VISUAL PRODUCT RECOGNITION FROM RETAIL SCENES

by

Alper Aydın

B.S., Electronics Engineering, Sabancı University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronic Engineering
Boğaziçi University

2017

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Ceyhun Burak Akgül for his help. His ideas and guidance were indispensible in this thesis work. He has been an example to me with his academic and professional life. I consider myself very lucky to benefit from his vision and experience.

I would like to thank Assoc. Prof. Burak Acar for accepting to work on this thesis, and for his patience. It is a honour to be his student. I am also thankful to the rest of my thesis committee: Assist. Prof. İlke Ercan, Assoc. Prof. Engin Erzin and for their review and criticism.

I am thankful to Vispera members. Working with them is a pleasure. I am especially grateful to Dr. Erdem Yörük for his immense guidance and detailed review. I also would like to thank Kaan Can Fidan, Levent Özyıldırım, and Kaan Öner for sharing their extensive knowledge.

I would like to thank my friends for believing in me and for their moral support.

Most importantly, I would like to thank my parents Aynur and Ömer Aydın, my sister Nisan Aydın for their unending love and support.

ABSTRACT

VISUAL PRODUCT RECOGNITION FROM RETAIL SCENES

In this thesis project, we explore the problem of recognizing the products in retail scenes. We proposed a method where we localize the products by detecting their separating boundaries and recognize the product images using supervised learning methods.

We experimented with various image descriptors like Histogram of Oriented Gradients, Local Binary Pattern Histograms, and response features to a Root Filter Set. Concurrently we tried various classifiers like Quadratic Classifier, AdaBoost with single level decision trees and Support Vector Machines. We found out that using Histograms of Oriented Gradients as descriptors and classifying them with a Support Vector Machine enabled us to achieve the highest accuracy with a query time constraint. We achieved 88.5% accuracy at classifying a set of sparsely collected descriptors from boundary and non-boundary regions. Using the trained SVM classifier we calculated the probability of the existence of a product boundary for all possible locations with a sliding window. We localized the boundaries using probability peak detection and achieved a product boundary detection precision and recall over 75%. The same method is used to detect shelf boundaries with 97% precision.

We described the product images with sparse and dense SIFT descriptors and achieved 96% accuracy using a bag of words approach on a 20 class subset of products. We experimented with various techniques and verified that using dense SIFT descriptors and random codebook generation instead of K-means based clustering yields better performance at recognizing retail products.

ÖZET

PERAKENDE İMGELERİNDEN GÖRSEL ÜRÜN TANINMASI

Bu tez projesinde perakende sahnelerinde ürünleri tanıma sorununu araştırdık. Ürünleri ayıran sınırları tespit ederek ürünleri lokalize ettiğimiz ve öğretici ile öğrenme yöntemlerini kullanarak ürün görsellerini tanıyabildiğimiz bir method önerdik.

Yönlü Gradyan Histogramları, Yerel İkili Örüntü Histogramları ve Root Filter Set yanıt tanımlayıcı gibi çeşitli resim tanımlayıcılarını denedik. Eş zamanlı olarak, Kuadratik Sınıflandırıcı, Destek Vektör Makinesi, ve tek kademeli karar ağaçları ile oluşturulan AdaBoost Sınıflandırıcı gibi çeşitli sınıflandırıcıları denedik.Bir sorgu süresi kısıtlaması altında Yönlü Gradyan Histogramları'nı tanımlayıcı olarak kullanıp bunları bir Destek Vektör Makinesi ile sınıflandırmamızın, en yüksek doğruluğu elde etmemizi sağladığını gördük. Ürün sınırı bölgelerinden ve ürün sınırı olmayan bölgelerden seyrek şekilde toplanmış tanımlayıcıların sınıflandırılmasında %88.5 doğruluk elde ettik. Bu SVM sınıflandırıcıyı kullanarak, test görsellerinde tüm olası yerler için, bir ürün sınırının var olma ihtimalini, kayan bir pencere ile hesapladık. Oluşan olasılık sinyalinde tepe detektörü kullanarak ürün sınırlarını lokalize ettik ve %75'ten yüksek bir ürün sınırı algılama kesinliği ve geri getirme yüzdesi elde ettik. Aynı yöntemi raf sınırlarını %97 kesinlik ile tespit etmek için kullandık.

Elde ettiğimiz ürün imgelerini seyrek ve yoğun SIFT tanımlayıcıları ile tanımadık ve 20 sınıflık bir ürün alt kümesinde, kelime torbası yaklaşımı kullanarak %96 doğruluk elde ettik. Ceşitli tekniklerle denemeler yaptık ve yoğun SIFT tanımlayıcıları kullanmanın ve K-ortalamalı kümeleme yerine rastgele kod tablosu üretmemizin perakende ürünlerini tanıma probleminde daha iyi performans sağladığını doğruladık.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES viii
LIST OF TABLES xvii
LIST OF SYMBOLS xxii
LIST OF ACRONYMS/ABBREVIATIONS xxiii
1. INTRODUCTION 1
2. LITERATURE REVIEW 6
2.1. Instance Recognition 6
2.2. Product Recognition 8
2.3. Boundary Detection 10
3. SYSTEM OVERVIEW 12
3.1. Boundary Detection 12
3.2. Product Recognition 15
4. BOUNDARY DETECTION 19
4.1. Visual Descriptors 19
4.2. Classifiers 22
4.3. Experiments with Filter Response Features from The Root Filter Set . 28
4.4. Experiments with Histograms of Oriented Gradients 47
4.5. Experiments with Local Binary Pattern Histograms 68
4.6. Score Signal Peak Detection 76
4.7. Comparison to Benchmark Method 91
4.8. Shelf Boundary Detection 93
5. VISUAL PRODUCT RECOGNITION 97
5.1. Datasets 97
5.2. Descriptor Matching with Geometric Verification 99
5.3. Image Retrieval Using Visual Words 105
5.4. Image Retrieval Using Visual Words with Non-Geometric Re-Ranking . 114

5.5. Image Retrieval Using Visual Words with Geometric Re-Ranking . . . 117
5.6. Image Retrieval Using Random Sampled Densely Extracted Visual Words 121
6. DISCUSSION . . . 126
6.1. Boundary Detection . . . 126
6.2. Product Recognition . . . 128
7. CONCLUSION . . . 130
REFERENCES . . . 131
APPENDIX A: DISCRETE ADABOOST . . . 137
APPENDIX B: BOUNDARY DETECTION RESULTS . . . 138

LIST OF FIGURES

Figure 1.1. Sample retail scene including soft drinks 2

Figure 1.2. Shelf boundaries, product boundaries and products of various classes 2

Figure 1.3. Proposed algorithm pipeline 3

Figure 1.4. Shelf line detection and region extraction 3

Figure 1.5. Product boundary detection and product image extraction 4

Figure 3.1. Types of boundary regions 13

Figure 3.2. A sample shelf image with corresponding boundary annotations . 14

Figure 3.3. Boundary detection pipeline 14

Figure 4.1. Root filter set 21

Figure 4.2. AdaBoost block diagram 25

Figure 4.3. A sample shelf image 29

Figure 4.4. A filter and corresponding response map from the sample image . 29

Figure 4.5. A sample stripe response map 30

Figure 4.6. Positive and negative annotations 30

Figure 4.7. 2 vertical cell division and 8 vertical cell division patterns 31

Figure 4.8. AdaBoost weak classifier number vs. AdaBoost Accuracy in Stripe width and Cell Division Experiment 34

Figure 4.9. Extended Root Filter Set 35

Figure 4.10. AdaBoost weak classifier number vs. AdaBoost Accuracy in Filter Set Size Experiment 37

Figure 4.11. Cell division patterns for new variants 38

Figure 4.12. AdaBoost weak classifier number vs. AdaBoost Accuracy in Cell Division and Filter Size Experiment. 40

Figure 4.13. AdaBoost weak classifier number vs. AdaBoost Accuracy graph. 41

Figure 4.14. Query time vs. Accuracy of FRF Variant - Classifier couples. . . . 44

Figure 4.15. Query time vs. Accuracy of FRF Variants for QC 45

Figure 4.16. Query time vs. Accuracy of FRF Variants for AdaBoost 100 . . . 45

Figure 4.17. Query time vs. Accuracy of FRF Variants for AdaBoost 150 . . . 46

Figure 4.18. Query time vs. Accuracy of FRF Variants for SVM 46

Figure 4.19. Gradient binning pattern with a magnitude threshold 49

Figure 4.20. Gradient binning pattern for HOG variant V1 50

Figure 4.21. Gradient binning pattern for HOG variant V2 50

Figure 4.22. AdaBoost weak classifier number vs. AdaBoost accuracy graph . . 52

Figure 4.23. Gradient binning pattern for HOG variant V4 53

Figure 4.24. Gradient binning pattern for HOG variant V5 53

Figure 4.25. A sample gradient magnitude image 57

Figure 4.26. Cumulative intensity histogram of the sample magnitude images . 57

Figure 4.27. The cell division patterns 60

Figure 4.28. Histogram phase binning schemes for Alternative Histogram Binnings Experiment 61

Figure 4.29. Query time vs. Accuracy of Variant - Classifier couples 66

Figure 4.30. Query time vs. Accuracy of Variants for QC 66

Figure 4.31. Query time vs. Accuracy of Variants for AdaBoost 67

Figure 4.32. Query time vs. Accuracy of Variants for SVM 67

Figure 4.33. A sample shelf image 68

Figure 4.34. LBP image visualization for the sample shelf image 68

Figure 4.35. AdaBoost weak classifier number vs. AdaBoost accuracy graph . . 72

Figure 4.36. Query time vs. Accuracy of LBP Variant - Classifier couples . . . 75

Figure 4.37. Description of stripes and score calculation 77

Figure 4.38. Score histograms for test descriptors 78

Figure 4.39. Posterior probability of being a positive test sample 79

Figure 4.40. Best fit sigmoid over the posterior probabilities 80

Figure 4.41. Score signal and corresponding probability signal 80

Figure 4.42. Classifier decision for different windows 81

Figure 4.43. Effect of smoothing for a query image 82

Figure 4.44. Distribution of annotation widths 83

Figure 4.45. Some annotation examples 83

Figure 4.46. Smoothed probability image with detected peak indices 84

Figure 4.47. Precision, and Recall vs. Probability threshold 85

Figure 4.48. Precision vs. Recall curve at varying thresholds 86

Figure 4.49. A sample shelf image (Soft drinks) 87

Figure 4.50. A sample shelf image (Personal care) 87

Figure 4.51. A sample shelf image (Confectionery) 88

Figure 4.52. A sample shelf image (Confectionery) 88

Figure 4.53. A sample shelf image (Soft drinks) 89

Figure 4.54. A sample shelf image (Confectionery) 89

Figure 4.55. A sample shelf image (Confectionery) 90

Figure 4.56. A sample shelf image (Personal care) 90

Figure 4.57. A sample shelf image (Confectionery) 91

Figure 4.58. Horizontal gradients 91

Figure 4.59. Benchmark comparison example (Personal hygiene products) . . . 92

Figure 4.60. Benchmark comparison example (Chocolate) 92

Figure 4.61. Benchmark comparison example (Confectionery) 93

Figure 4.62. Benchmark comparison example (Biscuit) 93

Figure 4.63. A sample retail scene and corresponding shelf boundary annotations 94

Figure 4.64. Smoothed probability signal, detected peaks drawn over a sample query image 95

Figure 5.1. Sample images from Dataset 1 98

Figure 5.2. Sample images from Dataset 2 98

Figure 5.3. Training phase 99

Figure 5.4. Matching test and training descriptors 100

Figure 5.5. SSE graph for a test image from 2. class. 101

Figure 5.6. SSE graph for a test image from 6. class. 102

Figure 5.7. Number of elements for 500 centers 106

Figure 5.8. Number of elements for 2000 centers 107

Figure 5.9. Number of elements for 5000 centers 107

Figure 5.10. Number of elements for 500 clusters 112

Figure 5.11. Number of elements for 5000 clusters 112

Figure 5.12. Re-ranking accuracy vs. Number of re-ranked close matches 117

Figure 5.13. Re-ranking accuracy vs. Number of re-ranked mathces 120

Figure 5.14. Number of elements for 5000 random visual words 123

Figure 5.15. Sample images from Class 1 and Class 2 124

Figure A.1. Discrete AdaBoost Pseudocode 137

Figure B.1. Product boundary detection sample result 1 138

Figure B.2. Product boundary detection sample result 2 138

Figure B.3. Product boundary detection sample result 3 139

Figure B.4. Product boundary detection sample result 4 139

Figure B.5. Product boundary detection sample result 5 140

Figure B.6. Product boundary detection sample result 6 140

Figure B.7. Product boundary detection sample result 7 141

Figure B.8. Product boundary detection sample result 8 141

Figure B.9. Product boundary detection sample result 9 142

Figure B.10. Product boundary detection sample result 10 142

Figure B.11. Product boundary detection sample result 11 143

Figure B.12. Product boundary detection sample result 12 143

Figure B.13. Product boundary detection sample result 13 144

Figure B.14. Product boundary detection sample result 14 144

Figure B.15. Product boundary detection sample result 15 145

Figure B.16. Product boundary detection sample result 16 145

Figure B.17. Product boundary detection sample result 17 146

Figure B.18. Product boundary detection sample result 18 146

Figure B.19. Product boundary detection sample result 19 147

Figure B.20. Product boundary detection sample result 20 147

Figure B.21. Product boundary detection sample result 21 148

Figure B.22. Product boundary detection sample result 22 148

Figure B.23. Product boundary detection sample result 23 149

Figure B.24. Product boundary detection sample result 24 149

Figure B.25. Product boundary detection sample result 25 150

Figure B.26. Product boundary detection sample result 26 150

Figure B.27. Product boundary detection sample result 27 151

Figure B.28. Product boundary detection sample result 28 151

Figure B.29. Product boundary detection sample result 29 152

Figure B.30. Product boundary detection sample result 30 152

Figure B.31. Product boundary detection sample result 31 153

Figure B.32. Product boundary detection sample result 32 153

Figure B.33. Product boundary detection sample result 33 154

Figure B.34. Product boundary detection sample result 34 154

Figure B.35. Product boundary detection sample result 35 155

Figure B.36. Product boundary detection sample result 36 155

Figure B.37. Product boundary detection sample result 37 156

Figure B.38. Product boundary detection sample result 38 156

Figure B.39. Product boundary detection sample result 39 157

Figure B.40. Product boundary detection sample result 40 157

LIST OF TABLES

Table 3.1. Inverted file structure 17

Table 4.1. SVM parameters found using 5 fold grid search 32

Table 4.2. 5 fold cross validation average accuracies of classifiers over the training sets 33

Table 4.3. Accuracy of classifiers over the test sets 33

Table 4.4. SVM parameters found using 5 fold grid search 35

Table 4.5. 5 fold cross validation average accuracies of classifiers over the training sets 36

Table 4.6. Accuracy of classifiers over the test sets 36

Table 4.7. 5 fold cross validation average accuracies of classifiers over the training sets 38

Table 4.8. Accuracy of classifiers over the test set 39

Table 4.9. Accuracy of classifiers over the training set 40

Table 4.10. Accuracy of classifiers over the test set 40

Table 4.11. Average query times for Filter Response Features 42

Table 4.12. FRF Variants and classifier accuracies 43

Table 4.13. Average of 5 fold accuracies of classifiers over the training set . . . 51

Table 4.14. Accuracy of classifiers over the test sets 51

Table 4.15. Average of 5 fold accuracies of classifiers over the training set . . . 54

Table 4.16. Accuracy of classifiers over the test sets 54

Table 4.17. Average of 5 fold accuracies of classifiers over the training sets . . 55

Table 4.18. Accuracy of classifiers over the test sets 55

Table 4.19. Average of 5 fold accuracies of classifiers over the training sets . . 56

Table 4.20. Accuracy of classifiers over the test sets 56

Table 4.21. Average of 5 fold accuracies of classifiers over the training sets . . 58

Table 4.22. Accuracy of classifiers over the test sets 58

Table 4.23. Average of 5 fold accuracies of classifiers over the training sets . . 59

Table 4.24. Accuracy of classifiers over the test sets 59

Table 4.25. Average of 5 fold accuracies of classifiers over the training sets . . 60

Table 4.26. Accuracy of classifiers over the test sets 61

Table 4.27. Average of 5 fold accuracies of classifiers over the training sets . . 62

Table 4.28. Accuracy of classifiers over the test sets 62

Table 4.29. Average query times for HOG variant - classifier combinations in Phase Binning Experiment and Unsigned Phase Binning Experiment 63

Table 4.30. Average query times for HOG variant - classifier combinations in Image Height Experiment 63

Table 4.31. Average query times for HOG variant - classifier combinations in Magnitude Weighted Histogram Experiment 64

Table 4.32. Average query times for HOG variant- classifier combinations in Alternative Histogram Binnings Experiment 65

Table 4.33. LBP examples 70

Table 4.34. Average of 5 fold accuracies of classifiers over the training sets . . 71

Table 4.35. Accuracy of classifiers over the test sets 71

Table 4.36. Average of 5 fold accuracies of classifiers over the training sets . . 72

Table 4.37. Accuracy of classifiers over the test sets 73

Table 4.38. Average query times for LBP variants and classifiers 74

Table 4.39. LBP variants and classifier accuracies 74

Table 4.40. Optimal variant, optimal classifier, query time, and accuracy for different descriptor types 76

Table 5.1. Confusion matrix for Experiment on Dataset 1 103

Table 5.2. Confusion matrix for Experiment on Dataset 2 103

Table 5.3. Average number of SIFT descriptors and test times for Experiment on Dataset 2 104

Table 5.4. Accuracy for different implementations 108

Table 5.5. Average test times for different number of centers 110

Table 5.6. Confusion matrix for the best implementation 110

Table 5.7. Ratio of test images with at least one relevant match among top retrieved training images 111

Table 5.8. Accuracy for different implementations 113

Table 5.9. Average test times for different number of centers 114

Table 5.10. Accuracy for Image Retrieval Using Visual Words with Non-Geometric Re-Ranking 115

Table 5.11. Average test times for different number of centers 116

Table 5.12. Accuracy and average test times for K=5000, T_{stop}=1500 116

Table 5.13. Accuracy for different ϵ_n parameters 119

Table 5.14. Average test times for K=5000, T_{stop}=1500 119

Table 5.15. Optimum results for different methods 121

Table 5.16. Confusion matrix 123

Table 5.17. Ratio of test images which have at least one correct close match among retrieved training images 124

LIST OF SYMBOLS

h	Classification result of a weak classifier
v	Visual word
x_i	Feature vector with index i
y_i	Label of feature vector with index i
α	Weight of weak classifier
μ_y	Mean of the feature vectors with label y
ξ	Slack variable
Σ_y	Covariance matrix of the feature vectors with label y

LIST OF ACRONYMS/ABBREVIATIONS

FRF	Filter Response Features
LBP	Local Binary Patterns
QC	Quadratic Classifier
RFS	Root Filter Set
SIFT	Scale Invariant Feature Transform
SSE	Sum Squared Error
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
tf-idf	term frequency - inverse document frequency

1. INTRODUCTION

The main challenge of computer vision is to automate the tasks the human visual system can easily perform [1]. Recent developments in the fields of image descriptors, machine learning, and computing hardware technologies push the boundaries of practical applications where computer vision is not only helpful but also a necessary tool. Any task that requires the processing of visual information is of interest for this interdisciplinary field. The visual object recognition problem is one of the most important of those tasks. Its importance is due to its complexity and the broad scope of areas it can be applied to.

Retail stores are one of the physical spaces in which computers can automate the aforementioned recognition task. A sample retail scene is shown in Figure 1.1 and Figure 1.2. The counts and locations of stock keeping units (SKU)[1] are of huge importance in retail sector since the product placement has an effect in sales volume. This problem is particularly relevant for fast moving consumer goods (FMCG) and retailer companies and its solution via computer vision-based recognition tools is of great economic benefit. Humans are charged to recognize the products on shelves in order to make sure that the placement is in compliance with a pre-planned placement schematic, which is called "planogram". The developments in the field of computer vision can enable us to achieve faster, cheaper and more accurate product recognition.

In this thesis project we developed and tested visual recognition methods to parse shelf scenes and then to recognize the products in the parsed shelf regions. We implemented extensive tests using the data provided by Vispera Information Technologies [3]. The pipeline for the proposed algorithm is given in Figure 1.3.

[1] "A stock-keeping unit, or sku is an individually identifiable item stored in a specific location and tracked by an inventory system" [2].



Figure 1.1. Sample retail scene including soft drinks



Figure 1.2. Shelf boundaries, product boundaries and products of various classes

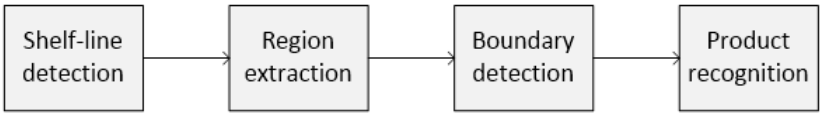


Figure 1.3. Proposed algorithm pipeline

Given a rectified retail scene image, firstly, the shelf lines are needed to be detected to find regions of interests. In order to do that, a shelf line detector is developed. For all shelf regions, the boundaries between products are detected with the same approach used in the shelf line detection step. The retail scene images are pre-rectified so that the shelf lines are horizontal and the shelf walls are vertical.

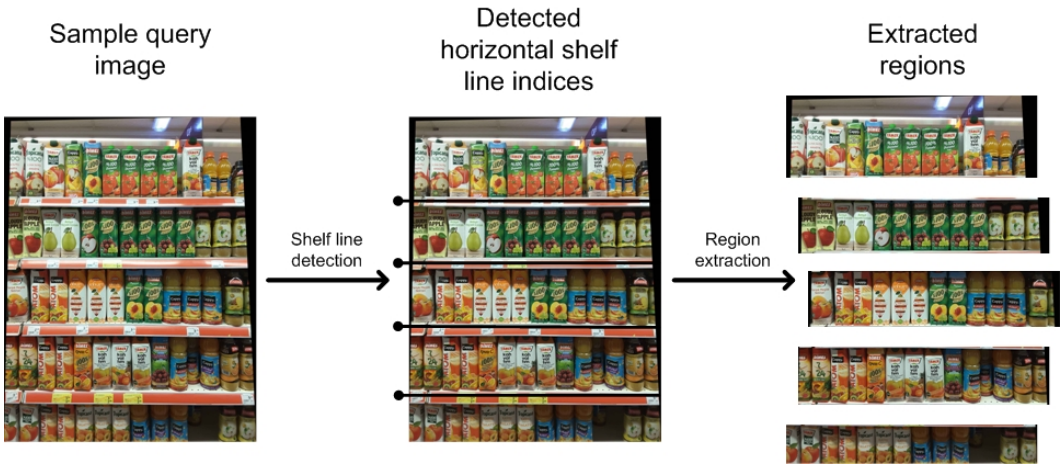


Figure 1.4. Shelf line detection and region extraction

After the boundaries between products are detected, product images are extracted. For completeness, we implement various recognition methods on a subset of products and document their performances. The region extraction and recognition tools are used in a system that constructs the so-called "realograms"[2] from retail scenes.

[2]The term realogram refers to the actual arrangement of products in a retail scene.

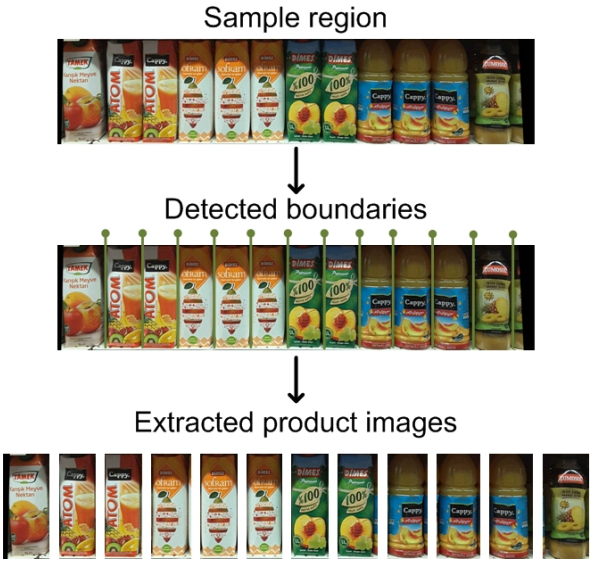


Figure 1.5. Product boundary detection and product image extraction

In the following chapters we explain the work we carried out during the thesis.

Chapter 2 provides a literature review in the field. First, the vast body of literature in visual recognition methods are reviewed. Then, we narrow down our interest to product recognition. Later, we investigate boundary detection methods.

Chapter 3 presents an overview of the overall system. We first describe our novel boundary detection methodology. Later we describe a standard product recognition engine for completeness.

In Chapter 4, we describe our boundary detection experiments in detail. Various image descriptors and classifiers are experimented with, in order to find the most practical algorithm for boundary detection. The algorithm is used in both shelf boundary detection and product boundary detection. After the lines which separate the products are detected, we coarsely segment product items into image crops, which will be used as image data to be recognized by the subsequent product recognition engine.

In Chapter 5, we put forward the methods and experiments for which we came up with a system that can recognize the product classes.

We take each set of experiments and discuss the results in Chapter 6.

Then we summarize the method and the important findings of the experiments in Chapter 7.

Our contributions are as follows:

- We came up with a novel approach to detect the shelf lines and product boundaries in retail scene images. We achieved 96% accuracy in shelf line detection and 75% accuracy in product boundary detection. An algorithm is developed which can be used in a system that localizes the product boundaries in retail scenes.
- We implemented standard recognition methods from the image recognition literature on a subset of extracted product images and documented their performances. We implemented a bag of visual words approach in product recognition. We showed that we can achieve 96% accuracy using random sampled densely extracted SIFT descriptors at the codebook generation step.

2. LITERATURE REVIEW

Object recognition deals with the task of recognizing distinct objects in a scene. It can be split into two main topics: Instance recognition and category recognition. In the former case, the aim is to re-recognize an object in a new image. The object in this new image may be viewed from a different viewpoint, in a different illumination condition, with partial occlusion, and/ or at a different scale. In the "category recognition" case the aim is to recognize objects within different general classes, such as dogs, pedestrians, or shoes. The difference in the latter case is that the elements in such classes may show great variance. Thus, the latter problem is harder than the former one.

Given a rectified retail scene, we want to recognize the SKU's of distinct retail products in the scene. The products corresponding to a specific SKU are 3D objects which are visually clones of each other (almost every time). This is why our problem falls into the instance recognition case. For this purpose we reviewed the instance recognition literature first. After reviewing the general instance recognition literature, we narrowed down our interest into the product recognition literature.

2.1. Instance Recognition

The earlier approaches focused on matching features like lines, and contours between unknown and known objects [4]. However, lately, viewpoint invariant features are more widely used. Thanks to viewpoint invariant features, it becomes possible to re-recognize an image in a new scene even if it is viewed from a different point.

In these approaches, the viewpoint invariant features such as SURF [5], or SIFT [6] are extracted. Usually, the next step is to match the features from a new image to the features from database images. At this step, along with invariance, we may want uniqueness as well. For instance a gradient encoded feature extracted from a low gradient patch would be of no use since it would not match uniquely between images. SIFT [6] and SURF [5] detectors are found to be successful in detecting unique and

invariant structural features [7]. A very helpful next step is to check for geometric alignment between the matching visual features. Checking for geometric alignment can help us eliminate the wrong matches. In [6] Lowe suggests a method where every matching feature couple votes for a location, scale, and orientation for an object. This is possible since every SIFT descriptor has its location, orientation and scale. A similar method is implemented by [8] with full local affine frames.

Test and training feature matching is done by comparing descriptor vectors of features and calculating the distance between them. However, calculating the distance between all test and training feature combinations may be exhaustive. Different algorithms are developed to speed up the process, such as tree-based [9], and hashing based algorithms [10].

If the number of database images are too high, it would be exhaustive to search for matching descriptors at test time. A new approach is proposed in [11]. The idea is to use text (or generally, information) retrieval techniques. One of the most common approaches in text retrieval literature is to describe documents with a vector of "word frequency" [12]. [11] uses SIFT descriptors as visual words. They calculate these descriptors around Harris feature points.

In order to calculate a word frequency vector, one needs to construct a vocabulary. Every visual word in the vocabulary represent a bin in the feature space. At the vocabulary construction step, we define which visual words we will count per image. In the original paper [11] they use k-means clustering on the set of descriptors extracted from training images. [13] uses vocabulary trees to obtain the vocabulary. [14] uses random forest of k-d trees. They claim it is a better idea than using vocabulary trees. Methods like using a stop-list of very frequent words, and using inverse document frequencies of words are used in text retrieval literature in order to reduce the effect of very frequent words such as "and", "of", etc. [12]. Similar approaches are proposed to be used by [11] as well. The similarity between two images can be calculating the distance between the frequency vector space. [15] shows that using dot product is the best way to measure similarity.

Re-ranking the most similar images can be useful than directly retrieving the most similar image in the database. For this purpose [14] uses an affine transformation model. The geometric locations of the matching features are checked whether they map with an affine transformation. Most of the time there is no perfect affine mapping between the matching feature locations. However the optimal affine transformation is estimated using error minimization techniques and even outlier matches can be eliminated by methods such as RANSAC [16]. [15] uses a spatial consistent model where they use the spatial distances between matching features.

2.2. Product Recognition

The advances in the visual recognition technologies, enabled us to use this technology in many different areas. Recently there has been an interest to use this technology in retail product recognition. In this chapter we review the recent works on product recognition and retail product recognition in particular.

One of the early works is [17]. This paper's first contribution is the construction of a database of 120 grocery classes. The database includes both images captured at ideal conditions (in vitro), and at retail scenes (in situ). They tried various visual descriptors and dissimilarity measures to recognize and localize products in retail scenes using in vitro images as database images. Recognition rate and localization accuracy are presented using different methods like color histogram matching, SIFT descriptor matching, and boosted Haar-like features [18]. They report that overall best performing algorithm is SIFT while boosted Haar-like features is the worst.

In [19] they focus on product recognition in general, using mobile phones. The products are generally packaged products, in "rigid boxes with printed labels, such as CDs, DVDs, and books". Due to communication constraints of mobile phones, they use low bit-rate local descriptors like Compressed Histogram of Gradients (CHoG) [20]. They use a bag of words approach using CHoG's and at vocabulary construction step they quantized the visual words with vocabulary trees. They use fast geometric re-ranking proposed by [21] in order to re-rank the closest matches from the database.

In [22] they propose a mobile product image search system. In the work, two datasets are used. First one is a sports product database which includes t-shirts, sports shoes etc. The second database consists of general products sampled from Caltech-256 object database Caltech [23]. The training images consists of single products with black backgrounds, so they are masked. The method consists of two steps: segmentation and retrieval. In order to segment images, the test and training descriptors are extracted and matched to the visual vocabulary which is pre-computed by a hierarchical k-means algorithm. For segmentation, the masks of closest training images are used. Later, the segmented object is described using a BoW approach and this helps remove the description of background clutter. The visual words which are used in this work is dense and sparse SIFT features.

A recent work which focused on recognizing and localizing retail products in shelves is [24]. From test and training images, they compute dense SIFT descriptors and match those descriptors. Top k classes are retrieved and it is assumed that these are the SKU's which are present in the test image. For localization, they implement dense pixel matching from the top k classes to the query images. In this work, the test images consists of 1 or 2 shelves and the training images are single product images.

In more recent works, using context to achieve higher performance gained more popularity. For instance in [25] they encode temporal and spatial context with the approach called Visual Co-occurrence Network. On a retail scene dataset, they establish the relationship between observed objects and improved their detection performance and accuracy. In a similar manner, in [26] improves the accuracy of a multi class SVM based retail product classifier as well. The context free classifier uses a Bag of Words approach with dense SIFT features. In order to model the co-occurrence and adjacency of retail products they construct a probabilistic graphical model based on Hidden Markov Models. This approach is especially very helpful at discriminating visually very similar products.

Usually, the places of the objects to be recognized are not known in general object recognition problems. However, the retail products are usually placed on shelves

and side by side. Knowing the specific locations of different products, reduces the complexity of exhaustively searching for products in a retail scene. Thus we decide to, segment the separate product images first and then later recognize them. In order to separate the retail products, we need to find the shelf and product boundary lines. We direct our research into boundary detection literature.

2.3. Boundary Detection

In order to segment the products in a shelf, we need to localize the boundaries between them. Since our aim is obtaining the rectangular crop which includes a single product, we just need to find the horizontal lines which separate the shelves and vertical lines which separate two neighboring products in a shelf. Given a retail scene, detection of shelf lines needs to be done first and detection of product boundaries needs to be done later. However since it is harder to detect the product boundaries, we focused on product boundary detection first, and extended the method to include the detection of shelf lines.

A boundary is usually defined as a separating surface between two objects. In literature, there are works to decide whether an image pixel is an element of a boundary or not. Martin *et al.* defines a boundary as "a contour in the image plane that represents a change in pixel ownership from one object or surface to another" [27]. So in their paper "Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues" they focus on features responding characteristic changes corresponding to brightness, color, and texture in order to detect natural boundaries. At boundaries, the pixel ownership changes, thus focusing on visual feature changes is wise. *In our case we are not interested in a pixel level boundary detection. Our aim is to detect the product (or shelf) separating vertical (horizontal for shelf case) lines.*

There have been works to solve a similar problem in literature; book boundary detection. Conventional methods use line detection approaches since book boundaries can be modeled as lines [28]. This approach is not useful for our case since only some product boundaries can be modeled as lines. We are interested in more complex

boundaries as well.

Iwanowski *et al.* [29] propose a line detection approach to product boundary detection in their paper as well. Their method uses a directional gradient calculation step, and after morphological operations, for every column they collect the number of horizontal gradients and compare the sum to a threshold. This method may work well on detection of boundaries of rectangular products.

3. SYSTEM OVERVIEW

In general object detection problems, usually the locations of the objects are random. However in our case, the objects are aligned in racks. Thus we took an approach where we first divide the image into sub images which include separate products, and later recognize them. The pipeline consists of 2 main parts; product boundary detection and product recognition. For both parts we experimented with different methods and parameters to achieve high performance.

3.1. Boundary Detection

From a rectified retail scene, first step into our method is detecting the shelf boundaries to parse the shelves. It is important to remember that the retail scene images are geometrically transformed so that the shelf lines are horizontal and side walls of the shelves are vertical. The rectification step is not covered in this thesis and we assume the retail scene images to be pre-rectified. After the shelf boundaries are localized, from the segmented shelf images, we detect product boundaries. However, since the product boundaries display more visual diversity, the product boundary detection in shelves is assumed to be the bottleneck of the product parsing. Thus, we decided to focus our attention to first boundary detection from shelf images and then adapt the method to shelf line detection in retail scenes. This was possible since we already had an extracted and boundary annotated shelf image database. After we prove the concept in product boundary detection, we implement the method also for the shelf boundary detection.

Given a shelf image, we need to localize the product boundaries. A product boundary is defined as a vertical line separating two products (or separating a product from the background). This is needed to crop the product regions to later recognize them. The ultimate goal of this part is to find the column indices where product boundaries occur. We use a dataset of shelf images and corresponding hand annotated boundary column indices as ground truth. The dataset consists of 42000 shelf images

with annotation lists. The boundaries may occur between two neighbor products or between a product and background. The first type of boundary region is shown with blue and the second type of boundary is shown with red in Figure 3.1.

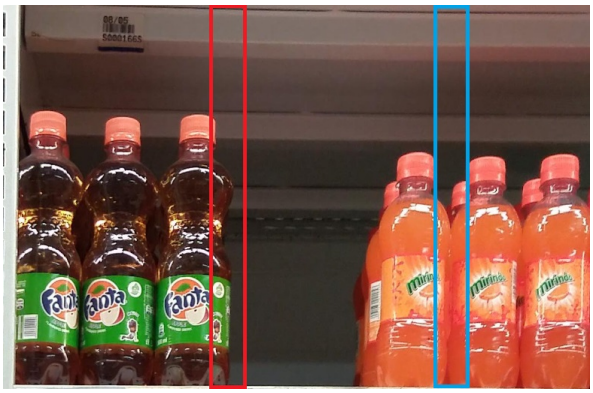


Figure 3.1. Types of boundary regions

The proposed method is as follows:

(i) A training image set and corresponding annotation list set are sampled from the given dataset.

(ii) For convenience, all images are resized to the same height; preserving aspect ratio. The annotations are adjusted accordingly.

(iii) With a sliding window of fixed width, passing through the training images, stripe images are extracted and their image descriptors are calculated. Different image descriptor types are tried to find out which ones capture the essence of the boundaries better.

(iv) Using the ground truth data, we sample positive and negative descriptors and record the corresponding labels.



Figure 3.2. A sample shelf image with corresponding boundary annotations marked with green vertical lines

(v) A classifier is trained, using the positive and negative training descriptors. This is called supervised learning. For our case, labeling is not enough, the classifier has to be able to decide on a score for a descriptor from a stripe image. Various classifiers are tried.

(vi) Given a query image, with a sliding window of the same width used in training phase, the classifier grades descriptors from all stripes. A score signal is the output of this step.

(vii) We process the signal and detect peaks and declare the strong ones as boundaries.

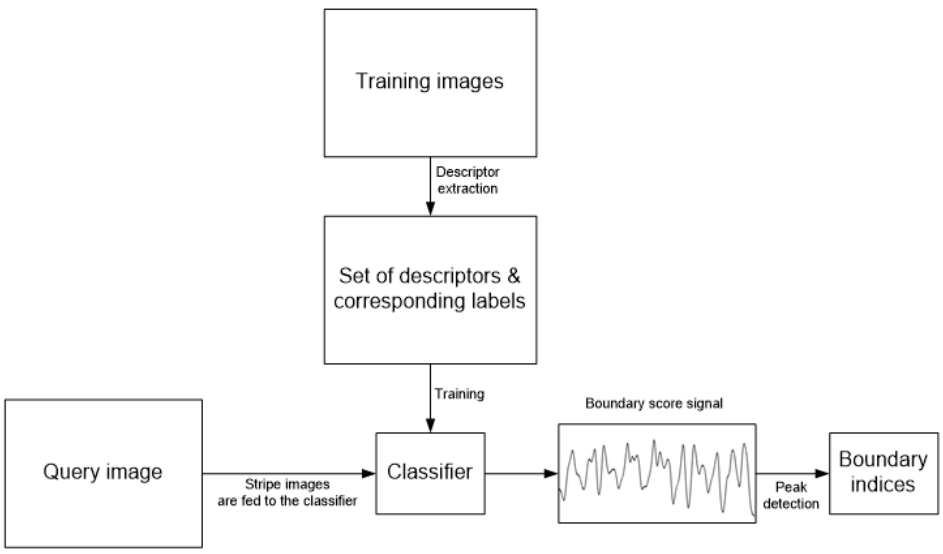


Figure 3.3. Boundary detection pipeline

In Chapter 4 we will begin by exploring two aspects concurrently: The image descriptor, which captures the boundary features best; and the classifier, which dis-

criminates between boundary and non-boundary descriptors best. After we decide on a descriptor-classifier couple, hopefully we will be able to extract "boundary score signal" from which we will detect the peaks and localize the detected boundaries.

3.2. Product Recognition

After we segment the rectangle regions which include separate products with the help of boundary detection step, our aim is to recognize these products. In this step, our proposition is to use an image retrieval method which is proposed in the paper "Video Google: A Text Retrieval Approach to Object Matching in Videos" [11].

In this method visual descriptors are extracted from training images and a visual vocabulary is constructed. Images are described using a weighted word frequency vector [12]. Training images are ranked with respect to their inter-descriptor distances to query image. The method consists of 3 main steps: Vocabulary construction, database construction and query. We are given,

- The corresponding set of annotations $Y = \{y^{(n)}\}_{n=1}^{N}$ where $y^{(n)}$ stands for the label of the object present in image $I^{(n)}$,
- A set of query images $J = \{J^{(q)}\}_{q=1}^{Q}$ containing objects with unknown labels at test time .

(A) Vocabulary Construction – offline

(A1) A set of stable image descriptors $\{d = (p, s, \theta, u)\}$ from a set of training images $I = \{I^{(n)}\}_{n=1}^{N}$ are extracted. The descriptors are chosen to be SIFT descriptors. An image descriptor d consists of a position vector $p = (x, y)$ in the image domain, a scalar s for characteristic scale, a scalar θ for dominant orientation and a visual descriptor vector $u \in \mathbb{R}^d$. We will try both using a SIFT detector and using dense SIFT at this step to compare these two methods.

(A2) The set of descriptor vectors $\{u\}$ is clustered into a set of visual words $V = \{v^{(k)} \in \mathbb{R}^d\}_{k=1}^{K}$ making note of membership information for each descriptor u, e.g. $c(u) = v^{(k)}$. For clustering, the K-means algorithm is used [30]. The number of visual words is chosen at different values. Instead of clustering, we will also try randomly selecting visual words. This approach is proposed in [31] by Novak *et al.* and claimed to be a wiser vocabulary constructed method.

(A3) Too frequent visual words in the dictionary are detected, i.e., the ones such that $|M(v^{(k)})| > T_{stop}$ where $M(v^{(k)})$ denote the set of descriptors $\{u\}$ assigned to the visual word $v^{(k)}$ and $|A|$ the number of elements in the set A. The set of such visual words is denoted as V_{stop} . For every selected K value multiple T_{stop} values are tried to compare their performances.

(B) Database Construction – offline

(B1) For each visual word $v^{(k)} \notin V_{stop}$,

- The number of images in which it is present as N_k and,
- The number of its occurrences in each image $I^{(n)}$ as m_{kn}

are recoreded.

(B2) The number of visual words present in each image $I^{(n)}$ is recorded as $m_n = \sum_{k:v^{(k)} \notin V_{stop}} m_{kn}$.

(B3) For each image $I^{(n)}$, its weighted word frequency vector $z^{(n)} = [z_k^{(n)}], \forall v^{(k)} \notin V_{stop}$ is constructed.

$$z_k^{(n)} = \frac{m_{kn}}{m_n} \log \frac{N}{N_k} \tag{3.1}$$

$z_k^{(n)}$ is the term frequency - inverse document frequency of the word v_k in the image $I^{(n)}$ [12]. $\frac{m_{kn}}{m_n}$ is the term frequency part. It measures the frequency of the visual word in an image. $\log \frac{N}{N_k}$ is the inverse document frequency part. It measures the rarity of the images which includes the visual word v_k. The vector $z^{(n)}$ is called a term frequency-inverse document frequency (tf-idf) vector. This vector – the signature of image $I^{(n)}$ is then normalized by rendering its L1 and L2-norm unity. Both normalization methods are implemented to compare performances.

(B4) The inverted file structure is constructed as depicted below:

Table 3.1. Inverted file structure.

	$I^{(1)}$	$I^{(2)}$	$I^{(n)}$	$I^{(N)}$
	${z_k}^{(1)}$	${z_k}^{(2)}$	${z_k}^{(n)}$	${z_k}^{(N)}$
$\forall v^{(k)} \notin V_{stop}$	${(p,s,\theta)_{1,k}}^{(1)}$	${(p,s,\theta)_{1,k}}^{(2)}$	${(p,s,\theta)_{1,k}}^{(n)}$	${(p,s,\theta)_{1,k}}^{(N)}$
	${(p,s,\theta)_{2,k}}^{(1)}$	${(p,s,\theta)_{2,k}}^{(2)}$	${(p,s,\theta)_{2,k}}^{(n)}$	${(p,s,\theta)_{2,k}}^{(N)}$
	...	...	...	...
${(p,s,\theta)_{i,k}}^{(n)}$ stands for the i'th feature from the n'th database image ($I^{(n)}$) and whose descriptor u is assigned to the k'th visual word. Also a list of the visual words in V_{stop} is kept as well – test features assigned to such visual words will not be taken into account.				

(C) Image Retrieval – online (test time)

(C1) A set of stable image descriptors $\{d = (p, s, \theta, u)\}$ from the query image J is detected using the same algorithm as in the visual dictionary construction stage.

(C2) Each query image descriptor is assigned to a visual word in the dictionary V, identifying the ones assigned to a visual word in the stop list V_{stop}. The tf-idf vector $q = [q_k], \forall v^{(k)} \notin V_{stop}$ of the query image is constructed in the same way as in (B3).

(C3) The database images are ranked based on the L1 and L2-distance of their tf-idf vectors $z^{(n)}$ to the query vector q where L1 and L2 distance measures are evaluated respectively as follows:

$$l^1(q, z^{(n)}) = \sum_{k:v^{(k)} \notin V_{stop}} |q_k - z_k^{(n)}| \tag{3.2}$$

$$l^2(q, z^{(n)}) = \sqrt{\sum_{k:v^{(k)} \notin V_{stop}} |q_k - z_k^{(n)}|^2} \tag{3.3}$$

For the methods in which the weighted word frequency vector $z^{(n)}$ of the training images and tf-idf vector q of the test images are L1 normalized, L1 distance is used at this step. For L2 normalized vectors, L2 distance is used.

(C3) A number of highest ranking training images are retrieved and optionally re-ranked.

(C4) The test image is labeled same as the highest ranking training image's label.

4. BOUNDARY DETECTION

In this chapter, we put forward our experiments on boundary detection. These experiments can be divided in 2 parts:

(i) Finding the best visual descriptor and classifier couple: Here, we extract visual descriptors from boundary and non boundary regions and we train classifiers to classify them. These experiments are done with 3 different descriptors and 3 different classifiers. The visual descriptors are explained in Section 4.1 and the classifiers are explained in Section 4.2. Also various parameters are tried. For every visual descriptor and classifier couple, we present accuracy and query times. We search for the visual descriptor, classifier, and parameter set combination which maximizes accuracy in a reasonable query time. These experiments are presented in Sections 4.3, 4.4, and 4.5.

(ii) Implementation on query images and parameter selection: Here, we use the optimal descriptor-classifier couple to extract "boundary score signal" from which we will detect the peaks and localize the detected boundaries. We try to achieve high precision and recall with various signal processing techniques. These experiments are presented in Section 4.6.

In Section 4.7 we compare our boundary detection method to a benchmark method and in Section 4.8 we implement the product boundary detection pipeline to shelf line detection problem and report the results of this experiment.

4.1. Visual Descriptors

We take a different approach where we start off with the hypothesis that the image locations where a product boundary occurs have some distinct image features. We decide to describe boundary including image stripes with global image descriptors. The descriptors are global to the image stripes, not in terms of scene images. Our approach to boundary detection is a supervised learning method where we use image

features to describe the boundaries and train a classifier to learn the statistical decision function which takes a image descriptor and decides whether the descriptor is of a boundary region or not. We reviewed image descriptors for this purpose. A global image descriptor is an array of attributes which is used to describe an image. We keep that in mind that we can always introduce locality by describing local patches in an image distinctly.

We need to find descriptors which capture what means to be a product boundary. We know that at product boundaries pixel ownership changes in the horizontal direction. Thus intuitively we look for descriptors that encode a change, or direction of change in a feature channel. We decided to use the below descriptors to capture the product boundaries.

4.1.1. Filter Response Features from Root Filter Set

Filters are useful in determining if a pattern exists in an image and where it exists. We thus searched for a texture filter set to capture the direction of intensity change. We decided that, responses to Root Filter Set can be used to describe the boundary and non boundary stripes [32]. The filter set consists of 38 texture filters very similar to "Leung-Malik filter bank". The filters in the set are all sized 49 by 49 pixels, including;

- An edge filter at 6 orientations and 3 scales; $(\sigma_x, \sigma_y) \in \{(1,3),(2,6),(4,12)\}$,
- A bar filter at 6 orientations and 3 scales; $(\sigma_x, \sigma_y) \in \{(1,3),(2,6),(4,12)\}$,
- A symmetric Gaussian filter at 1 scale; $(\sigma_x, \sigma_y) \in \{(10,10)\}$,
- A symmetric Laplacian of Gaussian filter at 1 scale; $(\sigma_x, \sigma_y) \in \{(10,10)\}$.

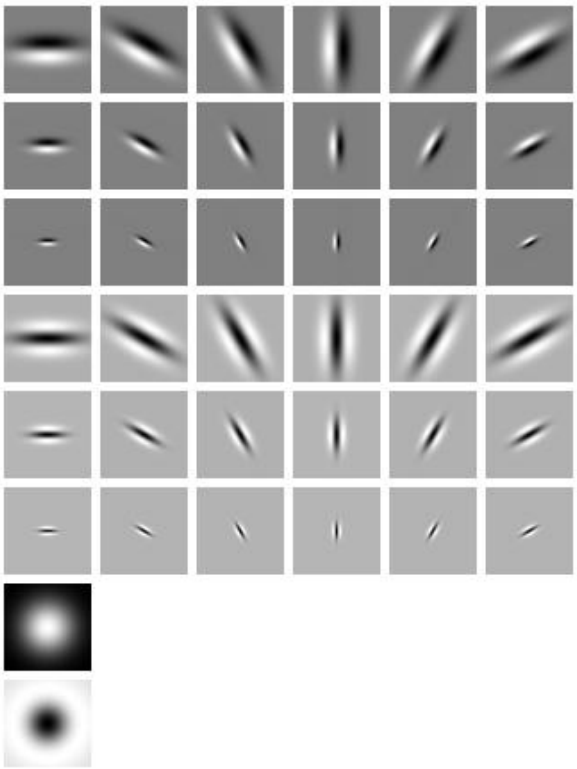


Figure 4.1. Root filter set

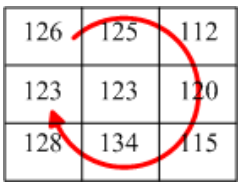
4.1.2. Histograms of Oriented Gradients

One of the descriptors that we can use to describe the image stripes is the well known Histograms of Oriented Gradients (HOG) [33]. A HOG descriptor encodes the occurrence frequency of different gradient vectors. Localized HOG descriptors are successfully used human detection in literature. It is wise to assume the gradient orientation distribution of boundary regions are distinctive and we may take this approach in our work. To describe an image patch with a HOG descriptor, first the gradient map of the image patch is extracted. Then, for different gradient interval bins, the number of occurrences in the gradient map are counted and accumulated in the corresponding histogram bins. Later the histogram is L1 normalized and the frequency histogram is obtained.

4.1.3. Local Binary Pattern Histograms

Histogram of local texture descriptors are also used in literature to describe images. A very popular texture descriptor is Local Binary Patterns. In the survey Two Decades of Local Binary Patterns various applications and variants are surveyed [34].

A local binary pattern is a texture operator, labeling a pixel by comparing its intensity level to the neighbor intensity level. Neighbor intensities are visited clockwise, compared to the pixel, and a binary code is constructed recording which neighbor pixel intensities are greater than the center pixel intensity. A sample patch of pixel intensities is given below. To calculate the basic LBP label for the central pixel, we compare the 8 neighbors to it.



The binary code produced for the center pixel is "11000110". This binary code is then converted to decimal for convenience. Thus the label for the center pixel becomes 198. For a given image, an LBP image is calculated, recording all labels. This is the simplest form of an LBP, proposed by Ojala *et al.* [35].

In the next chapter we put forward the mathematical background of the classifiers which we used.

4.2. Classifiers

We need to find a classifier which learns the statistical decision function well. Using the train data, a classifier is going to be constructed. There are a lot of supervised learning algorithms in the literature. We searched for the ones which may be suitable to our problem of learning the distinction between boundary region descriptors and

non-boundary region descriptors.

4.2.1. Quadratic Classifier

Given an observation x, which is a feature vector in our case, we want to chose the label y which is more probable. In our case there are 2 labels: boundary (+1) and non-boundary (-1) so the decision rule is

$$\text{if } p(y=1|x) > p(y=-1|x), \text{ decide boundary;}$$

$$\text{else, decide non boundary}$$

Applying Bayes rule, the condition which we decide a sample x to be boundary can be written as,

$$\frac{p(x|y=1)p(y=1)}{p(x)} - \frac{p(x|y=-1)p(y=-1)}{p(x)} > 0 \tag{4.1}$$

We do not know the class conditional distributions of descriptors. We assume a Gaussian distribution. For Gaussian distributions in N dimensional vector space,

$$p(x|y=Y) = (2\pi)^{-N/2}|\Sigma_Y|^{-1/2}e^{-\frac{1}{2}(x-\mu_Y)^T|\Sigma_Y|^{-1}(x-\mu_Y)} \tag{4.2}$$

where μ_Y is the mean and Σ_Y is the covariance matrix of training samples with label Y. Under Gaussian assumption for $p(x|y=1)$ and $p(x|y=-1)$, we get the rule below:

$$p(y=1)|\Sigma_1|^{-1/2}e^{-\frac{1}{2}(x-\mu_1)^T|\Sigma_1|^{-1}(x-\mu_1)} - p(y=-1)|\Sigma_2|^{-1/2}e^{-\frac{1}{2}(x-\mu_2)^T|\Sigma_2|^{-1}(x-\mu_2)} > 0 \quad (4.3)$$

When we take natural logarithm, we get;

$$-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - \frac{1}{2}ln(|\Sigma_1|) + ln(p(y=1)) + \frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) + \frac{1}{2}ln(|\Sigma_2|) - ln(p(y=-1)) > 0 \quad (4.4)$$

The maximum likelihood estimate of Gaussian distribution parameters μ and Σ are given below.

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad (4.5)$$

$$\hat{\Sigma} = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \hat{\mu})^T \quad (4.6)$$

where x_i's are training observations. The training phase consists of calculating the maximum likelihood estimates $\hat{\mu}$ and $\hat{\Sigma}$, given a set of training descriptors and labels (+1, -1). This is enough to obtain the decision rule for a given query x.

4.2.2. AdaBoost with Single Level Decision Trees

Boosting refers to a method of finding an effective and accurate classification rule by combining multiple rules that are moderately inaccurate [36]. These moderately inaccurate classifiers are called weak classifiers. A weak classifiers only needs to be

slightly better than random guessing [37]. Boosting was first proposed by Schapire in the paper "The Strength of Weak Learnability" and gained much attention since then [38]. One of the most popular boosting schemes is Discrete AdaBoost which we implemented in our work [39].

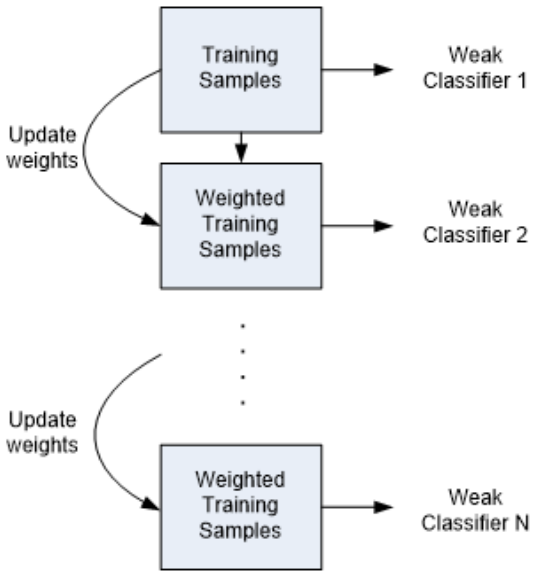


Figure 4.2. AdaBoost block diagram

The final decision rule is:

$$\text{decide 'class is +1' if and only if } \sum_{k=1}^{N} \alpha_k h_k(x) > 0$$

where $h_k(x)$ is the classification result of Weak Classifier k and α_k's are weak classifier weights. α_k's are higher for more accurate classifiers and lower for less accurate classifiers. Weak classifiers have at least 50% accuracy and the weights are positive. We calculate the weight α of the classifier as below.

$$\alpha = \frac{1}{2} \ln(\frac{accuracy}{1 - accuracy}) \tag{4.7}$$

The training samples are weighted at every step. Initially the weights are equal for all training samples. The weights are updated so that wrongly classified training samples have more weights at the next iteration.

AdaBoost can be used to boost any learning algorithm which can take weighted training samples as input. A very popular weak classifier which can be used in this procedure is a "one level decision tree" which is also called a stump [40]. The optimal stump at each step is found by finding the feature dimension and the threshold at that dimension which results in the highest separation accuracy over the weighted training descriptor set. At each step, we find the optimal stump which classifies the weighted training data with highest accuracy. The pseudo code for AdaBoost is given in Appendix A.

4.2.3. Support Vector Machines

A Support Vector Machine is a classifier which finds the optimal hyper plane which separates the training samples from different classes in a augmented feature space. It is introduced in 1995 by Vapnik [41]. This chapter provides a brief glimpse to the SVM theory. Only binary classification is considered. We have benefited from [42] and [43] for the research in this chapter. More detailed information on SVM can be found in the book "The elements of statistical learning: data mining, inference, and prediction" [37].

SVM takes its roots from the idea of finding the optimal hyper plane which separates samples of different classes. Given a set of descriptors x_i's and corresponding labels y_i's where $y_i \in$ {-1,+1} a separating hyper plane $f(x) = w^T x + w_0 = 0$ in the feature space holds the condition $y_i f(x_i) > 0$ for all x_i, y_i training descriptor label couples (if a separating hyper plane exists). Among the infinite separating hyper planes our aim is to find the one which has the maximum margin where the margin of a hyper plane is defined the minimum distance to the closest training vector. For a given vector

x_i the distance to the hyper plane is

$$\frac{w^T x_i + w_0}{||w||} \tag{4.8}$$

Since $y_i(w^T x_i + w_0) > 0$ and $y_i \in \{-1, +1\}$, we can rewrite it as

$$\frac{y_i(w^T x_i + w_0)}{||w||} \tag{4.9}$$

The margin M is the smallest value the distance can take, thus

$$\frac{y_i(w^T x_i + w_0)}{||w||} \geq M \tag{4.10}$$

We can normalize M by $||w||$ since M's scale is not important. Thus the optimization problem becomes:

$$\arg\min_{w^T, w_0} \frac{1}{2}(w^T w) \tag{4.11}$$

$$\text{subject to constraint } y_i(w^T x_i + w_0) \geq 1 \text{ for all i}$$

If the descriptors are not linearly separable, we relax the constraint by introducing slack variables $\xi_i > 0$. However since we do not want the constraint to relax too much, we want to keep the slack variables small as well. Now the optimization problem becomes

$$\arg\min_{w^T, w_0} \frac{1}{2}(w^T w) + C \sum \xi_i \tag{4.12}$$

$$\text{subject to constraint } y_i(w^T x_i + w_0) \geq 1 \text{ for all k}$$

The parameter C controls the trade of between norm minimization and margin relaxation.

Up to this point we stayed in the descriptor space. However there may be cases where there is no competent hyper plane (soft or hard margin) but a competent hyper surface. We can introduce it by mapping the samples to a higher dimensional space and searching for a hyper plane there. However mapping each point to a higher dimensional space is computationally expensive. It also turns out to be unnecessary since solving the minimization problem in the new space only requires the corresponding inner product,

$$K(x_k, x_l) =< \psi(x_k), \psi(x_j) > \tag{4.13}$$

where $K(x_k, x_l)$ is the kernel function and ψ is the mapping to the new space. The kernel function enables us to calculate the inner product of two vectors without explicitly mapping them to the higher dimensional space. This is called a kernel trick. Various kernel functions are used in literature.

4.3. Experiments with Filter Response Features from The Root Filter Set

In this subchapter, we summarize the experiments in which the boundaries are represented using the Filter Response Features. The filter set used to produce responses is the Root Filter Set which is very similar to "Leung-Malik filter bank" [32]. The filter output images are divided into cells and max-pooling implemented to collect features across cells. The implemented method is explained in detail in below sections. Our implementation of the Root Filter Set is taken from the paper "A Statistical Approach to Texture Classification from Single Images" [44]. In the paper, the set is referred as MR8 filter bank since they only collect 8 responses to achieve rotation invariance.

4.3.1. Sample Description

Images are resized to fit an image height of 128, preserving aspect ratio. And they are converted to grayscale.



Figure 4.3. A sample shelf image

Feature response maps are produced by filtering the images with Root Filters. From one image, 38 filter response maps are obtained.

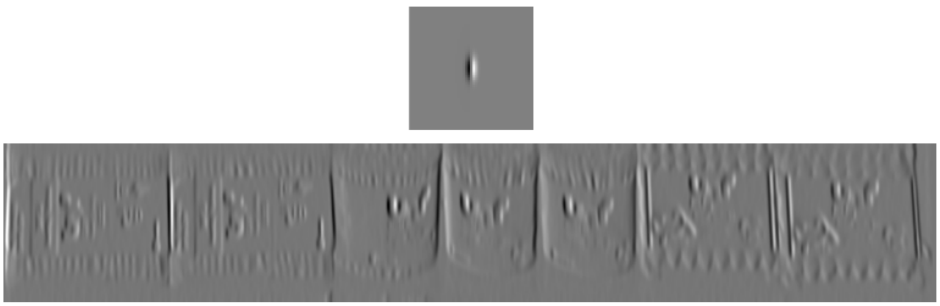


Figure 4.4. A filter and corresponding response map from the sample image

For a positive boundary annotation, 38 stripe response maps are striped from all filter response maps. The central pixel of the stripes corresponds to the boundary annotations. The stripes are divided into cells and maximum filter responses in cells are accumulated respectively from all stripe filter response maps. This is done for all positive annotations.

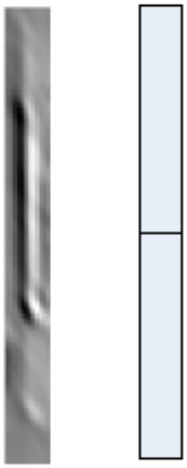


Figure 4.5. A sample stripe response map of width 8 and a cell division pattern

Negative annotations are randomly selected from the 30% middle intervals between positive annotations. Same feature vector generating pattern is used to collect negative samples. The positive annotations and negative annotations for the example image are shown in Figure 4.6. From 1 positive boundary annotation, 1 positive feature vector is created and added to positive sample set. From 1 negative annotation (a non-boundary location), 1 negative feature vector is created and added to negative sample set.



Figure 4.6. Positive (blue) and negative (red) annotations

Since only the maximum response is collected as a feature from a cell, the dimensionality of the feature vector can be calculated as Number of filters x Number of cells.

4.3.2. Experiments

In this chapter, experiments on FRF variants are documented.

4.3.2.1. Stripe width and Cell Division Experiment. 4 variants of Root Filter Response Features are used for classification trials. Variants differ in 2 aspects.

(i) Stripe width: Two different stripe width options are tried; 8 and 12. Stripe width parameter does not affect the dimensionality of feature vectors.
(ii) Cell division pattern: Two different cell division patterns are used; 2 vertical cell division and 8 vertical cell division. The dimensionality of the feature vector is directly proportional to the number of cells.

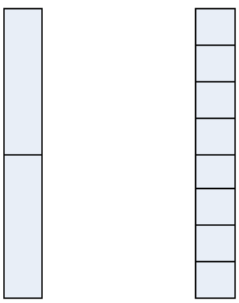


Figure 4.7. 2 vertical cell division and 8 vertical cell division patterns

From 42114 shelf images, random 1000 and 500 images are sampled and used as training image set (Set 2.6) and test image set (Set 2.13) respectively. The sets are discrete. 8107 positive training samples and 8238 negative training samples are collected. 3943 positive test samples and 3992 negative test samples are collected. This is separately done for different variants.

Four types of classifiers are trained and tested; a Quadratic classifier, an AdaBoost with 100 weak classifiers, an AdaBoost with 150 weak classifiers, and an SVM with Radial Basis Function kernel. The weak classifiers of the AdaBoost classifiers are one

level decision trees.

In order to find the optimal SVM parameters -Gamma and C-, 5 fold grid search is implemented on the training descriptor sets. The steps are as follows:

(i) The training set is partitioned into 5 subsets.

(ii) Every subset is tested by training the SVM on the union of the 4 remaining subsets.

(iii) The tests are implemented with different C and Gamma combinations. Logarithmic step is used in the grid search. 6 steps are tried per parameter. The minimum value for C is chosen as 0.1. Logarithmic step is chosen as 5. Thus the range for C is from 0.1 to 312.5. The minimum value for Gamma is chosen as 1e-5. Logarithmic step is chosen as 15. Thus the range for Gamma is from 0.00001 to 7.59375.
These grids are default grids in the OpenCV library SVM training function trainAuto() [45].

(iv) The average accuracy maximizing C and Gamma combination is found.

For the variants in this experiment, average accuracy maximizing SVM parameters are found and given in Table 4.1 .

Table 4.1. SVM parameters found using 5 fold grid search.

	V1	V2	V3	V4
Stripe width	8	8	12	12
Cell division	2x1	8x1	2x1	8x1
C	2.5	2.5	2.5	2.5
Gamma	0.50625	0.03375	0.50625	0.03375

For all variant - classifier combinations, 5 fold cross validation is implemented. That is; the training data is divided into 5 subsets. Every subset is tested using

the union of the 4 remaining subsets as a training set. The average accuracies are recorded. Later, new classifiers are trained over the training sets (without folding) and the descriptors in the discrete test sets are classified.

The accuracies of classifiers for different variants are given in below tables. Table 4.2 shows the 5 fold average accuracies over of the training descriptor sets. Table 4.3 shows the accuracies of classifiers trained over the training descriptor set and tested over the test descriptor set. The SVM's are trained with the optimal parameters found in the 5 fold grid search.

Table 4.2. 5 fold cross validation average accuracies of classifiers over the training sets.

	V1	V2	V3	V4
Stripe width	8	8	12	12
Cell division	2x1	8x1	2x1	8x1
Quadratic classifier	83.77%	87.43%	82.00%	87.04%
AdaBoost with 100 iterations	85.76%	87.84%	85.19%	87.58%
AdaBoost with 150 iterations	86.30%	88.60%	85.50%	88.10%
SVM	87.25%	90.65%	87.54%	89.61%

Table 4.3. Accuracy of classifiers over the test sets.

	V1	V2	V3	V4
Stripe width	8	8	12	12
Cell division	2x1	8x1	2x1	8x1
Quadratic classifier	82.18%	82.13%	80.63%	80.56%
AdaBoost with 100 iterations	85.13%	87.34%	84.53%	86.76%
AdaBoost with 150 iterations	85.14%	87.46%	84.58%	87.06%
SVM	87.17%	89.83%	86.59%	89.56%

The accuracies in the two tables are consistent. They validate each other. It is observed that using a stripe width of 8, results in higher test set accuracy for all classifiers. Except for Quadratic classifier, increasing the vertical cell number yields higher accuracy as well . Thus, the optimal variant seems to be V2 which has 304 feature dimensions. AdaBoost classifier performs better than Quadratic classifier for all variants. For the optimal variant V2, increasing the number of weak classifiers from 100 to 150, resulted in 0.12% higher accuracy over the test set. The accuracy during AdaBoost training iterations can be seen below.

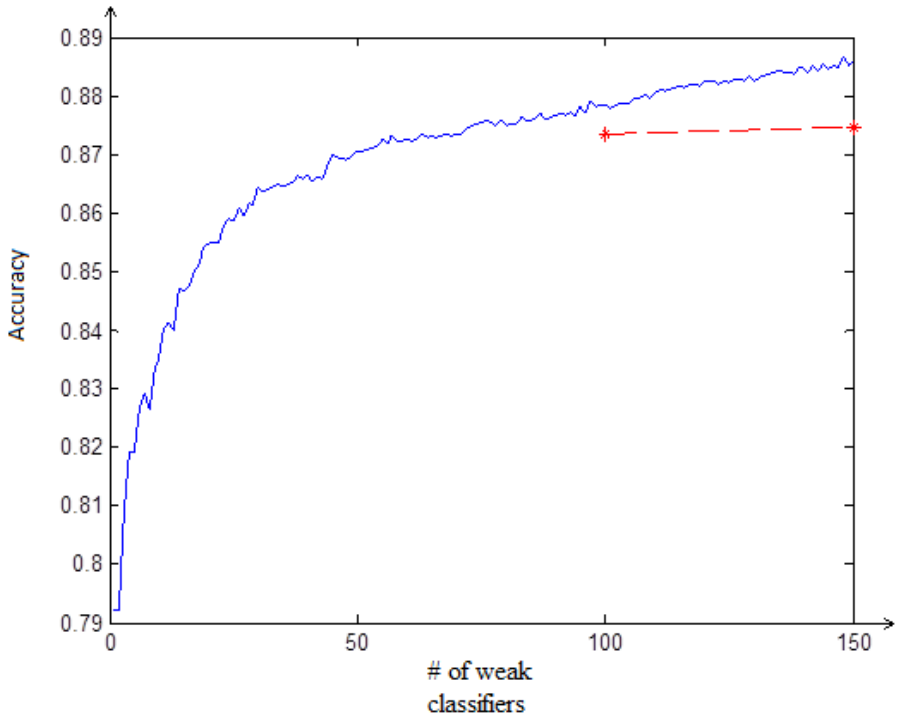


Figure 4.8. AdaBoost weak classifier number vs. AdaBoost Accuracy in Stripe width and Cell Division Experiment. Blue: Training samples, Red: Test samples

We can see that raising the number of iterations from 100 to 150, increased average training sample classification accuracy from 87.84% to 88.60% (by 0.76%). However, test sample classification accuracy increased from 87.34% to 87.46% (only by 0.12%). This may show that; for this feature variant, raising the number further from 150 will not add much classification accuracy. The classifier seems to be saturated at this iteration number.

For all variants, SVM performed better than other classifiers. The best combination for this experiment is Variant 2 with SVM. Accuracy is 89.83% for this combination.

4.3.2.2. Filter Set Size Experiment. In order to be able to work on more features, the filter set is extended to include the negative versions of the filters. 4 more variants are obtained with this extended filter set.

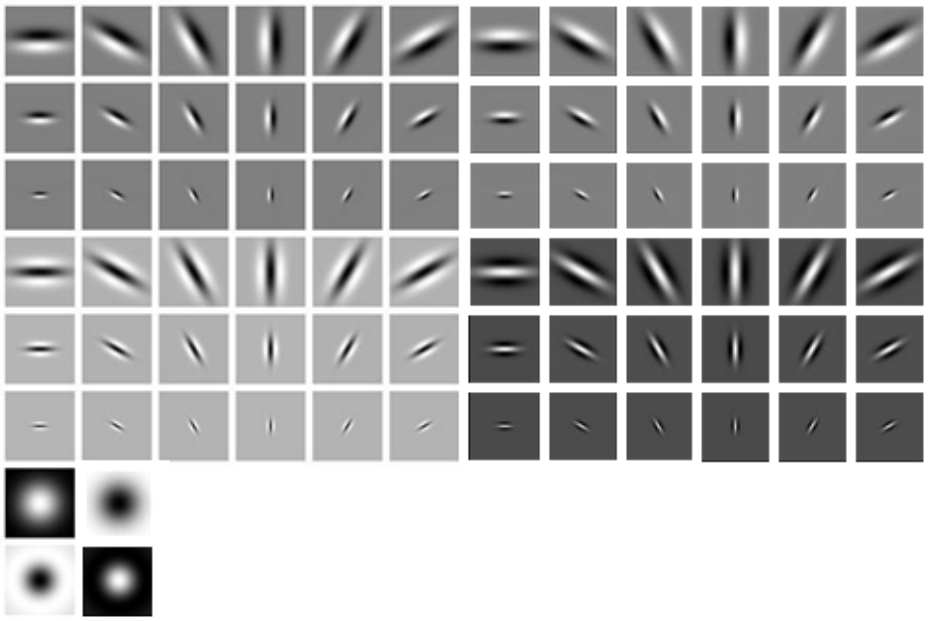


Figure 4.9. Extended Root Filter Set

The optimal SVM parameters are found as below.

Table 4.4. SVM parameters found using 5 fold grid search.

	V5	V6	V7	V8
Stripe width	8	8	12	12
Cell division	2x1	8x1	2x1	8x1
C	12.5	2.5	12.5	2.5
Gamma	0.03375	0.03375	0.03375	0.03375

Classification accuracies for these variants are given in below tables.

Table 4.5. 5 fold cross validation average accuracies of classifiers over the training sets.

	V5	V6	V7	V8
Stripe width	8	8	12	12
Cell division	2x1	8x1	2x1	8x1
Quadratic classifier	83.64%	82.54%	83.00%	85.35%
AdaBoost with 100 iterations	86.49%	88.54%	85.71%	87.55%
AdaBoost with 150 iterations	87.03%	89.16%	86.19%	88.36%
SVM	89.91%	91.22%	88.44%	91.32%

Table 4.6. Accuracy of classifiers over the test sets.

	V5	V6	V7	V8
Stripe width	8	8	12	12
Cell division	2x1	8x1	2x1	8x1
Quadratic classifier	83.57%	83.49%	82.29%	83.23%
AdaBoost with 100 iterations	85.60%	87.47%	84.81%	86.68%
AdaBoost with 150 iterations	86.05%	87.75%	85.14%	87.23%
SVM	88.55%	90.26%	87.69%	90.09%

Again, SVM performed better than the other classifiers for all cases, and highest accuracy is again observed when stripe width is 8 and 8 vertical cell division pattern is used. The best combination is V6 with SVM which achieves an accuracy of 90.26

Comparing V2 to V6 (best variants of Experiments 1 and 2), we can observe that, extending the filter set increased highest SVM test set accuracy from 89.83% to

90.26 (by 0.43%) We see that using Extended Root Filter Set did not yield much more classification accuracy over the regular Root Filter Set. Although, the dimensionality is doubled due to the doubling of filter number. Another remark is that; comparing V5 to V6, we can see that increasing vertical cell divisions from 2 to 8, the dimensionality increased from 152 to 608 and test set classification accuracy of SVM increased from 88.55% to 90.26%.

For the best variant V6, the accuracy during AdaBoost training iterations can be seen below. For all cases, increasing the weak classifier number yielded better accuracy over the training samples.

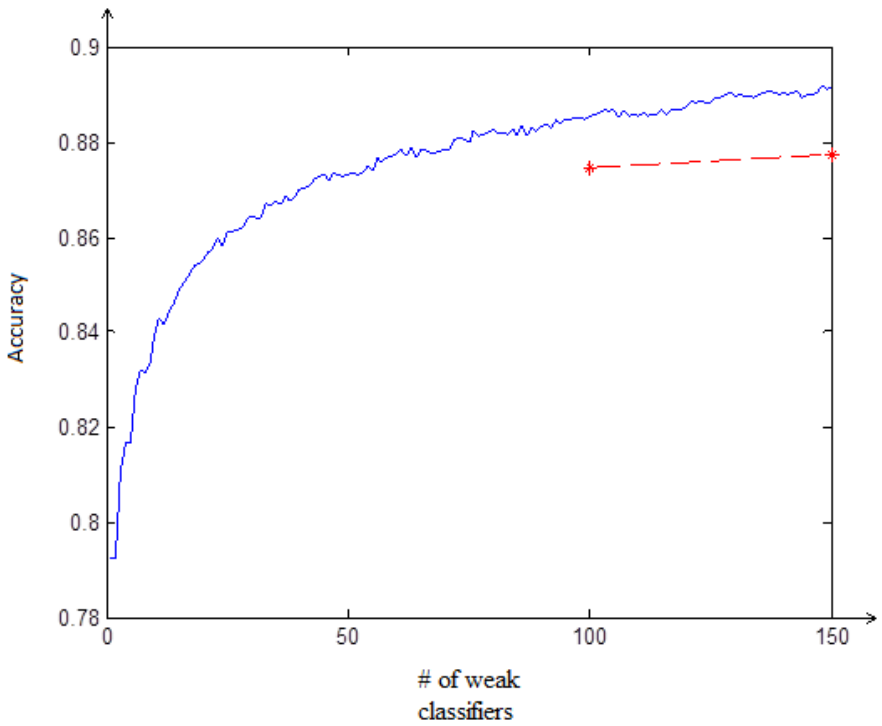


Figure 4.10. AdaBoost weak classifier number vs. AdaBoost Accuracy in Filter Set Size Experiment. Blue: Training samples, Red: Test samples

Raising the number of iterations from 100 to 150, increased training sample classification accuracy from 88.54% to 89.16% (by 0.62%). Test sample classification accuracy increased from 87.47% to 87.75% (by 0.28%).

4.3.2.3. Cell Division and Filter Set Size Experiment. 4 extra feature vector variants are tried. For these variants stripe width is fixed to 8 . V9: 16 vertical cells, no horizontal division. Non- extended root filter set is used. V10: 2 vertical cell division and 3 horizontal cell division. Cell widths are 3, 2, and 3 from left to right. Non-extended root filter set is used. V11: 16 vertical cells, no horizontal division. Extended root filter set is used. V12: 2 vertical cell division and 3 horizontal cell division. Cell widths are 3, 2, and 3 from left to right. Extended root filter set is used.

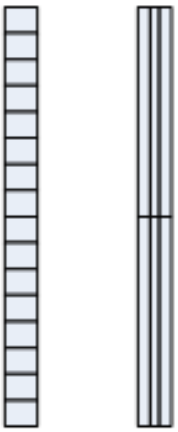


Figure 4.11. Cell division patterns for new variants

Classification accuracies for these variants are given in Tables 4.7 and 4.8.

Table 4.7. 5 fold cross validation average accuracies of classifiers over the training sets.

	V9	V10	V11	V12
Cell division	16x1	2 x3	16x1	2 x3
Extended filter set	No	No	Yes	Yes
Quadratic classifier	80.42%	83.43%	86.25%	85.39%
AdaBoost with 100 iterations	88.09%	87.15%	88.56%	87.54%
AdaBoost with 150 iterations	88.83%	87.54%	88.96%	88.33%
SVM	91.59%	89.21%	91.17%	90.51%

Table 4.8. Accuracy of classifiers over the test set.

	V9	V10	V11	V12
Cell division	16x1	2x3	16x1	2x3
Extended filter set	No	No	Yes	Yes
Quadratic classifier	80.32%	81.57%	82.94%	85.12%
AdaBoost with 100 iterations	87.79%	85.94%	88.19%	86.96%
AdaBoost with 150 iterations	88.28%	86.43%	88.09%	87.08%
SVM	90.01%	88.61%	90.27%	89.24%

Highest accuracy over the test sets is observed when SVM is used with the variant 11. However, it is almost equal to the previously observed highest accuracy. (increase is from 90.26% to 90.27%). Thus we see that increasing vertical cell division from 8 to 16 does not yield a performance increase.

Comparing V2 to V10, we can observe that 8 by 1 cell division is better than 2 by 3 cell division. The test sample classification accuracy of SVM for the former one is 89.83% and for the second one it is 88.61% (1.22% lower). The dimensionality of these variants are similar, V6 being 608 and V12 being 456. Thus there is no huge complexity difference between these feature variants.

AdaBoost test set classification accuracy is highest for Variant 9. AdaBoost accuracy vs. weak classifier number for V9 is shown in Figure 4.13.

Raising weak classifier number from 100 to 150, test sample classification accuracy increased from 87.79% to 88.28% by 0.49%. Since this may show there may be potential another AdaBoost with 200 weak classifiers is trained. This new classifier has higher accuracy on the training set. However classification of test set slightly dropped.

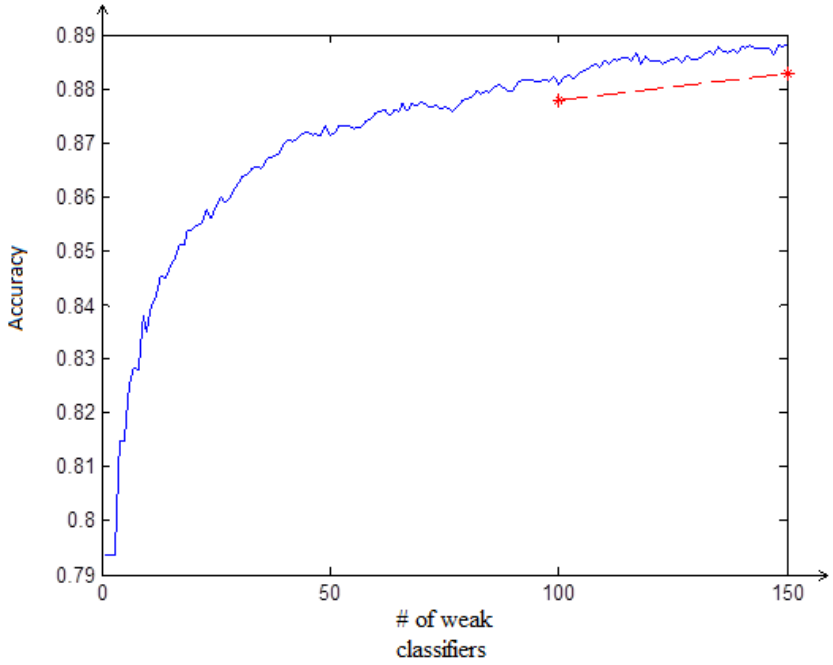


Figure 4.12. AdaBoost weak classifier number vs. AdaBoost Accuracy in Cell Division and Filter Size Experiment. Blue: Training samples, Red: Test samples

Table 4.9. Accuracy of classifiers over the training set.

	V9
Cell division	16x1
Filter set	Non-extended
AdaBoost with 150 iterations	88.83%
AdaBoost with 200 iterations	89.21%

Table 4.10. Accuracy of classifiers over the test set.

	V9
Cell division	16x1
Filter set	Non-extended
AdaBoost with 150 iterations	88.28%
AdaBoost with 200 iterations	88.20%

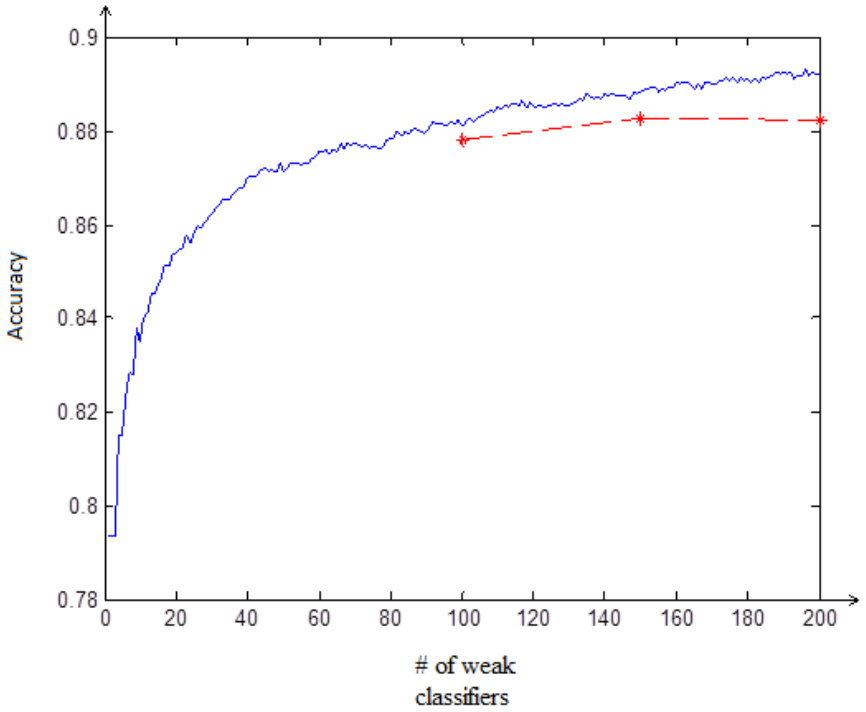


Figure 4.13. AdaBoost weak classifier number vs. AdaBoost Accuracy graph. Blue: Training samples, Red: Test samples

4.3.3. Query Time Analysis

The selected feature - classifier combination will classify all possible samples from a query shelf image. In this analysis, average query times are examined. The important steps in the query time are;

(i) Filtering with the filter set,

(ii) For every column

- Striping the response maps,
- Calculating the descriptor (max-pooling the cells),
- Classification by the classifier.

All experiments documented in this thesis are implemented on a computer with

below system specifications.

- Processor: Intel Core i7-4500U CPU @1.80 GHz, with maximum speed: 2.40 GHz
- Installed memory (RAM): 8.00 GB

Table 4.11. Average query times for Filter Response Features (in seconds).

Filter set: Non-extended						
Variant	V1	V2	V9	V10	V3	V4
Cell division	2x1	8x1	16x1	2x3	2x1	8x1
Stripe width	8	8	8	8	12	12
QC	0.35	0.40	0.75	0.37	0.36	0.40
AdaBoost 100	0.40	0.42	0.54	0.41	0.47	0.50
AdaBoost 150	0.43	0.50	0.60	0.47	0.48	0.59
SVM	0.72	1.41	2.33	1.18	0.73	1.43
Filter set: Extended						
Variant	V5	V6	V11	V12	V7	V8
Cell division	2x1	8x1	16x1	2x3	2x1	8x1
Stripe width	8	8	8	8	12	12
QC	0.55	1.06	2.35	0.81	0.54	1.10
AdaBoost 100	0.59	0.61	0.71	0.61	0.60	0.69
AdaBoost 150	0.64	0.70	0.76	0.67	0.67	0.74
SVM	1.19	2.32	3.84	1.98	1.22	2.42

We observe that,

(i) Stripe width does not affect query time much.

(ii) Using the Extended filter set, doubles the filtering time and descriptor calculation time. Except AdaBoost, it also increases classification time due to the dimension

increase.

(iii) AdaBoost classification time is affected by the number of weak classifiers only. N-comparisons are done for an N step AdaBoost.

(iv) 50 extra weak classifiers do not increase query time much.

4.3.4. Performance

Classifier accuracies over different variants are given in Table 4.12. The accuracies are of the classifiers trained on the training set and tested on the test sets.

Table 4.12. FRF Variants and classifier accuracies.

Filter set: Non-extended						
Variant	V1	V2	V9	V10	V3	V4
Cell division	2x1	8x1	16x1	2x3	2x1	8x1
Stripe width	8	8	8	8	12	12
QC	82.18%	82.13%	80.32%	81.57%	80.63%	80.56%
AdaBoost 100	85.13%	87.34%	87.79%	85.94%	84.53%	86.76%
AdaBoost 150	85.14%	87.46%	88.28%	86.43%	84.58%	87.06%
SVM	87.17%	89.83%	90.01%	88.61%	86.59%	89.56%
Filter set: Extended						
Variant	V5	V6	V11	V12	V7	V8
Cell division	2x1	8x1	16x1	2x3	2x1	8x1
Stripe width	8	8	8	8	12	12
QC	83.57%	83.49%	82.94%	85.12%	82.29%	83.23%
AdaBoost 100	85.60%	87.47%	88.19%	89.96%	84.81%	86.68%
AdaBoost 150	86.05%	87.75%	88.09%	87.08%	85.14%	87.23%
SVM	88.55%	90.26%	90.27%	89.24%	87.69%	90.09%

In the below figures, experiment results are plotted on a "query time x accuracy" space. X axis is query time (in seconds) and y-axis is classifier accuracies (in percent-

ages). QC is plotted in red plus signs, AdaBoost 100 is plotted in blue dots, AdaBoost 150 is plotted in magenta circles and, SVM is plotted in black cross signs. When we observe the figures, we see that SVM reaches higher accuracy, but the query time increases significantly. A query time more than 0.5 second is not very useful. Thus the optimal Filter Response Feature variant seems to be V9 and AdaBoost with 100 weak classifiers where we achieved 87.79% accuracy and query time for a test image is 0.54 seconds.

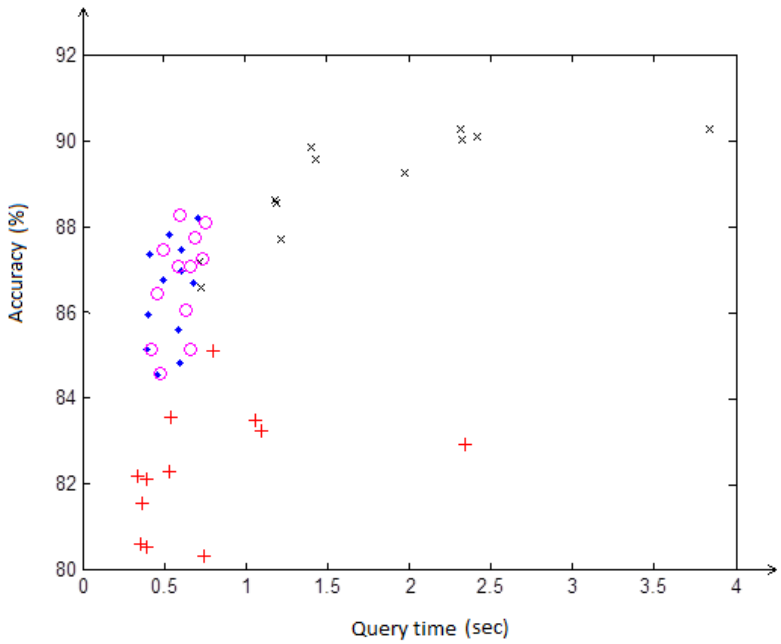


Figure 4.14. Query time vs. Accuracy of Variant - Classifier couples. QC: red pluses, AdaBoost 100: blue dots, AdaBoost 150: magenta circles, SVM: black crosses.

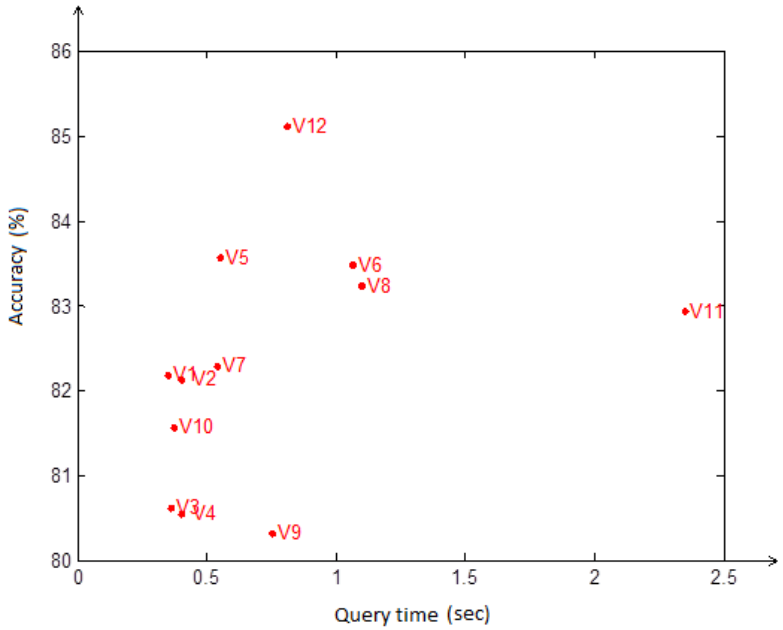


Figure 4.15. Query time vs. Accuracy of FRF Variants for QC

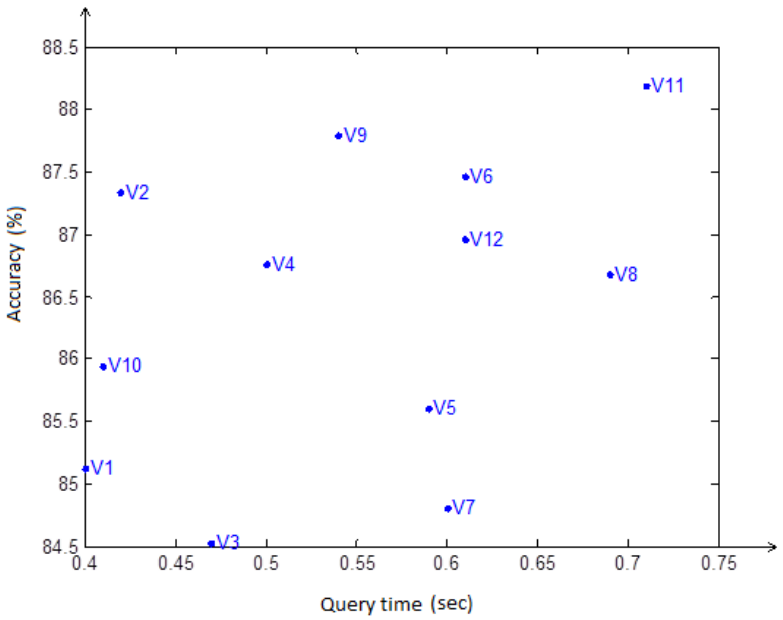


Figure 4.16. Query time vs. Accuracy of FRF Variants for AdaBoost 100

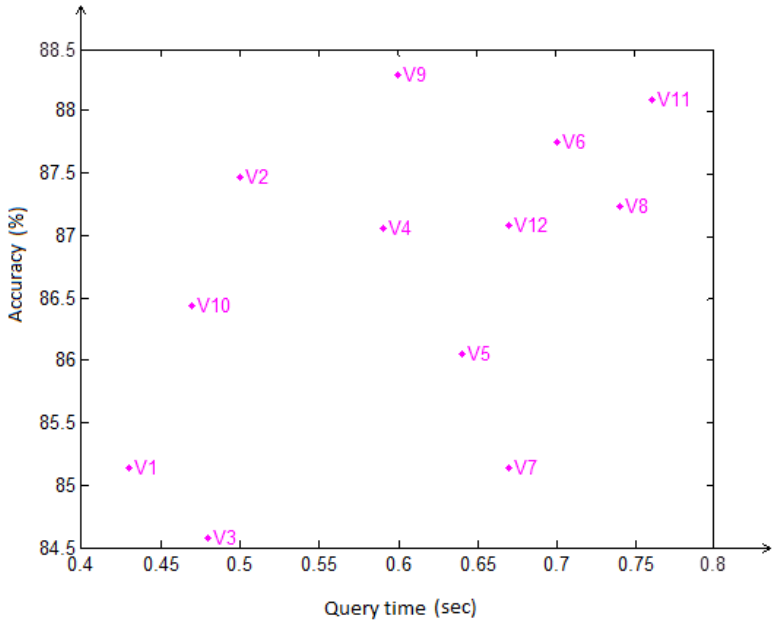


Figure 4.17. Query time vs. Accuracy of FRF Variants for AdaBoost 150

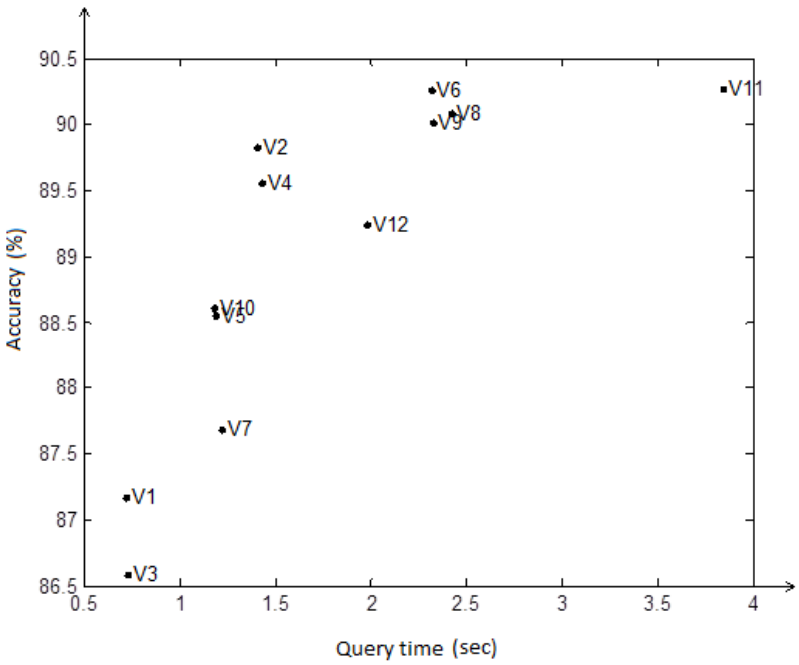


Figure 4.18. Query time vs. Accuracy of FRF Variants for SVM

4.4. Experiments with Histograms of Oriented Gradients

4.4.1. Sample Description

For every positive and negative boundary annotation, a HOG feature vector is calculated. (Negative annotations are randomly selected from the 30% middle intervals between positive annotations.) Feature vector calculation steps are explained below:

(i) The shelf images are converted to grayscale. Grayscale images are resized to fit an image height of 128, preserving aspect ratio. 2 different row sizes are tried as different variants. From this resized grayscale image (which has intensity values in the range [0-255]), x and y image gradients (I_x and I_y) are calculated using gradient operators. (x is for horizontal and y is for vertical dimension)

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

for I_x and

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

for I_y

(ii) Gradient magnitude image M and phase image P are calculated using I_x and I_y.

$$M = \sqrt{{I_x}^2 + {I_y}^2} \tag{4.14}$$

The range of M is dependent on the kernels used for gradient calculation. I_x and I_y have ranges [-255, 255]. Thus M has [0, 360.62).

$$P = \begin{cases} \arctan(\frac{I_y}{I_x}) & \text{if } I_y > 0 \\ \arctan(\frac{I_y}{I_x}) + \pi & \text{if } I_y < 0 \\ 0 & \text{if } I_y = 0 \text{ and } I_x > 0 \\ \pi & \text{if } I_y = 0 \text{ and } I_x < 0 \\ undefined & \text{if } I_y = 0 \text{ and } I_x = 0 \end{cases} \tag{4.15}$$

where arc tan has range [0,π). Thus P has range [0,2π)

(iii) Histogram bin maps are calculated using phase and magnitude images. For every pixel in the resized gray scale image, corresponding histogram bin is calculated using the phase and magnitude values for that pixel. The histogram bin indices are stored in the bin image. Using various histogram calculation methods, various bin maps are obtained.

- Histogram binning

  In order to produce a gradient histogram, the gradients which are distributed in the continuous interval [0,2π) need to be divided into discrete bins. Different histogram binning schemes are used in different variants. Uniform, non-uniform binning methods, signed or unsigned phases are tried. In some trials some bins are merged. The experimented methods will be explained in detail in Section 4.4.2.

- Magnitude threshold

  In the benchmark method, every gradient increases the bin its phase P_c (x,y) falls into, by 1. In some trials an extra bin -reserved for small magnitude gradients- is added to the histogram. The gradients with magnitudes greater than the magnitude threshold, increases the bin its phase falls into by 1. The gradients with magnitudes smaller than the magnitude threshold, increases the extra bin by 1. Various magnitude thresholds are tried. An illustrating figure is given in Figure 4.19 where $\theta = 2\pi/n$ results in n gradient bins and magnitude threshold T results in an extra radial bin.

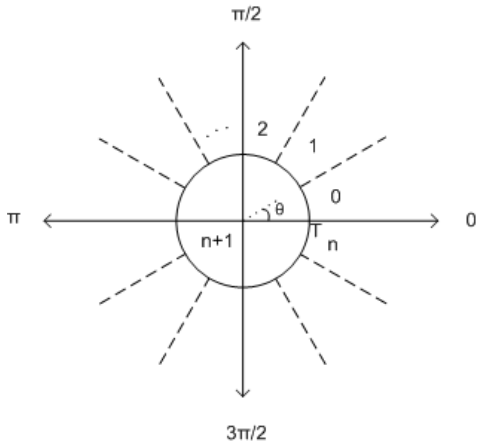


Figure 4.19. Gradient binning pattern with a magnitude threshold (bin indices are shown for different gradients)

- Magnitude weighted histograms

  A variant is tried where a gradient increases the bin its phase fall into not by 1 but by its magnitude. This bin increasing method is the standard version proposed in the original paper [33].

(iv) For every column index c, the stripe bin image is extracted centered around the sample column index-biased 0.5 pixel to the left side. This bias is due to the fact that we used an even number of pixels as stripe width. The center point of a stripe is fractional. Stripe bin images are extracted in a sliding window fashion for all possible columns.

(v) Stripe bin images are divided into cells vertically and horizontally. Various number of horizontal and vertical divisions are applied.

(vi) For every cell, the histogram of gradients is calculated. Histograms are concatenated and the resulting feature vector is L1 normalized.

4.4.2. Experiments

Different HOG descriptor variants are extracted from test and training sets. 3 different classifiers are trained using the descriptors in the training set: A quadratic classifier, an AdaBoost with 150 weak classifiers, and an SVM with Radial Basis Function kernel. For optimal SVM parameter selection 5 fold grid search is implemented

similar to the SVM trainings in the previous parts.

4.4.2.1. Phase Binning Experiment. In the first trials, the stripes are not divided into cells and no magnitude threshold is implemented. 3 different gradient phase binning approaches are tried:

V1. Uniform binning with phase bin width $\theta=\pi/4$. The interval is divided into 8 bins. Descriptor has 8 dimensions.

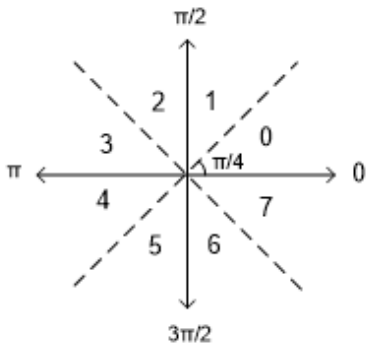


Figure 4.20. Gradient binning pattern for HOG variant V1

V2. Uniform binning with phase bin width $\theta=\pi/6$. The interval is divided into 12 uniform bins. Descriptor has 12 dimensions.

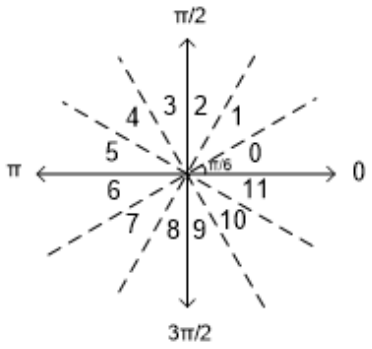


Figure 4.21. Gradient binning pattern for HOG variant V2

V3. Uniform binning with phase bin width $\theta=\pi/20$. The interval is divided into 40 uniform bins. Descriptor has 40 dimensions.

The training and test set classification accuracies of different classifier - HOG variant combinations are given in the Tables 4.13 and 4.13.

Table 4.13. Average of 5 fold accuracies of classifiers over the training set.

	V1	V2	V3
Quadratic classifier	83.04%	84.17%	82.19%
AdaBoost	83.64%	83.84%	85.75%
SVM	85.37%	86.03%	85.18%

Table 4.14. Accuracy of classifiers over the test set.

	V1	V2	V3
Quadratic classifier	83.83%	83.91%	82.67%
AdaBoost	84.07%	84.60%	85.32%
SVM	85.45%	85.66%	85.96%

For all variants, SVM achieved the highest accuracy over the test sets. It is observed that increasing the number of bins enabled the classifier to separate the positive and negative samples better. However, every extra bin increases the dimension of the HOG vector.

Comparing V1 to V2, we see that dimension increased from 9 to 12 and accuracy increased by 0.21%. Comparing V2 to V3, we see that dimension increased from 12 to 40 and accuracy increased by 0.30%. Although the accuracy increased by 30%, the increase in dimension is too much of a price for this performance raise. In later trials

we will divide the images into cells and every new cell will multiply the histogram dimension. Thus here the optimal decision here seems to be V2.

Figure 4.22 is the AdaBoost training sample set accuracy vs number of weak classifiers for the variant V2.

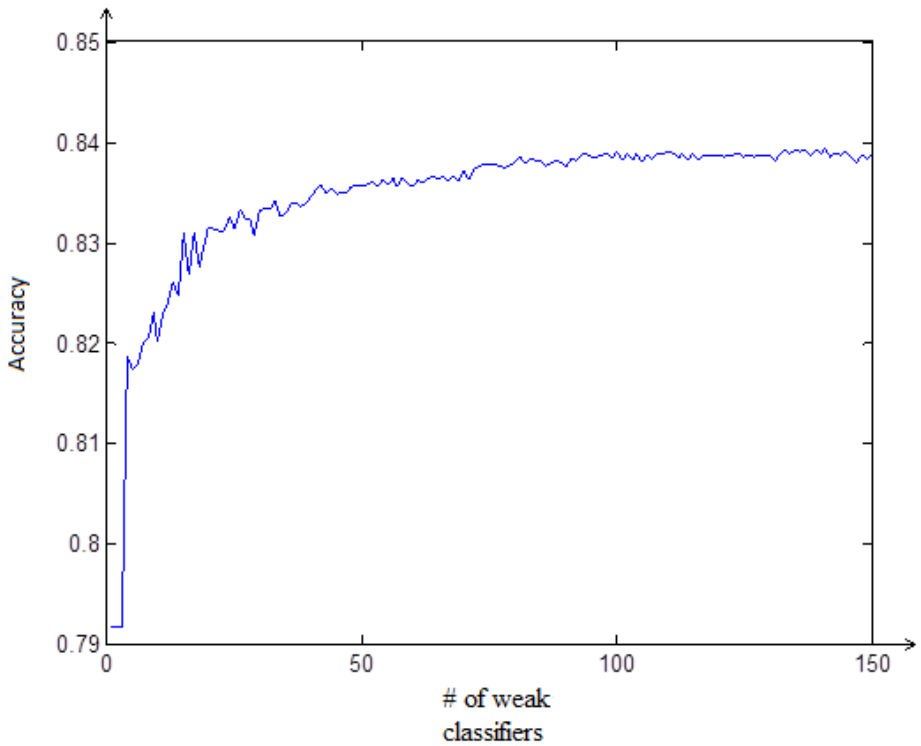


Figure 4.22. AdaBoost weak classifier number vs. AdaBoost accuracy graph

4.4.2.2. Unsigned Phase Binning Experiment. In this trials, gradients with same absolute phase are binned together i.e. light to dark and dark to light transitions in the same directions are treated the same. The same gradient interval binning degrees (θ's) in the previous experiment are used respectively.

V4: Uniform binning with $\theta=\pi/4$. The interval is divided into 4 bins. Descriptor has 4 dimensions.

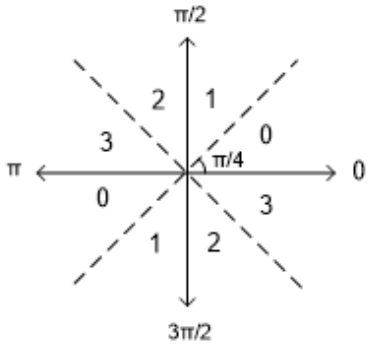


Figure 4.23. Gradient binning pattern for HOG variant V4

V5: Uniform binning with $\theta=\pi/6$. The interval is divided into 6 bins. Descriptor has 6 dimensions.

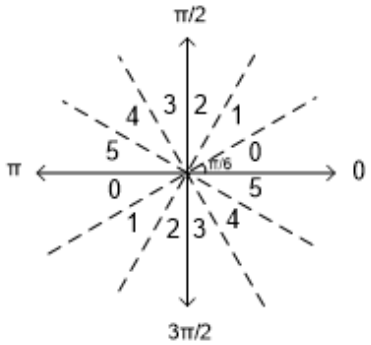


Figure 4.24. Gradient binning pattern for HOG variant V5

V6: Uniform binning with $\theta=\pi/20$. The interval is divided into 20 bins. Descriptor has 20 dimensions.

For all variant - classifier combinations, 5 fold cross validation is implemented. That is; the training data is divided into 5 subsets. Every subset is tested using the union of the 4 remaining subsets as a training set. The average accuracies are recorded. Later, new classifiers are trained over the training sets (without folding) and the descriptors in the discrete test sets are classified.

Table 4.15. Average of 5 fold accuracies of classifiers over the training set.

	V4	V5	V6
Quadratic classifier	81.00%	80.17%	80.10%
AdaBoost	81.54%	80.99%	81.75%
SVM	84.67%	83.02%	82.81%

Table 4.16. Accuracy of classifiers over the test sets.

	V4	V5	V6
Quadratic classifier	80.83%	80.90%	79.99%
AdaBoost	81.07%	81.76%	82.33%
SVM	83.04%	82.83%	83.01%

For all variants, SVM achieved the highest accuracy over the test sets. Using unsigned phase decreased the accuracy by approximately 3%. Here we see that when we use un-signed gradients, we lose significant information.

4.4.2.3. Image Height Experiment. In this experiments V2 is selected as benchmark variant and a compared to the case of resizing the images to a fit a row size of 256 (instead of 128). The training and test set classification accuracies of this new variant is given in Table 4.17 and 4.18.

Table 4.17. Average of 5 fold accuracies of classifiers over the training sets.

Variant	V2	V7
Row size	128	256
Quadratic classifier	82.17%	82.17%
AdaBoost	83.88%	80.99%
SVM	84.83%	81.72%

Table 4.18. Accuracy of classifiers over the test sets.

Variant	V2	V7
Row size	128	256
Quadratic classifier	83.91%	84.52%
AdaBoost	84.60%	84.96%
SVM	85.66%	85.69%

Working on larger images increased the description extraction, training and testing times a lot. Although, the increase in accuracy is almost negligible. Thus, in later experiments we stick with 128 as the row size.

4.4.2.4. Magnitude Weighted Histogram Experiment. In previous experiments, every gradient increases its corresponding bin by 1 before L1 normalization. In this experiment we try a new method; every gradient increase its corresponding bin by its magnitude. Here the descriptor vector is not a regular frequency vector but a weighted frequency vector; the weights being the magnitudes. The training and test set classification accuracies of this new variant is given in Table 4.19 and 4.20.

Table 4.19. Average of 5 fold accuracies of classifiers over the training sets.

Variant	V2	V8
Bin increase method	Unit step	Magnitude step
Quadratic classifier	82.17%	79.11%
AdaBoost	83.88%	81.74%
SVM	84.83%	84.63%

Table 4.20. Accuracy of classifiers over the test sets.

Variant	V2	V8
Bin increase method	Unit step	Magnitude step
Quadratic classifier	83.91%	80.14%
AdaBoost	84.60%	82.28%
SVM	85.66%	84.22%

We observe no accuracy boost in magnitude step bin increasing. Thus we use unit step bin increasing in the later experiments.

4.4.2.5. Magnitude Threshold with Extra Histogram Bin Experiment. In this experiments, an extra bin -reserved for small magnitude gradients- is added to the histogram. The gradients with magnitudes greater than the magnitude threshold, increases the bin its phase falls into by 1. The gradients with magnitudes smaller than the magnitude threshold, increases the extra bin by 1.

Before selecting the magnitude threshold parameter, the training data is observed. The magnitude images usually do not span their entire range. Because the differences between neighboring pixels in natural images are not usually high, the gradient magnitude values tend to be smaller than the possible maximum values. A

gradient magnitude image may have an intensity value of 360 theoretically. However a typical gradient magnitude image usually have a maximum intensity of around 160.

100 Sample images are randomly selected to observe the magnitude image histograms. The cumulative histogram of the average magnitude image is shown in Figure 4.25.



Figure 4.25. A sample gradient magnitude image (brightness inverted for visualization)

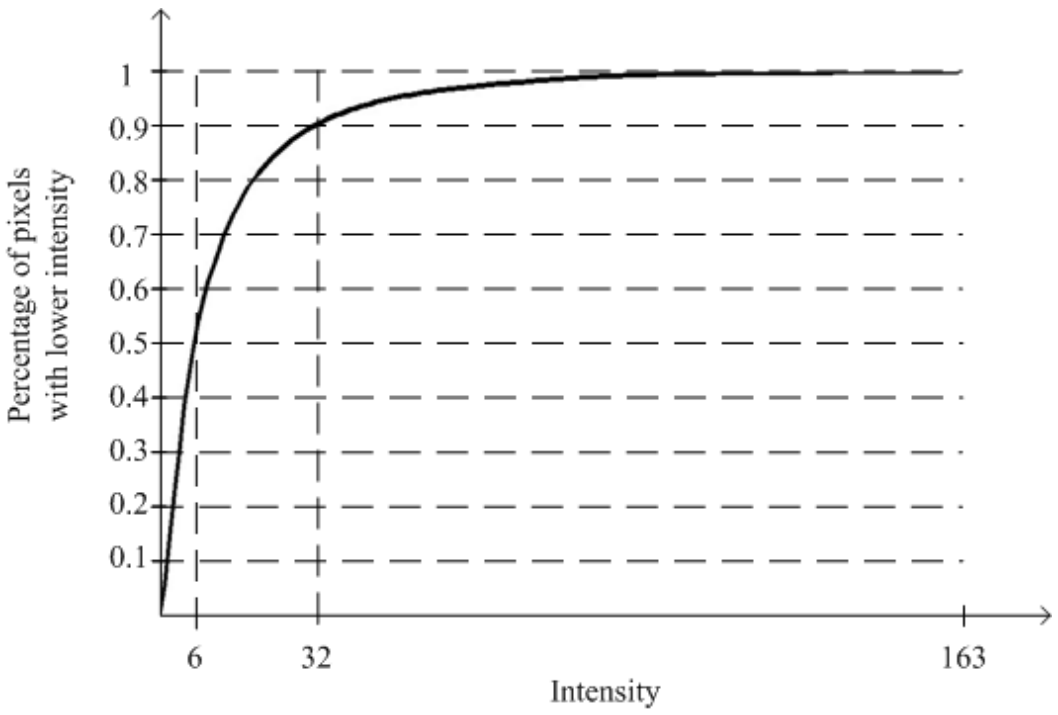


Figure 4.26. Cumulative intensity histogram of the sample magnitude images

Moreover, most of the intensity values are swarmed in the lower side of the histogram. Even though the highest intensity for the sample gradient images is 163,

90 percent of the pixels have intensity values less than 32. A magnitude threshold parameter of 6 will put approximately half of the pixels in the extra bin.

For benchmark variant, V2 is selected. Various magnitude thresholds are tried and classification accuracies of classifiers are compared to the benchmark variant V2.

Table 4.21. Average of 5 fold accuracies of classifiers over the training sets.

Variant	V2	V9	V10	V11	V12	V13
Magnitude threshold	0	2	4	6	24	32
Quadratic classifier	82.17%	82.62%	82.56%	82.45%	80.23%	80.02%
AdaBoost	83.88%	84.12%	84.39%	84.30%	83.11%	82.50%
SVM	84.83%	85.11%	85.36%	84.80%	82.07%	81.76%

Table 4.22. Accuracy of classifiers over the test sets.

Variant	V2	V9	V10	V11	V12	V13
Magnitude threshold	0	2	4	6	24	32
Quadratic classifier	83.91%	84.02%	83.95%	82.87%	80.17%	79.89%
AdaBoost	84.60%	85.17%	85.12%	84.91%	81.98%	81.81%
SVM	85.66%	86.02%	86.27%	85.07%	82.20%	82.11%

Again, SVM performed best in this experiments. It is observed that using a magnitude threshold of 4 enabled the SVM to achieve the highest accuracy. The increase in accuracy is 0.61%.

4.4.2.6. Magnitude Threshold Without Extra Histogram Bin Experiment. In this experiments, the extra bin -reserved for small magnitude gradients- is removed before normalization. The gradients with magnitudes greater than the magnitude threshold,

increases the bin its phase falls into by 1. The gradients with magnitudes smaller than the magnitude threshold are not counted. The same threshold options used in the previous experiment are tried. Classification accuracies of classifiers are compared to the benchmark variant V2.

Table 4.23. Average of 5 fold accuracies of classifiers over the training sets.

Variant	V2	V14	V15	V16	V17	V18
Magnitude threshold	0	2	4	6	24	32
Quadratic classifier	82.17%	81.95%	81.22%	81.50%	78.12%	79.33%
AdaBoost	83.88%	82.06%	83.84%	82.30%	82.94%	80.74%
SVM	84.83%	84.05%	83.21%	83.20%	80.63%	80.46%

Table 4.24. Accuracy of classifiers over the test sets.

Variant	V2	V14	V15	V16	V17	V18
Magnitude threshold	0	2	4	6	24	32
Quadratic classifier	83.91%	81.77%	83.05%	80.45%	79.07%	77.04%
AdaBoost	84.60%	84.14%	84.81%	83.94%	79.99%	80.86%
SVM	85.66%	84.65%	85.86%	83.96%	81.15%	80.43%

When we compare the results of Experiment 5 and 6, it is observed that ignoring the ratio of the gradients having smaller magnitude, mixed the positive and negative samples more. This is probably because the variants in the Experiment 6 are affected by noise a lot. The classifiers in this experiment are very unsuccessful at labeling some of the non-boundary samples which has very low gradients mostly, with a small region with strong horizontal gradients. Since we normalize the frequency vector after ignoring the weak gradients, the effect of the small horizontal structure is amplified, making the descriptor resemble that of a positive descriptor. Even a slight vertical

shadow on an empty shelf is labeled as a product boundary. We learn that the ratio of weak gradients is important in boundary discrimination.

4.4.2.7. Cell Division Experiment. The experiments up to this point showed us that V10 with SVM achieves the highest accuracy (86.27%) without increasing the dimension a lot. This variant is has 13 dimensions. The first 12 are reserved for different phase bins for strong gradients and the last one is reserved for the weak gradients. The magnitude threshold separating the weak and strong gradients is 4. In this experiment different cell division patterns are tried. Benchmark variant is V10.

Four different cell division patterns are used. 1 by 4, 2 by 2 ,4 by 1, and 4 by 2. HOG descriptors are calculated for all cells and concatenated in the same order. Normalization is done after concatenation.

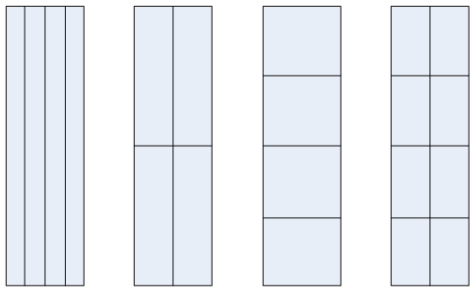


Figure 4.27. The cell division patterns used. 1x4, 2x2, 4x1, and 4x2 consecutively

The classifier accuracies are given in Tables 4.25 and 4.26.

Table 4.25. Average of 5 fold accuracies of classifiers over the training sets.

Variant	V10	V19	V20	V21	V22
Cell division	1x1	1x4	2x2	4x1	4x2
Quadratic classifier	82.56%	81.46%	82.43%	84.63%	84.11%
AdaBoost	84.39%	84.72%	85.89%	86.28%	86.23%
SVM	85.36%	93.64%	95.86%	95.61%	97.28%

Table 4.26. Accuracy of classifiers over the test sets.

Variant	V10	V19	V20	V21	V22
Cell division	1x1	1x4	2x2	4x1	4x2
Quadratic classifier	83.95%	82.59%	83.96%	84.38%	81.99%
AdaBoost	85.12%	85.67%	86.19%	86.61%	86.09%
SVM	86.27%	87.65%	88.51%	87.93%	87.65%

Here we see that dividing the stripes into cells increased accuracy for all classifiers. The highest accuracy is obtained when SVM is used on V20. The accuracy increased by 2.24% when we divide the stripes into 2 by 2 cells.

4.4.2.8. Alternative Histogram Binnings Experiment. The 20th variant is taken as benchmark and two more different histogram binning schemes tried. The schemes are shown in Figure 4.28.



Figure 4.28. Histogram phase binning schemes for Alternative Histogram Binnings Experiment (V23 and V24 respectively)

Table 4.27. Average of 5 fold accuracies of classifiers over the training sets.

Variant	V20	V23	V24
Quadratic classifier	82.43%	79.67%	79.00%
AdaBoost	85.89%	86.08%	85.22%
SVM	88.86%	87.07%	88.05%

Table 4.28. Accuracy of classifiers over the test sets.

Variant	V20	V23	V24
Quadratic classifier	83.96%	80.57%	81.14%
AdaBoost	86.19%	85.74%	86.04%
SVM	88.51%	87.21%	87.89%

This new binning structures are tried in order to decrease dimension. The decrease in dimension is an advantage. However, accuracy decreased as well.

4.4.3. Query Time Analysis

The selected feature - classifier combination will classify all possible samples from a query shelf image. In this analysis, average query times are examined. The important steps in the query time are;

(i) Bin image calculation,

(ii) For every column

- histogram calculations from stripe bin image cells, and
- classification by the classifier.

4.4.3.1. Query Time Analysis for Signed and Unsigned Phase Experiment. In this experiment we use signed and unsigned gradients. Also we used different bin widths. We saw that SVM is affected by the feature dimension. Other than that, no significant query time variance is observed.

Table 4.29. Average query times for HOG variant - classifier combinations in Phase Binning Experiment and Unsigned Phase Binning Experiment (in seconds).

Variant	V1	V2	V3	V4	V5	V6
Phase bin width	$\pi/4$	$\pi/6$	$\pi/20$	$\pi/4$	$\pi/6$	$\pi/20$
Signed/Unsigned	Signed	Signed	Signed	Unsigned	Unsigned	Unsigned
Quadratic classifier	0.06	0.07	0.08	0.06	0.06	0.07
AdaBoost	0.24	0.25	0.30	0.23	0.24	0.27
SVM	0.11	0.16	0.24	0.10	0.10	0.18

4.4.3.2. Query Time Analysis for Image Height Experiment. In this experiment images were resized to a row size of 256 pixels. This does not change the dimensionality of the descriptors, thus the classification time is not affected much. However, bin image calculation times and histogram calculation times are doubled since the image region to be iterated is doubled.

Table 4.30. Average query times for HOG variant - classifier combinations in Image Height Experiment (in seconds).

Variant	V2	V4
Row size	128	256
QC query time	0.07	0.12
AdaBoost query time	0.25	0.30
SVM query time	0.16	0.22

4.4.3.3. Query Time Analysis for Magnitude Weighted Histogram Experiment. In the experiment the bins are not increased 1 by 1; but they are increased by the magnitudes of the gradients. No significant time change is observed.

4.4.3.4. Query Time Analysis for Magnitude Threshold Experiments. In these experiments, the gradient magnitudes are compared to a magnitude threshold level. No significant time change is observed.

4.4.3.5. Query Time Analysis for Cell Division Experiment. In these experiment, histograms are calculated for different cells and concatenated. This increased the dimension and thus, the classification times for QC and SVM.

Table 4.31. Average query times for HOG variant - classifier combinations in Magnitude Weighted Histogram Experiment (in seconds).

Variant	V10	V19	V20	V21	V22
Cell division	1x1	1x4	2x2	4x1	4x2
QC query time	0.07	0.08	0.08	0.08	0.10
AdaBoost query time	0.25	0.25	0.25	0.25	0.25
SVM query time	0.16	0.28	0.28	0.28	0.37

4.4.3.6. Query Time Analysis for Alternative Histogram Binnings Experiment. In the experiment, some histogram bins are merged together to attain histogram size reduction. As benchmark method, V10 is selected. This variant is 52 dimensional (4 cells x 13 bin histograms). 2 different merging methods decreased the histogram size from 13 to 4 and 6 respectively. Thus the descriptor sizes are 16 and 24 for 4 celled variants consecutively.

Table 4.32. Average query times for HOG variant- classifier combinations in Alternative Histogram Binnings Experiment (in seconds).

Variant	V10	V23	V24
Descriptor dimension	52	16	24
Quadratic classifier	0.08	0.07	0.08
AdaBoost	0.25	0.25	0.25
SVM	0.28	0.16	0.17

4.4.4. Performance

We value high accuracy and low query time. A query time under 0.3 seconds seems to be appropriate for our system. In the below figures, we see accuracies of different descriptor-classifier combinations. QC results are plotted in red pluses, AdaBoost results are in blue dots and SVM results are in black crosses. We can conclude that describing boundaries with V20 and using an SVM would be the best choice. For this variant, query time is 0.28 seconds and accuracy is 88.51%.

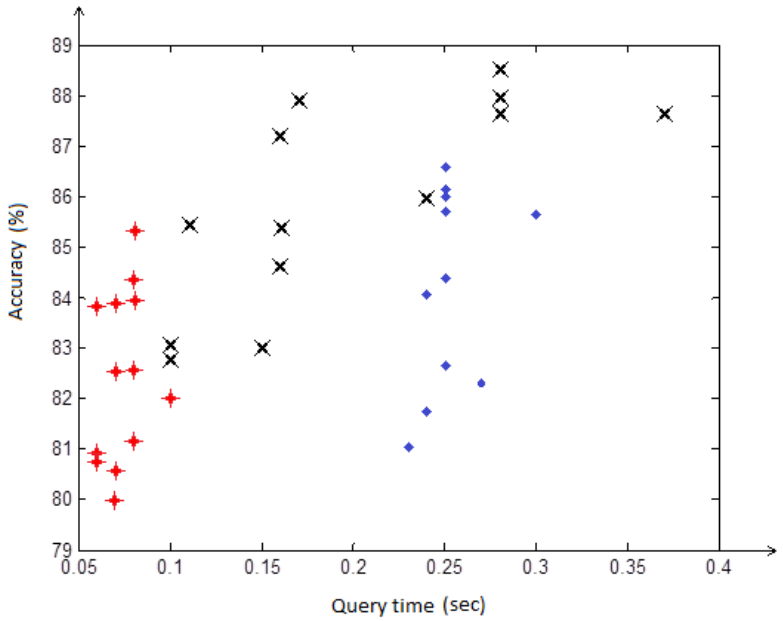


Figure 4.29. Query time vs. Accuracy of HOG Variant - Classifier couples. QC: red pluses, AdaBoost: blue dots, SVM: black crosses

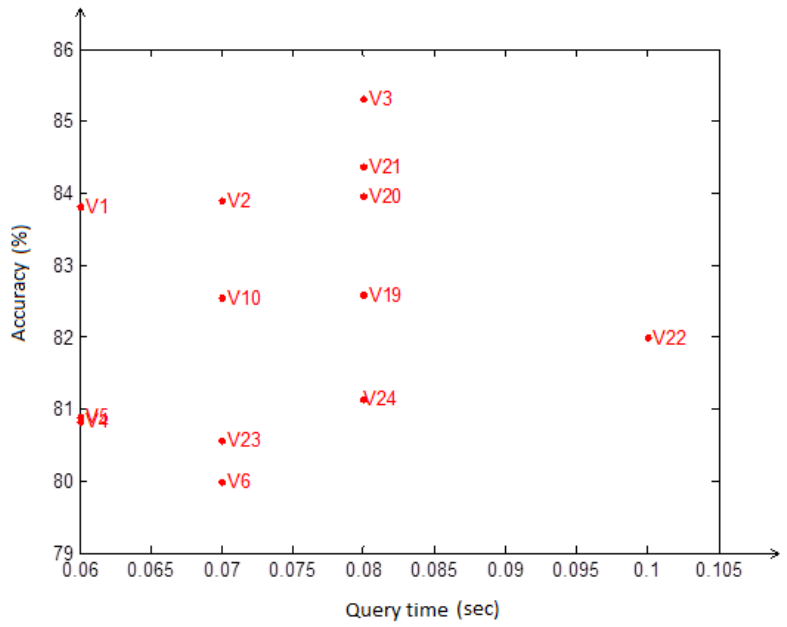


Figure 4.30. Query time vs. Accuracy of HOG Variants for QC

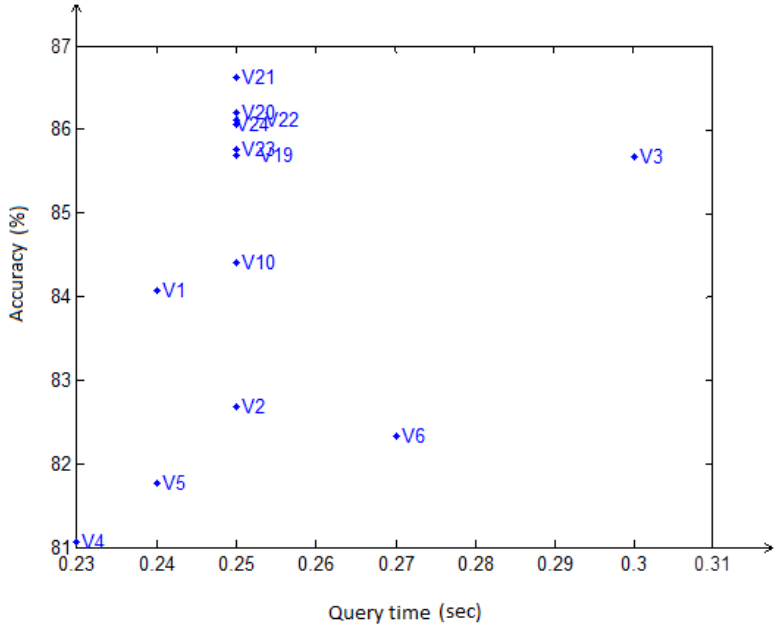


Figure 4.31. Query time vs. Accuracy of HOG Variants for AdaBoost

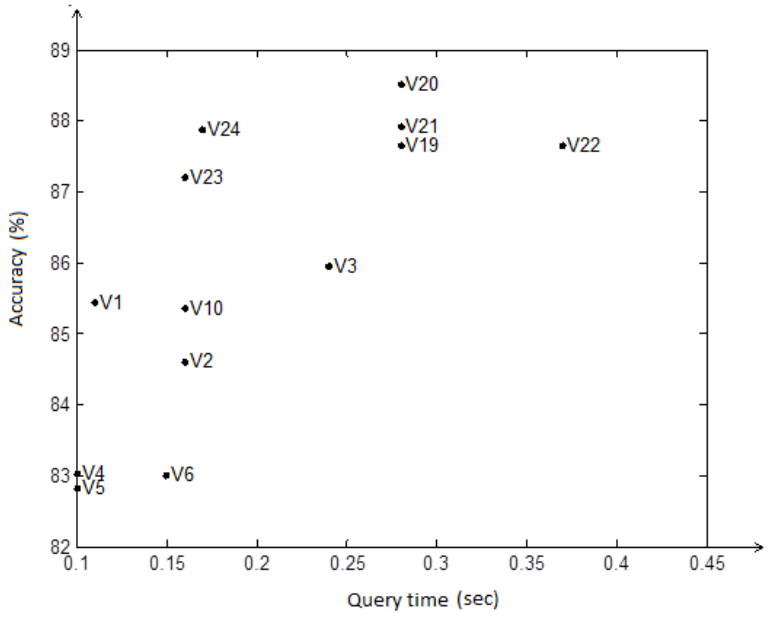


Figure 4.32. Query time vs. Accuracy of HOG Variants for SVM

4.5. Experiments with Local Binary Pattern Histograms

In this chapter, we summarize the experiments in which the boundaries are represented using the Local Binary Pattern histograms [35].

4.5.1. Sample Description

For every positive and negative boundary annotation, a histogram of Local Binary Patterns feature vector is calculated. (Negative annotations are randomly selected from the 30% middle intervals between positive annotations.) Feature vector calculation steps are explained below.

(i) Images are resized to fit an image height of 128, preserving aspect ratio. And they are converted to grayscale.



Figure 4.33. A sample shelf image

(ii) The LBP image is calculated. Since we compare 8 neighbors, he binary code has 8 digits. Converting the code to decimal, we have labels in the interval 0-255. For visualization, labels are represented with different gray values (0-255, black to white) in Figure 4.34.

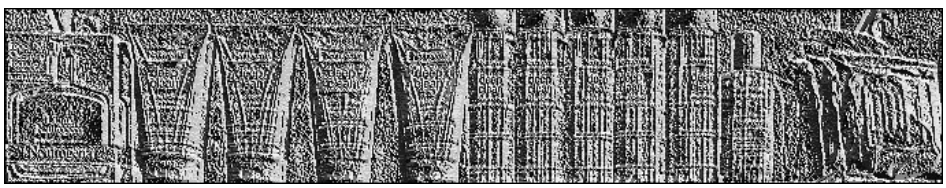


Figure 4.34. LBP image visualization for the sample shelf image

(iii) For every column index c, a descriptor vector is calculated. The steps are as follows:

- The stripe LBP image is extracted (centered around the sample column index-biased 0.5 pixel to the left side). Stripe width is selected as 8.
- Stripe LBP image is divided into cells vertically and horizontally. Various number of horizontal and vertical divisions are applied.
- For every cell, the histogram of LBP's is calculated. Different histogram generation variants are tried. The histograms are concatenated and then forming feature vector is L1 normalized.

Ahonen *et al.* (2006) achieved satisfactory results in face detection with concatenated histograms of LBP's [46].

(iv) After the descriptors are extracted, positive and negative samples descriptors are collected using annotations.

4.5.2. Experiments

Different LBP variants are extracted from test and training sets. 3 different classifiers are trained using the descriptors in the training set: A quadratic classifier, an AdaBoost with 150 weak classifiers, and an SVM with Radial Basis Function kernel. For optimal SVM parameter selection 5 fold grid search is implemented similar to the SVM trainings in the previous parts.

4.5.2.1. Non Uniform LBP Histogram Experiment. In this experiment no cell divisions are applied. The benchmark method explained above is compared to the case where non-uniform LBP's are counted in the same histogram bin. Uniform and non-uniform LBP's are explained below.

Uniform LBP's are the ones where the number of binary transitions (0 to 1 or 1 to 0) in the binary code are at most 2. Some examples are given below.

Table 4.33. LBP examples.

0 binary transitions	1 binary transition	2 binary transitions	3 to 7 transitions
00000000	00111111	00111000	01000100
11111111	11110000	11111101	10010000
	00000001	11100001	01010100
	⋮	⋮	⋮

The uniform LBP's are the ones where brighter neighbor pixels and darker neighbor pixels are not intertwined, but on separate sides. Among 256 LBP's 58 of them are uniform and 198 of them are non- uniform. However it is observed that around 90% of the LBP's in the texture images are uniform [47]. Thus a new type of histogram is used where every uniform LBP has a discrete bin and all non-uniform LBP's are counted in one bin. With this method, histogram size drops to 59 from 256. In our experiments we observed that the ratio of non uniform LBP's is around 20%.

The training and test set classification accuracies of two LBP variant and 3 classifier combinations are given in the Tables 4.34 and 4.35. For all variant - classifier combinations, 5 fold cross validation is implemented. That is; the training data is divided into 5 subsets. Every subset is tested using the union of the 4 remaining subsets as a training set. The average accuracies are recorded. Later, new classifiers are trained over the training sets (without folding) and the descriptors in the discrete test sets are classified.

First table shows the 5 fold average accuracies over of the training descriptor sets. Second table shows the accuracies of classifiers trained over the training descriptor set and tested over the test descriptor set. The SVM's are trained with the optimal parameters found in the 5 fold grid search.

Table 4.34. Average of 5 fold accuracies of classifiers over the training sets.

Variant	V1	V2
QC	74.49%	81.59%
AdaBoost	84.60%	84.52%
SVM	86.12%	86.34%

Table 4.35. Accuracy of classifiers over the test sets.

Variant	V1	V2
QC	73.41%	82.47%
AdaBoost	84.67%	84.76%
SVM	86.24%	86.42%

Here we observe that, using uniform LBP's adds significant complexity decrease by lowering the dimension more than 75% while maintaining the accuracy - even increasing it slightly. Thus in later experiments uniform LBP's are used.

The accuracy during the AdaBoost training over the V2 training set is given in Figure 4.35.

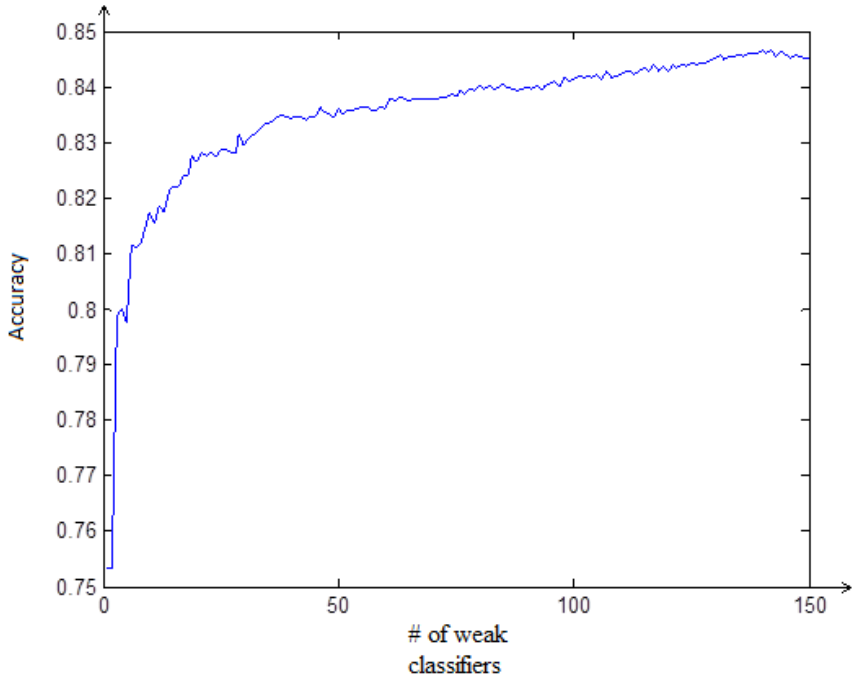


Figure 4.35. AdaBoost weak classifier number vs. AdaBoost accuracy graph (on training sample set)

4.5.2.2. Cell Division Experiment. In this experiment we use 2 by 2 cell division, and compare the results to the no cell division case.

Table 4.36. Average of 5 fold accuracies of classifiers over the training sets

Variant	V2	V3
QC	81.59%	85.28%
AdaBoost	84.52%	85.69%
SVM	86.34%	88.77%

Table 4.37. Accuracy of classifiers over the test sets.

Variant	V2	V3
QC	82.47%	83.81%
AdaBoost	84.76%	85.51%
SVM	86.42%	88.24%

Cell division caused the descriptor dimension to increase from 59 to 236. However, it increased the accuracy for all classifiers. For SVM, it increased by 2% and reached to 88.24%.

4.5.3. Query Time Analysis

The selected variant - classifier combination will classify all possible samples from a query shelf image. In this analysis, average query times are examined. The important steps in the query time are;

(i) LBP image calculation,
(ii) For every column
- histogram calculations from stripe LBP image cells
- classification by the classifier.

For the tried variants, LBP image calculation and histogram calculation times do not differ. The difference in time is observed due to different classification times.

Table 4.38. Average query times for LBP variants and classifiers (in seconds).

Variant	V1	V2	V3
QC	0.17	0.15	0.16
AdaBoost	0.24	0.25	0.25
SVM	1.54	0.53	1.50

4.5.4. Performance

Classifier accuracies over different variants are given in Table 4.39. The accuracies are of the classifiers trained on the training set and tested on the test sets.

Table 4.39. LBP variants and classifier accuracies.

Variant	V1	V2	V3
QC	73.41%	82.47%	83.81%
AdaBoost	84.67%	84.76%	85.51%
SVM	86.24%	86.42%	88.24%

In Figure 4.36, experiment results are plotted on a "query time x accuracy" space. X axis is query time (in seconds) and y-axis is classifier accuracies (in percentages). QC is plotted in red pluses, AdaBoost with 150 weak classifiers is plotted in blue crosses and, SVM is plotted in black dots.

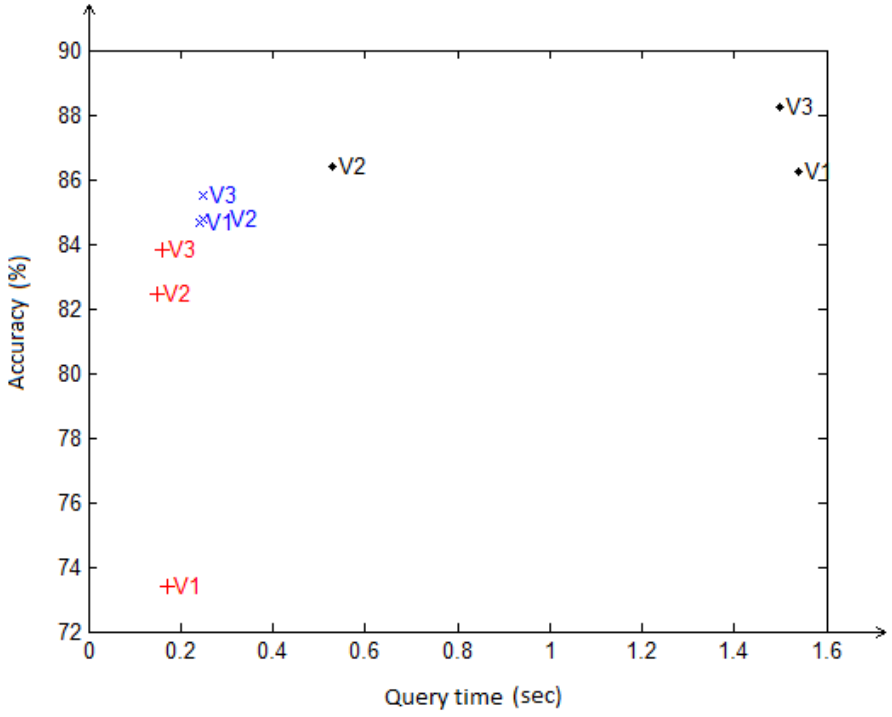


Figure 4.36. Query time vs. Accuracy of LBP Variant - Classifier couples

When we observe the figures, we see that V3 variant (uniform LBP's of 2 by 2 cells) reaches the highest accuracy for all classifiers. And for all descriptor variants SVM reaches the highest accuracies. However our constraint of 0.5 seconds does not enable us to use V3 with SVM. Thus we conclude that the optimal variant is chosen as V2 with SVM classifier.

Overall, LBP histograms tried in this experiment did not reach satisfactory accuracy levels. Another remark is that for AdaBoost, V1 and V2 are approximately the same. This is probably because AdaBoost weak classifiers are mostly interested in the features corresponding the histogram bins of uniform LBP's.

4.6. Score Signal Peak Detection

In the previous 3 sections, we searched for a descriptor and classifier which achieves high accuracy and small query time. For every descriptor type, we selected the optimal variant which achieves the highest accuracy at a query time under 0.6 seconds. Here it is important to mention the fact that we had actually achieved a classification accuracy of 90.26% using SVM with the Filter response feature V6. However the query time was 2.32 seconds. Considering the query time is 8 times higher and the accuracy increase is less than 2%, we did not consider this variant to better.

Table 4.40. Optimal variant, optimal classifier, query time, and accuracy for different descriptor types.

Description method	Optimal variant	Optimal classifier	Query time (sec)	Accuracy
Filter response features	V9	AdaBoost	0.54	87.79%
HOG	V20	SVM	0.28	88.51%
LBP Histograms	V2	SVM	0.53	86.42%

The best combination is HOG V20 with SVM classifier since it achieves the highest accuracy and the smallest query time. After we decide on a descriptor type, descriptor variant and a classifier, we are able to process query images. In order to detect the boundaries in a query image, we consider every stripe as a boundary candidate, calculate the descriptor for that stripe and feed it to the classifier. In our case we are not only interested in the label decided by the classifier but also the assuredness of classifier on this label. The classifiers we use, decide on a class of a query vector by comparing a score to a threshold (usually 0). For instance, the quadratic classifier calculates the Mahalanobis distances to the means of positive and negative training feature vectors in the descriptor space. Decision is made by comparing the differences of distances to zero. Thus we can use the difference between the distances

as a assuredness measure. For SVM the score is the signed distance to the separating hyper plane in the augmented feature space. And for AdaBoost it is the weighted average of the decisions of weak classifiers.

The score signal for a query image is calculated with the below steps.

(i) A sliding window passes through the query image.
(ii) For every window, descriptor is calculated.
(iii) The descriptor is fed to the descriptor and a score value is obtained.
(iv) The scores are recorded and a score signal is obtained.

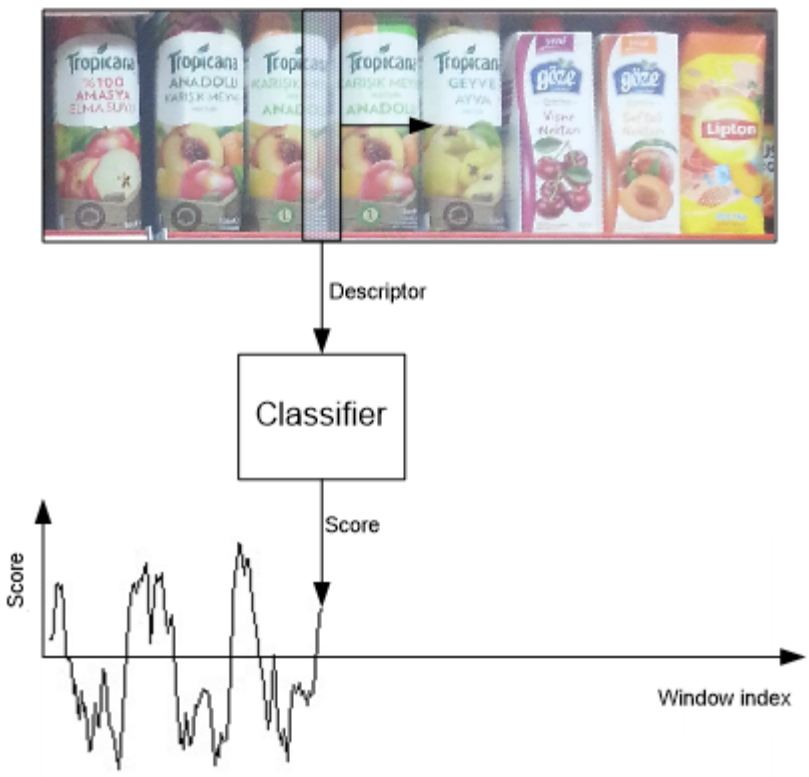


Figure 4.37. Description of stripes and score calculation

4.6.1. Probability Estimation from Classifier Output Scores

It is safe to assume that there is a correlation between the scores and the probability of including a boundary. However we need to observe the relationship and model it properly. For this purpose, the distribution of the scores for the test descriptors are plotted. We divide the score interval into 100 bins and for every bin, we count the number of positive and negative test descriptors scoring in that interval. Thus we obtain the score histograms of positive and negative test descriptors' score distribution.

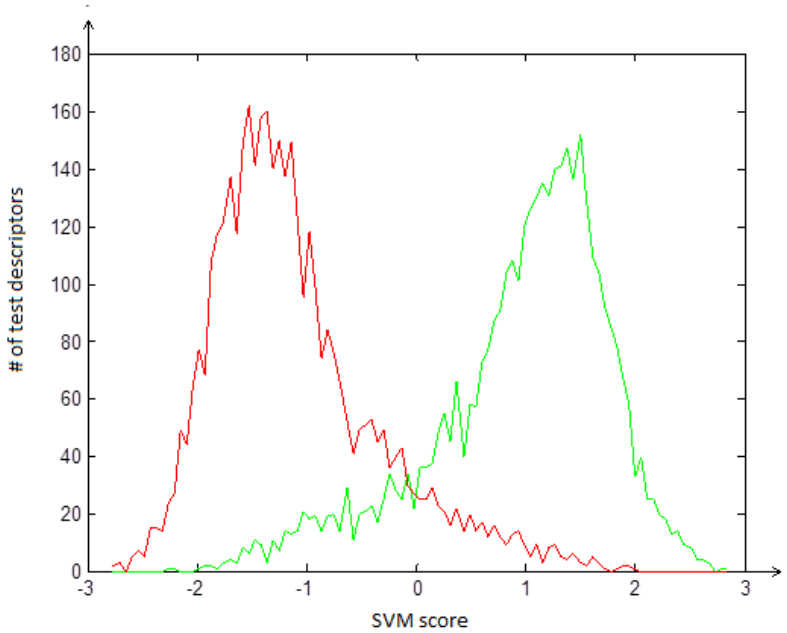


Figure 4.38. Score histograms for test descriptors (Green is positive, red is negative)

We can use the ratio of positive ones among the test descriptors scoring in an interval, as a probability approximation for that score interval. For this purpose score vs. positive test sample ratio is observed.

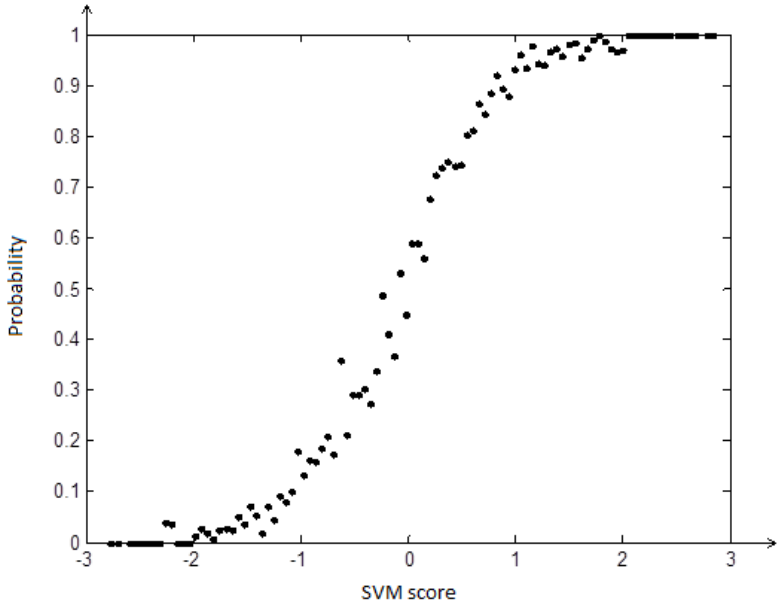


Figure 4.39. Posterior probability of being a positive test sample for score bins of width 0.06

Observing the figure, we see that we can use regression to obtain the function that maps the scores to probabilities. In the paper "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods", a method is presented to calculate the class probabilities from SVM outputs [48]. The method is not specific to SVM and can be adapted to any classifier which outputs a score. It is basically an iterative method to fit a sigmoid using the SVM decision values and sample labels. The output of the algorithm is the sigmoid parameters A and B where f is the SVM output score.

$$p(y = 1|f) = \frac{1}{1 + e^{Af+B}} \tag{4.16}$$

We collect positive and negative descriptors from a different image set and obtain classifier scores for them. When we implement the algorithm with the descriptor labels and classifier output scores we obtain the sigmoid shown in Figure 4.40. It is drawn in green over the posterior probabilities.

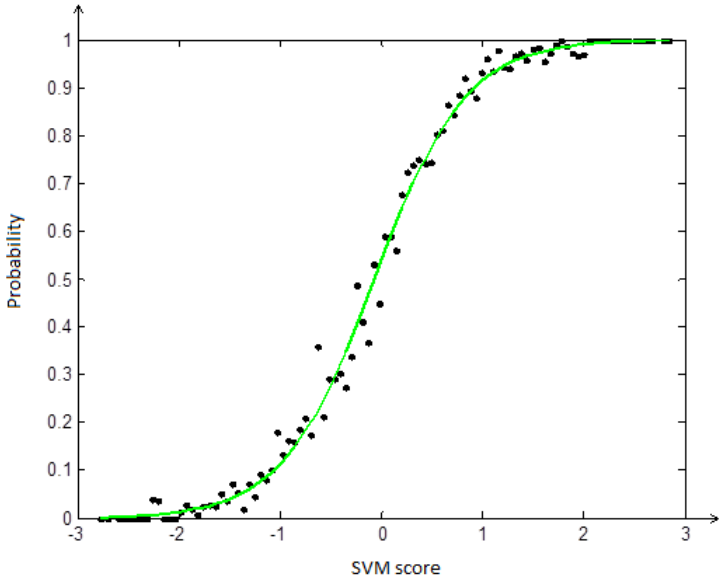


Figure 4.40. Best fit sigmoid over the posterior probabilities

The sigmoid maps the scores in the interval [-3,3] to the probabilities [0,1]. Using the mapping we convert the score signal to a probability signal where we can work more robustly.

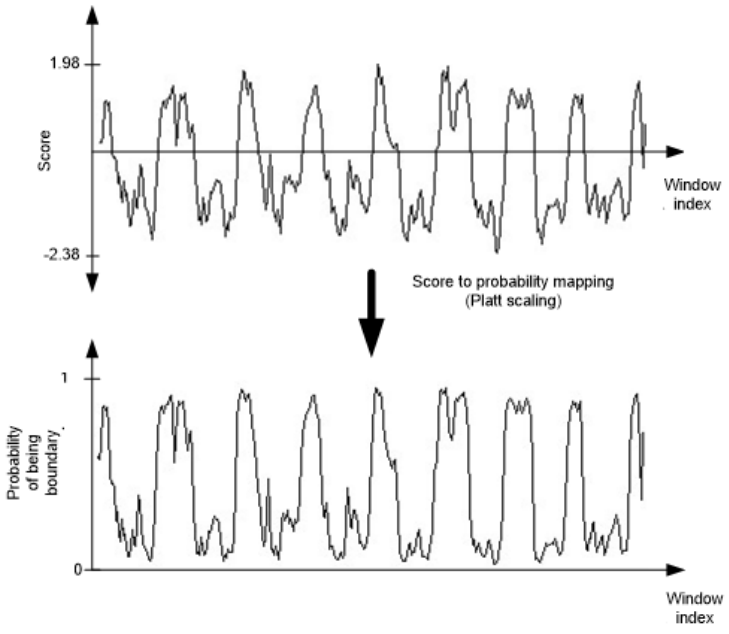


Figure 4.41. Score signal and corresponding probability signal

After the conversion we have a more meaningful signal. Different classifiers output different score scales, so comparing them is hard. For instance SVM scores are usually between -3 and 3. However, QC output scores are at a much smaller scale, they are around 0.001's. Comparing probability signals are more convenient.

4.6.2. Non Maximum Suppression and Peak Detection

Comparing the probability values with 0.5, we obtain the classifier outputs {+1,-1}. This way, we can obtain the classifier output signals. The corresponding classifier output signal for the probability signal in Figure 4.41 is given in Figure 4.42.

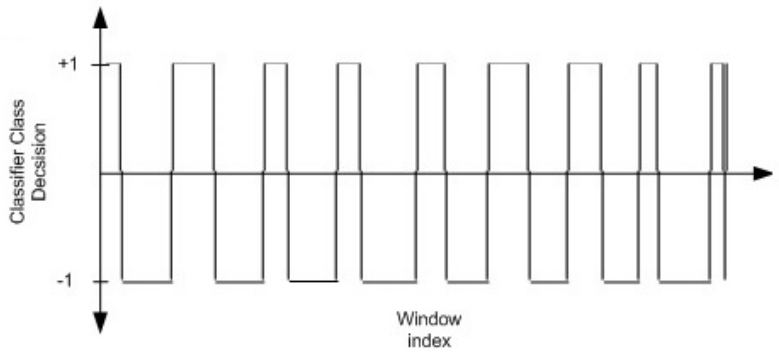


Figure 4.42. Classifier decision for different windows

Since boundaries should be an object apart, we need to select only a single position as boundary within islands of +1 responses. We will detect the classifier score peaks and declare the center column of the corresponding stripes as boundaries. We make sure we are addressing the below cases:

(i) We need to suppress close peaks in order not to detect multiple boundaries between two products.
(ii) We need a peak to occur close to the middle column between two products.
(iii) The peaks with low probability of being a boundary need to be ignored.

To achieve these goals, we implement these steps.

(i) Smoothing the probability signal:

This enables the very close multiple peaks to merge in the middle. By visual inspection, we decide on a smoothing box filter of width 15.

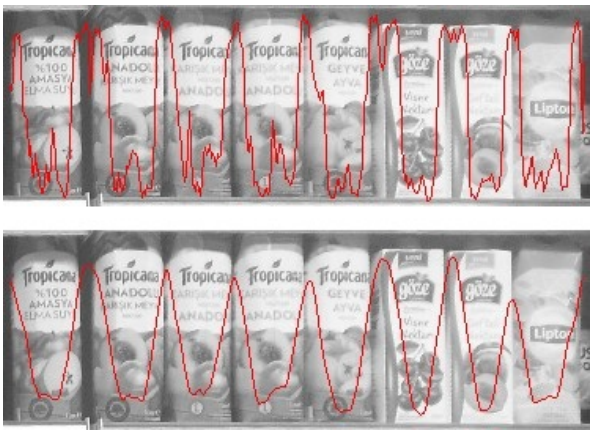


Figure 4.43. Effect of smoothing for a query image

(ii) Peak detection:

We detect the peaks which are the maximums in a window. That is, a stripe index is decided to be a peak if the probability concurring at that peak is greater than the neighbor probabilities in a window. We also check if the peak is more likely to be a boundary by comparing the probability to 50%.

$$\text{x is peak if } p(x)>p(x+a),\ \forall\ a \in \mathbb{N} \text{ s.t.} |a| \leq a_n,\ a \neq 0 \text{ and } p(x)>0.5$$

We need to decide on a window width a_n. For this purpose we observe the data and record the widths between annotations (after resizing to fit the image height of 128). We see that usually the annotations are more than 25 pixels away from their neighbor annotations. Thus we use a window width of 16 as a starting point.

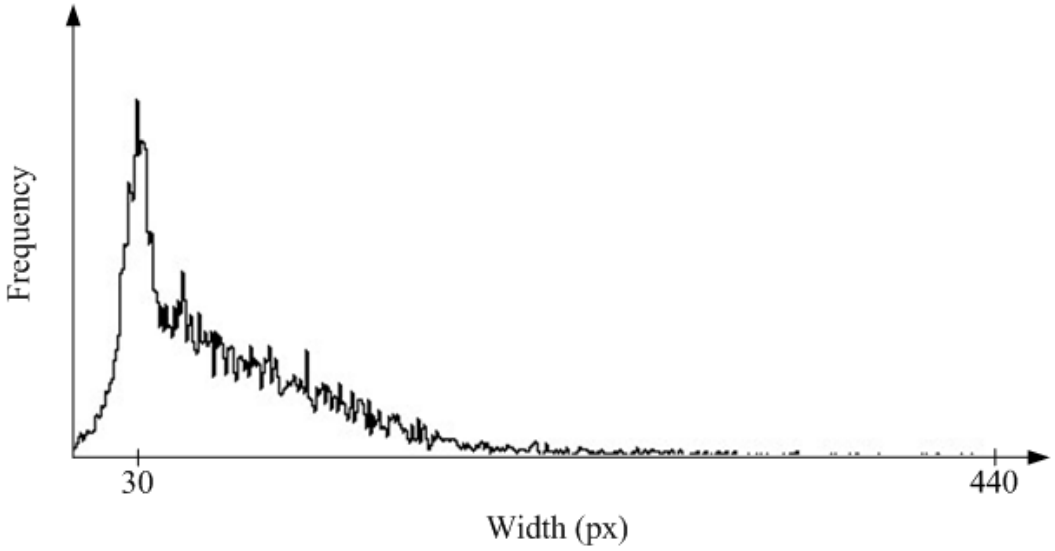


Figure 4.44. Distribution of annotation widths



Figure 4.45. Some annotation examples (drawn in red) closer than 16 pixels

We see that the it is fine to just detect one of the boundaries which are closer than 16 pixels for most of the cases. However we need to keep that in mind during the boundary detection performance evaluation part. Approximately 2.82% of boundary annotation widths are closer than 16 pixels. When we implement non maximum

suppression with windows of 16 pixel widths, it is impossible for us to detect both two neighbor boundaries closer than 16 pixels. We can only detect the one with higher probability score. However, if we use a smaller window width, say 5, this time, we may need to deal with multiple peaks occurring at the same product boundary. We try different non maximum suppression window widths w and measure performance.

Another parameter we may need to calibrate is the threshold with which we compare the probability before declaring a peak. As a starting point 50% is used. This is logical since a probability below 50% indicates that the stripe is probably from a non-boundary location. However experimenting with different thresholds may help us achieve higher precision for the price of lower recall.

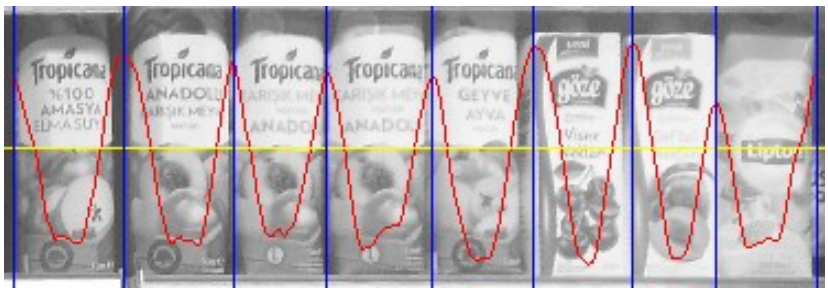


Figure 4.46. Smoothed probability image with detected peak indices (drawn in blue). Yellow line corresponds to 50% threshold.

4.6.3. Detection Performance

It is impossible to expect the machine detected boundaries to be at the pixel perfect annotated positions. We accept a detected peak as a true positive if the closest real boundary is closer than 8 pixels. The non maximum suppression window size 16 prevents us to detect 2 boundaries for a given real boundary so a real boundary can confirm at most one detected peak to be a true positive. For a given query image we define precision and recall as;

Precision=(Number of True Positive)/(Number of detected boundaries)

Recall=(Number of True Positive)/(Number of real boundaries)

We test our algorithm with 500 query images. The number of ground truth boundaries is 4295. 4323 boundaries are detected and 3288 of them are true positive. Thus total precision is 76.06% and total recall is 76.55%. Precision and recall are quite close thus we achieved equal error rate at this parameters. We change the probability threshold to observe the change of precision and recall. We see that 0.5 is a proper threshold. Precision and recall for different threshold values are given in Figure 4.47. The corresponding Precision vs. Recall curve is given in Figure 4.48.

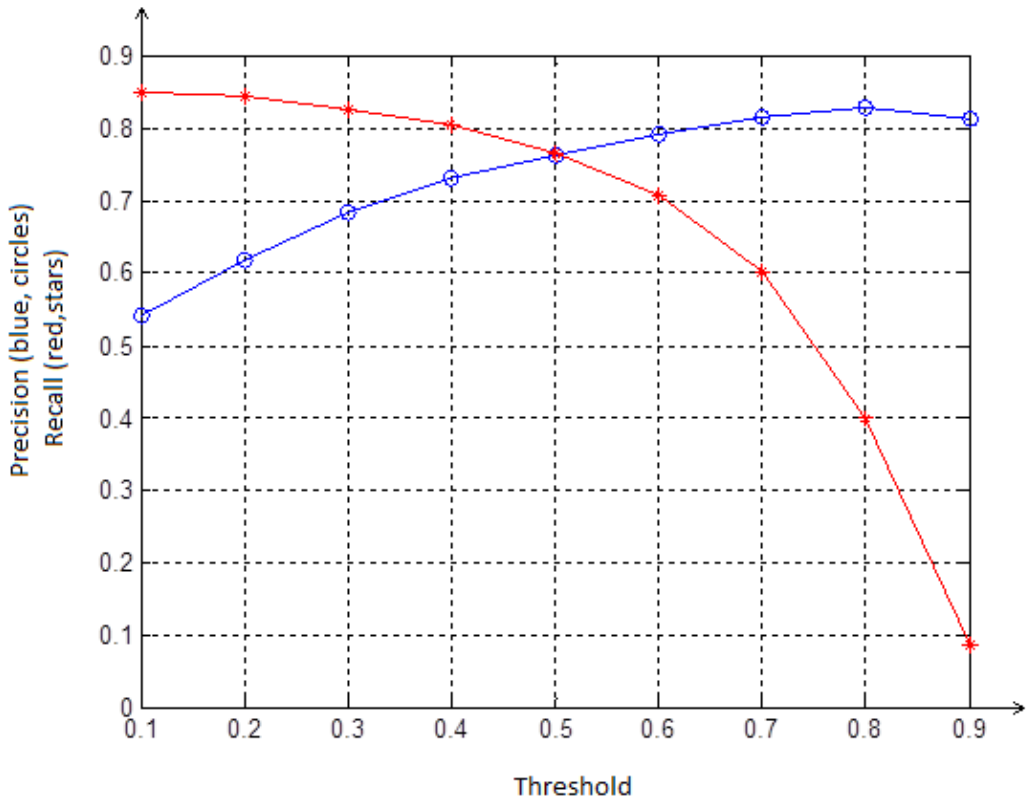


Figure 4.47. Precision (blue), and Recall (red) vs. Probability threshold

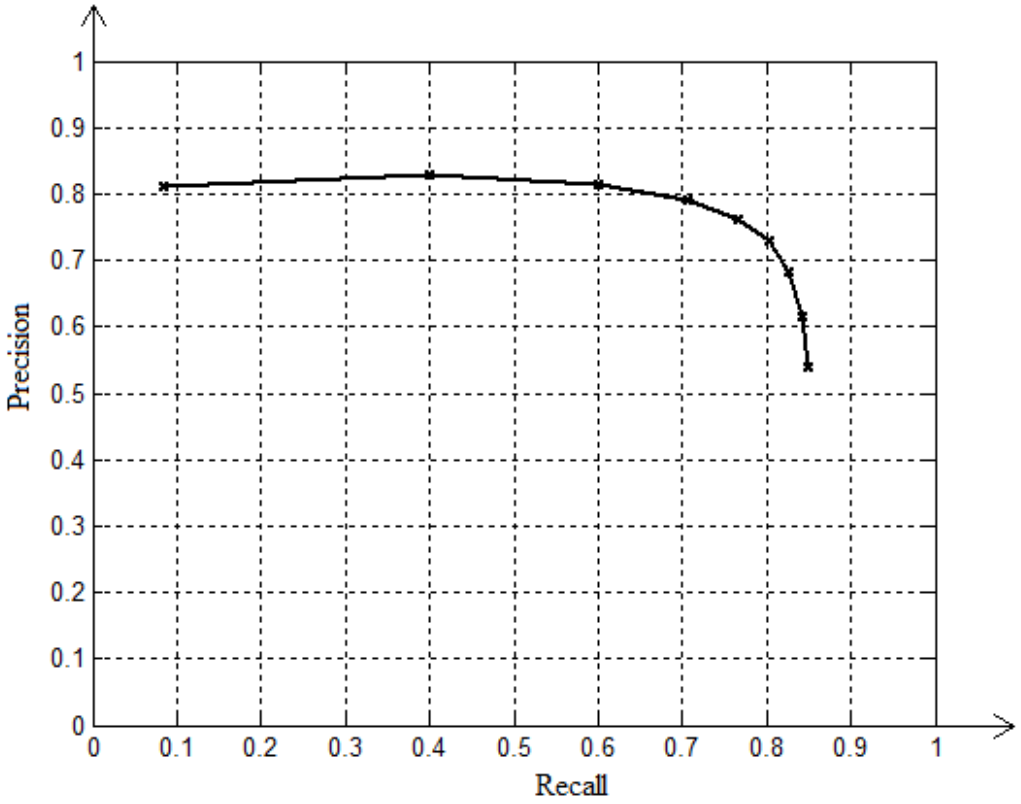


Figure 4.48. Precision vs. Recall curve at varying thresholds

Some sample cases are given below. More of them can be found in in Appendix B. Upper images: Probability signal, Middle images: Smoothed probability signal (red), detected peak indices (blue), threshold (yellow), Lower images: Detected true peak indices (green), Detected false peak indices (red), and Ground truth annotations (blue). Figure 4.49, 4.50, and 4.51 are some of the examples where our algorithm achieved perfect detection performance.



Figure 4.49. A sample shelf image (Soft drinks)



Figure 4.50. A sample shelf image (Personal care)



Figure 4.51. A sample shelf image (Confectionery)

In Figure 4.52 we observe 5 false positives. 2 of them are very close to image boundaries, 1 of them has a very close ground truth annotation however the distance is still more than our threshold and it is evaluated as a false positive.



Figure 4.52. A sample shelf image (Confectionery)

In Figure 4.53 we see that, due to smoothing, we lost the rightmost boundary. And at the rightmost bottle, a vertical shining created a false positive.

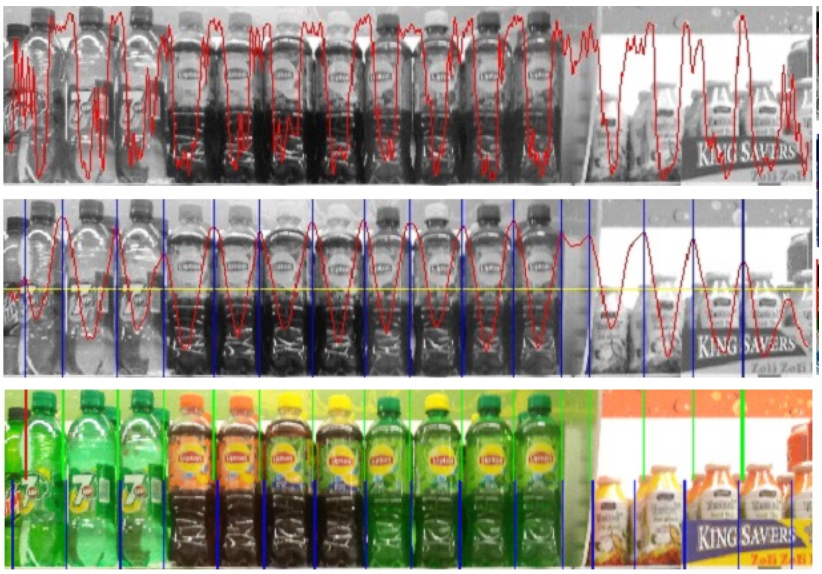


Figure 4.53. A sample shelf image (Soft drinks)

In Figure 4.54 we see that a non-product occlusion created a false boundary. Also we failed at detecting 3 product boundaries at the right side of the shelf.



Figure 4.54. A sample shelf image (Confectionery)

Sometimes vertical shadows are detected as false positives. We see examples of this case in Figure 4.55.



Figure 4.55. A sample shelf image (Confectionery)

The products with vertical line features also create false boundary detections. We see an example in Figure 4.56 where the strong horizontal intensity change on the toothpaste package creates a boundary like pattern.

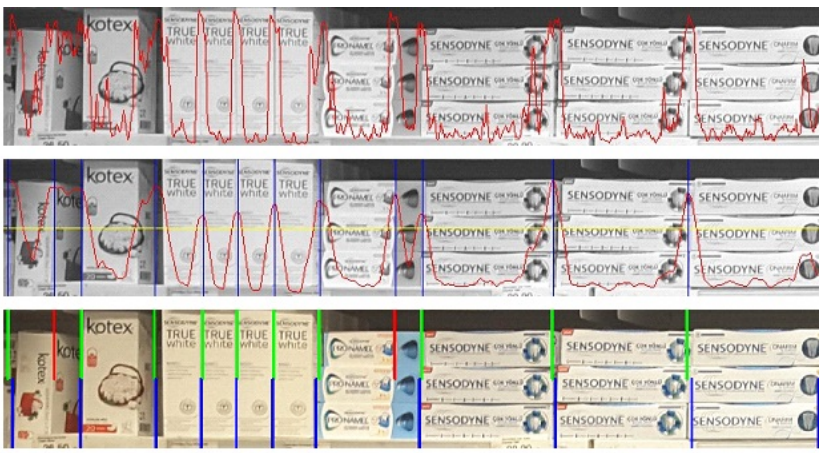


Figure 4.56. A sample shelf image (Personal care)

In some cases, the fences holding the products are detected as false positives. We see an example in Figure 4.57.

The developed boundary detection algorithm can be used in a system where users are tasked to mark the image boundaries by hand. Our algorithm can facilitate this

process by presenting the shelf image with the automatically detected boundaries and a user is only expected to correct the problematic cases.

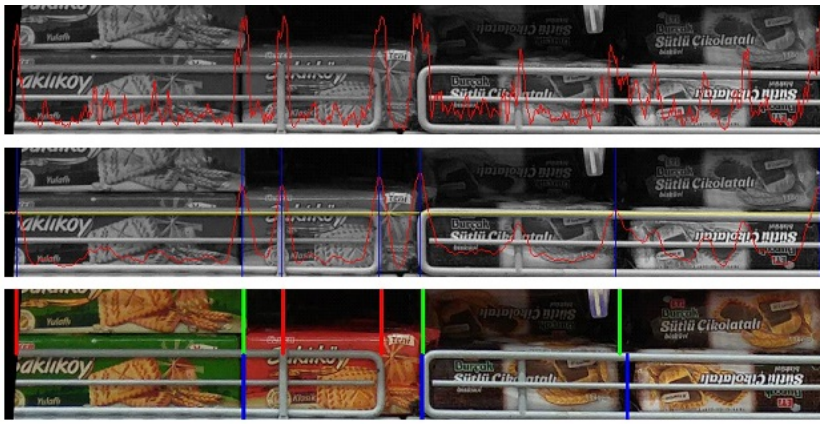


Figure 4.57. A sample shelf image (Confectionery)

4.7. Comparison to Benchmark Method

We compared the proposed boundary detection method to a benchmark method. In the benchmark method, for every column, we calculate the ratio of horizontally oriented gradients (corresponding to vertical lines). In this context, we define a gradient vector is called a horizontally oriented gradient if the gradient vector is at least 5 degrees away to horizontal line. The subspace for horizontal gradients is given in Figure 4.58.

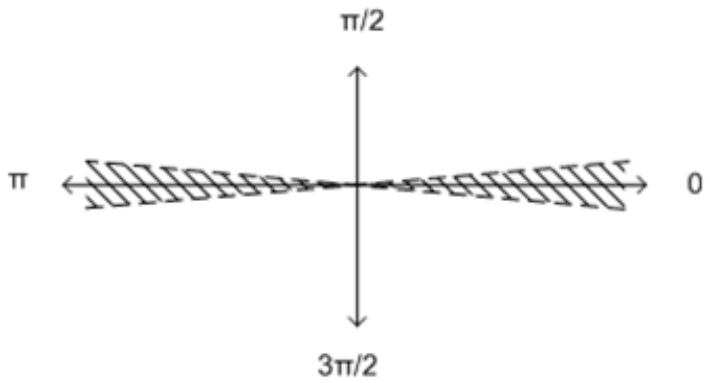


Figure 4.58. Horizontal gradients

Later, similar to our method, we produce a signal, smooth it, implement a threshold to it and perform peak detection. The smoothing and threshold parameters are optimized so that we achieve highest possible precision and recall at equal error rate. Precision is 69.28% and recall is 68.32% for the optimal case. It is approximately 7% than the performance of our method. There is no machine learning approach implemented in the benchmark method. Some figures are given below for comparison. The upper signals are produced by our proposed method and the below signals are produced by the benchmark method. The signals produced with the benchmark method is amplified (multiplied by 3) for better visualization.

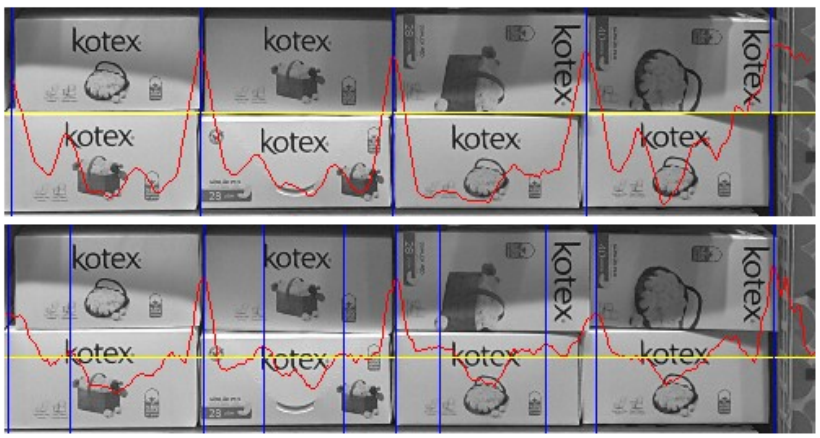


Figure 4.59. Benchmark comparison example (Personal hygiene products)

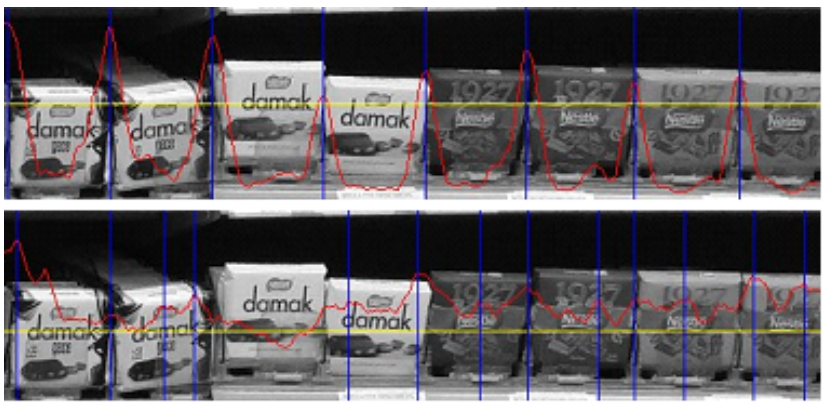


Figure 4.60. Benchmark comparison example (Chocolate)

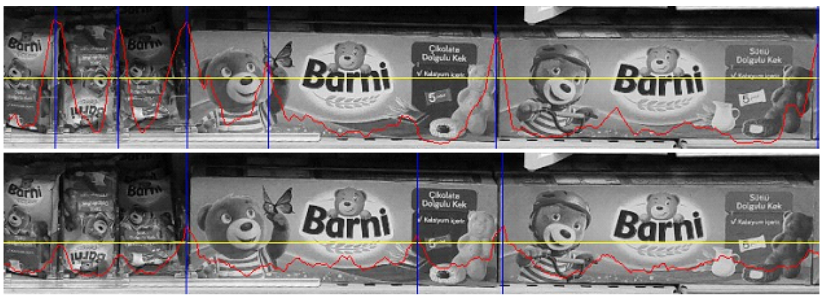


Figure 4.61. Benchmark comparison example (Confectionery)

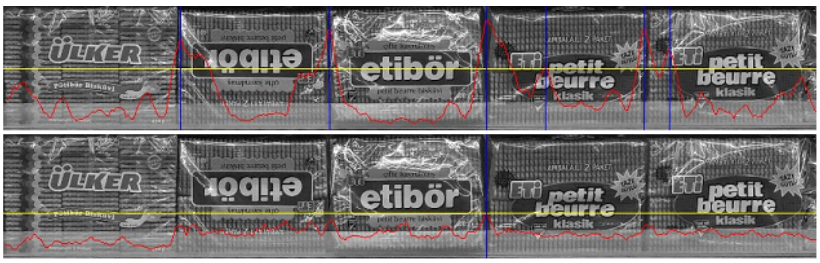


Figure 4.62. Benchmark comparison example (Biscuit)

4.8. Shelf Boundary Detection

The boundary detection pipeline is also implemented to detect shelf boundaries for shelf parsing. To be able to work on the same bench, we transposed the retail scene images, so that the shelf boundaries are vertical lines similar to product boundaries.

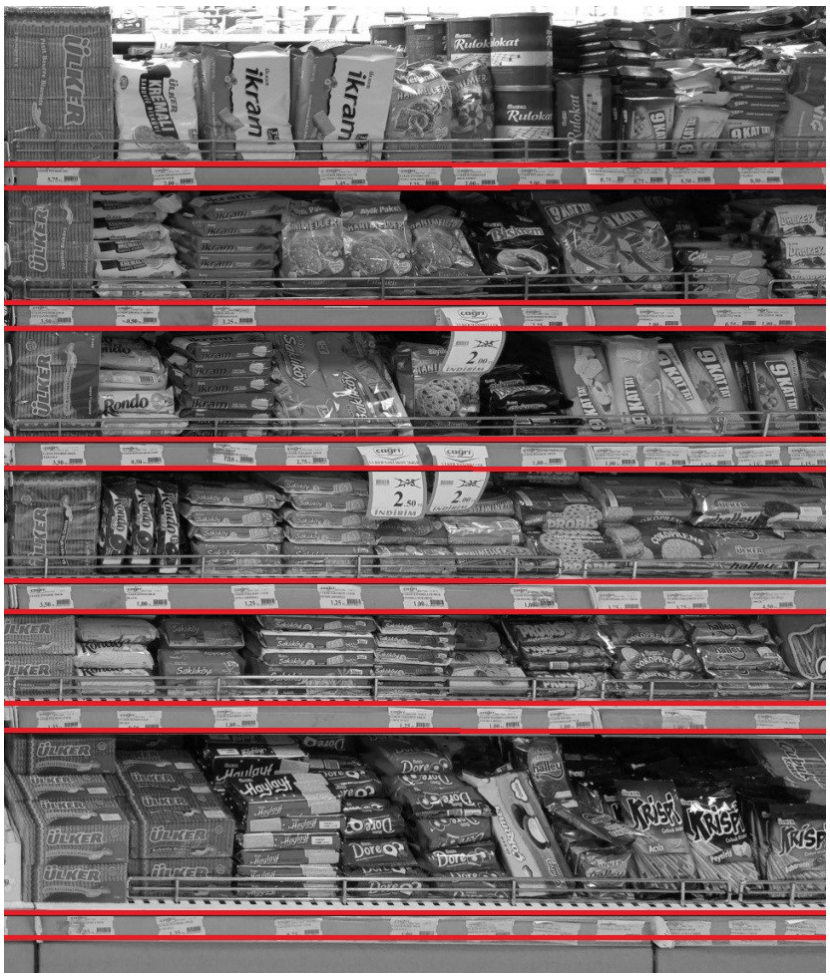


Figure 4.63. A sample retail scene and corresponding shelf boundary annotations (red lines)

After the transposition, similar to the boundary detection approach, the below steps are implemented to train an SVM,

(i) Images are resized to fit the same height (preserving aspect ratio),
(ii) HOG descriptors are extracted for all stripes from training images,
(iii) Positive and negative descriptors are sampled with their labels, using the annotations,
(iv) 5 fold validation is implemented to find the optimal SVM parameters.
(v) An SVM is trained with the optimal parameters.

After the SVM is trained, using another set of 500 images, we calculated the sigmoid parameters which we use to convert score values to probability values.

With the SVM classifier and the sigmoid parameters, we tested 500 images and reported the performance. Note that for one shelf, we have 2 annotations; one for the upper side of the shelf base and one for the bottom side of the shelf base. However since our aim is to parse the shelves, detecting only one of them is sufficient, even better. Thus we use a broader signal smoothing and non maximum peak suppression to avoid detecting both shelf lines. The smoothing filter size is increased to 20 from 15. NMS window size is increased to 24 from 16.

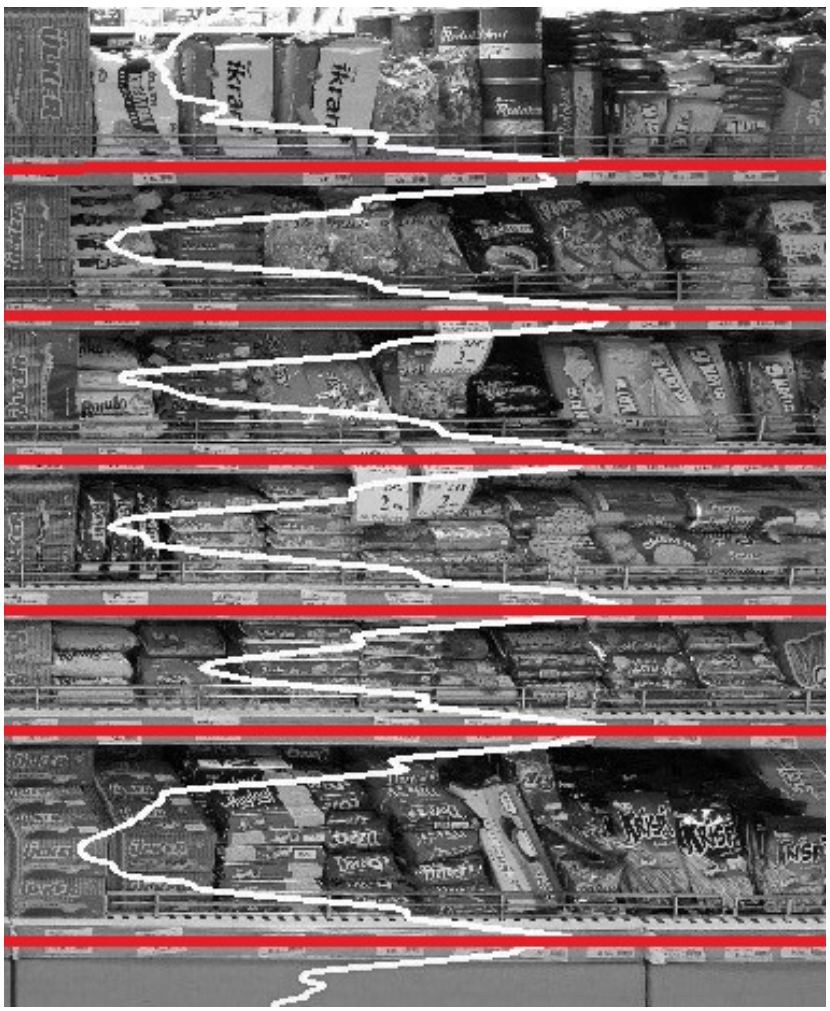


Figure 4.64. Smoothed probability signal (in white), detected peaks (in red) drawn over a sample query image

Since our aim is detecting 1 shelf boundary for 2 annotations, we are only interested in the precision performance. The precision is calculated as 96.87%. This is the ratio of real shelf boundaries among the detected shelf boundaries. We are not interested in recall for this case. Recall is 50.94% since we only detect one shelf boundary per two ground truth annotations. The shelf line detection algorithm facilitated the task of shelf parsing. In a system where users were marking shelf separating lines, users can now be supplied with the detected shelf lines and their task can be reduced to fixing the false detections and missing shelf lines. Comparing to product boundary detection, we achieved higher precision in shelf line detection. This is because shelf boundaries exhibit less variance. Product boundaries vary in a wider range and describing them is harder.

5. VISUAL PRODUCT RECOGNITION

After we obtain the shelf and product boundaries, we are able to parse the products. Since the boundary detection is not 100% accurate, we still need human interaction. The detected lines are corrected by a human user and using the corrected shelf and product boundaries we segment the rectangles that include separate products.

Our recognition paradigm uses a supervised learning phase again, where we learn the of visual feature statistics of distinct product classes and develop a recognition algorithm for query product images. We used several methods to achieve better recognition performance.

In the first method, we used a descriptor to descriptor matching phase. Later we used a bag of words approach which enabled us to reach shorter query times. Then we introduced a geometric verification step to achieve higher recognition accuracy. The methods and their performances are explained in the next subchapters. Note that in this chapter we focused our attention to recognizing products from soft drinks category.

5.1. Datasets

We worked on 2 datasets of soft drinks products. The details are given below.

5.1.1. Dataset 1

There are 10 different product classes in this dataset. This dataset is randomly divided into test and training sets. A training set consists of approximately 65 training images per class and a test set consists of 30 test images per class. Sample test images are shown in Figure 5.1.



Figure 5.1. Sample images from Dataset 1 (in order)

5.1.2. Dataset 2

There are 20 different product classes in this dataset. This dataset is randomly divided into test and training sets as well. A training set consists of approximately 123 training images per class and a test set consists of 67 test images per class. Sample test images are shown in Figure 5.2.



Figure 5.2. Sample images from Dataset 20 (in order)

5.2. Descriptor Matching with Geometric Verification

This method uses a descriptor to descriptor nearest neighbor based matching step. Later the matches are accepted or declined using spatial correspondence between neighbor descriptor and their matches. The method can be summed up in below steps.

(i) Using previously labeled training images, all training descriptors are found and tagged as the label of their mother image. For example a descriptor from a training image from Class 2 is tagged as 2. The descriptors are SIFT descriptors [6].

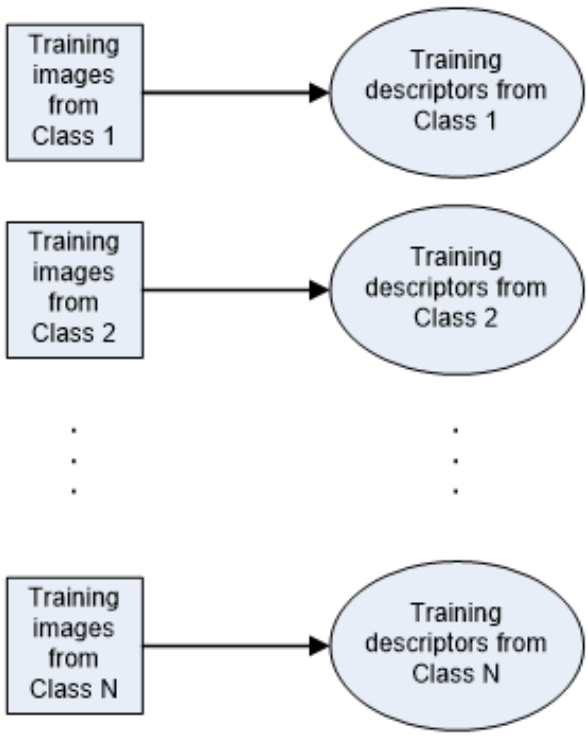


Figure 5.3. Training phase

(ii) Given a query image, its SIFT descriptors are extracted.

(iii) All test descriptors are matched to a training descriptor and tagged as the matching training descriptor's label. The matching is done by Euclidian distance minimization. That is, every test descriptor is matched to the closest training descriptor in the 128 dimensional SIFT descriptor space.

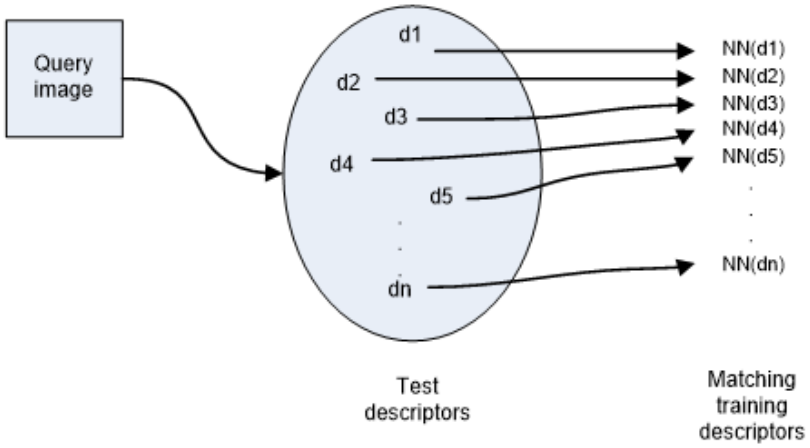


Figure 5.4. Matching test and training descriptors

A test descriptor is tagged as its matching training descriptor's tag.

$$\forall f_n; Tag(d_n) = Tag(NN(d_n)) \tag{5.1}$$

(iv) All test descriptor tags are either accepted or declined. To accept the tag of a test descriptor,

- The neighbor test descriptors (descriptors in the test image with spatial distance less than ϵ_n) which also have the same tag as the test descriptor are found. The set of neighbor same tag descriptors of a test descriptor d_n is

$$A = \{d | \forall d; Tag(d) = Tag(d_n), dist(d, d_n) < \epsilon_n\} \tag{5.2}$$

  where ϵ_n is the neighborhood distance parameter. The best affine (SSE minimizing) mapping which transforms these neighbor test descriptor to their matching training descriptors is found.
- Resulting sum squared error is calculated.
- If the sum squared error is below threshold ϵ_{sse}, the tag of the given descriptor is accepted. The image is decided to be in the class which has the highest number of accepted test descriptors.

5.2.1. Parameter Selection

There are 2 parameters in the algorithm. The neighborhood threshold (ϵ_n) and sum squared error (SSE) threshold (ϵ_{sse}). The neighborhood parameter is chosen to be 20 initially. That is, a test descriptor is neighbor to another test descriptor if the L2 distance between their locations is less than 20 pixels. In order to choose the other parameter ϵ_{sse}, the sum squared errors of best affine mappings per test descriptor neighborhood is calculated and observed. For instance, a test image from class 2 produced 335 SIFT descriptors. The SSE for every test descriptor neighborhood is graphed in Figure 5.5. Another test image from class 6 produced a SSE graph shown in Figure 5.6. Then, observing the graphs, the SSE threshold after which the SSE values grow rapidly is observed to be around 4000.

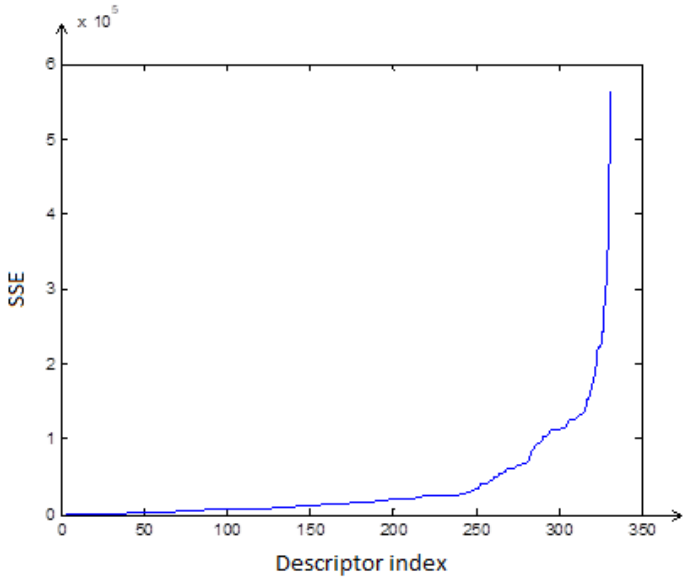


Figure 5.5. SSE graph for a test image from 2. class.

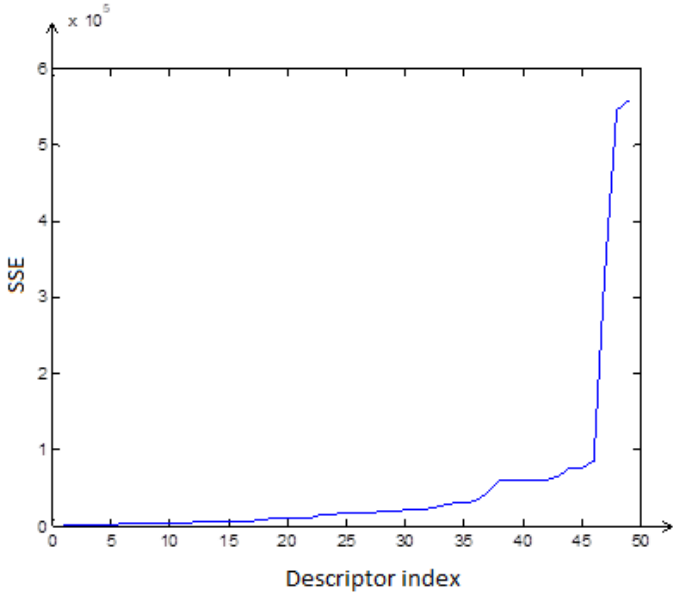


Figure 5.6. SSE graph for a test image from 6. class.

Observing the graphs, the optimal threshold is chosen to be 40000.

5.2.2. Experiments

The proposed method is implemented on both datasets.

5.2.2.1. Experiment on Dataset 1. This experiment is implemented on Dataset 1. From all training images (647 images), all training features are found using SIFT. At this step vl_sift is used from vl_feat library [49]. A total of 403112 training SIFT features are found. The confusion matrix is given in Table 5.1. Overall accuracy is 95.33%.

5.2.2.2. Experiment on Dataset 2. The method in the previous experiment is implemented on the Dataset 2. Compilation times are also recorded for each test image.

From all training images (2692 images), all training features are found using SIFT. At this step vl_sift function is used from vl_feat library [49]. This function runs a SIFT detector on a given image with the method explained in the original paper

by Lowe [6]. A total of 2192535 training SIFT features are found. There are 1346 test images overall. The confusion matrix and the true positive rates are given below. Overall accuracy is 93.68%.

Table 5.1. Confusion matrix for Experiment on Dataset 1.

Class name	Image number	Number of test images classified as											Accuracy (%)
		1	2	3	4	5	6	7	8	9	10	Not classified	
1	30	29	0	0	0	0	0	0	0	1	0	0	96.67
2	30	0	30	0	0	0	0	0	0	0	0	0	100
3	30	0	0	29	0	0	0	0	0	1	0	0	96.67
4	30	0	0	0	28	1	0	0	0	0	0	1	93.33
5	30	0	0	0	1	29	0	0	0	0	0	0	96.67
6	30	0	0	0	0	0	26	0	0	1	0	3	86.67
7	30	0	0	0	0	0	0	30	0	0	0	0	100
8	30	0	0	0	0	0	0	0	27	0	1	2	90
9	30	0	0	0	0	0	0	0	0	30	0	0	100
10	30	0	0	0	0	0	0	0	0	2	28	0	93.33
													95.33

Table 5.2. Confusion matrix for Experiment on Dataset 2.

Class name	Image number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Not classified	Accuracy
1	65	55	8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	84.62%
2	65	5	56	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0	0	0	0	86.15%
3	66	0	0	61	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	92.42%
4	66	0	0	0	57	0	0	1	0	0	5	0	0	0	0	0	0	0	0	0	0	3	86.36%
5	62	0	0	0	0	55	0	0	0	0	3	2	0	0	0	2	0	0	0	0	0	0	88.71%
6	66	0	0	0	0	0	59	0	0	0	5	0	0	0	0	0	0	0	0	0	0	2	89.39%
7	65	1	0	0	5	0	0	53	0	0	3	0	0	0	0	0	0	0	0	0	0	3	81.54%
8	70	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
9	68	0	1	0	0	0	0	0	0	65	0	0	0	0	0	0	0	0	0	0	0	2	95.59%
10	67	0	0	0	0	0	0	1	0	0	65	0	0	0	0	1	0	0	0	0	0	0	97.01%
11	72	0	0	0	1	0	0	0	0	0	0	67	0	0	0	1	0	0	0	0	0	3	93.06%
12	68	0	0	0	0	0	0	0	0	0	0	0	68	0	0	0	0	0	0	0	0	0	100.00%
13	69	0	0	0	0	0	0	0	0	0	2	0	0	66	0	0	1	0	0	0	0	0	95.65%
14	69	0	0	0	0	0	0	0	0	0	1	0	0	0	63	0	0	0	0	0	0	5	91.30%
15	65	0	0	0	0	0	0	0	0	0	2	0	0	0	0	61	1	0	0	0	1	0	93.85%
16	71	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	71	0	0	0	0	0	100.00%
17	69	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	67	0	0	0	1	97.10%
18	68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	68	0	0	0	100.00%
19	68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	67	1	0	98.53%
20	67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	67	0	100.00%
																							93.68%

Table 5.3. Average number of SIFT descriptors and test times for Experiment on Dataset 2.

Class name	Average number of SIFT descriptors per image	Average test time per image (sec)
1	679	61.5
2	884	80.2
3	550	43.2
4	1519	133.8
5	483	52.3
6	1609	130.9
7	779	90.5
8	844	57.4
9	447	29.1
10	2416	207.6
11	472	54.1
12	688	57.4
13	1318	108.5
14	216	17.3
15	1122	96.9
16	700	67.6
17	205	19.0
18	396	37.1
19	376	34.5
20	1107	130.2

Matching the test features to the training features takes most of the algorithm time. Average test time is 74.7 seconds and average matching time is 71.7 seconds. Approximately 97% of the test time is spent on the matching step. The test time

increases with the number of detected SIFT descriptors per test image.

5.2.3. Discussion

The initial accuracy rates seem encouraging. However average test time is too high. The matching step takes the most of the time. Complexity of brute-force match time per test image is O(NM) where N is the number of test features and M is the number of training (database) features. Thus, instead of matching to the full training (database) feature, matching into a set of learned prototypes (words) is assumed to reduce the complexity of brute force matching. Next method makes use of this idea of using a bag of words approach [50].

5.3. Image Retrieval Using Visual Words

In this method, we extracted SIFT descriptors from training images with a SIFT detector and we constructed a visual vocabulary by clustering the training SIFT descriptors into centers. For every image, we detect the SIFT features and we constructed tf-idf vectors to represent the image. A query is labeled as the inter-descriptor distance minimizing training image label. Step by step description of the method can be found in Section 3.2.

5.3.1. Experiments

5.3.1.1. Experiment on Dataset 1. This method is implemented on Dataset 1. From a total of 647 training images, 403112 training SIFT descriptors are obtained using vl_sift function with default parameters. The variants in this experiment differ in three steps; [49]

(i) Choosing the number of clusters K. 3 values are selected for this step; 5000, 2000, and 500. It takes 65, 35 and 8 minutes consecutively to obtain the centers and membership information using vl_kmeans on 403112 training features [49].

(ii) Choosing the parameter T_{stop} which determines the threshold which centers with higher number of members are discarded.

- When the training features are clustered in 500 centers, the most popular center has 2729 members. T_{stop} parameters are chosen to be 1250, 1500, 2500, 3000. Out of 500 clusters they put 37, 12, 3, and 0 centers in the set V_{stop} consecutively. Number of elements and chosen T_{stop} parameters are shown in Figure 5.7.
- When the training features are clustered in 2000 centers, the most popular center has 1216 members. T_{stop} parameters are chosen to be 180, 400, 800, 1500. Out of 2000 clusters they put 991, 18, 11, and 0 centers in the set V_{stop} consecutively. Number of elements and chosen T_{stop} parameters are shown in Figure 5.8.
- When the training features are clustered in 5000 centers, the most popular center has 673 members. T_{stop} parameters are chosen to be 150,200,350,700. These parameters determine the number of elements in V_{stop}. Out of 5000 clusters they put 336, 136, 24, and 0 centers in the set V_{stop} consecutively. Number of elements and chosen T_{stop} parameters are shown in Figure 5.9.

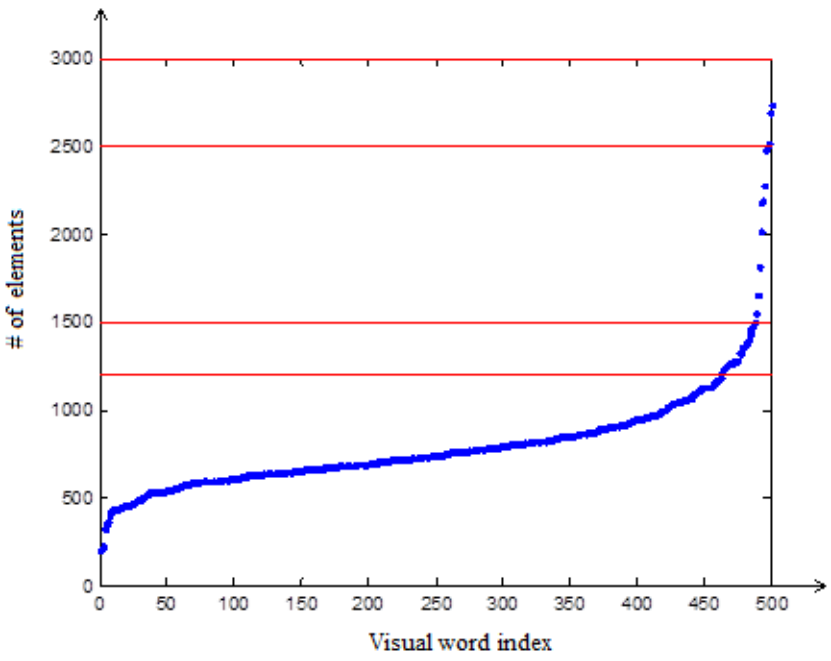


Figure 5.7. Number of elements for 500 centers

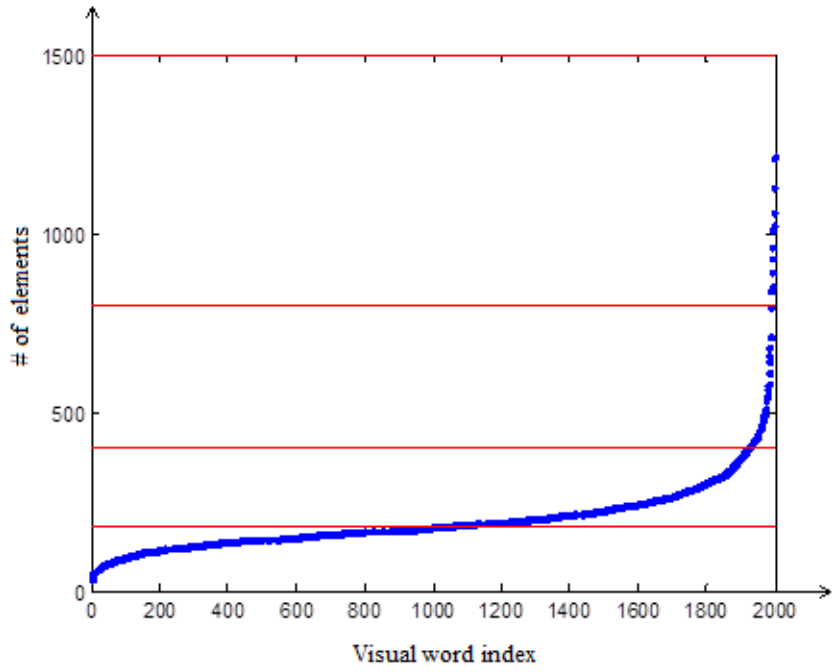


Figure 5.8. Number of elements for 2000 centers

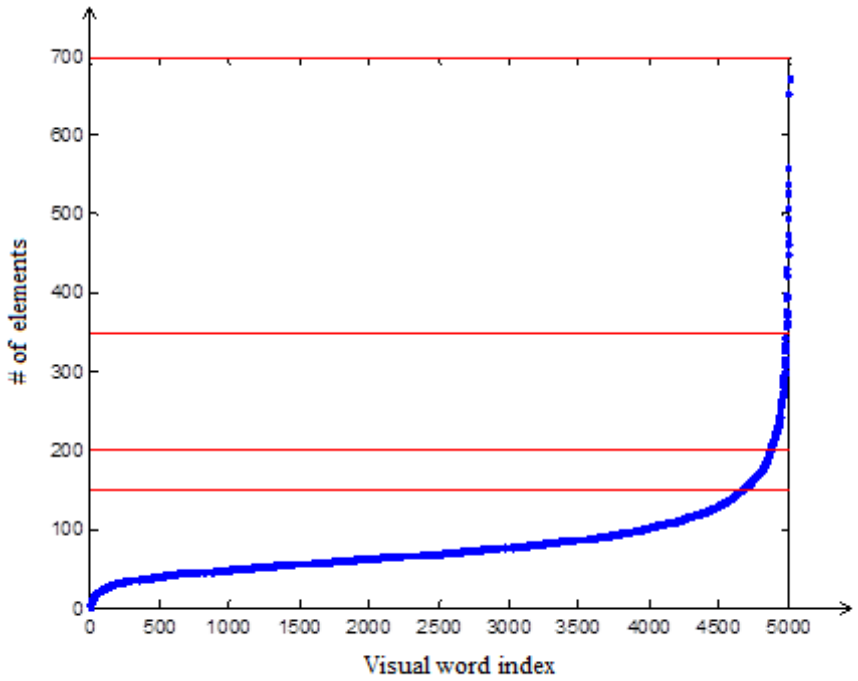


Figure 5.9. Number of elements for 5000 centers

(iii) The method for normalizing the vectors and measuring distance between them. 2 methods are used for this step: L1 and L2 normalizations and distances.

These variations are combined together to 24 different implementations. True positive rates are given in Table 5.4. The values in parenthesis' shows the number of remaining centers after discarding the most popular centers which have higher members than the given T_{stop} parameter i.e. $K - |V_{stop}|$.

Table 5.4. Accuracy (%) for different implementations.

K	500				2000				5000			
T_{stop}	3000	2500	1500	1250	1500	800	400	180	700	350	200	150
# of remaining centers	500	497	488	463	2000	1989	1922	1009	5000	4976	4864	4664
L1 dist.	76.67	76.00	75.67	75.00	88.00	86.00	85.00	85.00	92.67	92.67	93.00	91.33
L2 dist.	72.33	72.00	71.00	72.00	75.33	76.00	74.00	75.00	79.00	79.67	78.67	78.67

The highest overall accuracy 93% is observed when K is 5000, T_{stop} is 200 and L1 distance is used. An overall accuracy rate of 95.33% was obtained with the previous method which uses feature to feature distance instead of image to image distance and uses geometric alignment. The previous method and implementation details are given in the previous reports. Geometric alignment will be used at future work to re-rank a number of close matching training images.

For all K and T_{stop} parameter pairs, using L1 normalization and distance calculation yielded better performance than using L2 normalization and distance calculation. The performance also increases with the initial number of centers.

In order to observe the effect of weighting multiplier of the word frequency vectors and tf-idf vectors, some different multipliers are used with this optimal parameter combination. So far, the histogram vectors are constructed as below;

$$z_k^{(n)} = \frac{m_{kn}}{m_n} \log \frac{N}{N_k} \tag{5.3}$$

The multiplier $\log \frac{N}{N_k}$ is smaller for centers observed at more training images and it is bigger for centers observed at less training images. Thus it enables us to add more weight to the centers observed at less training images. It is a number between 1 (for centers appearing in all training images) and $\log N$ (for centers appearing in only one training image)

Three different weighting functions other than the original one are tried:

- 1 i.e. not using a weighting function
- $\log \frac{N}{N_k}^2$
- $\frac{1}{N_k}$

When the multiplier is omitted, i.e. when the vectors are constructed as $z_k^{(n)} = \frac{m_{kn}}{m_n}$, the overall accuracy drops from 93% to 91.67%. When the multiplier is squared, i.e. when the vectors are constructed as $z_k^{(n)} = \frac{m_{kn}}{m_n}(\log \frac{N}{N_k}^2$, the overall accuracy drops from 93% to 91.33%. When the direct multiplicative reciprocal is used instead of its logarithm, i.e. when the vectors are constructed as $z_k^{(n)} = \frac{m_{kn}}{m_n}\frac{1}{N_k}$, the overall accuracy rate drops from 93% to 91.33%. Note that omitting the number of training images N from the formula does not affect the vectors since they are normalized later.

Looking at the results, it can be seen that none of the alternative weighting functions yield better performance at the specific combination of parameters which we obtained the highest accuracy rate.

Only the parameter K affects the test time for a query image. The other parameters and change of methods does not affect the test time more than 2%. Average test times for different K values are given below.

Table 5.5. Average test times for different number of centers .

K	Average test time (sec)
500	0.22
200	0.31
5000	0.53

When $K = 5000$, $T_{stop} = 200$ and L1 norm is used, 21 test images out of 300 are wrongly labeled. For these images, 10 closest ranked training images are found for later work.

For 5 of these 21 wrongly labeled test images there are no same class training images among their nearest 10 neighbor training images. For 7 of them, the second closest training image has the correct class information. A number of neighbor training images will be used in the further work.

Confusion table for the implementation with the highest overall accuracy rate is given below.

Table 5.6. Confusion matrix for the best implementation.

Class name	Image number	Decided Class										
		1	2	3	4	5	6	7	8	9	10	Accuracy
1	30	27	0	0	0	1	0	0	0	2	0	90.00%
2	30	0	30	0	0	0	0	0	0	0	0	100.00%
3	30	0	0	30	0	0	0	0	0	0	0	100.00%
4	30	0	0	0	26	3	0	0	0	1	0	86.67%
5	30	0	0	0	0	29	0	0	0	1	0	96.67%
6	30	0	0	0	0	1	24	0	0	0	5	80.00%
7	30	0	0	0	0	0	0	30	0	0	0	100.00%
8	30	0	0	0	1	0	0	0	28	0	1	93.33%
9	30	0	0	0	0	1	0	0	0	29	0	96.67%
10	30	0	0	1	2	0	0	0	1	0	26	86.67%
												93.00%

Table 5.7. Ratio (%) of test images with at least one relevant match among top retrieved training images.

Top 1	93.00
Top 2	95.33
Top 3	96.33
Top 4	96.33
Top 5	96.33
Top 6	97.00
Top 7	97.00
Top 8	97.67
Top 9	98.00
Top 10	98.33

5.3.1.2. Experiment on Dataset 2. The previous experiment is also implemented on Dataset 2. Only L1 normalization is used for this dataset thus implemented methods differ in only two steps;

(i) Choosing the number of clusters K. 2 values are selected for this step; 5000, and 500. It takes 484 and 40 minutes consecutively to obtain the centers and membership information using vl_kmeans on 2192535 training features [49].

(ii) Choosing the parameter T_{stop} which determines the threshold which centers with higher number of members are discarded.

   (a) When the training features are clustered in 500 centers, the most popular center has 13896 members. T_{stop} parameters are chosen to be 6000, 8000, and 14000. Out of 500 clusters they put 41, 12, and 0 centers in the set V_{stop} consecutively. Number of elements and chosen T_{stop} parameters are shown in Figure 5.10.

(b) When the training features are clustered in 5000 centers, the most popular center has 2785 members.T_{stop} parameters are chosen to be 1000, 1500, 1500. Out of 5000 clusters they put 106, 25, and 0 centers in the set V_{stop} consecutively. Number of elements and chosen T_{stop} parameters are shown in Figure 5.11.

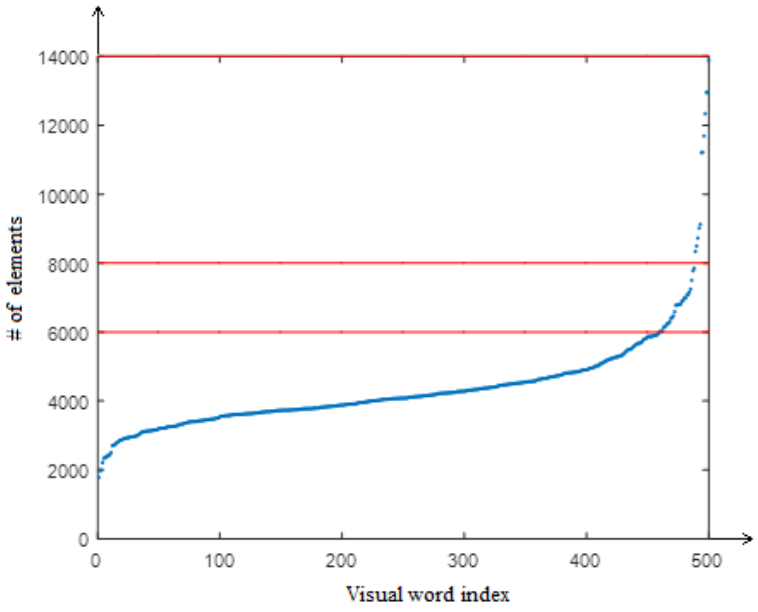


Figure 5.10. Number of elements for 500 clusters

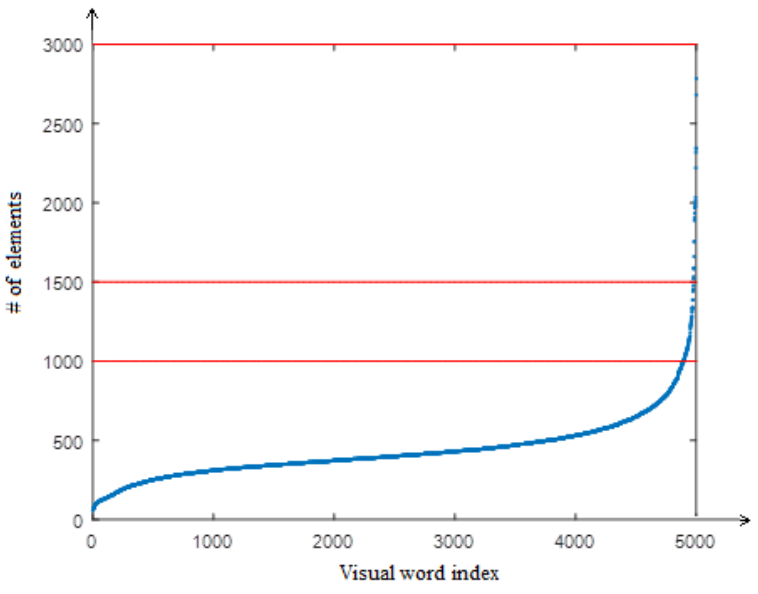


Figure 5.11. Number of elements for 5000 clusters

These variations are combined together to 6 different implementations. Accuracy rates are given in the below table. The values in parenthesis' shows the number of remaining centers after discarding the most popular centers which have higher members than the given T_{stop} parameter i.e. $K - |V_{stop}|$.

Top-5 accuracy performances are also calculated. For this calculation the 5 closest training images are found and if at least one of the retrieved training image has the same label as the test image, the result is accepted to be correct.

Table 5.8. Accuracy (%) for different implementations.

K	500			5000		
T_{stop}	14000	8000	6000	3000	1500	1000
Number of remaining centers	500	488	459	5000	4975	4894
Accuracy (Top 1)	64.41	63.00	63.08	85.36	85.44	85.36
Accuracy (Top 5)	80.76	80.39	79.72	95.77	95.77	95.84

The highest Top-1 performance 85.44% is observed when K is 5000, and T_{stop} is 1500. The highest Top-5 performance 95.84% is observed when K is 5000, and T_{stop} is 1000. The performance increases with the initial number of centers. No significant performance difference is observed between the selected T_{stop} parameters.

Only the parameter K affects the test time for a query image. The other parameters or the number of retrieved images do not affect the test time more than 2%. Average test times for different K values are given below.

Table 5.9. Average test times for different number of centers.

K	Average test time (sec)
500	0.26
5000	0.75

5.4. Image Retrieval Using Visual Words with Non-Geometric Re-Ranking

Another method is implemented in order to re-rank the 5 closest matches. It is a coarse approach just matching the SIFT descriptors between the test and training images and scoring the top-5 closest training images with their number of matching features to the test image.

Steps:

(i) Find 5 closest training images via the same approach used in the previous method. (Image Retrieval Using Using Visual Words)

(ii) Use vl_ubcmatch in the with default threshold parameter 1.5 to match these 5 images to the test image [49]. Training and test features are previously found. vl_ubcmatch matches a test feature to the training feature which minimizes the Euclidian distance if the ratio of distance to second-closest to closest neighbors of each key point is more than a threshold parameter.

$$\text{Match } f \text{ to } f_{closest} \text{ if } \frac{dist(f,f_{second_closest})}{dist(f,f_{closest})} > threshold$$

(iii) Label the test image as the training image with the highest number of matching features.

Obviously the previous Top-5 accuracy rate is a bounding number for this method's accuracy rate. If there is no same class training image among the 5 closest training

images, it is impossible to obtain a correct match via this method.

Note: Re-ranking only favors number of matching features. Top xx training images are treated the same. Tf-idf distances are not used in re-ranking. Having smaller tf-idf distance has no advantage here.

This method does not make use of the geometric locations of the SIFT features.

5.4.1. Experiment

This method is only implemented on Dataset 2. Accuracies for different parameters are given below.

Table 5.10. Accuracy (%) for Image Retrieval Using Visual Words with Non-Geometric Re-Ranking.

K	500			5000		
T_{stop}	14000	8000	6000	3000	1500	1000
Number of remaining centers	500	488	459	5000	4975	4894
Accuracy (Top 1)	64.41	63.00	63.08	85.36	85.44	85.36
Accuracy (Top 5)	80.76	80.39	79.72	95.77	95.77	95.84
Accuracy (Re-rank Top 5)	75.04	74.59	73.63	90.19	90.42	90.12

The highest performance 90.42% is observed when K is 5000, and T_{stop} is 1500. Naturally accuracy increased and stayed below the Top-5 accuracy rates.

Again, only the parameter K affects the test time for a query image. The other parameters do not affect the test time more than 2%. Average test times for different K values are given below. Here K only affects the algorithm time at the first step of this method. Average test time increased by 1.2 seconds on average. This time difference

mainly resulted from the SIFT feature matching step.

Table 5.11. Average test times for different number of centers.

K	Average test time (sec)
500	1.58
5000	1.87

We increased the number of closest matches to re-rank. The aim is finding the training image (among top candidates) with the highest number of SIFT feature matches to the query image. This is implemented for the case of K = 5000, and T_{stop}= 1500 only. Results are below.

Table 5.12. Accuracy (%) and average test times (sec) for K=5000, T_{stop}=1500.

	Accuracy (%)	Average test time (sec)
Closest match	85.44	0.75
Re-rank first 5 matches	90.19	1.87
Re-rank first 10 matches	92.27	4.1
Re-rank first 15 matches	93.09	4.5
Re-rank first 30 matches	94.21	11.3
Re-rank first 60 matches	94.95	12.5

We see that the optimal case is re-ranking the first 15 closest matches; since further re-ranking increases time while bringing small increases to accuracy.

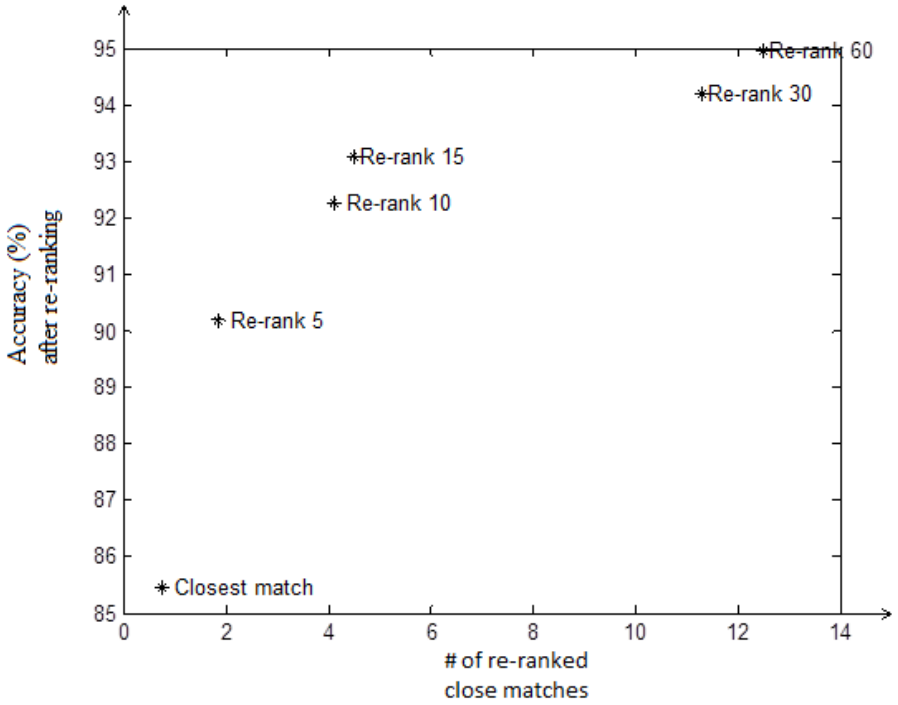


Figure 5.12. Re-ranking accuracy vs. Number of re-ranked close matches

5.5. Image Retrieval Using Visual Words with Geometric Re-Ranking

In order to re-rank the top training image candidates, geometric alignment of training images to the query image is used. [14] For every training image candidate, a geometric transformation is found. We find the geometric transformation, using the locations, scales, and orientations of SIFT matching SIFT descriptors. Steps of finding the geometric transformation for a pair of images (the test image and a candidate training image) are given below. The found transformations are used to re-rank the training images.

(i) Match the SIFT features of test and training images using vl_ubcmatch with default threshold parameter 1.5 (same as the first step of the previous method).

(ii) Construct the set of transformation hypotheses from the list of corresponding feature pairs. Each pair provides such an hypothesis as follows:

$$t_x = x_A - x_B \tag{5.4}$$

$$t_y = y_A - y_B \tag{5.5}$$

$$s = \frac{s_A}{s_B} \tag{5.6}$$

$$\theta = \theta_A - \theta_B \tag{5.7}$$

Consequently, the transformation from the point $p_B = (x_B, y_B, 1)^T$ in image B to the point $p_A = (x_A, y_A, 1)^T$ in A is given by

$$p_A = s \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} (p_B - \begin{pmatrix} t_x \\ t_y \\ 1 \end{pmatrix}) \tag{5.8}$$

(iii) For each transformation hypothesis:

- Apply the transformation on the remaining pairs of corresponding features.
- Identify the inliers using one-way transformation error.
- Identify the inliers (the test features which are transformed to a point in the ϵ_n neighborhood of their matching training feature)
- Take note of the number of inliers.

(iv) Choose the hypothesis with the greatest number of inliers among all hypotheses.

(v) Solve for the affine transformation using the output inlier set of step (3).

(vi) Recalculate the inliers using the final affine transformation. If the inlier set has four or more elements, then the match is geometrically verified. A match is deemed as inlier if transformation maps a test feature to a point in the ϵ_n neighborhood of their matching training feature.

Choosing the hypothesis with the greatest number of inliers is standard RANSAC algorithm [16]. However the randomness is removed since each transformation hypoth-

esis is tried. Thus, the procedure is deterministic [14].

After these steps, two different approaches are used to determine the class of the test image.

5.5.1. Approach 1: Re-ranking by the number of inliers

Select the training image with the highest number of inliers recalculated at step (6). Label the test image with this training image's class. This method is implemented with the parameters K=5000, T_{stop}=1500, (# of remaining centers=4894) on 20 class dataset. 3 different ϵ_n parameters are tried.

Table 5.13. Accuracy (%) for different ϵ_n parameters.

ϵ_n	80	60	20
Re-rank first 5 matches	89.60	88.86	88.78
Re-rank first 10 matches	91.98	91.16	90.49
Re-rank first 15 matches	93.37	93.16	91.25
Re-rank first 30 matches	94.13	94.13	91.75
Re-rank first 60 matches	94.43	94.43	92.57

Table 5.14. Average test times (sec) for K=5000, T_{stop}=1500.

Re-rank first 5 matches	3.2
Re-rank first 10 matches	4.1
Re-rank first 15 matches	5.3
Re-rank first 30 matches	10.9
Re-rank first 60 matches	14.5

Again, re-ranking the first 15 closest matches seems optimal. We reached an accuracy of 93.37% at 5.3 seconds test time.

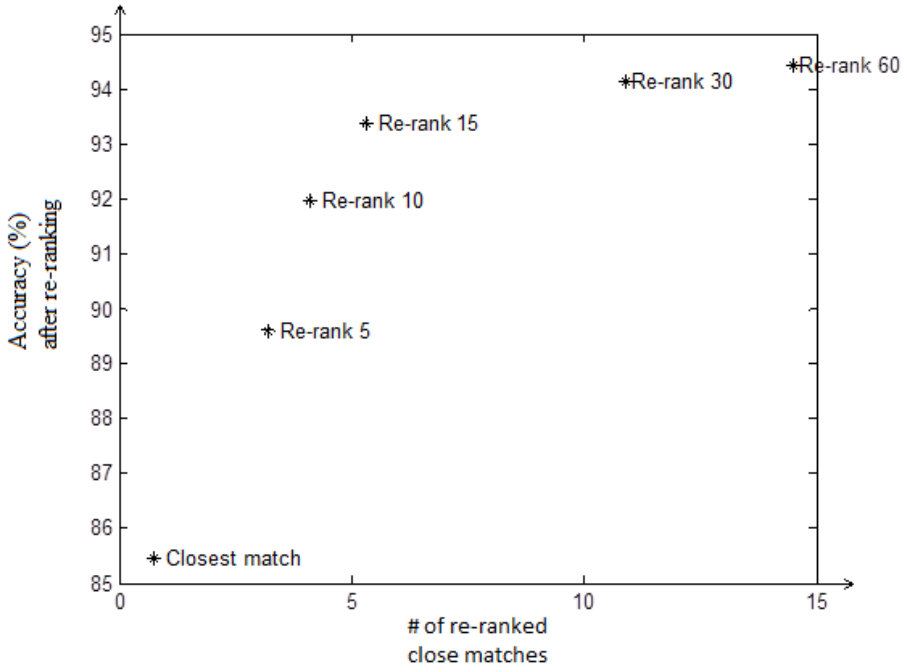


Figure 5.13. Re-ranking accuracy vs. Number of re-ranked mathces

5.5.2. Approach 2: Re-ranking by the tf-idf sum of inliers

- Spatial (geometric) verification is deemed "successful" if the final transformation gives at least 4 inlier correspondences.
- Images are re-ranked by scoring them equal to the sum of the idf values for the inlier words. Geometrically verified images are placed above the unverified ones in the re-ranked list.
- Re-ranking is aborted if there is no "success" 20 images in a row in the initial ranked list.
- Select the training image with the highest score. Label the test image with this training image's class.

This method is implemented with the parameters K=5000, T_{stop}=1500, (# of remaining centers=4894) on 20 class dataset. Only first 15 images are re-ranked yet and ϵ_n is chosen as 80. 92.49% accuracy is obtained.

5.5.3. Observation and Discussion

Optimum results for different methods with the same parameter combination (K=5000, T_{stop}=1500) on 20 class dataset are below.

Table 5.15. Optimum results for different methods.

Section 5.3		Section 5.4	Section 5.5 (Approach 1)	Section 5.5 (Approach 2)
% of test images with correct closest match label	% of test images which lhave at east 1 relevant match among top 15	Accuracy after non-geometric re-ranking (# of matches)	Accuracy after geometric re-ranking (# of inliers)	Accuracy after geometric re-ranking (tf-idf sum of inliers)
85.36%	98.51%	93.09%	93.37%	92.49%
0.75 sec		4.5 sec	5.3 sec	5.5 sec

Geometric re-ranking was expected to yield better results. However accuracy stayed almost same. Test time for geometric re-ranking is slightly higher than non-geometric re-ranking.

5.6. Image Retrieval Using Random Sampled Densely Extracted Visual Words

One of the important findings in the paper "Sampling strategies for bag-of-features image classification" is that random sampled visual words are more discriminative than the detected ones [49]. To this end, we modified the method in Section 5.3 (Image Retrieval Using Visual Words) so that a SIFT detector is not used and instead, we calculated SIFT descriptors at specific locations and at a fixed scale and orientation. We extracted SIFT descriptors from a grid of pixels. And instead of clustering the visual words, we randomly sampled them as centers. We calculated tf-idf vectors

and described the test and training images with this vectors.

5.6.1. Experiment

This method is implemented with the parameters K=5000, T_{stop}=1500, (# of remaining centers=4919) on 20 class dataset. The steps are explained below:

(i) Resize all images to 256 x 256 pixels.

(ii) Using the training images, calculate SIFT descriptors at every 8 pixels. This results in 29 horizontal and 29 vertical steps and thus from one image, 841 SIFT descriptors are collected. The selected scale is 2.66 and orientation is 0 degrees. It takes 0.13 seconds to extract 841 descriptors from an image.

(iii) From 2708 training images, 2277428 descriptors are collected.

(iv) We sampled 5000 of these training descriptors randomly to construct a vocabulary.

(v) Every image is described with a tf-idf vector. We used a T_{stop} value of 1500 and the number of remaining centers become 4919.

(vi) For a given test image, we sort the training images with respect to tf-idf distances.

With this method we reached a closest match accuracy of 95.54%, at 0.67 seconds average test time. This is the highest accuracy compared to all the previous experiments. In Section 5.3, the highest closest match accuracy was 85% and re-ranking the first 15 matches increased the accuracy up to 93%. However dense descriptor collection and random vocabulary construction enabled us to achieve 95.54% accuracy without any re-ranking.

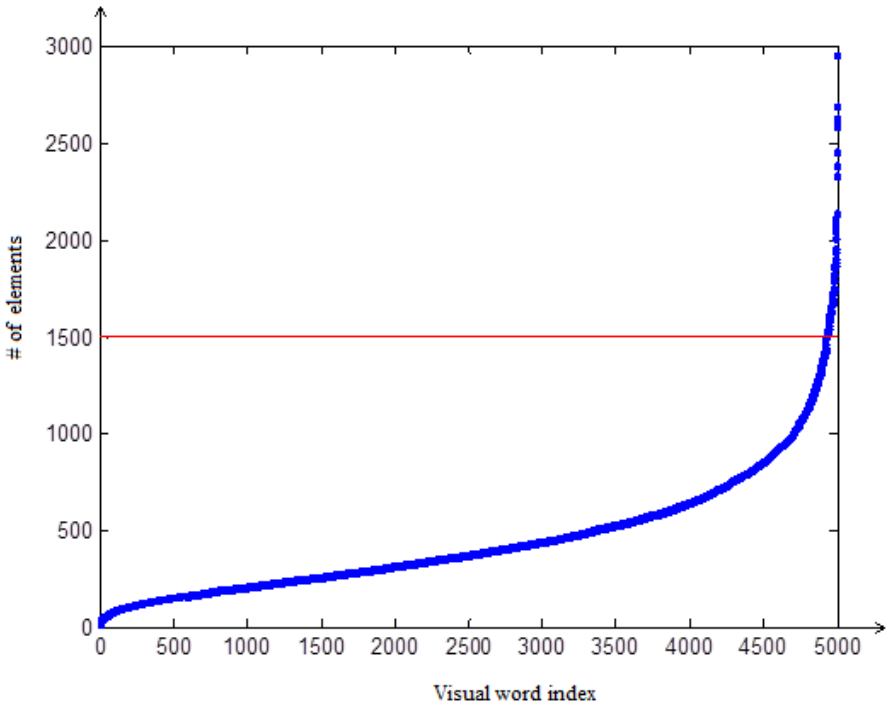


Figure 5.14. Number of elements for 5000 random visual words

Table 5.16. Confusion matrix.

Class name	Image number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Accuracy
1	65	53	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	81.54%
2	65	14	51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78.46%
3	66	0	0	66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
4	66	0	0	0	56	0	1	7	0	0	0	0	0	2	0	0	0	0	0	0	0	84.85%
5	62	0	0	0	0	61	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	98.39%
6	66	1	1	0	0	0	60	0	0	0	2	1	0	0	0	0	0	0	1	0	0	90.91%
7	65	0	0	1	2	0	0	60	0	0	1	1	0	0	0	0	0	0	0	0	0	92.31%
8	70	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
9	68	0	0	0	0	1	0	0	0	65	0	0	0	0	1	0	0	0	0	1	0	95.59%
10	67	0	0	0	0	0	0	2	0	0	65	0	0	0	0	0	0	0	0	0	0	97.01%
11	72	0	0	0	1	0	0	0	0	0	0	71	0	0	0	0	0	0	0	0	0	98.61%
12	68	0	0	0	0	0	0	0	0	0	0	0	68	0	0	0	0	0	0	0	0	100.00%
13	69	0	0	0	0	0	0	0	0	0	1	1	0	67	0	0	0	0	0	0	0	97.10%
14	69	0	0	1	0	0	0	0	0	0	0	0	0	0	67	0	0	1	0	0	0	97.10%
15	65	0	0	0	0	0	0	0	0	0	0	0	0	1	0	64	0	0	0	0	0	98.46%
16	71	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	71	0	0	0	0	100.00%
17	69	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	69	0	0	0	100.00%
18	68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	68	0	0	100.00%
19	68	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	67	0	98.53%
20	67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	67	100.00%
																						95.54%

Observing the table we see that 60 of the 1346 test images are wrongly labeled with the closest retrieved image's label. Almost half of the problematic cases are the Class 1 and Class 2 image mix ups. This is due to the fact that these 2 classes are visually very similar.



Figure 5.15. Sample images from Class 1 (left) and Class 2 (right)

We see that if we are going to use re-ranking, top 5 images would be enough since further re-ranking will not increase the "ratio of test images which have at least one relevant retrieved image among retrieved training images" more than 1%.

Table 5.17. Ratio (%) of test images which have at least one correct close match among retrieved training images.

Top 1	95.54
Top 2	97.85
Top 3	98.66
Top 4	98.96
Top 5	99.11

We decided to implement the same approach used in Section 5.4 (non-geometric re-ranking) to re-rank the first 5 matches. We achieved 96.73% at an average test time of 1.43 seconds.

5.6.2. Result Evaluation

Since we did not use a SIFT detector at different scales and orientations, SIFT descriptor collecting time dropped drastically (to 6 minutes). Also by random codebook generation, we bypassed the clustering phase which was taking approximately 8 hours. So the effect on training time is immense.

Using dense descriptor collection and random descriptor sampling, we achieved the highest accuracy 95.54%, highest top 5 accuracy 99.11%, and highest accuracy after re-ranking 96.73%. We verified the findings by Nowak *et al.* [31]. In a recognition problem of bigger scope, the accuracies would be lower. Since we could not achieve an accuracy of 99% top 1 image matching, we still need to use human validation in a system which needs to recognize the images at almost perfect precision. A number of retrieved images can be of great use to a human who is tasked to recognize the images.

6. DISCUSSION

In this thesis, we proposed a system which consists of different blocks, and we implemented various experiments to optimize each block. In this chapter, we will discuss the results of these experiments.

6.1. Boundary Detection

These experiments can be divided in two parts,

(i) Descriptor - Classifier selection,
(ii) Peak detection on "Boundary probability signal".

6.1.1. Descriptor - Classifier selection

We collected visual descriptors from boundary and non boundary stripe regions. In these experiments 3 different visual descriptor types are used to describe stripe regions:

(i) Filter response features,
(ii) Histograms of Oriented Gradients,
(iii) Local Binary Pattern Histograms.

For every type of descriptor, we used 3 different type of classifiers:

(i) Quadratic classifier,
(ii) AdaBoost,
(iii) SVM.

6.1.1.1. FRF experiments. We observed that SVM achieves highest accuracy (90.46%) for these descriptors. However the query time 2.32 seconds is very high. This is because the descriptor dimension is very high. The second highest accuracy is obtained by AdaBoost (87.79%). The query time 0.54 seconds is somewhat acceptable. It is 8 times lower than SVM. AdaBoost only focuses on the important dimensions and this reduces the query time significantly for high dimensional descriptors such as our Root Filter Set filter response features.

6.1.1.2. HOG experiments. In these experiments we observed that HOG descriptors are better at encoding what it means to be boundary. It makes sense since the gradient distribution behavior changes very much between boundary and non boundary regions. With SVM, we were able to achieve an accuracy of 88.51% accuracy at a very reasonable query time of 0.28 seconds.

6.1.1.3. LBP experiments. We could not achieve very successful results with LBP histograms. Even using uniform LBP's we were not able to reduce the dimension significantly. With SVM, we were able to achieve 86.42% accuracy at 0.53 seconds query time. If we ignore the time constraint, we were able to achieve 88.24% accuracy with SVM at 1.5 seconds query time. However even this accuracy is below the accuracy we achieved with HOG at a very low query time.

6.1.2. Peak detection on "Boundary probability signal"

Using the HOG - SVM combination, we extracted boundary probability values for different image slices and extracted a boundary probability signal. Our proposition was to detect the peaks and declare them as boundaries. We did not experiment with many parameters here. We tried different probability thresholds and found that 0.5 is a proper threshold to achieve equal error rate. This threshold is also logical since the a probability value below 0.5 means that the classifier would label the corresponding stripe as non boundary and vice versa.

6.2. Product Recognition

In this part we tried some image recognition method from the literature to see how we achieve a satisfactory recognition performance on a subset of retail product images. Mainly 5 different methods are tried.

6.2.1. Descriptor Matching with Geometric Verification

This was a raw method in which we match the SIFT features from test and training images and verify the matches with geometric alignment. The accuracy was 93%. However recognizing a single query image takes 75 seconds. This is because we are brute force matching the SIFT descriptors from the query image to a large dataset of training descriptors. After this experiment, we came into the conclusion that, using visual words would be more proper for our case.

6.2.2. Image Retrieval Using Visual Words

In these experiments, we implemented a Bag of Visual Words approach with various parameters. Using a SIFT detector we extracted SIFT descriptors from training images and we constructed a visual vocabulary by clustering these descriptors. We were able to achieve very little query times (0.75 seconds) at the cost of longer training times by representing test and images with tf-idf vectors and matching them. However, the recognition performance was not very high (85.44%). Thus we focused on increasing it by re-ranking top retrieved database images.

6.2.3. Image Retrieval Using Visual Words with Non-Geometric Re-Ranking

Using the parameter combination which yielded highest accuracy, we introduced an extra step. We re-ranked the closest training images with respect to the number of matching SIFT descriptors to the query image. Here we observed that when we increased the number of close matches, the accuracy after re-ranking increases. However, the marginal benefit drops after a point. The accuracy does not increase so much;

moreover, the test time increases linearly. This is why we decided that re-ranking 15 close matches is the best decision. We were able to achieve 93.09% accuracy at 4.5 seconds query time.

6.2.4. Image Retrieval Using Visual Words with Geometric Re-Ranking

Here, we implemented a more complex re-ranking strategy. For a number of close matches, we calculated the best affine transformation and instead of the number of descriptors matches, we count the number of inliers. This did not increased the accuracy much, compared to the previous method. Re-ranking 15 closest matches took 5.3 seconds and the accuracy becomes 93.37%.

6.2.5. Image Retrieval Using Random Sampled Densely Extracted Visual Words

In these experiments, we implemented a different approach. Instead of using a SIFT detector, we sampled SIFT descriptors at specific locations (Dense SIFT). And at vocabulary construction step, instead of clustering, we used random sampling. We introduced these two differences to be able to cover all kinds of descriptors. We observed that with this method we were able to create a more descriptive vocabulary. A SIFT detector is successful at finding features to match between images. However, at vocabulary construction, we probably need more diverse samples in order to represent the descriptor space. The accuracy reached to 95.54% at 0.67 seconds query time.This is the highest accuracy at shortest query time yet. We also tried re-ranking with non-geometric verification and achieved 96.73% accuracy at 1.43 seconds query time.

7. CONCLUSION

This thesis focuses on the implementation of an image recognition method to the case of retail product recognition. After reviewing the literature on image and product recognition literature, we proposed a method where we first detect the retail product separating shelf and product boundaries and later recognize the extracted image crops.

Our boundary detection method is a novel approach where we use a sliding window to calculate the probability of including a boundary for image stripes using supervised learning techniques. We experimented with different image descriptors to describe image stripes and various machine learning algorithms to come up with a stable classifier. We used the classifier to calculate a "boundary probability signal" and detect the boundaries with peak localization. This approach is used in shelf line detection and product boundary detection and very promising results are obtained. The developed algorithm can facilitate the task of cropping distinct product sub images in a system where human users are assigned.

At product recognition step, we experimented with state of the art image description methods. We worked with SIFT descriptors and used image to image nearest neighbor classifier to decide on a label for a query image. We saw that describing the images with a bag of words approach is very useful to minimize the query time. The most important finding of these experiments was the verification of the proposition that dense SIFT descriptor collection and random word sampling yielded much better accuracy at faster training and testing times [31].

It is important to mention that an artificial neural network approach would probably attain better results in both boundary detection and product recognition. We considered deep learning as plausible future work. As an initial step, we trained a 60 epoch AlexNet [51] based convolutional neural network to discriminate boundary and non boundary stripe regions and achieved 85.87% accuracy. We assume a deeper network with more training data may help us construct a better boundary detector.

REFERENCES

1. Huang, T., "Computer Vision: Evolution and Promise", *1996 CERN SCHOOL OF COMPUTING*, p. 21, 1996.

2. Sawaya, W. and W. Giauque, *Production and operations management*, Harcourt Brace Jovanovich, 1986.

3. "Vispera Information Technologies, Istanbul, Turkey", `http://www.vispera.co`, accessed at March 2017.

4. Mundy, J. L., "Object recognition in the geometric era: A retrospective", *Toward category-level object recognition*, pp. 3–28, Springer, 2006.

5. Bay, H., T. Tuytelaars and L. Van Gool, "Surf: Speeded up robust features", *Computer vision–ECCV 2006*, pp. 404–417, 2006.

6. Lowe, D. G., "Distinctive image features from scale-invariant keypoints", *International journal of computer vision*, Vol. 60, No. 2, pp. 91–110, 2004.

7. Mikolajczyk, K. and C. Schmid, "A performance evaluation of local descriptors", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 27, No. 10, pp. 1615–1630, 2005.

8. Obdrzalek, S. and J. Matas, "Object recognition using local affine frames on maximally stable extremal regions", *Toward Category-Level Object Recognition*, Vol. 4170, pp. 83–104, 2006.

9. Friedman, J. H., J. L. Bentley and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time", *ACM Transactions on Mathematical Software (TOMS)*, Vol. 3, No. 3, pp. 209–226, 1977.

10. Har-Peled, S., P. Indyk and R. Motwani, "Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality.", *Theory of computing*, Vol. 8, No. 1, pp. 321–350, 2012.

11. Sivic, J. and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos", *Proceedings of the international conference on computer vision*, p. 1470, IEEE, 2003.

12. Baeza-Yates, R., B. Ribeiro-Neto *et al.*, *Modern information retrieval*, Vol. 463, ACM press New York, 1999.

13. Nister, D. and H. Stewenius, "Scalable recognition with a vocabulary tree", *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, Vol. 2, pp. 2161–2168, IEEE, 2006.

14. Philbin, J., O. Chum, M. Isard, J. Sivic and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching", *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

15. Sivic, J. and A. Zisserman, "Efficient visual search of videos cast as text retrieval", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 31, No. 4, pp. 591–606, 2009.

16. Fischler, M. A. and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, Vol. 24, No. 6, pp. 381–395, 1981.

17. Merler, M., C. Galleguillos and S. Belongie, "Recognizing groceries in situ using in vitro training data", *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

18. Viola, P. and M. Jones, "Rapid object detection using a boosted cascade of simple features", *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceed-*

ings of the 2001 IEEE Computer Society Conference on, Vol. 1, pp. I–I, IEEE, 2001.

19. Tsai, S. S., D. Chen, V. Chandrasekhar, G. Takacs, N.-M. Cheung, R. Vedantham, R. Grzeszczuk and B. Girod, "Mobile product recognition", *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1587–1590, ACM, 2010.

20. Chandrasekhar, V., G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk and B. Girod, "CHoG: Compressed histogram of gradients a low bit-rate feature descriptor", *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2504–2511, IEEE, 2009.

21. Tsai, S. S., D. Chen, G. Takacs, V. Chandrasekhar, R. Vedantham, R. Grzeszczuk and B. Girod, "Fast geometric re-ranking for image-based retrieval", *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 1029–1032, IEEE, 2010.

22. Shen, X., Z. Lin, J. Brandt and Y. Wu, "Mobile product image search by automatic query object extraction", *Computer Vision–ECCV 2012*, pp. 114–127, 2012.

23. Griffin, G., A. Holub and P. Perona, *Caltech-256 Object Category Dataset*, Tech. Rep. 7694, California Institute of Technology, 2007, `http://authors.library.caltech.edu/7694`.

24. George, M. and C. Floerkemeier, "Recognizing products: A per-exemplar multi-label image classification approach", *European Conference on Computer Vision*, pp. 440–455, Springer, 2014.

25. Advani, S., B. Smith, Y. Tanabe, K. Irick, M. Cotter, J. Sampson and V. Narayanan, "Visual co-occurrence network: using context for large-scale object recognition in retail", *Embedded Systems For Real-time Multimedia (ESTIMedia), 2015 13th IEEE Symposium on*, pp. 1–10, IEEE, 2015.

26. Baz, I., E. Yoruk and M. Cetin, "Context-aware hybrid classification system for fine-grained retail product recognition", *Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), 2016 IEEE 12th*, pp. 1–5, IEEE, 2016.

27. Martin, D. R., C. C. Fowlkes and J. Malik, "Learning to detect natural image boundaries using brightness and texture", *Advances in Neural Information Processing Systems*, pp. 1279–1286, 2003.

28. Taira, E., S. Uchida and H. Sakoe, "Book boundary detection from bookshelf image based on model fitting", *International Symposium on Information Science and Electrical Engineering*, pp. 601–604, 2003.

29. Iwanowski, M., B. Zieliński, G. Sarwas and S. Stygar, "Identification of products on shop-racks by morphological preprocessing and feature-based detection", *International Conference on Computer Vision and Graphics*, pp. 286–293, Springer, 2014.

30. Lloyd, S., "Least squares quantization in PCM", *IEEE transactions on information theory*, Vol. 28, No. 2, pp. 129–137, 1982.

31. Nowak, E., F. Jurie and B. Triggs, "Sampling strategies for bag-of-features image classification", *Computer Vision–ECCV 2006*, pp. 490–503, 2006.

32. Leung, T. and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons", *International journal of computer vision*, Vol. 43, No. 1, pp. 29–44, 2001.

33. Dalal, N. and B. Triggs, "Histograms of oriented gradients for human detection", *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1, pp. 886–893, IEEE, 2005.

34. Pietikäinen, M. and G. Zhao, "Two decades of local binary patterns: A survey", *Advances in Independent Component Analysis and Learning Machines*, pp. 175–

210, 2015.

35. Ojala, T., M. Pietikäinen and D. Harwood, "A comparative study of texture measures with classification based on featured distributions", *Pattern recognition*, Vol. 29, No. 1, pp. 51–59, 1996.

36. Freund, Y., R. Schapire and N. Abe, "A short introduction to boosting", *Journal-Japanese Society For Artificial Intelligence*, Vol. 14, No. 771-780, p. 1612, 1999.

37. Nordhausen, K., "The elements of statistical learning: data mining, inference, and prediction, by Trevor Hastie, Robert Tibshirani, Jerome Friedman", *International Statistical Review*, Vol. 77, No. 3, pp. 482–482, 2009.

38. Schapire, R. E., "The strength of weak learnability", *Machine learning*, Vol. 5, No. 2, pp. 197–227, 1990.

39. Freund, Y., R. E. Schapire *et al.*, "Experiments with a new boosting algorithm", *icml*, Vol. 96, pp. 148–156, 1996.

40. Iba, W. and P. Langley, "Induction of one-level decision trees", *Proceedings of the ninth international conference on machine learning*, pp. 233–240, 1992.

41. Cortes, C. and V. Vapnik, "Support-vector networks", *Machine learning*, Vol. 20, No. 3, pp. 273–297, 1995.

42. Andersson, C., *Multivariate signal approaches for object detection using microwave sensing*, Master's Thesis, Chalmers University of Technology, Gothenburg, Sweden, 2013.

43. Abdul, S. and A. Basha, *SVM Classification and Analysis of Margin Distance on Microarray Data*, Ph.D. Thesis, University of Akron, 2011.

44. Varma, M. and A. Zisserman, "A statistical approach to texture classification from single images", *International Journal of Computer Vision*, Vol. 62, No. 1, pp. 61–

81, 2005.

45. Bradski, G., *Dr. Dobb's Journal of Software Tools*, 2000.

46. Ahonen, T., A. Hadid and M. Pietikainen, "Face description with local binary patterns: Application to face recognition", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 28, No. 12, pp. 2037–2041, 2006.

47. Ojala, T., M. Pietikainen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 24, No. 7, pp. 971–987, 2002.

48. Platt, J. *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods", *Advances in large margin classifiers*, Vol. 10, No. 3, pp. 61–74, 1999.

49. Vedaldi, A. and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms", *Proceedings of the 18th ACM international conference on Multimedia*, pp. 1469–1472, ACM, 2010.

50. Gersho, A. and R. M. Gray, *Vector quantization and signal compression*, Vol. 159, Springer Science & Business Media, 2012.

51. Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, pp. 1097–1105, 2012.

APPENDIX A: DISCRETE ADABOOST

AdaBoost is a machine learning algorithm to combine weak classifiers into an accurate classifier. The pseudocode of the popular variant Disrete AdaBoost is given below.

Given $(x_1, x_2), ..., (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = \frac{1}{m}$ for $i = 1, \ldots, m$

for $k = 1$ to N **do**

Train weak learner using distribution D_k

Get weak hypothesis $h_k : X \mapsto Y$

Calculate $e_k = Pr_{i \sim D_k}[h_k(x_i) \neq y_i]$

Chose weak classifier weight $\alpha_k = \frac{1}{2} ln(\frac{1-e_k}{e_k})$

Update $D_{k+1}(i) = D_k(i) e^{-\alpha_k y_i h_k(x_i)} / Z_k$ where Z_k is a normalization factor (chosen so that D_{k+1} will be a distribution).

end for

Output the final hypothesis

$$H(x) = sign(\sum^{N} \alpha_k h_k(x))$$

Figure A.1. Discrete AdaBoost Pseudocode

APPENDIX B: BOUNDARY DETECTION RESULTS

Below are some sample results achieved by proposed product boundary detection method.

- Upper images: Probability signal,
- Middle images: Smoothed probability signal (red), detected peak indices (blue), threshold (yellow),
- Lower images: Detected true peak indices (green), Detected false peak indices (red), and Ground truth annotations (blue).



Figure B.1. Product boundary detection sample result 1



Figure B.2. Product boundary detection sample result 2

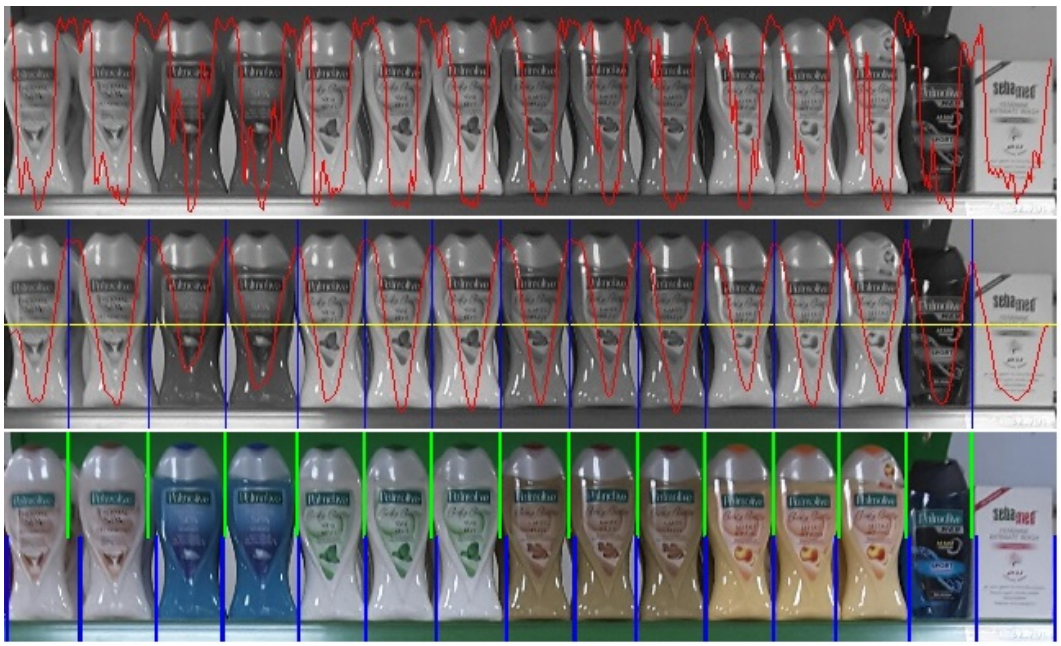


Figure B.3. Product boundary detection sample result 3

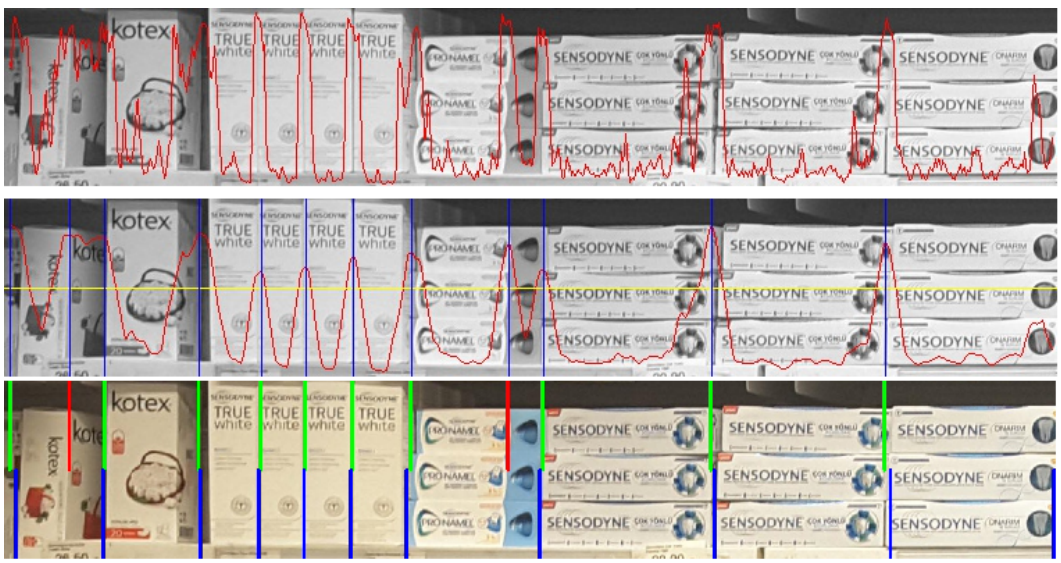


Figure B.4. Product boundary detection sample result 4

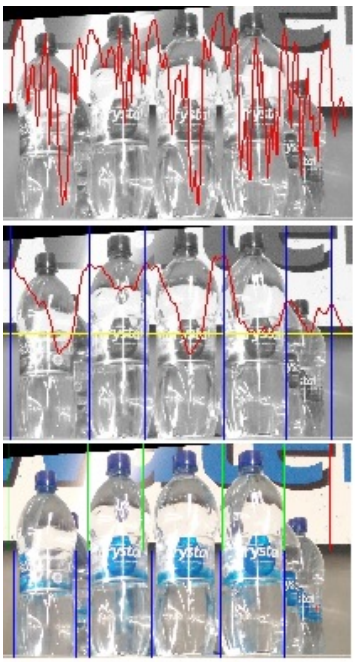


Figure B.5. Product boundary detection sample result 5



Figure B.6. Product boundary detection sample result 6

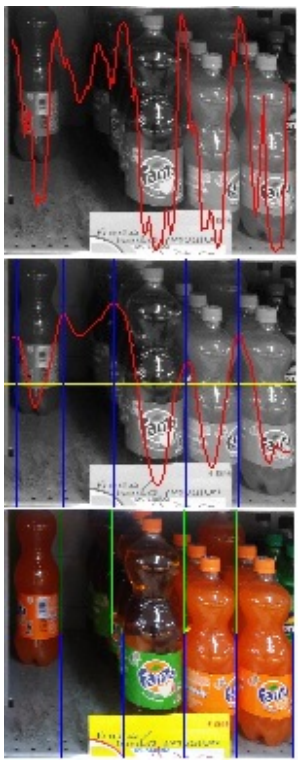


Figure B.7. Product boundary detection sample result 7

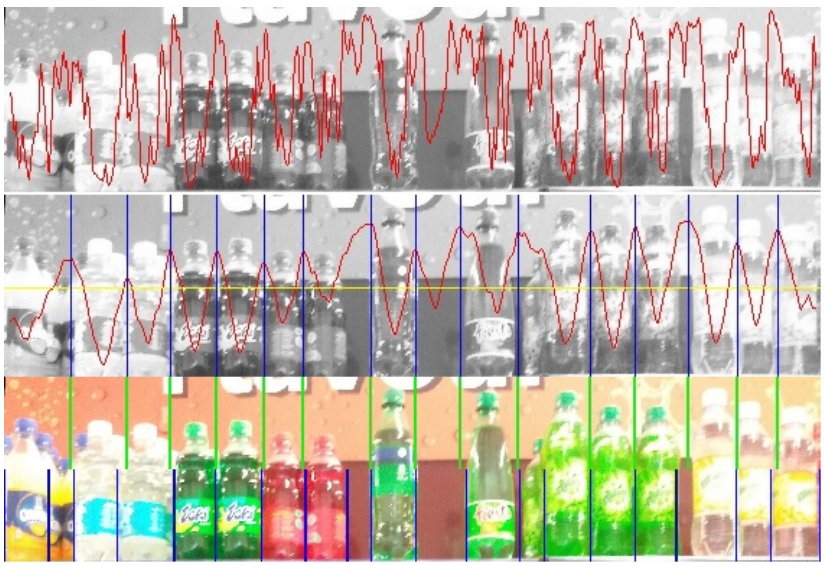


Figure B.8. Product boundary detection sample result 8

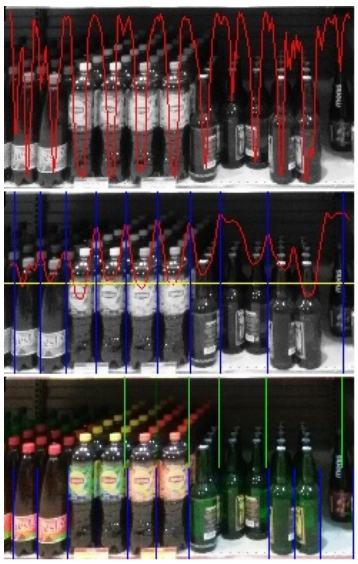


Figure B.9. Product boundary detection sample result 9

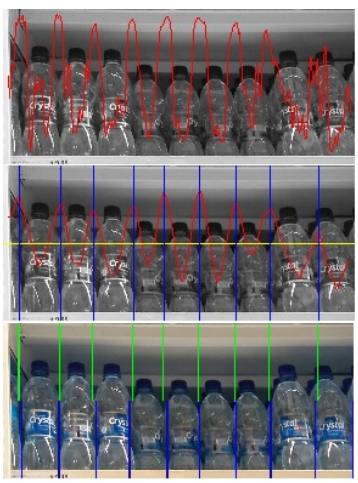


Figure B.10. Product boundary detection sample result 10



Figure B.11. Product boundary detection sample result 11

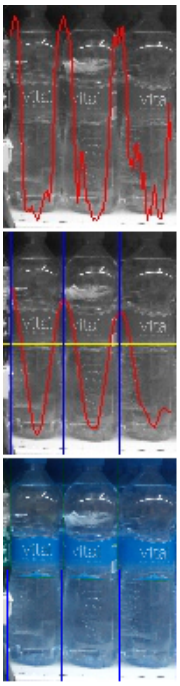


Figure B.12. Product boundary detection sample result 12

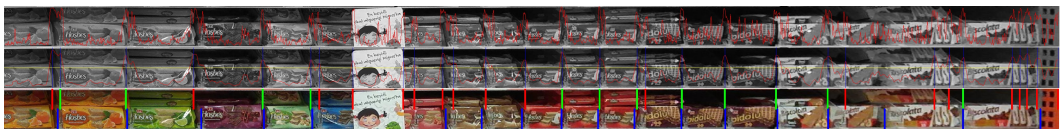


Figure B.13. Product boundary detection sample result 13

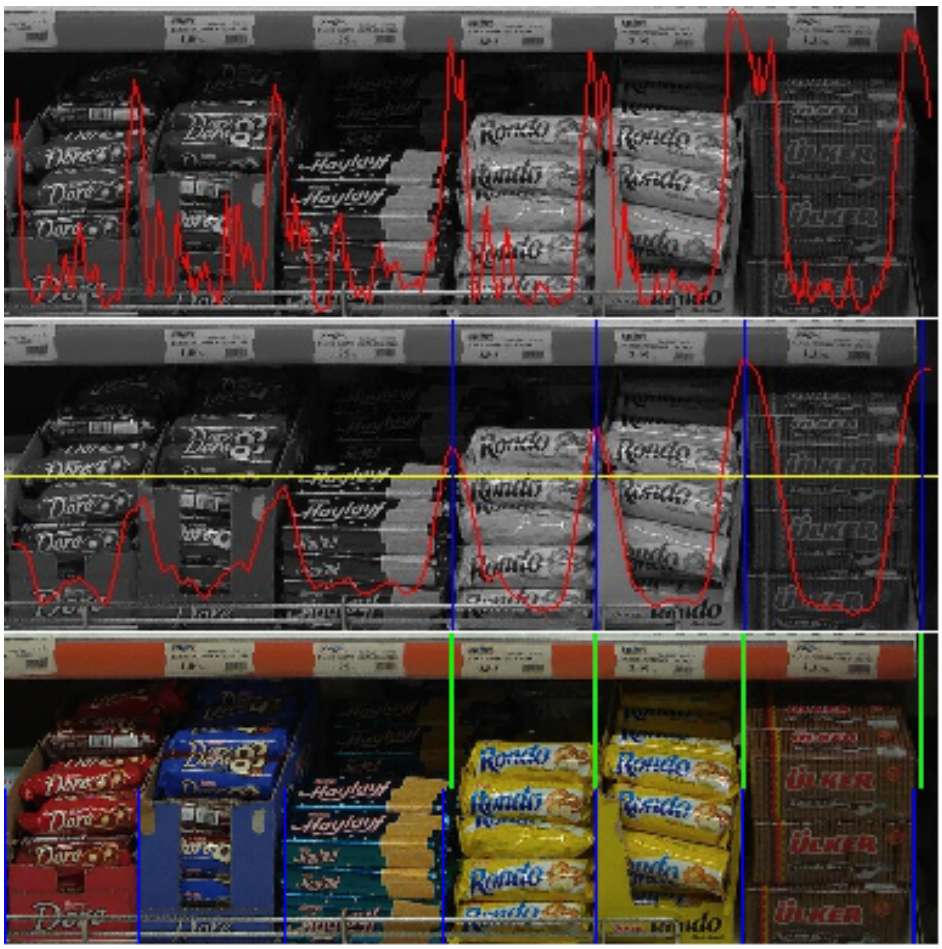


Figure B.14. Product boundary detection sample result 14



Figure B.15. Product boundary detection sample result 15

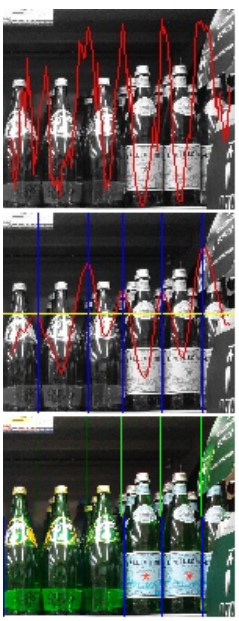


Figure B.16. Product boundary detection sample result 16

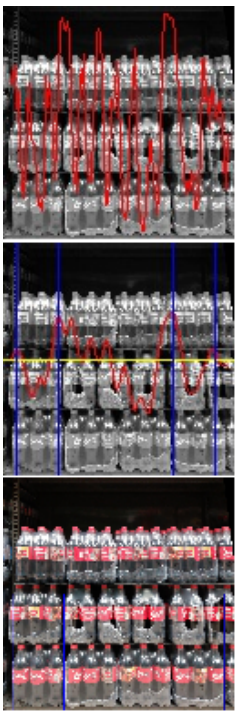


Figure B.17. Product boundary detection sample result 17



Figure B.18. Product boundary detection sample result 18

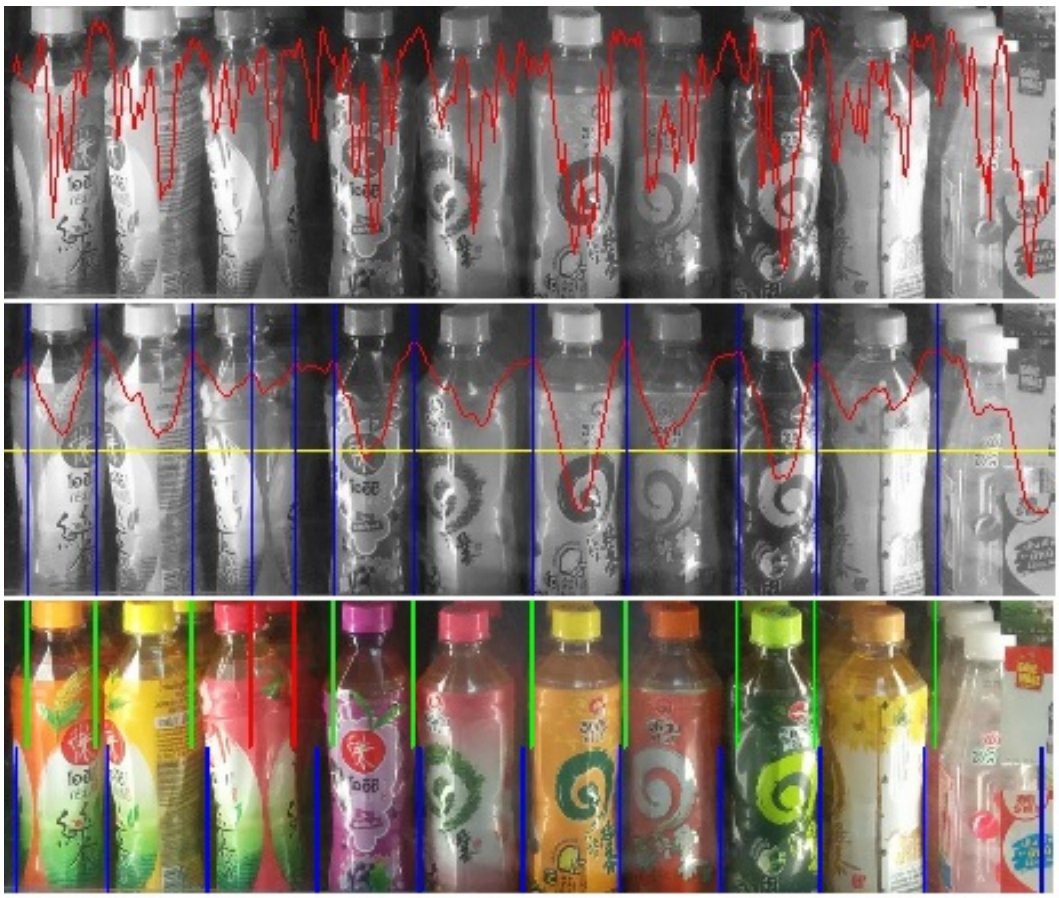


Figure B.19. Product boundary detection sample result 19

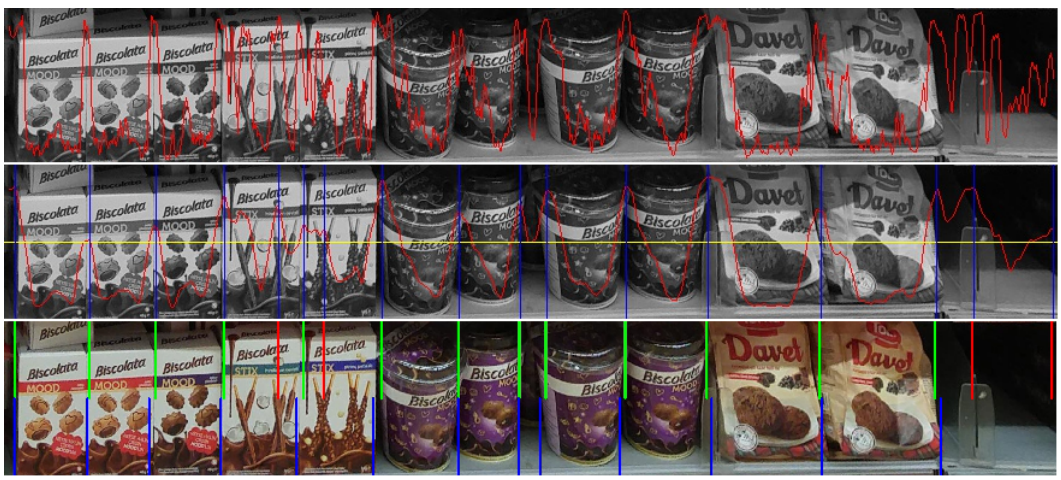


Figure B.20. Product boundary detection sample result 20

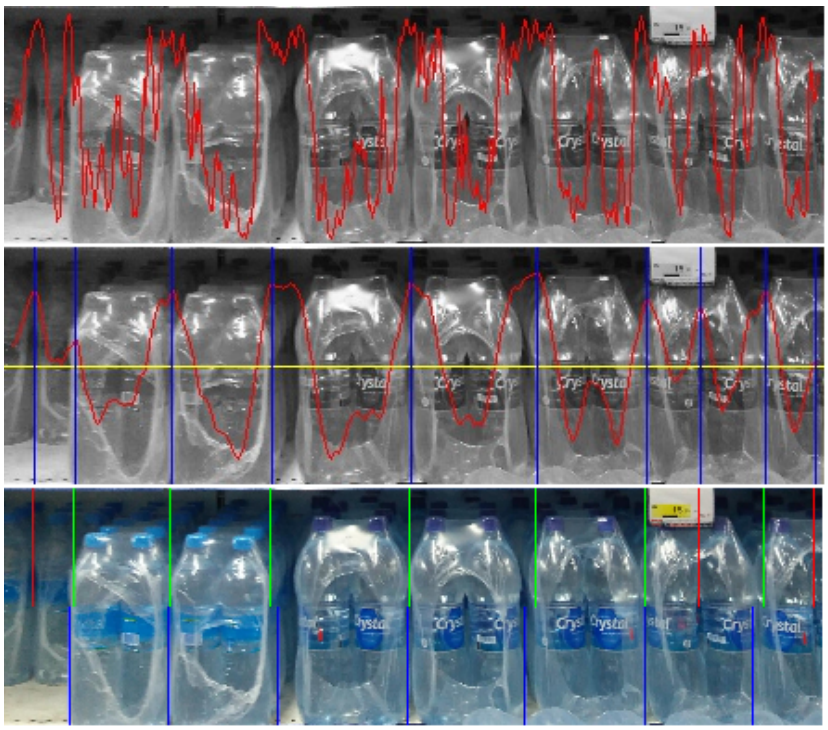


Figure B.21. Product boundary detection sample result 21



Figure B.22. Product boundary detection sample result 22

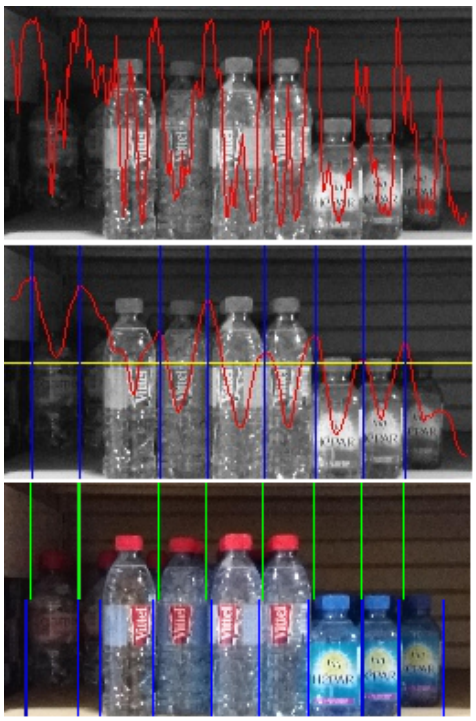


Figure B.23. Product boundary detection sample result 23

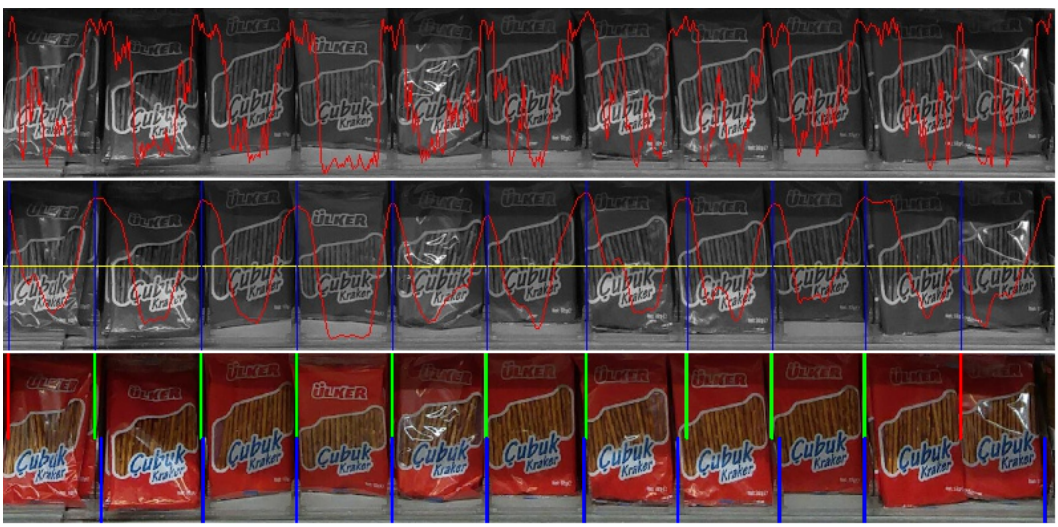


Figure B.24. Product boundary detection sample result 24



Figure B.25. Product boundary detection sample result 25



Figure B.26. Product boundary detection sample result 26



Figure B.27. Product boundary detection sample result 27

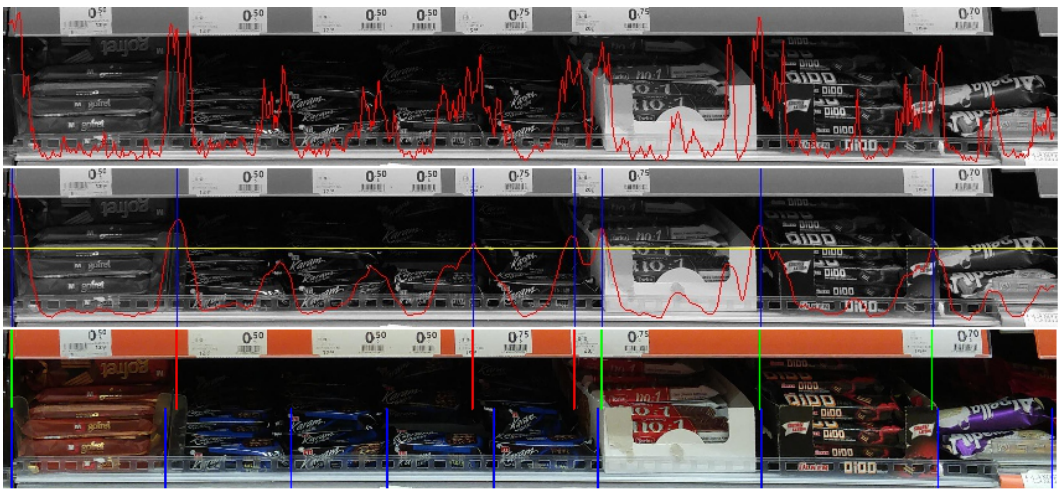


Figure B.28. Product boundary detection sample result 28

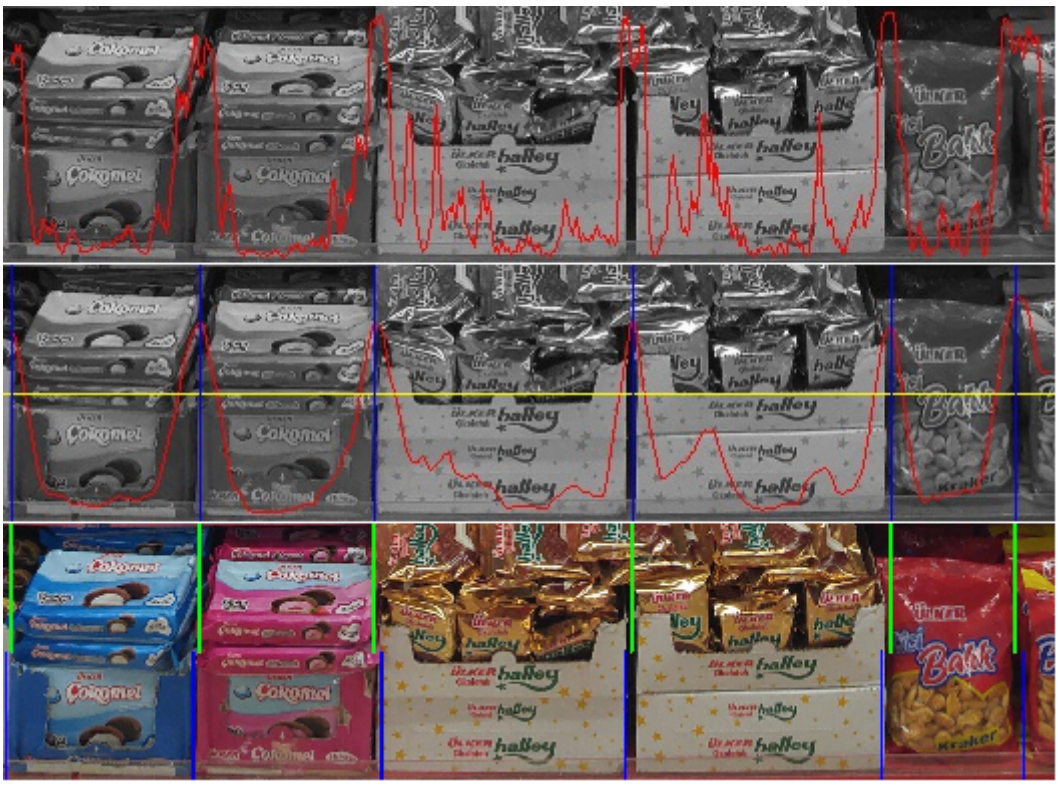


Figure B.29. Product boundary detection sample result 29

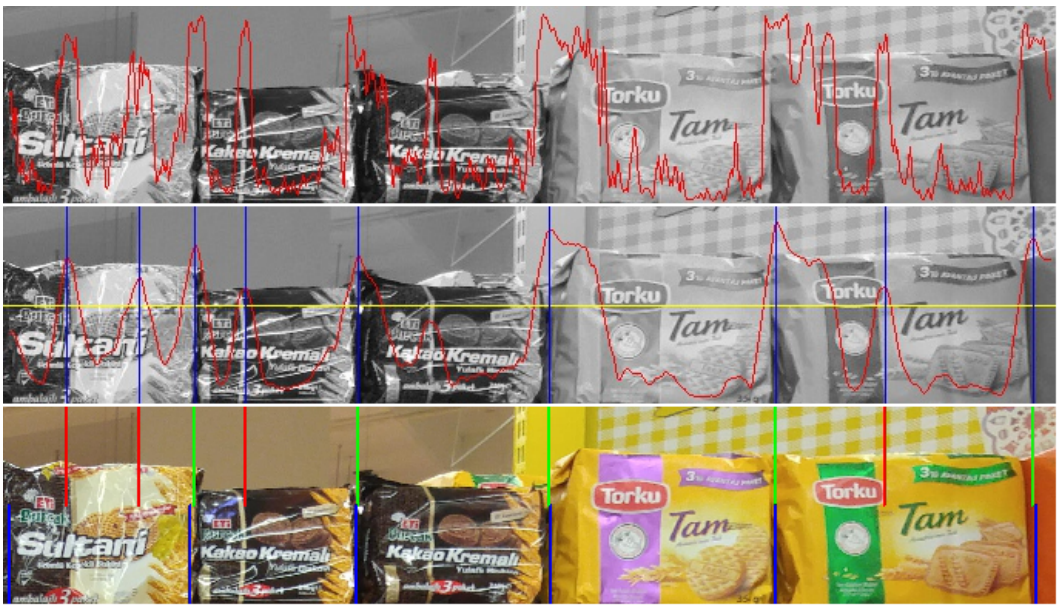


Figure B.30. Product boundary detection sample result 30

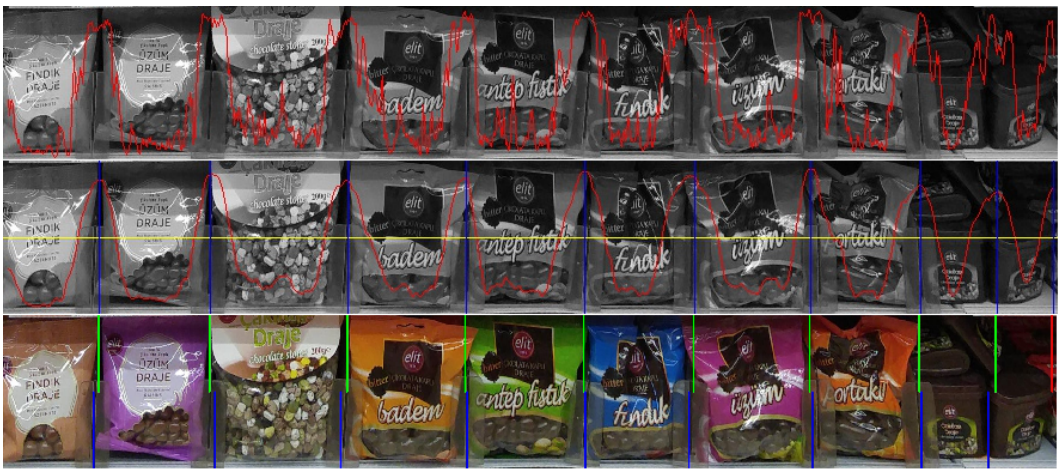


Figure B.31. Product boundary detection sample result 31

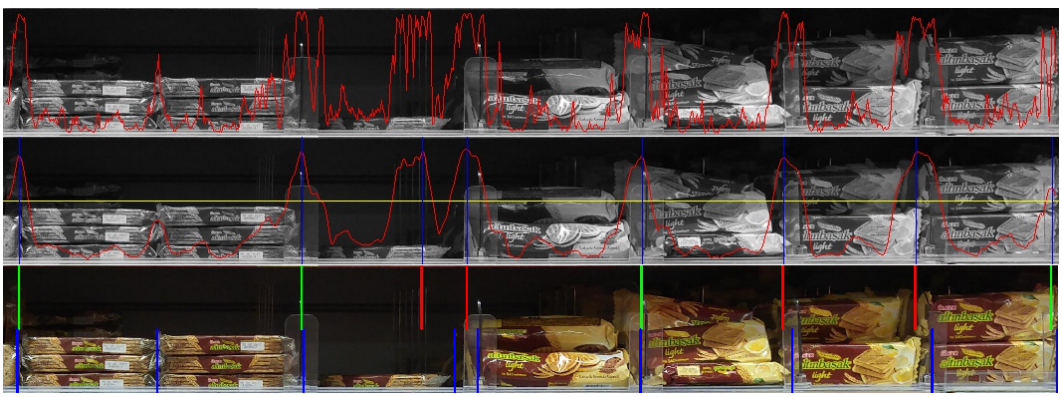


Figure B.32. Product boundary detection sample result 32

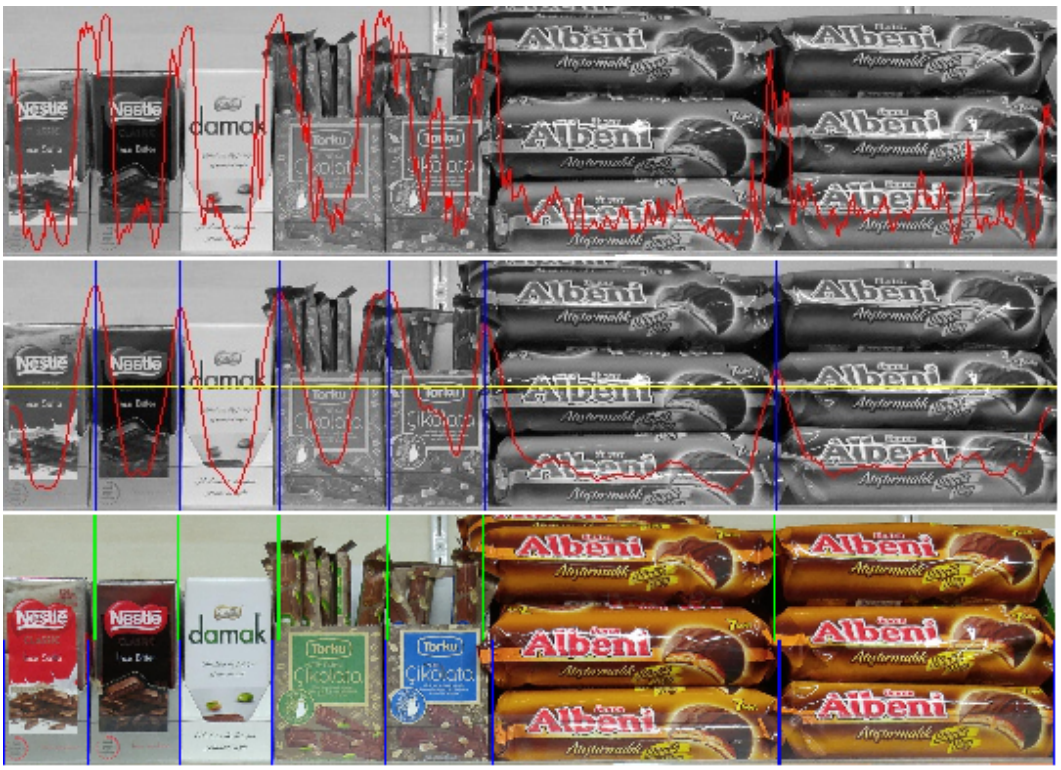


Figure B.33. Product boundary detection sample result 33

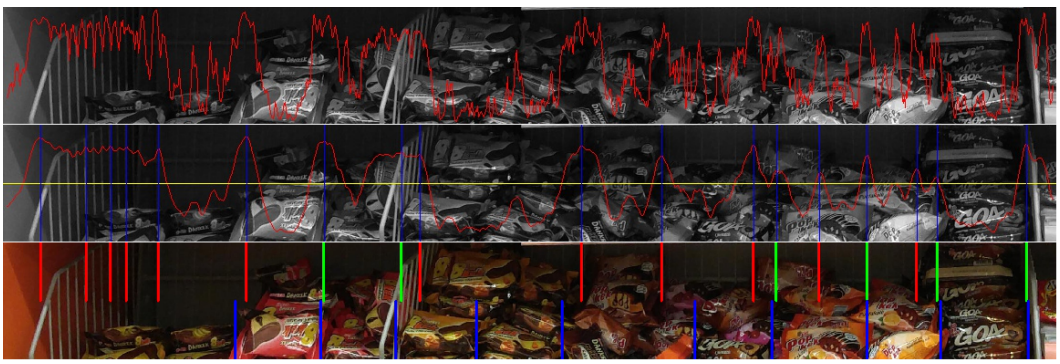


Figure B.34. Product boundary detection sample result 34

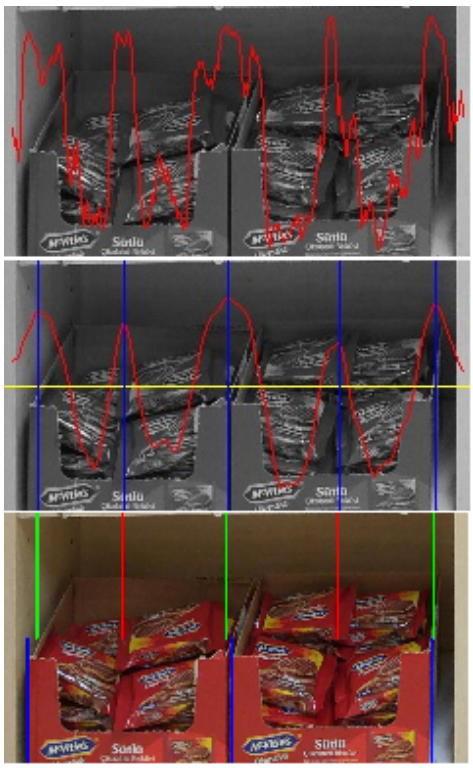


Figure B.35. Product boundary detection sample result 35

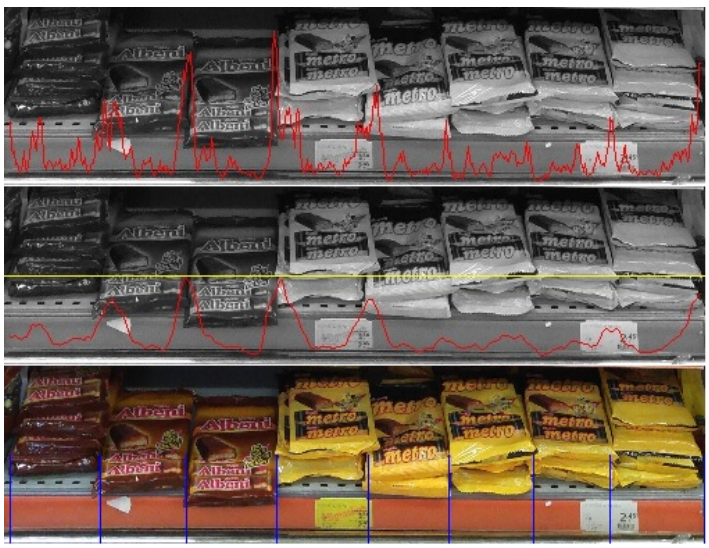


Figure B.36. Product boundary detection sample result 36

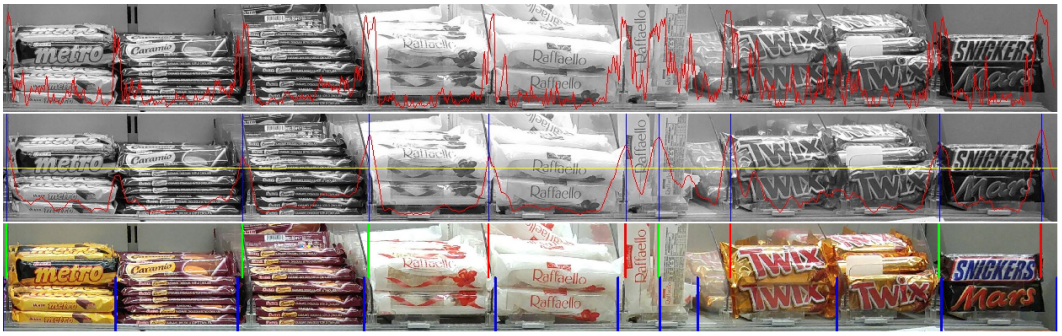


Figure B.37. Product boundary detection sample result 37



Figure B.38. Product boundary detection sample result 38



Figure B.39. Product boundary detection sample result 39



Figure B.40. Product boundary detection sample result 40