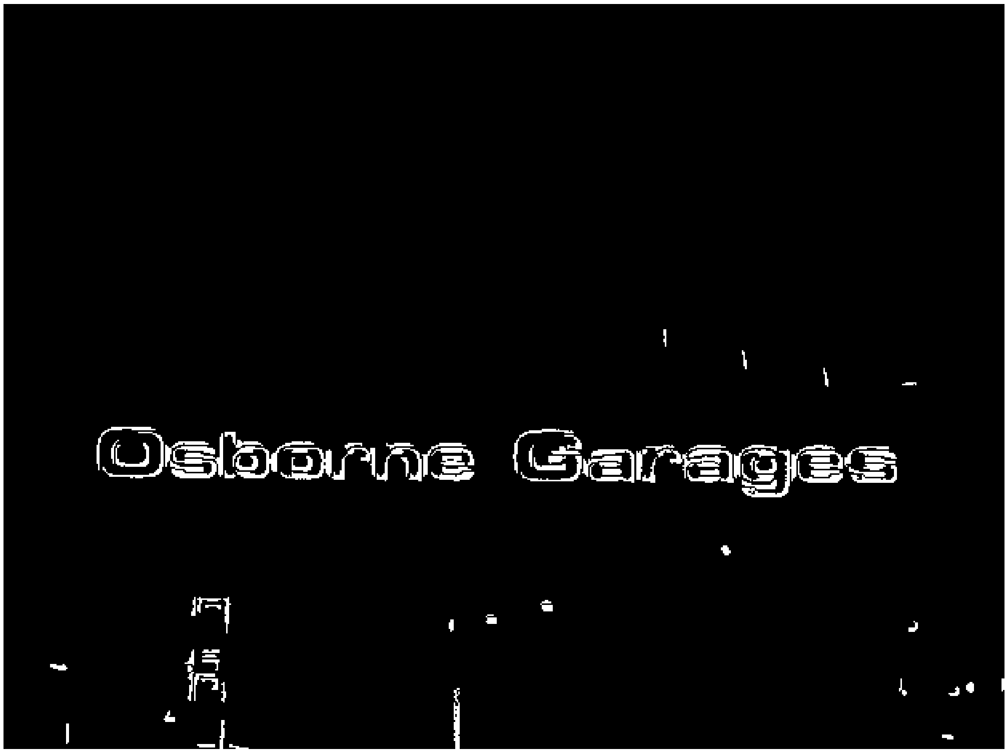


Figure 3.21: Results of the color consistency filter being applied to the previous step in the post processing chain for this classifier, shown in figure 3.16.

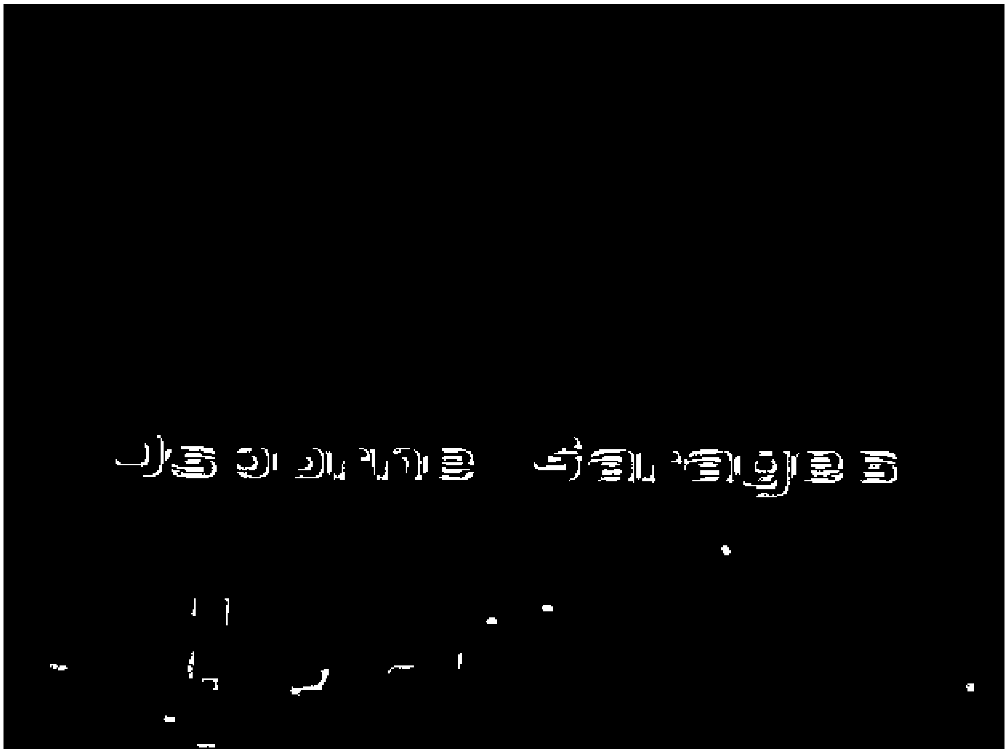


Figure 3.22: Results of the color consistency filter being applied to the previous step in the post processing chain for this classifier, shown in figure 3.16.

3.2.6 Edge Detection

Setting aside the DCT classification results, the edge detection post processing step makes use of the original Y channel intensity image. The Canny edge detector is applied using Matlab's default parameters, $\sigma = \sqrt{2}$ for the Gaussian filter, and high and low threshold values are automatically chosen. While it may be possible to achieve better results by modifying these parameters, keeping them fixed reduces the number of parameters needed by the overall system. The result of running the Canny edge detector on the "Osborne Garages" example is shown in figure 3.23.

Next the linearity filter is applied, removing any connected components which are below the set threshold. Prior to morphologically filling in closed connected components, a tree is created to represent nesting of components. We cannot simply fill in all connected components. The component which contains all other components in the image is labeled background. If a component is the parent of three or more other components, it is considered to be a container and is discarded. This is the case where a sign contains several letters and we are only interested in the letters, not the sign. Next we look at those components which contain one or two other components. If the interior components are within the threshold for color consistency of the background outside their parent, they are labeled background and are not filled in. In this way we are able to avoid over filling regions which should be background, including closed signs and the interior of letters. Color consistency is used to

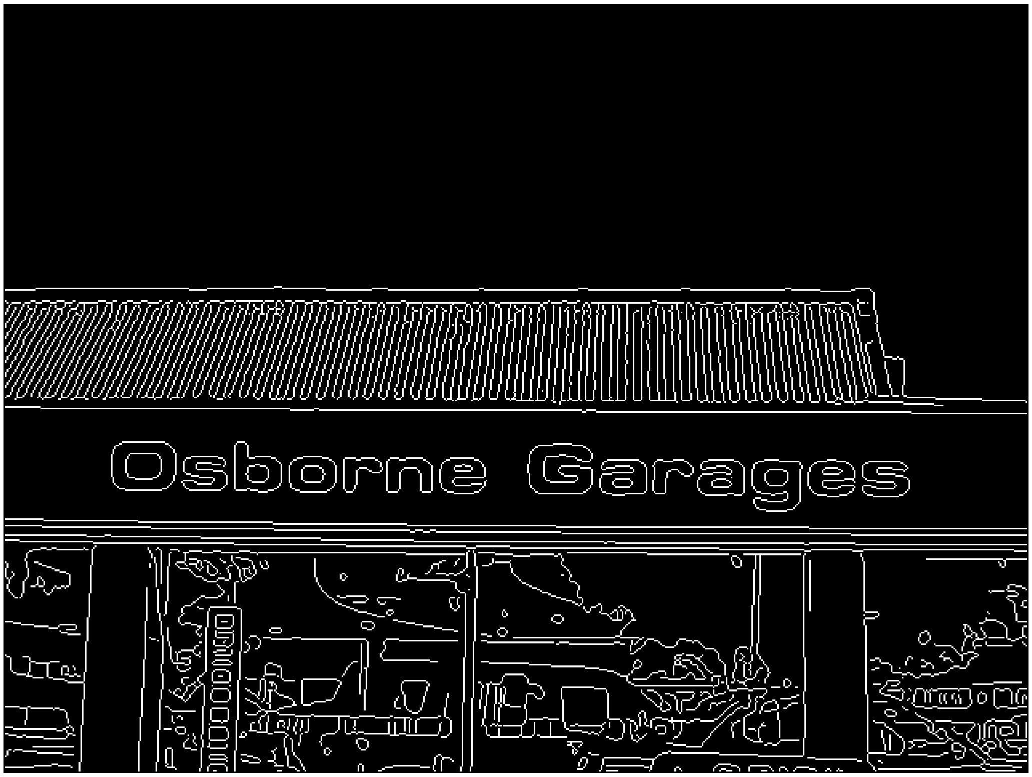


Figure 3.23: Canny edge detector applied to the grayscale version of the "Osborne Garages" image.

```
• Find all connected components
• For each connected component
    – If this component contains three or more other components, remove it
    – If this component contains two or one connected components, fill it
        ∗ If interior component is not within the threshold for color consistency with its containing component, don't fill it.
        ∗ Else, fill it
    – If this component contains no other components, fill it
```

Figure 3.24: Morphological filling of connected components

remove any additional components which are above the threshold for average color difference within a component. Pseudocode for this process is shown in Figure 3.24

Applying the linearity filter to our example results in figure 3.25. This image is morphologically filled according to the process outlined above, and the color consistency filter is applied, resulting in figure 3.26.

The mask which results from edge detection and region filling is intersected with the mask produced by the DCT classifier. Components which exist in both images are labeled text, however overlap is often small and usually does not include entire letters. Since edge detection provides a reasonable segmentation, filled connected components from the edge detection are included in the final classification mask where the two masks intersect. Note that this

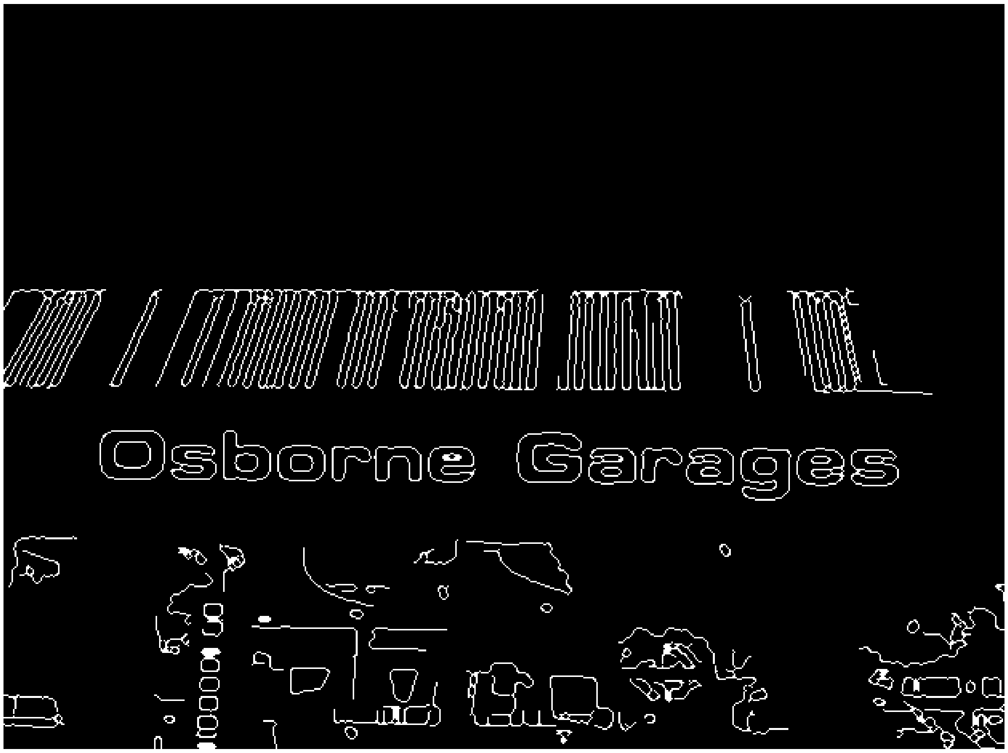


Figure 3.25: Result of running the linearity filter on the Canny edge detection image.

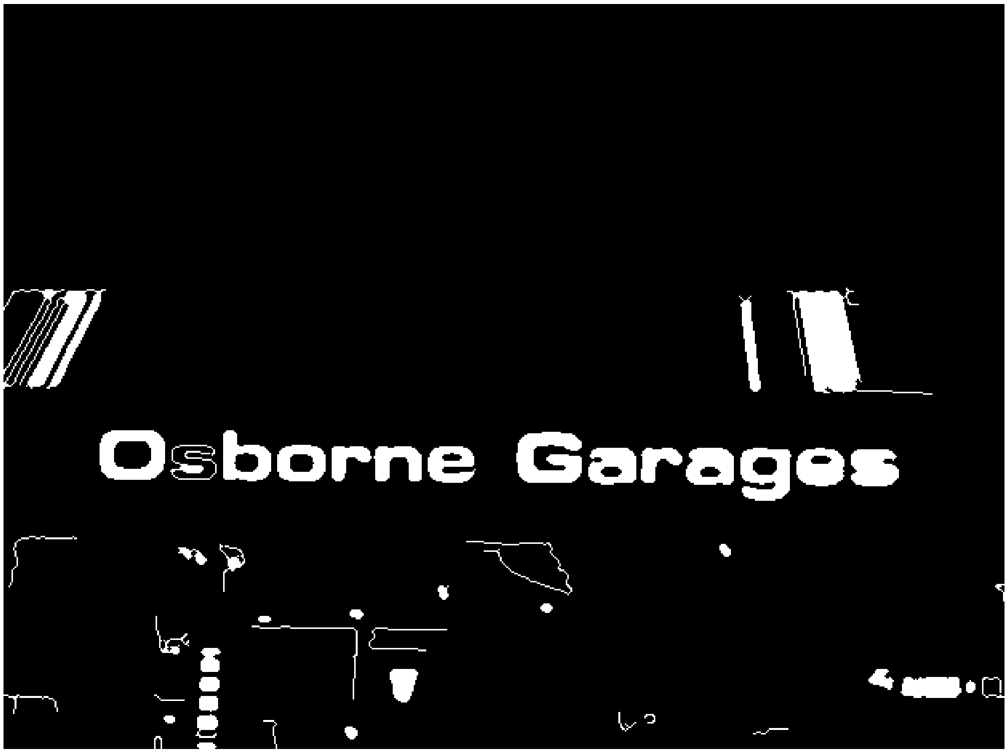


Figure 3.26: Morphologically filling the image shown in figure 3.25, according to the connected components tree method outlined above, followed by color consistency filtering.

process occurs within the WM step of our approach as it is applied to each classifier.

Returning to our running example, intersecting the second classifier after linearity filtering and color consistency filtering have been applied, with the edge detection result after the same filters have been applied we get the image shown in figure 3.27. This is done for each of the classification images, all of which are then summed together and thresholded at 0.5, retaining pixels which have received a majority vote from the weighted classifiers. This result is again compared with edge detection result. Connected components are retained in the final result if the result of weighted majority contains pixels in those components. The final result of this process for this example is shown in figure 3.28.

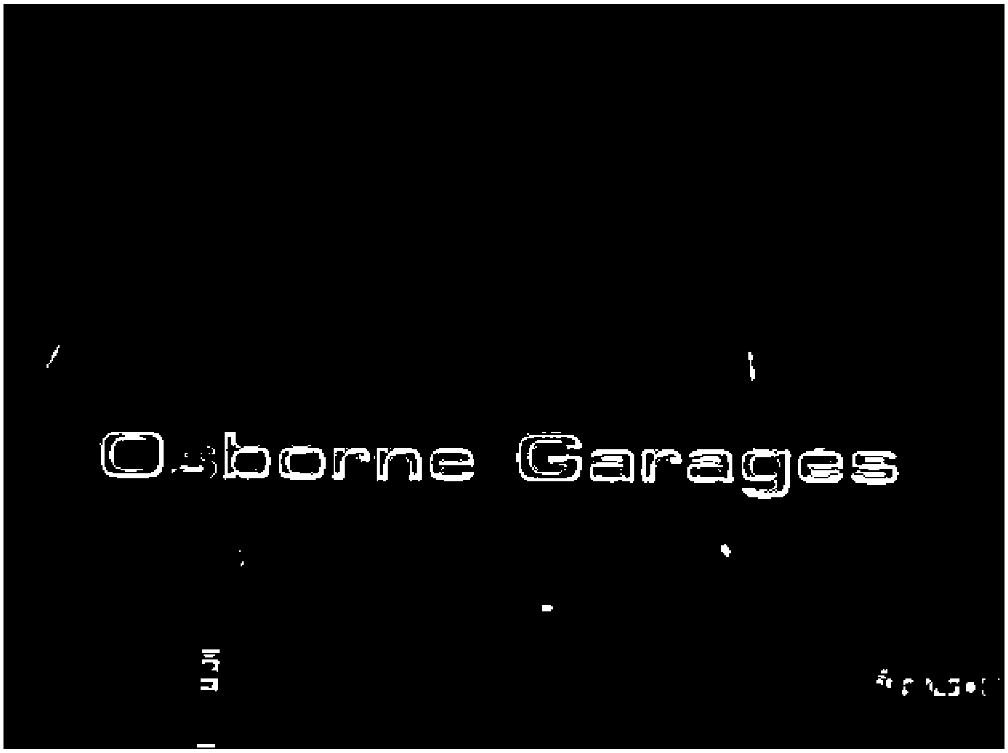


Figure 3.27: The intersection of the edge detection results and a classifier.

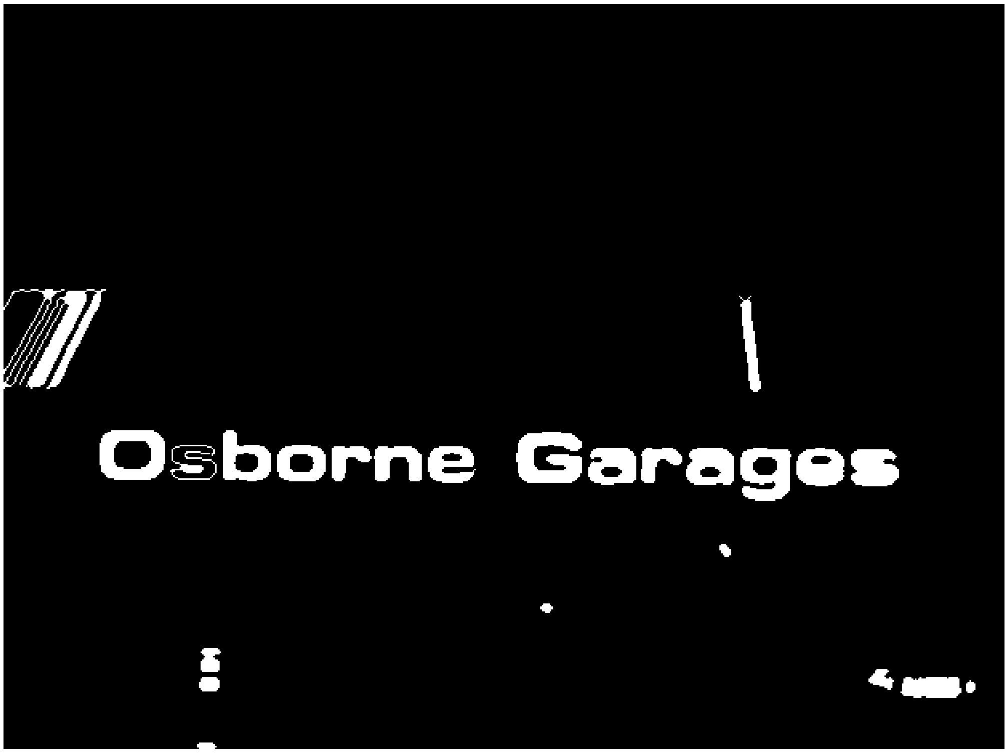


Figure 3.28: The final result. WM combination of intersection results is used to turn on or off connected components from the edge detection image.

Chapter 4

Experimental Results

Our algorithm was trained and tested using the dataset found in [31, 46]. Provided with the data is a word level precision, recall, and f-metric for scoring purposes. Ground truth provided for the dataset only captures the bounding boxes of words in the images. Since our goal is to perform text detection at the pixel level, we created new pixel level ground truth for training and testing our algorithm. Similarly, we measure precision, recall, and f-metric at the pixel level rather than at the word level. Because of this it is not possible to directly compare our results with the results of other researchers. Unlike many other scoring metrics, we do not include any overlap or tolerance. Since pixels in the result must exactly match pixels from the ground truth, this may slightly lower our scores than if we allowed for some tolerance in scoring.

In order to test our hypothesis that the weighted combination of filters followed by post processing shows better performance than any component does individually, the precision, recall, and f-metric of individual components and the entire system were computed. The Weighted Majority algorithm was trained using 250 training images. Similarly, those training images were used to compute thresholds used in post processing. These values were then used on a separate testing set of 250 images for scoring purposes. The scores reported are the results from the testing data.

Table 4.1 shows quantitative results for the entire system, as well as individual components. The complete ensemble is the weighted combination of all 18 DCT feature based classifiers, followed by post processing. Post processing represents only the post processing component; that is the Canny edge detector, linearity filter, color filter, and morphological connected component filling. Individual classifiers are single DCT feature based classifiers, followed by post processing. Note that the overall f-metric for the complete ensemble is higher than any other component, but that the precision of post processing only is higher than that of the complete ensemble.

Data from table 4.1 is visualized in the following graphs, shown in figures 4.1-4.3. In the graphs, classifiers 1-18 from left to right are individual classifiers with post processing. Classifier 19 is post processing only. Classifier 20 is the complete ensemble.

Histograms for each metric were computed for the testing set for the complete ensemble and for post processing reveal further information regarding the large variance observed. These are shown in figures 4.4-4.9.

Table 4.1: Classifier comparison; Individual DCT-based Classifiers

	mean precision	mean recall	mean f-metric
complete ensemble	0.31	0.44	0.36
post processing only	0.47	0.24	0.32
classifier 1	0.24	0.21	0.22
classifier 2	0.17	0.22	0.19
classifier 3	0.12	0.23	0.16
classifier 4	0.25	0.21	0.23
classifier 5	0.17	0.23	0.20
classifier 6	0.13	0.23	0.17
classifier 7	0.25	0.21	0.23
classifier 8	0.18	0.23	0.20
classifier 9	0.13	0.23	0.17
classifier 10	0.26	0.21	0.23
classifier 11	0.19	0.23	0.21
classifier 12	0.14	0.24	0.18
classifier 13	0.26	0.21	0.23
classifier 14	0.19	0.23	0.21
classifier 15	0.14	0.24	0.18
classifier 16	0.27	0.21	0.24
classifier 17	0.19	0.23	0.21
classifier 18	0.14	0.24	0.18

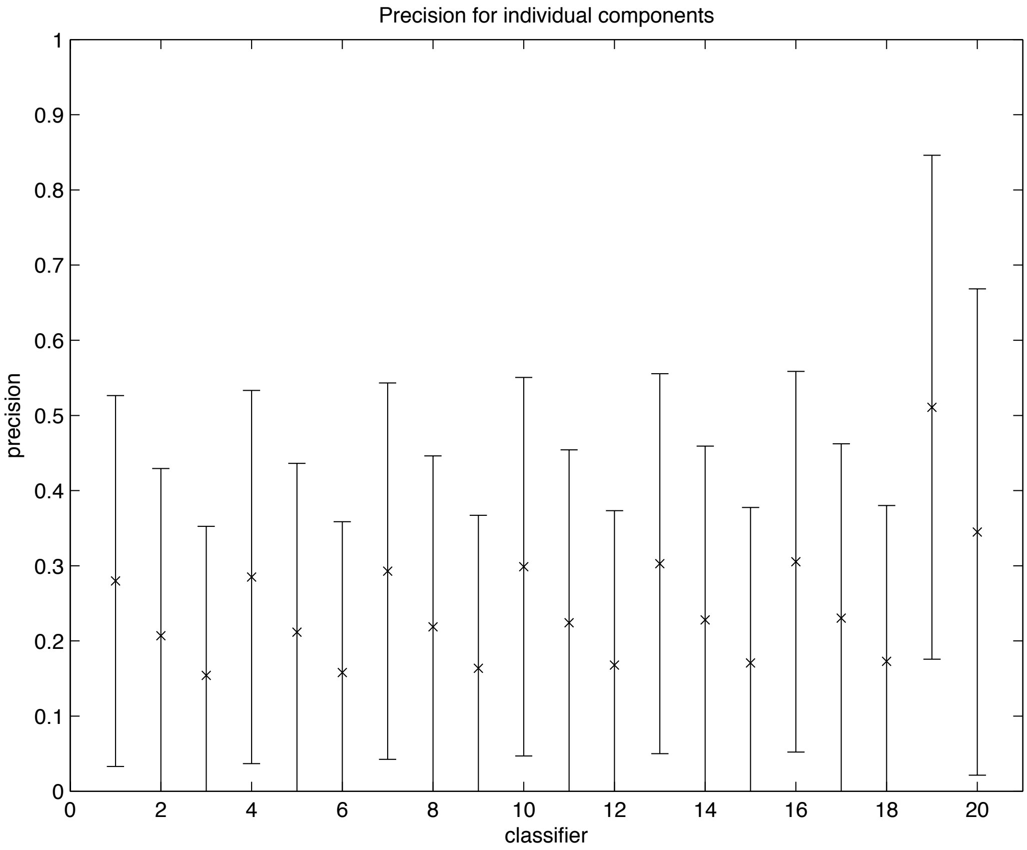


Figure 4.1: Precision for individual components. From left to right: classifiers 1-18 are individual classifiers with post processing; classifier 19 is post processing only; classifer 20 is the complete ensemble.

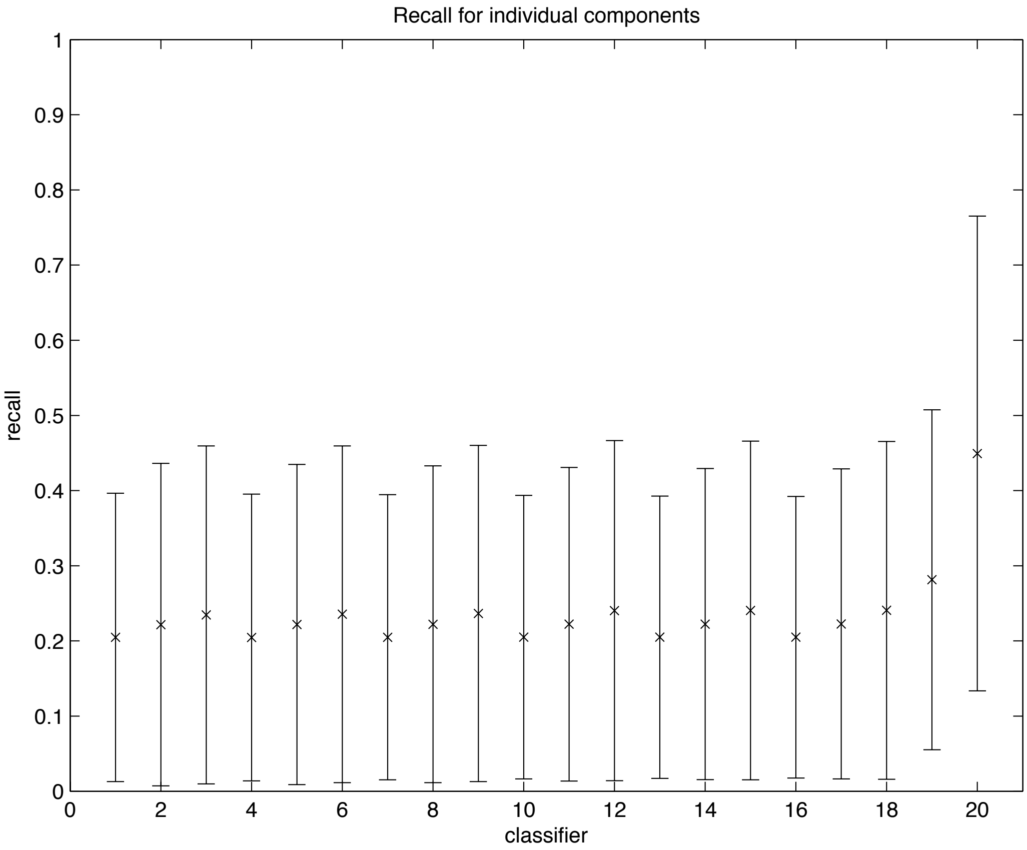


Figure 4.2: Recall for individual components. From left to right: classifiers 1-18 are individual classifiers with post processing; classifier 19 is post processing only; classifer 20 is the complete ensemble.

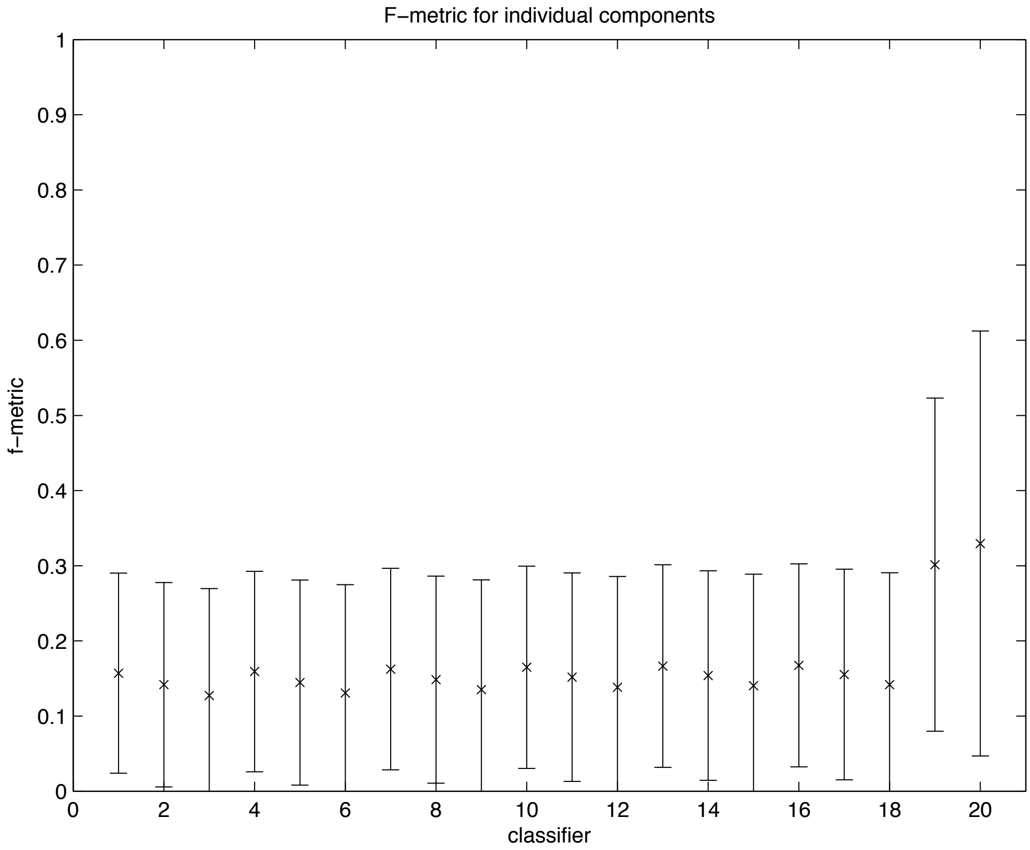


Figure 4.3: f-metric for individual components. From left to right: classifiers 1-18 are individual classifiers with post processing; classifier 19 is post processing only; classifer 20 is the complete ensemble.

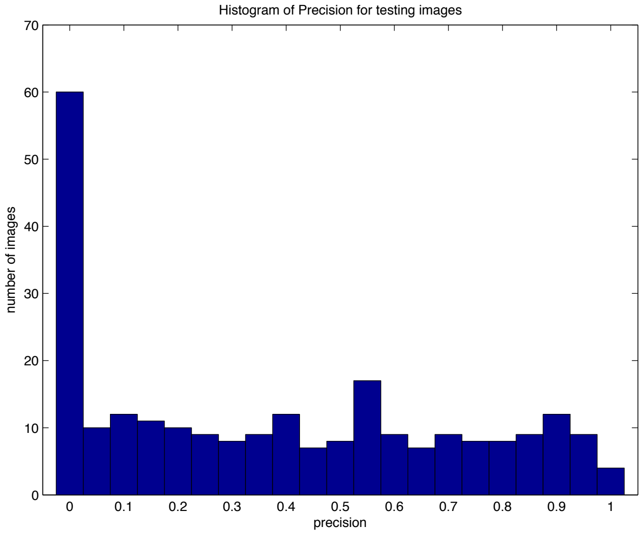


Figure 4.4: Histogram of precision metric on testing data for the complete ensemble.

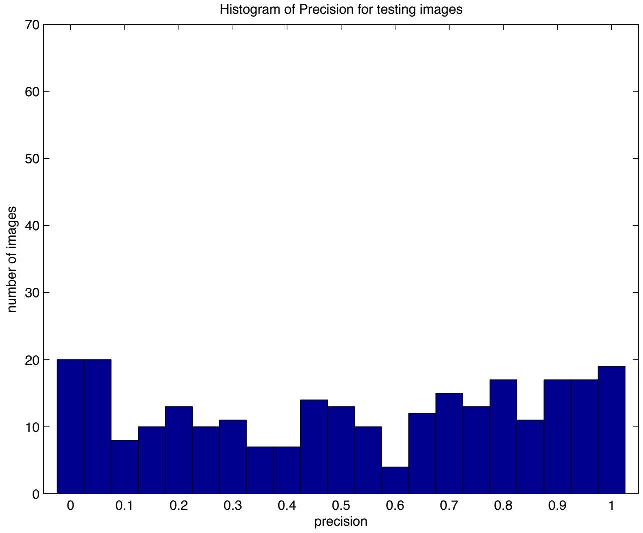


Figure 4.5: Histogram of precision metric on testing data for post processing only.

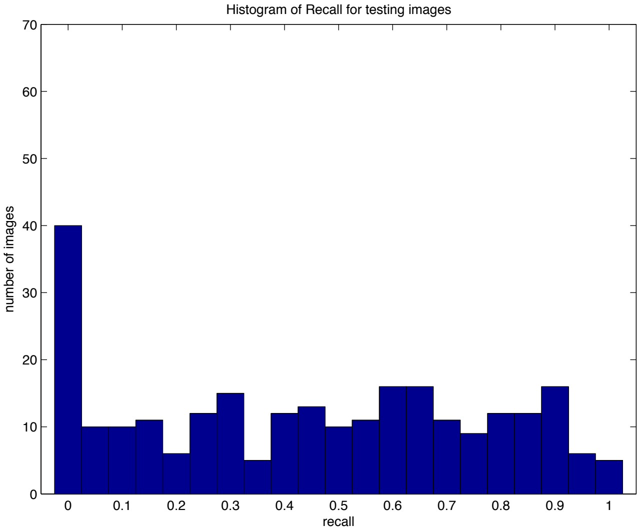


Figure 4.6: Histogram of recall metric on testing data for the complete ensemble.

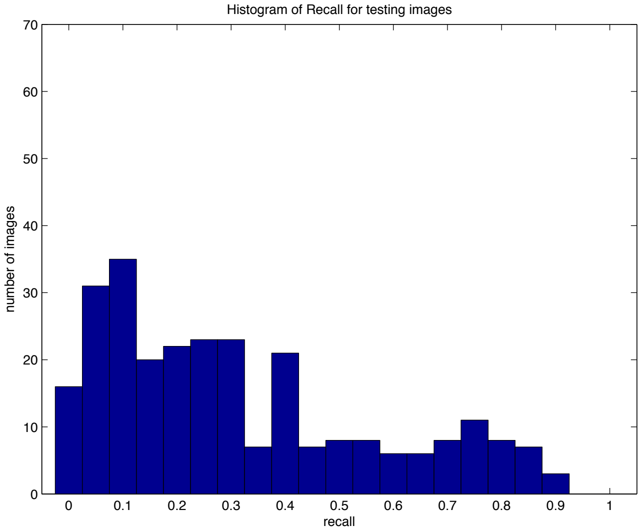


Figure 4.7: Histogram of recall metric on testing data for post processing only.

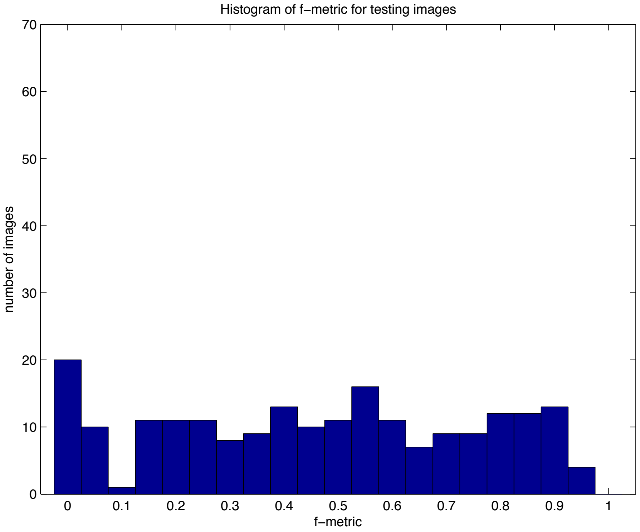


Figure 4.8: Histogram of f-metric on testing data for the complete ensemble.

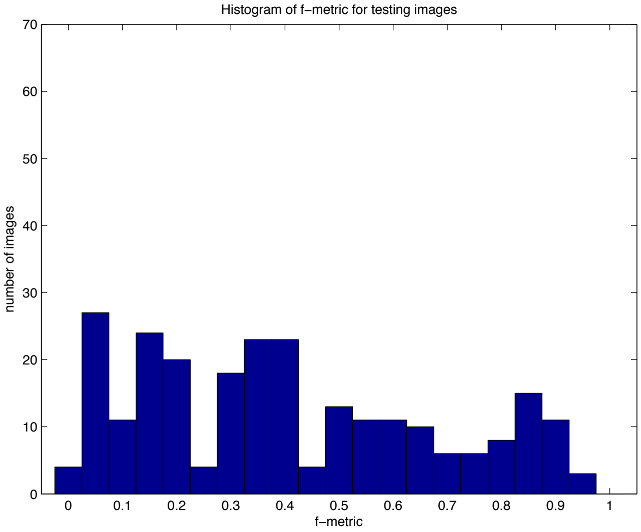


Figure 4.9: Histogram of f-metric on testing data for post processing only.

Given these metrics, in particular the f-metric, it is apparent that the combination of DCT based classifiers and post processing does outperform any individual component. However, when looking at precision and recall independently, the results are not as straight forward. Further, give the large amount of variation in the data, the performance improvement in terms of f-metric of the complete ensemble compared with only post processing may not be statistically significant. Post processing alone outperforms the combination in terms of precision, but the entire ensemble outperforms just post processing in terms of recall. Further, when looking at the performance of individual images rather than the mean of the set, the complete ensemble has a large number of images which fail completely, as compared with post processing alone which has more uniform distributions. These results indicate that while combining individual classifiers with Weighted Majority is successful, there may exist a better means of integrating post processing which allows for a more optimal combination of the two.

Example result images are provided to illustrate a fairly correct classification, figure 4.10, a partially correct classification, figure 4.11, and an incorrect classification, figure 4.12. In general, results tend to have greater numbers of missed detections than false alarms, with a few exceptions. False alarms are typically smaller objects or objects which are character-like but are not characters, such as icons or logos.

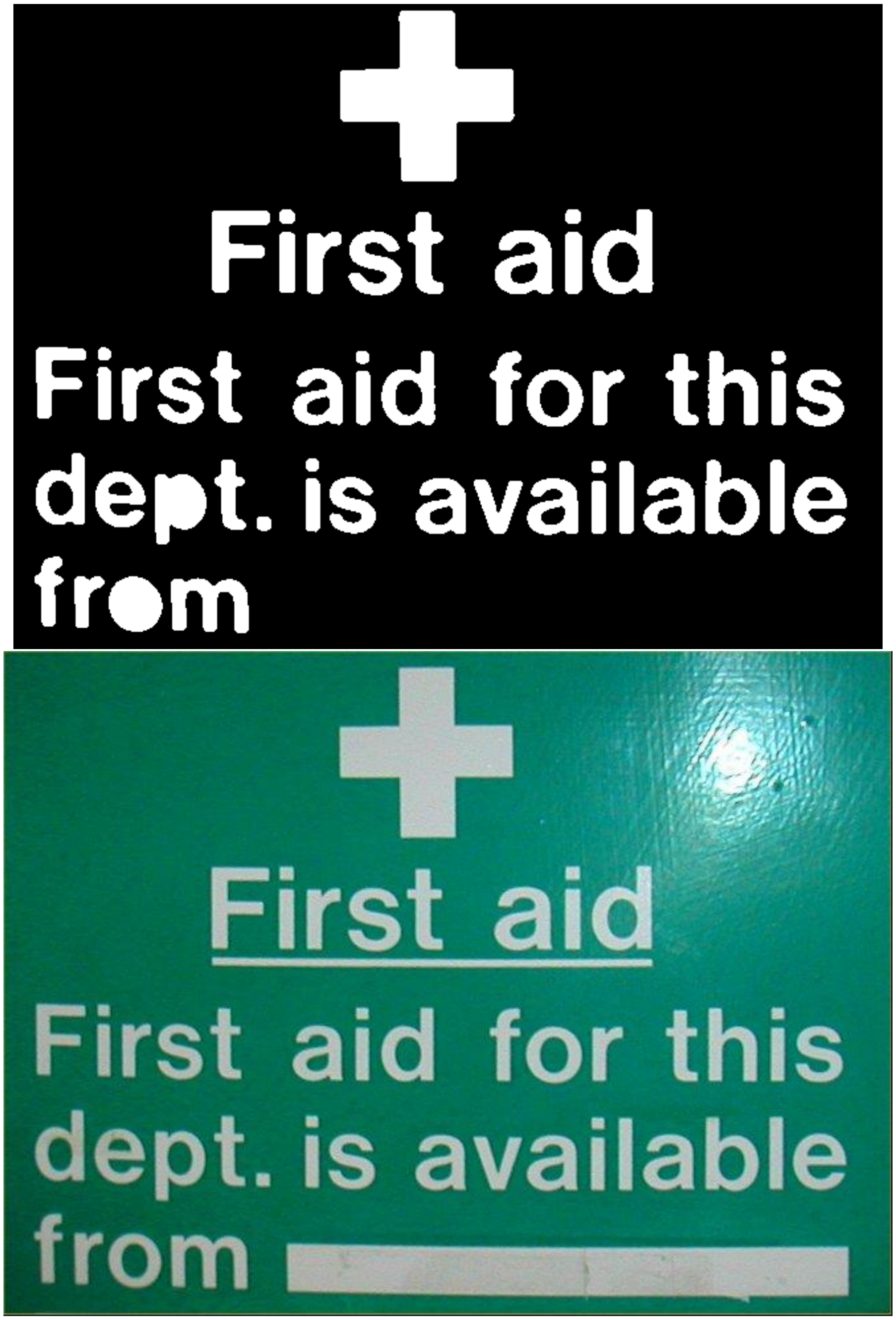


Figure 4.10: Example mostly correct image. Notice some incorrectly filled letters and the incorrect selection of the plus sign. Precision: 0.75 Recall: 0.97 f-metric: 0.85

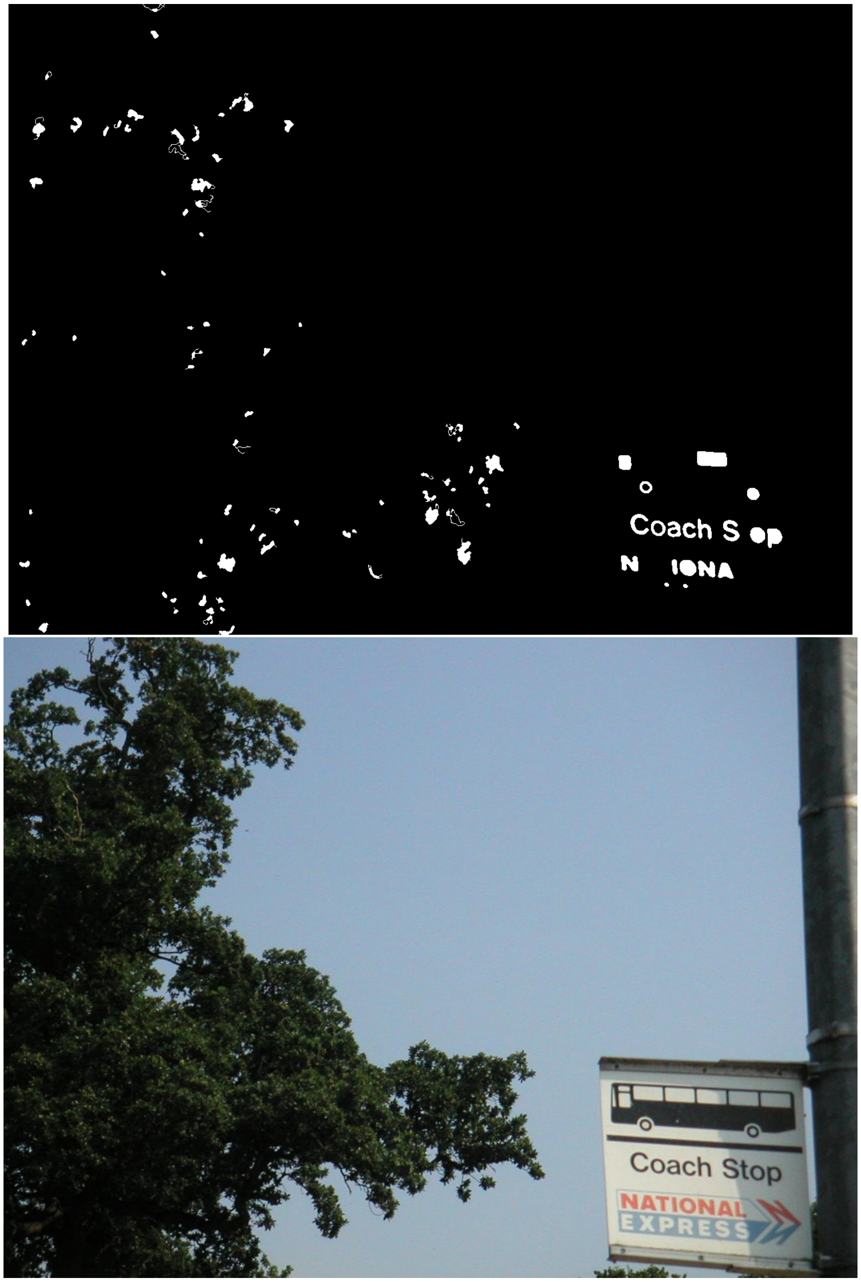


Figure 4.11: Example of a partially correct image. Some text is missing and a significant amount of non-text is incorrectly labeled as text. Precision: 0.36 Recall:0.29 f-metric: 0.32

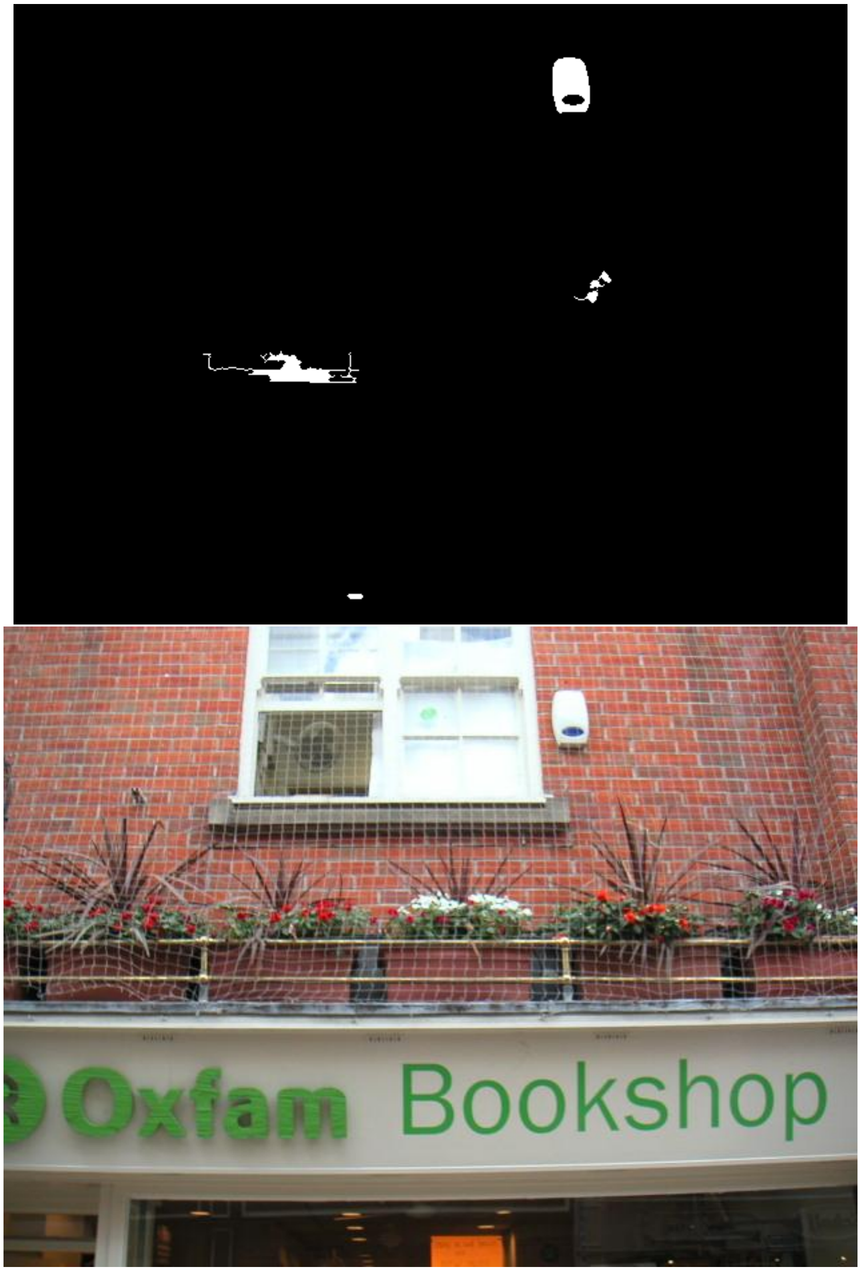


Figure 4.12: Example of a significantly incorrect image. None of the text has been selected, and some non-text has been incorrectly labeled as text. Precision: 0.00 Recall: 0.00 f-metric: 0.00

Chapter 5

Discussion

5.1 Features and Classifiers

Selection of features was motivated by related work as well as the intuition that filtering based on frequency information may allow for the extraction of text characters which are regularly spaced and contain a high concentration of edges. Initially the Sobel edge detector was also used to create classifiers, however, it was found to provide little additional information compared with that provided by DCT based frequency features. The use of bandpass filters may provide better results and was attempted, however this approach required too many parameters and was difficult to tune. Initially several additional highpass filters were used, but it became clear that they did not provide useful additional information than the final six selected.

After applying filters and generating six feature images, creation of classifiers from these filters is challenging. To reduce the number of parameters our system needed to learn, the f-metric was used to select reasonable thresholds based on performance. It is possible to use many additional thresholds, allowing the learning algorithm to decide how many provided useful information, however, the benefit of taking a brute force approach may not be worth the significant increase in computation time required. As a first attempt, it seemed more reasonable to limit the number of thresholds and thus reduce computation time required to produce results.

Figure 5.3 shows evolution of the classifier weights as new training data is introduced. While the weights of some classifiers end close to zero, others remain fairly even with each other. It is this behavior that allows the combination of classifiers to outperform any individual classifier. In the figure, the top four weights, accounting for nearly 95% of the weight of all classifiers correspond to the highpass filters, in order from least to most significant, where $\sigma = 1.5$, $\sigma = 2.0$, $\sigma = 2.5$, and $\sigma = 3.0$, each with a threshold value of 0.2. Interestingly, the lowest of the highpass filters, with $\sigma = 3.0$, is weighted higher than the other filters. This indicates the original f-metric criterion for choosing filters may need to be modified, and perhaps filters of size larger than 8×8 should be tested.

Also note from the figure that the final weights selected are weights which correspond to the lowest cumulative error for a set of weights. Weighted Majority has been run 10 times, and for each time the cumulative loss is computed. The final weights resulting from each run are stored, and the weights associated with the lowest cumulative loss are selected. Alternately this step can be omitted and the weights resulting from all 10 runs combined can be used.

Although this figure shows the results after ten iterations through the data, it is unclear if additional training runs or additional data will improve these weights or lead to overfitting. However, all of the weights except for the highest one seem to have leveled off or started declining, hinting that this is a good stopping point. Nevertheless, as the highest weight seems to still be increasing, additional runs may be needed. To address the concern of overfitting, it would perhaps be more beneficial to train on additional training data or a validation set rather than to continue to iterate on this set.

Altering post processing thresholds was attempted to see if there may be a quick solution to finding a good threshold for both the linearity filter and the color consistency filter, other than using the mean. Using thresholds corresponding to the most popular value from the training data, we get the results shown below. As shown in Figure 5.1 and Figure 5.2, these values seem too strict compared with using the mean. For the next iteration of this algorithm, a simple classifier or some other similar approach should be taken to identify better thresholds for these important features.

5.2 Weighted Majority Algorithm

The use of the Weighted Majority algorithm was motivated by several factors. Since classifiers are created prior to learning and we are interested in their relative performance, this algorithm is the natural choice. Additionally, WM is an online algorithm, that is after training data has been used, it is not needed in again in the future for further training. Instead, if new training data

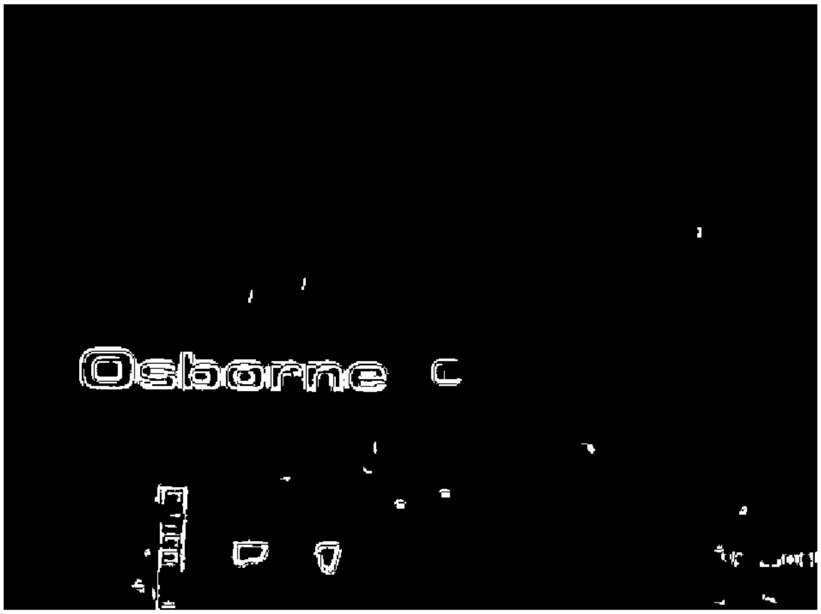


Figure 5.1: Applying a linearity threshold of 0.04 to our sample image results in a significant loss in the text selected.

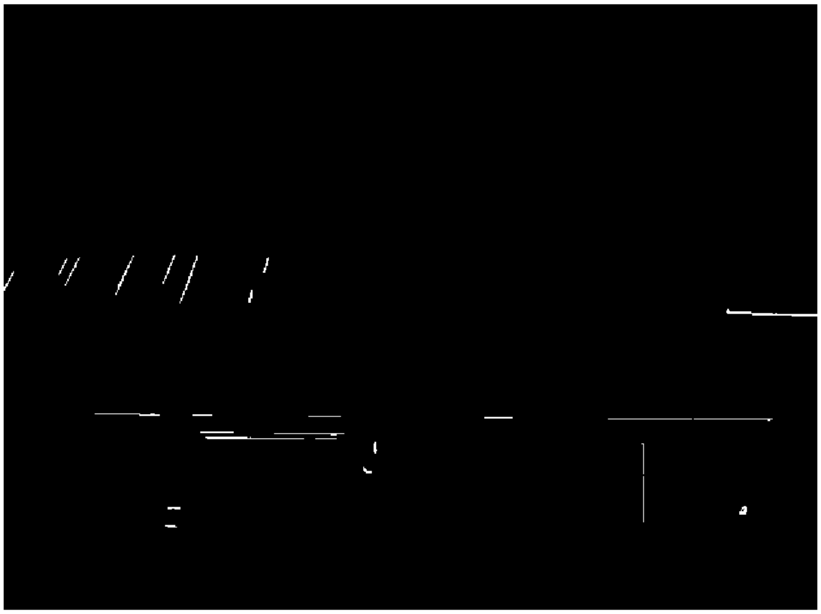


Figure 5.2: Applying a color consistency threshold of 2 to our sample image removes all text.

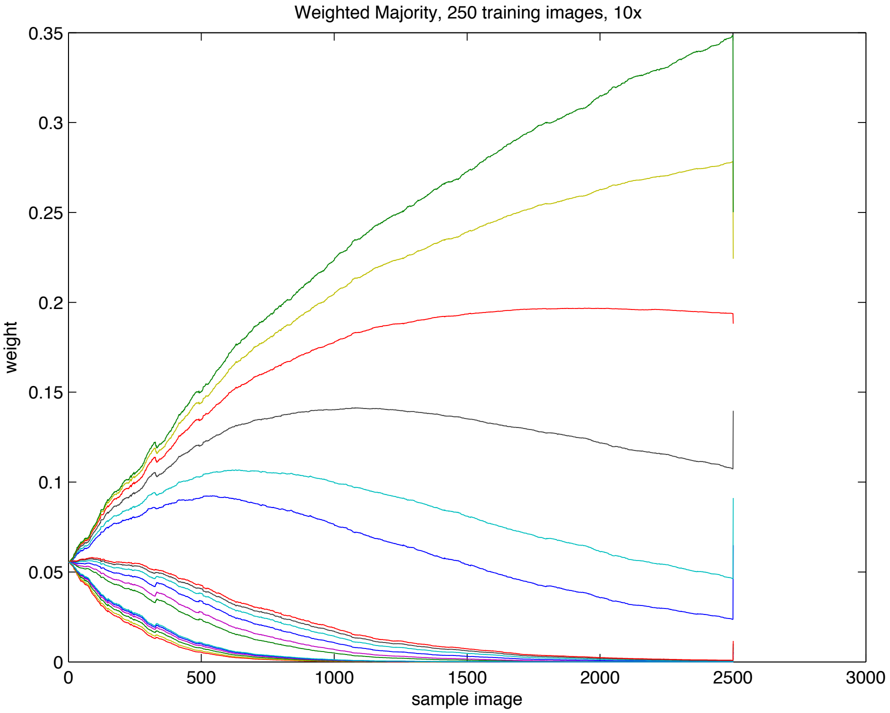


Figure 5.3: Change in classifier weight during training.

is available, the existing weights can be adjusted accordingly. This is particularly useful for training a text detection algorithm were data with ground truth is hard to come by. In the event of additional data being available, it is trivial to retrain the system to learn from this new information. On the other hand, more advanced approaches including adaptive boosting have been shown to produce impressive results when trained on seemingly trivial information. A potential improvement to our system is to explore other more advanced learning algorithms to gauge their utility for the text detection problem.

5.3 Post Processing

The post processing procedure resulted in the need to find a way to reduce the number of false positives as well as false negatives produced by the weighted combination of classifiers. Here we make assumptions about the aspect ratio and color consistency of text. Many false positives are due to the inclusion of small objects and fine lines, which are easily removed by a minimum size requirement and the requirement that letters are not too linear. In addition, to reduce the number of larger more circular objects from remaining, a color consistency constraint is applied. Unfortunately, some letters which are quite linear have a higher tendency of being rejected, as do letters which either intentionally contain several colors or appear to be multicolored due to illumination.

Earlier forms of our post processing did not include the Canny edge detector, instead relying on morphological processing to close gaps in letters. Due to the nature of a frequency based approach, the interior of a letter, which has relatively low frequency, is not likely to be selected by a series of highpass filters. This results in the need to fill in letters, however it was found that the majority of letters were only partially found by the classifiers, making a simple filling operation impossible. To overcome this problem, the Canny edge detector is used to segment letters more completely, which can then be filled in. On its own, the edge detector recovers all edges in an image, not just those associated with text, requiring the use of additional information. This extra information comes from the classifiers, and in those regions where both agree, pixels are labeled text.

Room for improvement exists in the way in which post processing is merged with results from the classifiers. By simply finding regions which intersect and opting to use filled in regions from the edge detection process, we assume the edge detection process worked perfectly. The Canny edge detector is a complex algorithm with many parameters available to tune. To keep things simple, these were fixed for our implementation. A potential improvement to the system would be to use multiple Gaussian filters in the Canny detector and to adjust the high and low thresholds used. Ideally these values would be extracted from the data, either by a statistical approach, or by a learning algorithm.

One key to using the Canny edge detector and region filling successfully turned out to be the ability to determine the nesting of connected components. If a region contains several components, it is more likely to be a sign or some other form of uniform background than text. On the other hand some text characters do contain interior regions, such as the letters “A”, “O”, and “P”. To account for this color consistency was used to determine if an interior region was more consistent with background or foreground. Overall it seems that this process worked quite well and provided a significant improvement in our results.

5.4 Analysis of Results

Looking at the numbers, it seems that our approach did not perform as well as is desired. While this is true to some extent, it is important to keep several things in mind when interpreting the numbers and comparing them to other text detection algorithms. First, unlike any other algorithm we could find, we focus on the pixel level and not on the word level. Since ground truth needed to be created for this task, but it is not practical to label every pixel in a 500 image dataset, we opted to use a semi-automated approach. While quite effective, this approach isn’t perfect and introduced some error in the ground truth. Also, since we started with the word level ground truth, we must assume that is correct. This is not actually the case. Since word level ground truth was created by hand, it too contains errors, including missing words, and incorrectly selected logos which are not selected in other images.

If we assume that the ground truth is correct, it is important to note that our metrics are pure precision, recall, and f-metric on the pixel level. It is quite common to compensate for ground truth errors by allowing some tolerance when computing results. Often metrics consider words found in by an algorithm to be correct if some percentage of overlap with a ground truth word is present. Although this approach can be justified, it can also lead to over inflating scores, especially when comparing to scores which have not been computed to include any tolerance.

If we assume that the ground truth and scoring metrics are correct, one final note to make is that this is quite a difficult problem, sometimes even for humans. Several example images and our classification results are provided below to illustrate this point. Due to the large variety of text, especially in-scene text, it is very difficult for any simple system to perfectly solve this problem. That said, many newer algorithms which have begun to appear over the last year continue to make advances in solving this task. Qualitatively, our results look quite promising and show interesting potential as well as some room for improvement.

5.5 Comparison With Other Algorithms

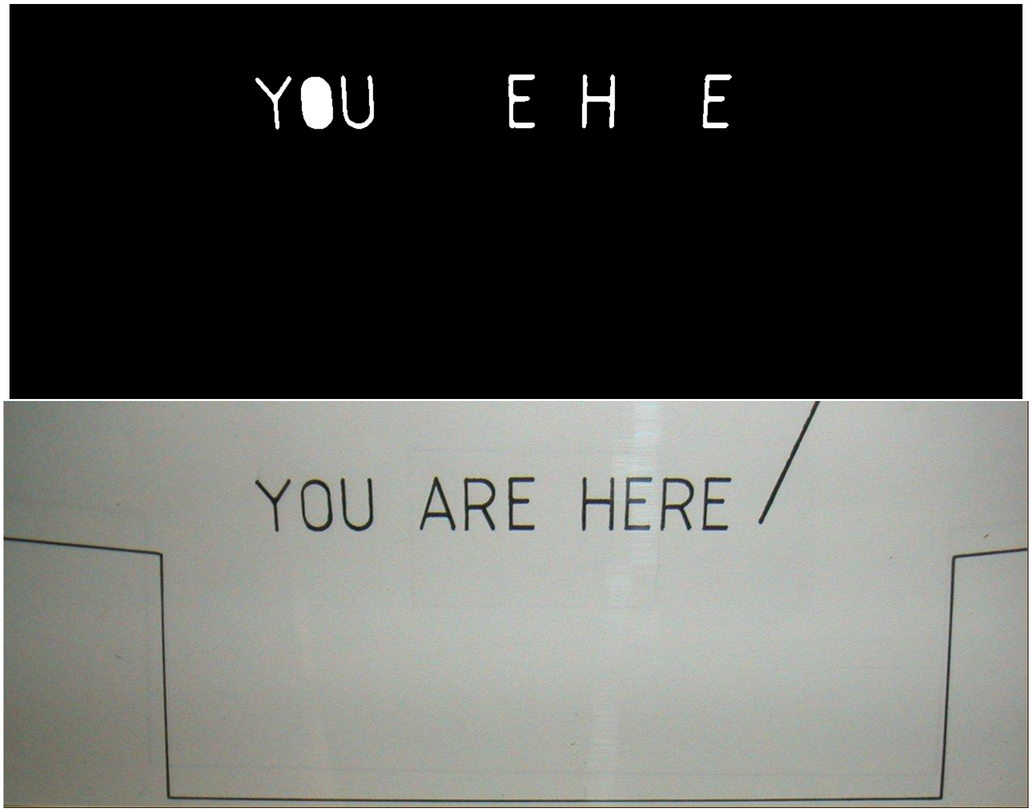


Figure 5.4: In this example, much of the text is correctly selected, however some letters are missing and the “O” is incorrectly filled in. This is a case where the extraction of word level bounding boxes may help performance.

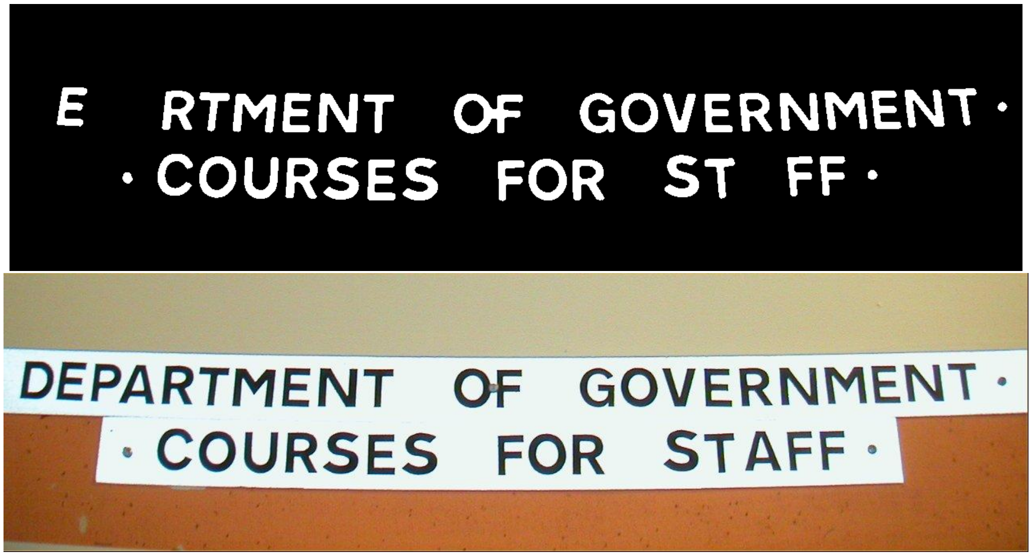


Figure 5.5: Similar to the previous example, many letters are correctly selected with only a few exceptions.

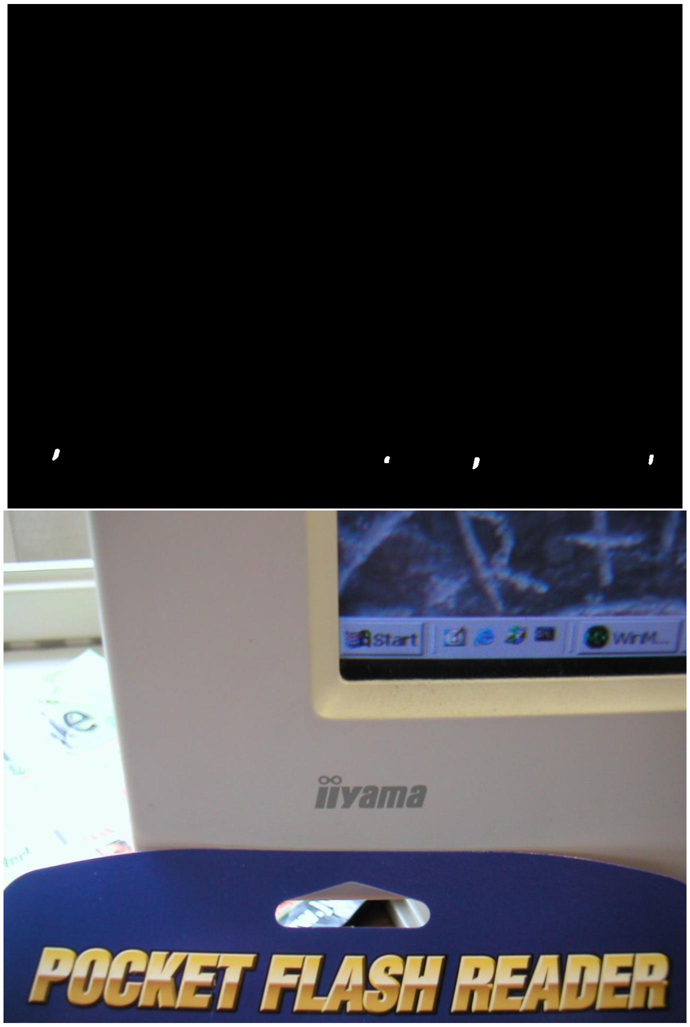


Figure 5.6: Unfortunately no text was successfully recovered from this example. Here our assumptions about the color consistency of text characters breaks down. Since color consistency is a fundamental assumption in our system, text is not successfully recovered. Figure 5.7 reveals further insight into the problem.

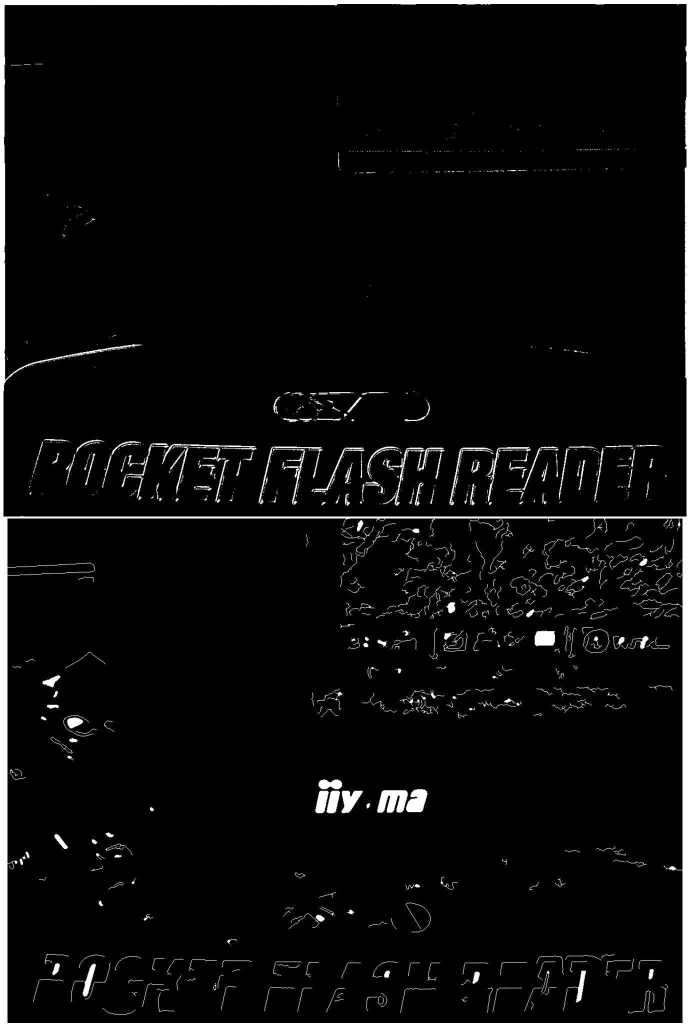


Figure 5.7: Top: sample classifier. Bottom: post processing. Notice the classifier does not correctly identify the text on the monitor, while the post processing does not correctly segment the words “pocket flash reader.”

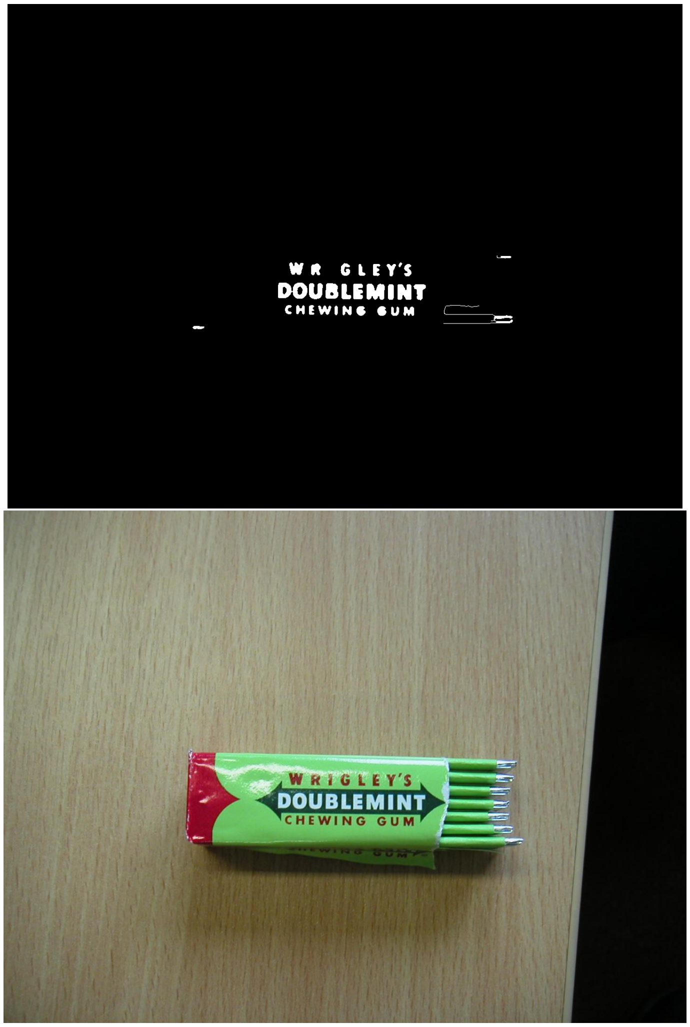


Figure 5.8: For this example, nearly all of the text is successfully detected, with only a minimal amount of incorrect detections.

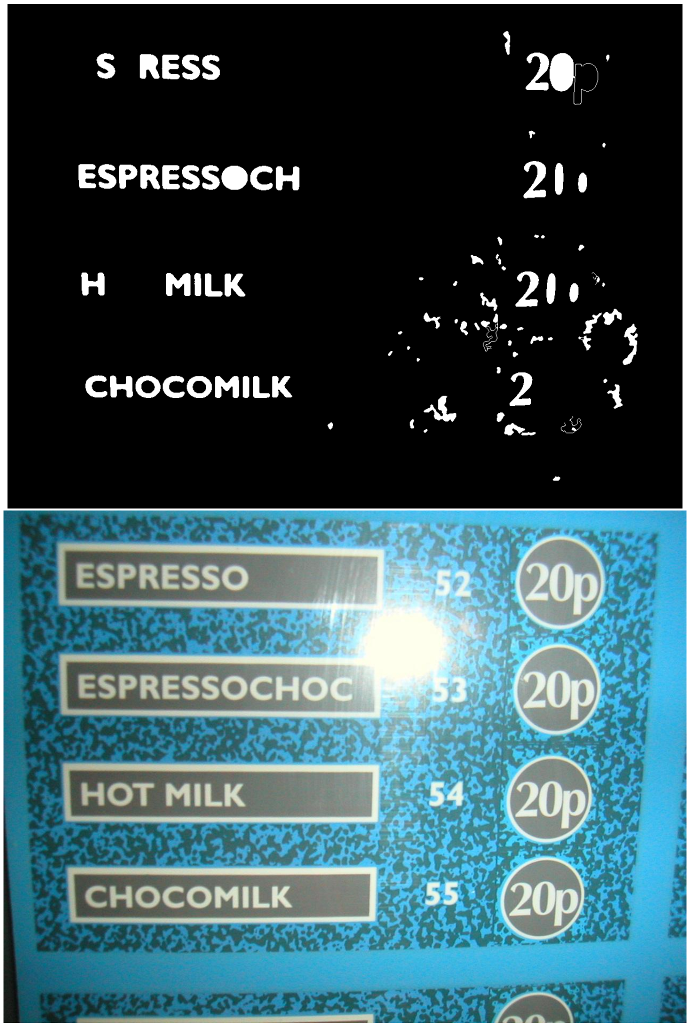


Figure 5.9: Notice the inability of the system to recover “52”, “53”, “54”, and “55”. Figure 5.10 provides further insight into the problem.

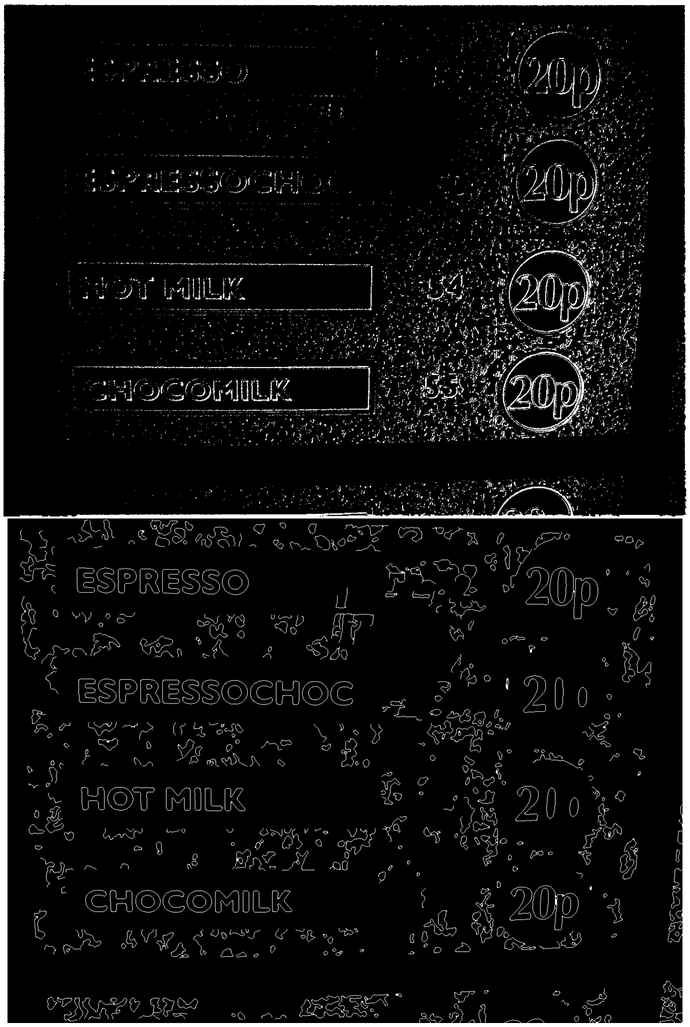


Figure 5.10: Top: sample classifier. Bottom: Canny edge detection results. Due to the complex background, the edge detector is unable to correctly segment the numbers “52”-“54”. Letters in words are lost by the classifiers.

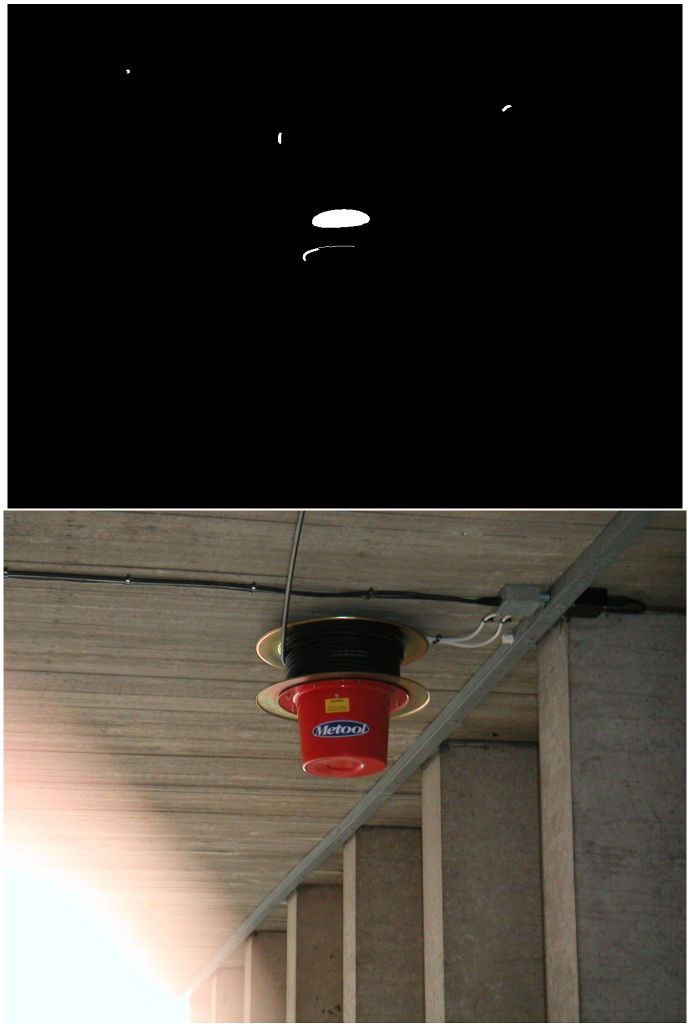


Figure 5.11: Interestingly, the system correctly identifies the location of the text in this image, but is unable to segment only letters, instead returning the entire region. If we were doing word level detection, this would be a successful result.

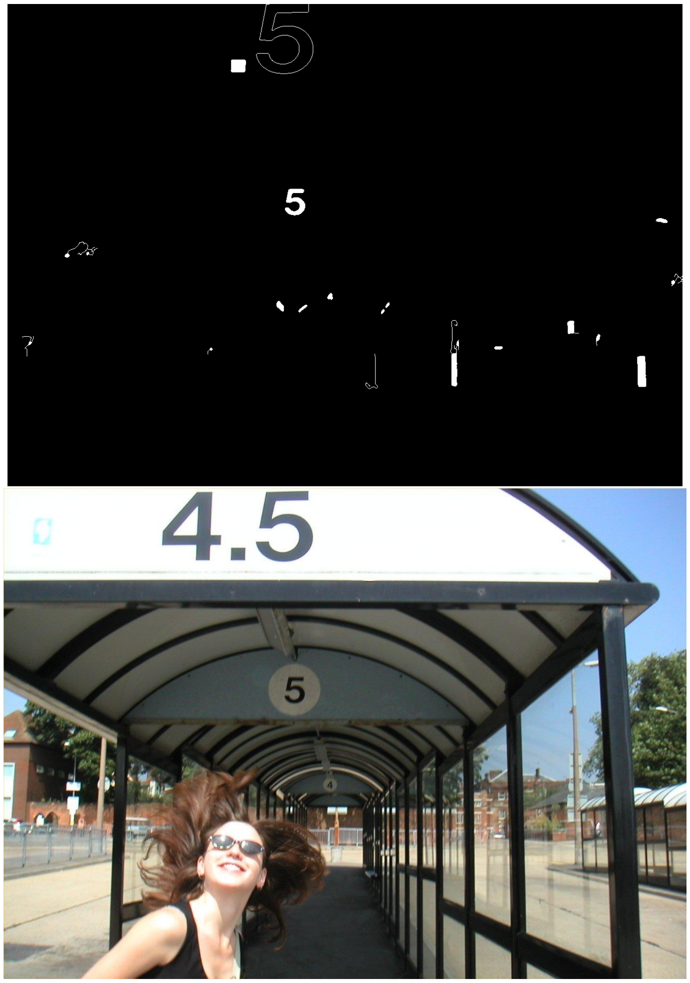


Figure 5.12: In this example the numbers “4.5” are partially missed, while other numbers are detected. Figure 5.13 shows a sample classifier and edge results to help explain this issue.

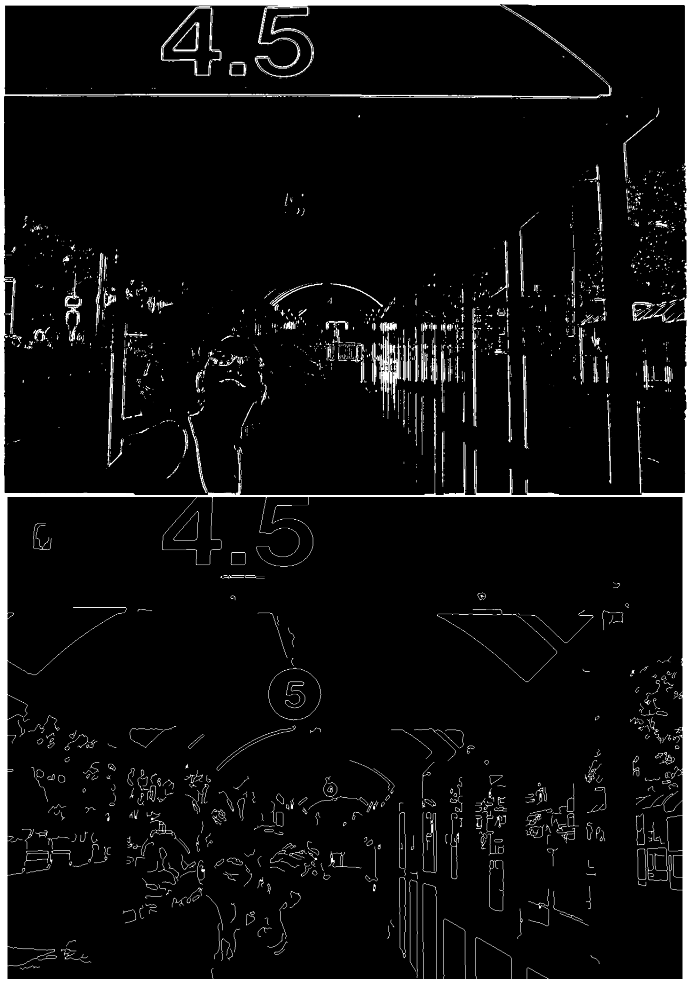


Figure 5.13: Both the sample classifier and the Canny edge detector perform quite well on this image. The issue instead resides in additional post processing steps. This may be corrected by improving post processing and reducing the number of static thresholds used.

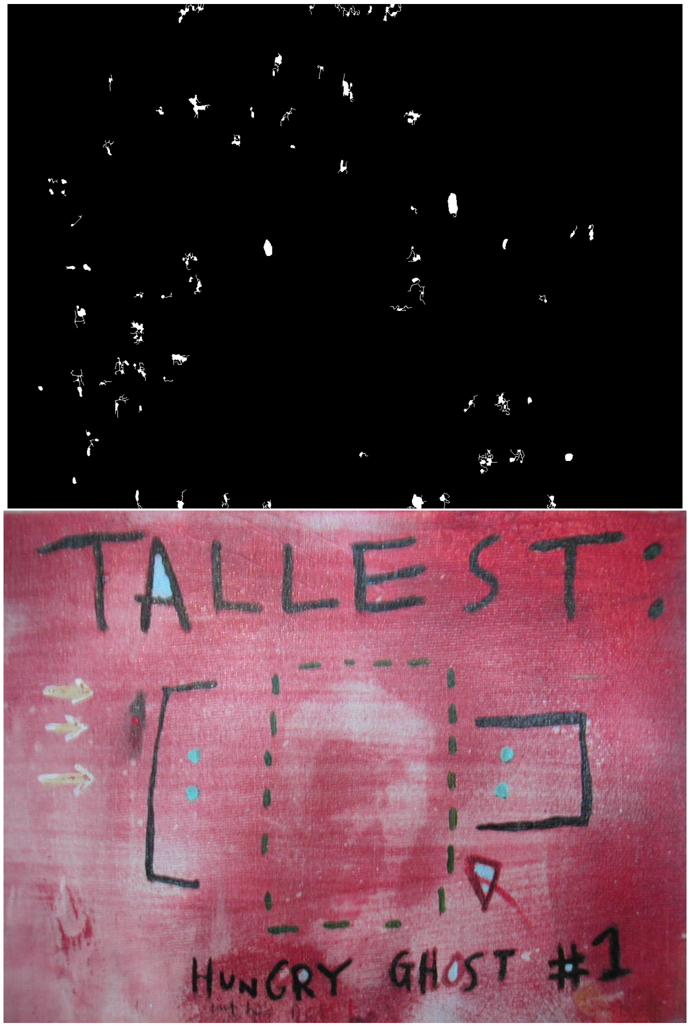


Figure 5.14: This example shows a very difficult image for our algorithm. Since this is a photo of a painting, lighting and texture causes changes in color within characters. Interiors are filled in some cases, and some letters are intentionally drawn in different colors. Neither the post processing step nor the classification step was able to successfully detect text in this image.

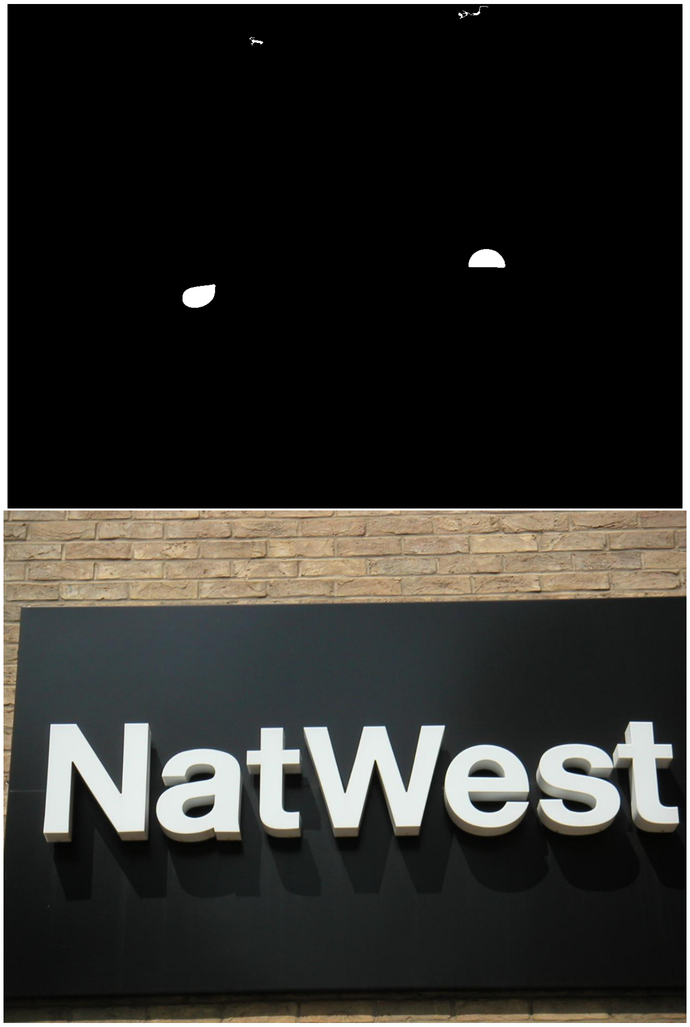


Figure 5.15: Although the classifiers performed well on this image, the Canny edge detector breaks down completely. Adjustment of Canny's parameters may help avoid this issue in the future. Figure 5.16 shows individual components to provide greater detail.

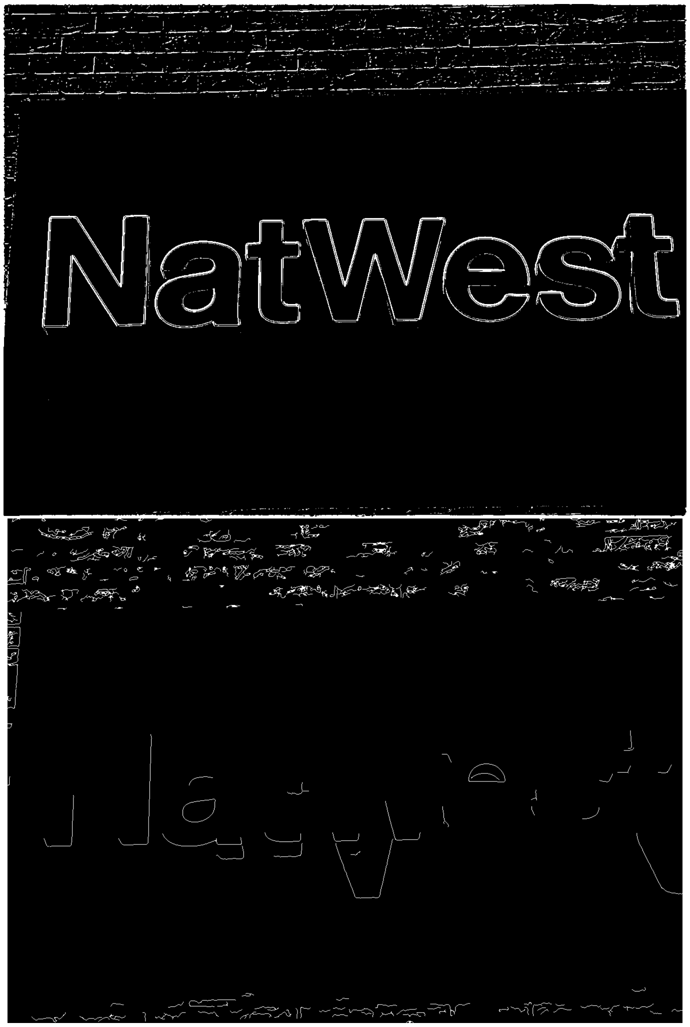


Figure 5.16: Top: sample classifier. Bottom: Canny edge detector. In this example, the edge detector fails to find the letters, causing the complete system to miss the text.

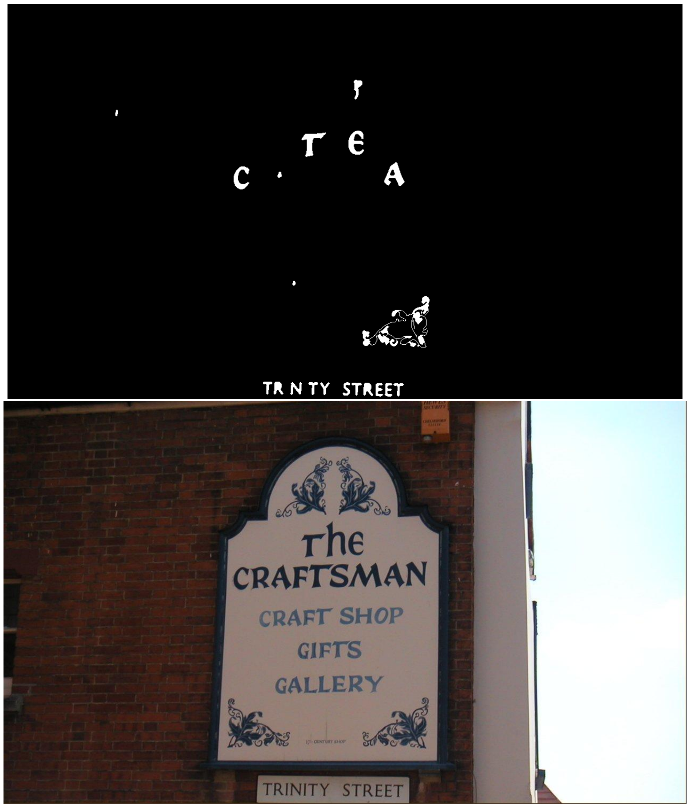


Figure 5.17: Here we see more typical behavior, some letters being selected and others missed, with an extra “letter like” region incorrectly labeled as text. Figure 5.18 shows further detail.

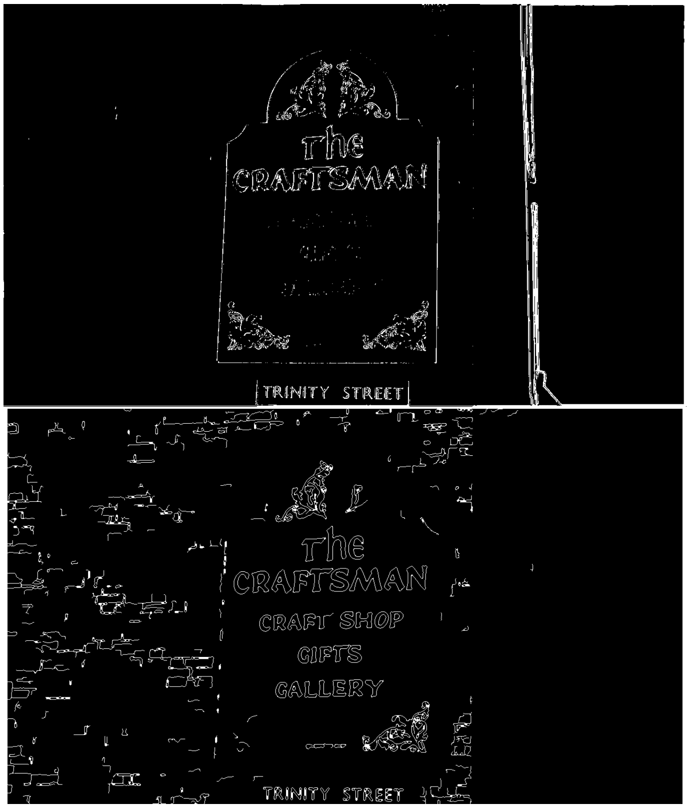


Figure 5.18: Here the edge detector (bottom) performs very well, however the DCT based classifiers (sample classifier, top) do not, resulting in missed letters.

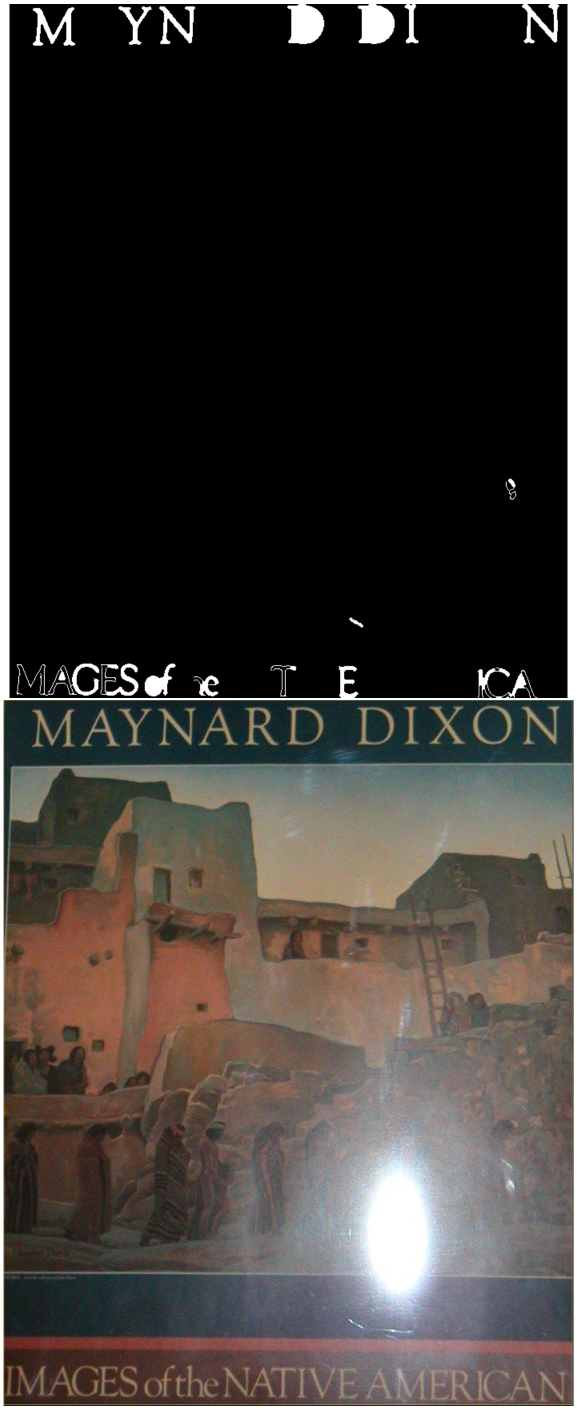


Figure 5.19: Similar to earlier examples, detection on the word level may improve this result, given that many characters are successfully detected, but most full words are not.

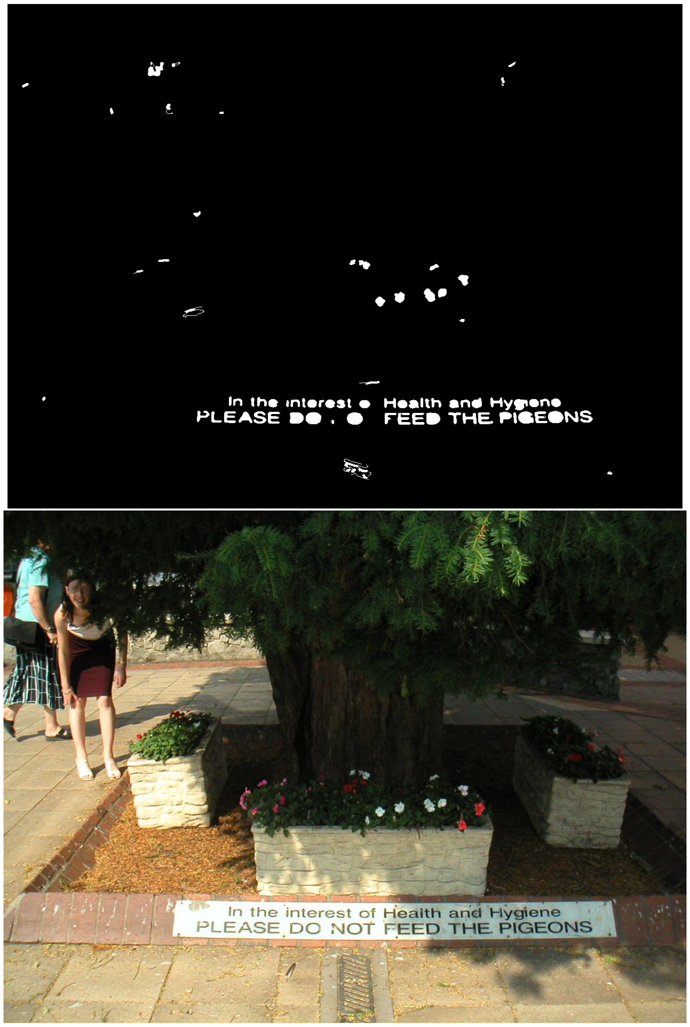

In the interest of Health and Hygiene
PLEASE DO NOT FEED THE PIGEONS

Figure 5.20: In this final example, we see the majority of letters are correctly selected, however there are quite a few false alarms. Adjustment of post processing or the inclusion of OCR may help address this situation.

Chapter 6

Conclusions and Future Work

We assert that the weighted combination of DCT based frequency filter classifiers and the addition of a post processing step produces better text detection results than any of the individual components do on their own. This appears to hold true when comparing the results of individual classifiers to the complete ensemble, but when comparing only the post processing step to the complete ensemble, results are mixed. For overall f-metric and recall, the complete ensemble outperforms only post processing, but post processing alone performs better in terms of precision. This result suggests there may be a better way of merging the classifiers with the post processing to create a complete ensemble which outperforms individual components according to all of the metrics.

Overall results, in particular the qualitative results, show that this approach is promising, but still leaves some room for improvement. Possible areas of improvement include integrating the system with OCR, improved threshold selection, and improved integration of classifiers with post processing. Extensions of this work include applying it to video and integrating a text removal technique for the video CAPTCHA application.

The integration of OCR with the system may reduce false alarms and help segment entire words more effectively. Likewise, by expanding the system to make use of video, redundant information between frames may be used to reduce false alarms. It is unclear how to increase the number of correct detections, however this may be possible by improving the selection of thresholds throughout the system. Currently many thresholds were chosen statistically, however it may be possible to use a learning algorithm to find better thresholds.

Potential improvements and changes aside, future work should also include application of this technique to video CAPTCHA since this was the original motivation for the project. Work needs to be done on extending the technique into video for detection, as well as in the area of text removal. Inpainting and other similar algorithms appear promising for this purpose.

This technique has shown interesting results so far. Qualitatively, it does a reasonable job on a wide range of images for the text detection task. Quantitatively, numbers are not as high as is desired, however word level metrics should be computed to get a better sense of relative performance to other algorithms. Areas where this approach shows difficulties can be further investigated and improved. Ideally with some minor modifications, this approach will be ready for its original task, text detection in video.

Bibliography

[1] http://algoval.essex.ac.uk/icdar/datasets.html.

[2] Roy S. Berns. *Billmeyer and Staltzman's Principles of Color Technology.* Wiley, 3rd edition, 2000.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 2000.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[5] D. Chen, H. Bourlard, and J.P. Thiran. Text identification in complex background using svm. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 621–626. IEEE Computer Society, 2001.

[6] D. Chen, J.M. Odobez, and H. Bourlard. Text detection and recognition in images and video frames. *Pattern Recognition*, 37(3):595–608, 2004.

[7] M. Cooper, T. Liu, and E. Rieffel. Video segmentation via temporal pattern classification. *IEEE transactions on multimedia*, 9(3):610–618, 2007.

[8] D. Crandall, S. Antani, and R. Kasturi. Extraction of special effects caption text events from digital video. *International Journal on Document Analysis and Recognition*, 5(2):138–157, 2003.

[9] R. Datta, J. Li, and J.Z. Wang. Exploiting the human-machine gap in image recognition for designing CAPTCHAs. *IEEE Transactions on Information Forensics and Security*, 4(3):504–518, 2009.

[10] R. Dugad and N. Ahuja. A fast scheme for altering resolution in the compressed domain. In *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on.*, volume 1, pages 213–218, 1999.

[11] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2963–2970. IEEE, 2010.

[12] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski. Coordinates for instant image cloning. *ACM Transactions on Graphics (TOG)*, 28(3):1–9, 2009.

[13] Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning*

theory, pages 23–37. Springer, 1995.

[14] Rafael C. Gonzalez and Ricard E. Woods. *Digital Image Processing*. Pearson Education, Inc., 3rd edition, 2008.

[15] R. Jiang, F. Qi, L. Xu, G. Wu, and K. Zhu. A learning-based method to detect and segment text from scene images. *Journal of Zhejiang University-Science A*, 8(4):568–574, 2007.

[16] Cheolkon Jung, Qifeng Liu, and Joongkyu Kim. Accurate text localization in images based on svm output scores. *Image and Vision Computing*, 27(9):1295 – 1301, 2009.

[17] K. Jung, K. In Kim, and A. K. Jain. Text information extraction in images and video: a survey. *Pattern Recognition*, 37(5):977–997, 2004.

[18] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):319–336, 2009.

[19] K.I. Kim, K. Jung, and J.H. Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1631–1639, 2003.

[20] K.A. Kluever and R. Zanibbi. Balancing usability and security in a video CAPTCHA. In *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, 2009.

[21] V. Kolmogorov and R. Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 26(2):147–159, 2004.

[22] L.I. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[23] C.C. Lee, Y.C. Chiang, and C.Y. Shih. Subject Caption Detection in News Video with Complex Picture. In *Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering*, pages 426–430. IEEE Computer Society, 2009.

[24] C.W. Lee, K. Jung, and H.J. Kim. Automatic text detection and removal in video sequences. *Pattern Recognition Letters*, 24(15):2607–2623, 2003.

[25] H. Li, D. Doermann, and O. Kia. Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing*, 9(1):147–156, 2000.

[26] H. Li, K.N. Ngan, and Q. Liu. FaceSeg: automatic face segmentation for real-time video. *IEEE Transactions on Multimedia*, 11(1):77–88, 2009.

[27] J. Li and J.Z. Wang. Real-time computerized annotation of pictures. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 911–920. ACM, 2006.

[28] R. Lienhart and W. Effelsberg. Automatic text segmentation and text recognition for video indexing. *Multimedia systems*, 8(1):69–81, 2000.

[29] R. Lienhart and A. Wernicke. Localizing and segmenting text in images and videos. *IEEE Transactions on circuits and systems for video technology*, 12(4):256–268, 2002.

[30] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, Feb 1994.

[31] S.M. Lucas. ICDAR 2005 text locating competition results. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 80–84. IEEE, 2005.

[32] M.R. Lyu, J. Song, and M. Cai. A comprehensive method for multilingual video text detection, localization, and extraction. *IEEE transactions on circuits and systems for video technology*, 15(2):243–255, 2005.

[33] S.G. Mallat et al. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.

[34] J.L. Mitchell, C. Fogg, W.B. Pennebaker, and D.J. LeGall. *MPEG video compression standard*. Kluwer Academic Publishers, 1996.

[35] Y. Nakajima, A. Yoneyama, H. Yanagihara, and M. Sugano. Moving-object detection from mpeg coded data. In *Proceedings of SPIE*, pages 988–996. Society of Photo-Optical Instrumentation Engineers, 1997.

[36] X. Qian, G. Liu, H. Wang, and R. Su. Text detection, localization, and tracking in compressed video. *Signal Processing: Image Communication*, 22(9):752–768, 2007.

[37] M. Rubinstein, A. Shamir, and S. Avidan. Improved seam carving for video retargeting. *ACM Transactions on Graphics-TOG*, 27(3):16–16, 2008.

[38] Z. Saidane and C. Garcia. Automatic scene text recognition using a convolutional neural network. In *Proceedings of the Second International Workshop on Camera-Based Document Analysis and Recognition (CB-DAR)*, 2007.

[39] Z. Saidane and C. Garcia. Robust binarization for video text recognition. In *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 874–879, 2007.

[40] T. Sato, T. Kanade, E.K. Hughes, and M.A. Smith. Video ocr for digital news archives. In *IEEE Intl. Workshop on Content-Based Access of*

Image and Video Databases, pages 52–60. IEEE, 1998.

[41] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *International Conference on Computer Vision*, pages 1154–1160. IEEE, 1998.

[42] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[43] Y. Shi and W.C. Karl. A real-time algorithm for the approximation of level-set-based curve evolution. *IEEE transactions on image processing*, 17(5):645–656, 2008.

[44] Y. Song and W. Wang. Text Localization and Detection for News Video. In *2009 Second International Conference on Information and Computing Science*, volume 2, pages 98–101. IEEE, 2009.

[45] M. Sonka, V. Hlavac, and R. Boyle. *Image processing, analysis, and machine vision second edition.* International Thomson, 1999.

[46] L.P. Sosa, SM Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 Robust Reading Competitions. In *In Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 682–687. Citeseer, 2003.

[47] K. Viswanath, J. Mukherjee, and PK Biswas. Image filtering in the block DCT domain using symmetric convolution. *Journal of Visual Communication and Image Representation*, 22(2):141–152, 2011.

[48] X. Wang, L. Huang, and C. Liu. A video text location method based on background classification. *International Journal on Document Analysis and Recognition*, 13(3):173–186, 2010.

[49] V. Wu, R. Manmatha, and EM Riseman. Textfinder: an automatic system to detect and recognize text inimages. *IEEE Transactions on pattern analysis and machine intelligence*, 21(11):1224–1229, 1999.

[50] W. Wu, X. Chen, and J. Yang. Detection of text on road signs from video. *IEEE Transactions on Intelligent Transportation Systems*, 6(4):378–390, 2005.

[51] L. Xu and K. Wang. Extracting text information for content-based video retrieval. *Lecture Notes in Computer Science*, pages 58–69, 2008.

[52] Q. Ye, Q. Huang, W. Gao, and D. Zhao. Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23(6):565–576, 2005.

[53] Y. Zhan, W. Wang, and W. Gao. A robust split-and-merge text segmentation approach for images. In *Proceedings of the 18th international*

Conference on Pattern Recognition, volume 2, pages 1002–1005. Citeseer, 2006.

[54] F. Zhong, X. Qin, and Q. Peng. Transductive segmentation of live video with non-stationary background. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2189–2196. IEEE, 2010.

[55] Y. Zhong, AK Jain, and M.P. Dubuisson-Jolly. Object tracking using deformable templates. *IEEE transactions on pattern analysis and machine intelligence*, 22(5):544–549, 2000.

[56] Y. Zhong, H. Zhang, and A.K. Jain. Automatic caption localization in compressed video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):385–392, 2000.

[57] H. Zhou, Y. Yuan, and C. Shi. Object tracking using SIFT features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009.

Vita

David K. Snyder was born in Rochester, New York. He attended Rochester Institute of Technology from 2005 to 2009 and received a Bachelor of Science in Imaging Science. He began work toward a Master of Science in Imaging Science at Rochester Institute of Technology in the Fall of 2009.

Permanent address: 945 Copper Kettle Road, Webster, NY 14580

This dissertation was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.