# EXTRACTION OF BUILDINGS IN SATELLITE IMAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MELİH ÇETİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

MAY 2010

Approval of the thesis:

**EXTRACTION OF BUILDINGS IN SATELLITE IMAGES**

submitted by **MELİH ÇETİN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen ________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen ________
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. A. Aydın Alatan ________
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Prof. Dr. Uğur Halıcı ________
Co-Supervisor, **Electrical and Electronics Engineering Dept., METU**

**Examining Committee Members**

Prof. Dr. Aydan Erkmen ____________
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. A. Aydın Alatan ____________
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Uğur Halıcı ____________
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Afşar Saranlı ____________
Electrical and Electronics Engineering Dept., METU

Dr. H. Burak Kaygısız ____________
Chief of Guidance and Control Division, TÜBİTAK-SAGE

**Date:** ____________

# ABSTRACT

EXTRACTION OF BUILDINGS IN SATELLITE IMAGES

Çetin, Melih
Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. A. Aydın Alatan
Co-Supervisor: Prof. Dr. Uğur Halıcı

May 2010, 197 pages

In this study, an automated building extraction system, which is capable of detecting buildings from satellite images using only RGB color band is implemented. The approach used in this work has four main steps: local feature extraction, feature selection, classification and post processing. There are many studies in literature that deal with the same problem. The main issue is to find the most suitable features to distinguish a building. This work presents a feature selection scheme that is connected with the classification framework of Adaboost. As well as Adaboost, four SVM kernels are used for classification. Detailed analysis regarding window type and size, feature type, feature selection, feature count and training set is done for determining the optimal parameters for the classifiers. A detailed comparison of SVM and Adaboost is done based on pixel and object performances and the results obtained are presented both numerically and visually. It is observed that SVM performs better if quadratic kernel is used than the cases using linear, RBF or polynomial kernels. SVM performance is better if features are selected either by Adaboost or by considering errors obtained on histograms of features. The performance obtained by quadratic kernel SVM operated on Adaboost selected features is found to be 38% in terms of pixel based performance criteria quality percentage and 48% in terms object based performance criteria correct detection with building detection threshold 0.4.

Adaboost performed better than SVM resulting in 43% quality percentage and 67% correct detection with the same threshold.

Keywords: Building detection, Feature selection, Textural features, Segmentation, Adaboost.

# ÖZ

UYDU GÖRÜNTÜLERİNDEN BİNALARIN ÇIKARILMASI

Çetin, Melih
Yüksek Lisans., Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. A. Aydın Alatan
Ortak Tez Yöneticisi: Prof. Dr. Uğur Halıcı

Mayıs 2010, 197 sayfa

Bu çalışmada, sadece RGB renk bantı kullanarak uydu görüntülerinden binaları tespit eden otonom bir sistem uygulanmıştır. Kullanılan yaklaşımın dört temel aşaması vardır: yerel öznitelik çıkarımı, öznitelik seçimi, sınıflandırma ve sınıflandırma sonrası işlemler. Literatürde bu sorun ile ilgili birçok çalışma vardır. Ana problem bir binayı ayırt etmek için en uygun öznitelikleri belirleyebilmektir. Bu çalışmada Adaboost sınıflandırıcısı tabanlı bir öznitelik seçim yöntemi kullanılmıştır. Sınıflandırılma için, Adaboost'un yanı sıra SVM algoritması dört farklı çekirdek tipiyle kullanılmıştır. Sınıflandırıcıların optimal parametrelerini belirlemek için, çerçeve türü ve boyutu , öznitelik tipi, öznitelik seçimi, öznitelik sayısı ve eğitim seti ile ilgili detaylı analiz yapılmıştır. Piksel ve nesne performanslarına dayalı detaylı bir SVM- Adaboost karşılaştırması yapılarak elde edilen sonuçlar hem sayısal hem görsel olarak sunulmaktadır. Buna göre, SVM, quadratik kernel kullanıldığında doğrusal, rbf veya polinom kernel kullanılan durumlara göre daha iyi performans göstermektedir. Öznitelikler Adaboost ile seçildiğinde veya öznitelik histogramlarından elde edilen hatalar göz önünde bulundurularak seçildiğinde SVM daha iyi performans göstermektedir. Bina algılama eşiği 0,4 iken, Adaboost ile seçilmiş öznitelikler üzerine quadratik çekirdek SVM işletildiğinde elde edilen performans: piksel tabanlı performans kriteri kalite yüzdesi bazında %38 ve nesne tabanlı performans kriteri doğru algılama bazında %48 olarak bulunmuştur. Aynı algılama eşiğinde, Adaboost %43 kalite yüzdesi ve %67 doğru algılama ile SVM’den daha iyi performans göstermiştir.

Anahtar Kelimeler: Bina algılama, Öznitelik seçimi, Dokusal öznitelikler, Bölütleme, Adaboost.

To my family,

To my love

# ACKNOWLEGMENTS

I am most thankful to my supervisor Prof. Dr. A. Aydın Alatan and my co-supervisor Prof. Dr. Uğur Halıcı for sharing their invaluable ideas and experiences on the subject of my thesis.

I would like to extend my thanks to all lecturers at the Department of Electrical and Electronics Engineering, who greatly helped me to store the basic knowledge onto which I have built my thesis.

I am very grateful to TÜBİTAK-SAGE for providing tools and other facilities throughout the production of my thesis.

I would like to forward my appreciation to all my friends and colleagues especially Mehmet Ocaklı, Haluk Erdem Bingöl and Caner Becer who contributed to my thesis with their continuous encouragement.

I would like to express my deep gratitude to my family, who has always provided me with constant support and help. My dear mother, I have always felt your endless love and best wishes with me. My dear father, your kind guidance, cooperation, and constructive criticism have been always enlightened me in establishing my life and career.

Special thanks to my love Mehtap Bilgin for all her help and showing great patience during the writing process of my thesis. I have always felt her endless support with me.

# TABLE OF CONTENTS

ABSTRACT .......... IV

ÖZ .......... VI

ACKNOWLEGMENTS .......... IX

TABLE OF CONTENTS .......... X

LIST OF TABLES .......... XIII

LIST OF FIGURES .......... XVII

LIST OF ABBREVIATIONS .......... XXIII

CHAPTERS

1 INTRODUCTION .......... 1

1.1 Literature Survey on Building Detection .......... 1

1.2 Motivation .......... 10

1.3 Objectives and Goals .......... 12

1.4 Scope of the Thesis .......... 14

1.5 Contribution of the thesis .......... 15

1.6 Organization of the Thesis .......... 16

2 BACKGROUND .......... 17

2.1 Color Models .......... 17

2.1.1 RGB Color Model .......... 17

2.1.2 HSV Color Model .......... 18

2.2 Features ........ 19

2.2.1 Basic Features ........ 19
2.2.2 Zernike Moments ........ 21
2.2.3 Circular-Mellin Features ........ 23
2.2.4 Fourier Power Spectrum ........ 24
2.2.5 Gabor Filters ........ 26
2.2.6 Haralick Features ........ 28
2.2.7 Wavelet Features ........ 29
2.2.8 HSV Color Space ........ 31

2.3 Adaboost Learning Algorithm ........ 31

2.4 Support Vector Machines ........ 34

2.4.1 Kernel Functions ........ 39
2.4.1.1 Polynomial ........ 39
2.4.1.2 Gaussian Radial Basis Function ........ 39

2.5 Morphological Operations ........ 40

2.5.1 Erosion and Dilation ........ 40

2.6 Performance Criteria ........ 41

3 THE PROPOSED APPROACH FOR BUILDING DETECTION ........ 44

3.1 Feature Extraction ........ 46

3.1.1 Basic Features ........ 46
3.1.2 Zernike Moments ........ 47
3.1.3 Circular Mellin ........ 48
3.1.4 Fourier Power Spectrum ........ 48
3.1.5 Gabor filters ........ 49
3.1.6 Haralick Features ........ 51
3.1.7 Wavelet Analysis ........ 52
3.1.8 HSV Domain Features ........ 53

3.2 Feature Selection ........ 54

3.3 Classification ........ 56

3.4 Post-processing ........ 57

4 EXPERIMENTAL RESULTS ........ 58

4.1 Sensitivity analysis ........ 58

4.1.1 Window Size Analysis ........ 65

4.1.2 Feature Analysis ........ 70

4.1.3 Iteration and feature count analysis ........ 75

4.1.3.1 Iteration and feature count analysis for all features ........ 76

4.1.3.2 Iteration and feature count analysis for intensity textural features ........ 94

4.1.3.3 Iteration and feature count analysis for HSV domain features. 98

4.1.3.4 Summary of iteration and feature count analysis ........ 101

4.1.4 Training set analyses ........ 103

4.2 Results ........ 122

4.2.1 Adaboost with intensity textural features ........ 123

4.2.2 Adaboost with all features ........ 136

4.2.3 SVM quadratic kernel with Adaboost selected features ........ 149

4.2.4 SVM quadratic kernel with HSV features ........ 162

4.3 Comparison of classifiers ........ 174

5 CONCLUSIONS ........ 179

REFERENCES ........ 183

APPENDIX A EXAMPLE FEATURE RESULTS ........ 189

# LIST OF TABLES

TABLES

Table 2.1: Zernike Moments and Their Respective Orders ................................ 22
Table 2.2 : Haralick Feature Measures ................................ 29
Table 4.1 : Example of Pixel Performance Values ................................ 63
Table 4.2: SF, MF, PBD and QP Values for 9 Classifier with Intensity Textural Features ................................ 71
Table 4.3: SF, MF, PBD and QP Values for 9 Classifier with HSV Domain Features ................................ 73
Table 4.4: SF, MF, PBD and QP Values for 9 Classifier with All Features ........... 74
Table 4.5: Features Selected by Adaboost over All Feature Set ................................ 78
Table 4.6: Histogram Errors of Features over All Features ................................ 83
Table 4.7: The Features Selected By Adaboost Over Intensity Textural Features ................................ 95
Table 4.8: Summary of Iteration Count Analysis ................................ 102
Table 4.9: Pixel Performance of Adaboost with Intensity Textural Features Using Image 1 in Training ................................ 104
Table 4.10: Object Performance of Adaboost with Intensity Textural Features Using Image 1 in Training ................................ 104
Table 4.11: Pixel Performance of Adaboost with Intensity Textural Features Using Image 2 in Training ................................ 105
Table 4.12: Object Performance of Adaboost with Intensity Textural Features Using Image 1 in Training ................................ 105
Table 4.13: Pixel Performance of Adaboost with Intensity Textural Features Using Image 3 in Training ................................ 106
Table 4.14: Object Performance of Adaboost with Intensity Textural Features Using Image 3 in Training ................................ 106
Table 4.15: Pixel Performance of Adaboost with Intensity Textural Features Using Image 4 in Training ................................ 107

Table 4.16: Object Performance of Adaboost with Intensity Textural Features Using Image 4 in Training ............ 107
Table 4.17: Pixel Performance of Adaboost with Intensity Textural Features Using Image 5 in Training ............ 108
Table 4.18: Object Performance of Adaboost with Intensity Textural Features Using Image 5 in Training ............ 108
Table 4.19: Pixel Performance of Adaboost with Intensity Textural Features Using All Images in Training ............ 109
Table 4.20: Object Performance of Adaboost with Intensity Textural Features Using All Images in Training ............ 109
Table 4.21: Pixel Performance of Adaboost with All Features Using Image 1 in Training ............ 110
Table 4.22: Object Performance of Adaboost with All Features Using Image 1 in Training ............ 110
Table 4.23: Pixel Performance of Adaboost with All Features Using Image 2 in Training ............ 111
Table 4.24: Object Performance of Adaboost with All Features Using Image 2 in Training ............ 111
Table 4.25: Pixel Performance of Adaboost with All Features Using Image 3 in Training ............ 112
Table 4.26: Object Performance of Adaboost with All Features Using Image 3 in Training ............ 112
Table 4.27: Pixel Performance of Adaboost with All Features Using Image 4 in Training ............ 113
Table 4.28: Object Performance of Adaboost with All Features Using Image 4 in Training ............ 113
Table 4.29: Pixel Performance of Adaboost with All Features Using Image 5 in Training ............ 114
Table 4.30: Object Performance of Adaboost with All Features Using Image 5 in Training ............ 114
Table 4.31: Pixel Performance of Adaboost with All Features Using All Images in Training ............ 115
Table 4.32: Object Performance of Adaboost with All Features Using All Images in Training ............ 115

Table 4.33: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 1 in Training ........................................................116
Table 4.34: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 1 in Training ........................................................116
Table 4.35: Pixel Performance of SVM Quadratic with Adaboost Selected Features Using Image 2 in Training ........................................................117
Table 4.36: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 2 in Training ........................................................117
Table 4.37: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 3 in Training ........................................................118
Table 4.38: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 3 in Training ........................................................118
Table 4.39: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 4 in Training ........................................................119
Table 4.40: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 4 in Training ........................................................119
Table 4.41: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 5 in Training ........................................................120
Table 4.42: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 5 in Training ........................................................120
Table 4.43: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using All Images in Training ....................................................121
Table 4.44: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using All Images in Training ....................................................121
Table 4.45: Pixel Performance of the 1. Set....................................................123
Table 4.46: Object Performance of the 1. Set For T = 0.4................................124
Table 4.47: Pixel Performance of the Images in 2. Set ....................................130
Table 4.48: Object Performance of the 1. Set for T = 0.4.................................131
Table 4.49: Pixel Performance of the 1. Set....................................................137
Table 4.50: Object Performance of the 1. Set for T = 0.4.................................138
Table 4.51: Pixel Performance of the 2. Set....................................................143
Table 4.52: Object Performance of the Images in 2. Set..................................144
Table 4.53: Pixel Performance Values of Image 1 to Image 5 ..........................150
Table 4.54: Object Performance of the 1. Set for T = 0.4.................................151

Table 4.55: Pixel Performance of the Images in 2. Set ..................................... 155
Table 4.56: Object Performance of the 2. Set for T = 0.4 .................................. 156
Table 4.57: Pixel Performance of the 1. Set ...................................................... 162
Table 4.58: Object Performance of the 1. Set for T = 0.4 .................................. 163
Table 4.59: Pixel Performance of the Images in 2. Set ..................................... 168
Table 4.60: Object Performance of the 2. Set For T = 0.4 ................................. 169
Table 4.61: Pixel performances of the selected classifiers over image set 1 ..... 175
Table 4.62: Object Performances of the Selected Classifiers over Image Set 1 (numbers given in brackets are obtained by dividing by ground truth) ........ 176
Table 4.63: Pixel Performances of the Selected Classifiers over Image Set 2... 177
Table 4.64: Object Performances of the Selected Classifiers over Image Set 2 (numbers given in brackets are obtained by dividing by ground truth) ........ 177
Table A.1: Basic Features .................................................................................... 189
Table A.2: Zernike Moments ............................................................................... 190
Table A.3: Circular Mellin Features ..................................................................... 191
Table A.4: Fourier Power Spectrum Features .................................................... 192
Table A.5: Gabor Features .................................................................................. 193
Table A.6: Haralick Features ............................................................................... 194
Table A.7: Wavelet Features ............................................................................... 196
Table A.8: HSV Domain Features ....................................................................... 197

# LIST OF FIGURES

FIGURES

Figure 2.1: Rgb Color Cube.....17
Figure 2.2: The HSV Color Model.....18
Figure 2.3: Ring and Wedge Shaped Regions On Frequency Domain.....25
Figure 2.4: Comparison of Complex and Real Gabor Filters. Complex Gabor Filters are not Symmetric over Frequency Domain While Their Real Counterparts are.....28
Figure 2.5: Single-Level Wavelet Decomposition.....30
Figure 2.6: Multi-Level Wavelet Decomposition.....30
Figure 2.7: Adaboost Algorithm.....33
Figure 2.8: (a) Linear Classifiers and (b) Best Linear Classifier.....35
Figure 2.9: SVM with Maximum Margin.....36
Figure 2.10: Kernel Function Mapping.....38
Figure 2.11: (a) Images of Squares of Size 1,3,5,7,9,15 Pixels on the Side. (b) Erosion of (a) with a Structuring Element of 1's,13 Pixels on the Side. (c) Dilation of (b) with the Same Structuring Element.....41
Figure 3.1: The Basic Block Schema Of The Proposed Building Detection Approach.....44
Figure 3.2: The Detailed Block Schema of the Proposed Building Detection Approach.....45
Figure 3.3: Example Histogram for Calculating Histogram Errors.....55
Figure 3.4: Algorithm for Computation of the Errors.....56
Figure 4.1: 1. Image (Quickbird Image of Resolution 0.6 M).....59
Figure 4.2: 2. Image (Quickbird Image of Resolution 0.6 M).....59
Figure 4.3: 3. Image (Quickbird Image of Resolution 0.6 M).....60
Figure 4.4: 4. Image (Quickbird Image of Resolution 0.6 M).....60
Figure 4.5: 5. Image (IKONOS Image of Resolution 1 M).....61

Figure 4.6: Example of Object Detection Performance: a) Ground Truth Objects of Example Image 1, b) Classifier Output for Image 1, c) Ground Truth Objects of Example Image 2, d) Classifier Output for Image 2 .................................. 64
Figure 4.7: Non-Overlapping Window; QP vs. Window Size Using All Features.. 66
Figure 4.8: Non-Overlapping Window; QP vs. Window Size Using Features Selected by Adaboost .......................................................................... 67
Figure 4.9: Non-Overlapping Window; QP vs. Window Size Using Features with Minimum Histogram Errors........................................................................ 67
Figure 4.10: Overlapping Window; QP vs. Window Size Using All Features ....... 68
Figure 4.11: Non-Overlapping Window; QP vs. Window Size Using Features Selected by Adaboost .......................................................................... 69
Figure 4.12: Non-Overlapping Window; QP vs. Window Size Using Features with Minimum Histogram Errors........................................................................ 69
Figure 4.13: QP and PBD Values vs. Iteration Count for Adaboost with All Features .......................................................................................... 81
Figure 4.14: Object Performance vs. Iteration Count for Adaboost with All Features .......................................................................................... 81
Figure 4.15: Pixel Performance of SVM Linear Kernel Features Selected Using Histogram Errors over the Entire Set Of Features ....................................... 84
Figure 4.16: Object Performance of SVM Linear Kernel Features Selected Using Histogram Errors over the Entire Set Of Features ....................................... 85
Figure 4.17: Pixel Performance of SVM Linear Kernel Features Selected Using Adaboost Over the Entire Set of Features .................................................... 85
Figure 4.18: Object Performance of SVM Linear Kernel Features Selected Using Adaboost Over the Entire Set of Features .................................................... 86
Figure 4.19: Pixel Performance of SVM Polynomial Kernel Features Selected Using Histogram Errors over the Entire Set of Features ............................... 87
Figure 4.20: Object Performance of SVM Polynomial Kernel Features Selected Using Histogram Errors over the Entire Set of Features ............................... 87
Figure 4.21: Pixel Performance of SVM Polynomial Kernel Features Selected Using Adaboost over the Entire Set of Features .......................................... 88
Figure 4.22: Object Performance of SVM Polynomial Kernel Features Selected Using Adaboost over the Entire Set of Features .......................................... 88

Figure 4.23: Pixel Performance of SVM RBF Kernel Features Selected Using Histogram Errors over the Entire Set of Features ........................................ 89
Figure 4.24: Object Performance of SVM RBF Kernel Features Selected Using Histogram Errors over the Entire Set of Features ........................................ 90
Figure 4.25: Pixel Performance of SVM RBF Kernel Features Selected Using Adaboost over the Entire Set of Features .................................................... 90
Figure 4.26: Object Performance of SVM RBF Kernel Features Selected Using Adaboost over the Entire Set of Features .................................................... 91
Figure 4.27: Pixel Performance of SVM Quadratic Kernel Features Selected Using Histogram Errors over the Entire Set of Features ............................... 92
Figure 4.28: Object Performance of SVM Quadratic Kernel Features Selected Using Histogram Errors over the Entire Set of Features ............................... 92
Figure 4.29: Pixel Performance of SVM Quadratic Kernel Features Selected Using Adaboost Over the Entire Set of Features .......................................... 93
Figure 4.30: Object Performance of SVM Quadratic Kernel Features Selected Using Adaboost Over the Entire Set of Features .......................................... 93
Figure 4.31: QP and PBD Values vs. Iteration Count for Adaboost with Intensity Textural Features ........................................................................... 97
Figure 4.32: Object Performance vs. Iteration Count for Adaboost with Intensity Textural Features ........................................................................... 98
Figure 4.33: Pixel Performance for Adaboost with HSV Domain Features .......... 99
Figure 4.34: Object Performance for Adaboost with HSV Domain Features........ 99
Figure 4.35: Pixel Performance of SVM Quadratic Kernel Features Selected Using Histogram Errors over the Set Of HSV Domain Features................. 100
Figure 4.36: Object Performance of SVM Quadratic Kernel Features Selected Using Histogram Errors over the Set Of HSV Domain Features................. 101
Figure 4.37: Object Performance vs. Object Detection Threshold ..................... 124
Figure 4.38: Visual Results for Adaboost with Intensity Textural Features for 1. Image ........................................................................................... 125
Figure 4.39: Visual Results for Adaboost with Intensity Textural Features for 2. Image ........................................................................................... 126
Figure 4.40: Visual Results for Adaboost with Intensity Textural Features for 3. Image ........................................................................................... 127

Figure 4.41: Visual Results for Adaboost with Intensity Textural Features for 4. Image ........ 128
Figure 4.42: Visual Results for Adaboost with Intensity Textural Features for 5. Image ........ 129
Figure 4.43: Object Performance vs. Object Detection Threshold ........ 130
Figure 4.44: Visual Results for Adaboost with Intensity Textural Features for 6. Image ........ 132
Figure 4.45: Visual Results for Adaboost with Intensity Textural Features for 7. Image ........ 133
Figure 4.46: Visual Results for Adaboost with Intensity Textural Features for 8. Image ........ 134
Figure 4.47: Visual Results for Adaboost with Intensity Textural Features for 9. Image ........ 135
Figure 4.48: Visual Results for Adaboost with Intensity Textural Features for 10. Image ........ 136
Figure 4.49: Object Performance vs. Object Detection Threshold ........ 137
Figure 4.50: Visual Results for Adaboost with All Features for 1. Image........ 139
Figure 4.51: Visual Results for Adaboost with All Features for 2. Image........ 140
Figure 4.52: Visual Results for Adaboost with All Features for 3. Image........ 141
Figure 4.53: Visual Results for Adaboost with All Features for 4. Image........ 142
Figure 4.54: Visual Results for Adaboost with All Features for 5. Image........ 143
Figure 4.55: Object Performance vs. Object Detection Threshold ........ 144
Figure 4.56: Visual Results for Adaboost with All Features for 6. Image........ 145
Figure 4.57: Visual Results for Adaboost with All Features for 7. Image........ 146
Figure 4.58: Visual Results for Adaboost with All Features for 8. Image........ 147
Figure 4.59: Visual Results for Adaboost with All Features for 9. Image........ 148
Figure 4.60: Visual Results for Adaboost with All Features for 10. Image........ 149
Figure 4.61: Object performance vs. object detection threshold ........ 150
Figure 4.62: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 1. Image ........ 152
Figure 4.63: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 2. Image ........ 152
Figure 4.64: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 3. Image ........ 153

Figure 4.65: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 4. Image ..... 154
Figure 4.66: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 5. Image ..... 155
Figure 4.67: Object Performance vs. Object Detection Threshold ..... 156
Figure 4.68: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 6. Image ..... 157
Figure 4.69: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 7. Image ..... 158
Figure 4.70: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 8. Image ..... 159
Figure 4.71: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 9. Image ..... 160
Figure 4.72: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 10. Image ..... 161
Figure 4.73: Object Performance Vs. Object Detection Threshold ..... 163
Figure 4.74: Visual Results for SVM Quadratic Kernel with HSV Features for 1. Image ..... 164
Figure 4.75: Visual Results for SVM Quadratic Kernel with HSV Features for 2. Image ..... 165
Figure 4.76: Visual Results for SVM Quadratic Kernel with HSV Features for 3. Image ..... 166
Figure 4.77: Visual Results for SVM Quadratic Kernel with HSV Features for 4. Image ..... 167
Figure 4.78: Visual Results for SVM Quadratic Kernel with HSV Features for 5. Image ..... 168
Figure 4.79: Object Performance vs. Object Detection Threshold ..... 169
Figure 4.80: Visual Results for SVM Quadratic Kernel with HSV Features for 6. Image ..... 170
Figure 4.81: Visual Results for SVM Quadratic Kernel with HSV Features for 7. Image ..... 171
Figure 4.82: Visual Results for SVM Quadratic with HSV Features for 8. Image ..... 172

Figure 4.83: Visual Results for SVM Quadratic with HSV Features for 9. Image ........................................................................................................ 173
Figure 4.84: Visual Results for SVM Quadratic Kernel with HSV Features for 10. Image ........................................................................................... 174

# LIST OF ABBREVIATIONS

| | |
|---|---|
| Adaboost | Adaptive Boosting |
| RGB | Red-Green-Blue |
| HSV | Hue Saturation Brightness |
| BABE | Built-up Area Building Extraction Based On Edges and Corners |
| SHADE | Shadow Detection |
| SHAVE | Shadow Verification |
| DSM | Digital Surface Model |
| ML | Maximum Likelihood |
| MRF | Markov Random Field |
| KMC | K-means Clustering |
| PCA | Principle Component Analysis |
| NDVI | Normalized Difference Vegetation Index |
| LIDAR | Light Detection And Ranging |
| SOM | Self Organizing Maps |
| SVM | Support Vector Machine |
| ATR | Automatic Target Recognition |
| UAV | Unmanned Air Vehicle |
| DWT | Discrete Wavelet Transform |
| OAM | The Overlapping Area Matrix |
| DFT | Discrete Fourier Transform |
| TP | True Positive |
| FP | False Positive |
| FN | False Negative |
| MF | Missing Factor |
| QP | Quality Percent |
| PBD | Percent of Building Determination |

# CHAPTER 1

# INTRODUCTION

In this chapter, first the previous studies in the field of building detection and extraction in digital satellite/aerial images are described briefly. Then motivation of thesis, objective and goals, scope of the thesis, contribution of the thesis and organization of the thesis is given.

## 1.1 Literature Survey on Building Detection

Building extraction has attracted many researchers' attention for many years. Extraction of buildings automatically from satellite/aerial imagery involves several different problems related to computer vision since in urban areas there are many other objects in close proximity such as trees, power lines vehicles, and parking lots. These problems make the automatic building extraction is a challenging problem.

Image segmentation is one of the important tasks in computer vision, and many fields of application are concerned with it, including robotics, remote sensing, medical imaging, etc. The prime objective of segmentation is to produce a partition of an image into homogeneous regions that one hopes to correspond to objects or part of objects in the real scene under study [1]. Building extraction is a popular application of image segmentation and it is widely used in numerous fields.

Extraction of buildings is not only challenging but also it is important for update of GIS systems. Therefore, many researchers focused on developing robust automatic building detection algorithms in satellite images. Mayer [2] surveyed object detection systems from aerial images, focusing on building detection.

Konecny and Schiewe [3] put emphasis on automatic city map generation. According to them, 33.5% of the world was mapped at 1:25,000 scale (around one meter per pixel resolution) as of 1993. This resolution is extremely important for developed crowded cities. The rate of manual map generation and updating is respectively 2.8% and 4.9%. It takes nearly 20 years for remapping the same area but a house can be built and destroyed in a very short time period compared with map generation rate. The current map update rates are inadequate but making this update with commercially available IKONOS images takes less than a month. Here the data collection rate exceeds the data process rate. That is automatic building detection is at high importance. As an example in [4], researchers automatically detected the unlicensed buildings that are on the forbidden areas of city İstanbul. At the time being these map generation issue is done by post-processing the images taken from satellite images but it will be possible in near future that the generation of maps, detection of illegal buildings etc. may be done in real time and changes in the man-made structures are easily recognized with the improvement of imaging sensors and developed algorithms.

Numerous approaches for building extraction have been proposed in the literature. Huertas and Nevatia [5] proposed a method using line, corner and shadow information. Their method works on medium density places and they have an assumption that buildings are rectangular or composed of rectangular components thus “box”, “T”,  “L”, ”E” are allowed. Line segments near 90 degree L-junctions are used to form rectangles.  They use height information obtained from shadow for verifying building hypothesis and disambiguating raised structures from structures on the ground (such as parking lots). The main problem of the method is due to the real structures in the image for example sides of buildings may have trees, vehicles and road markings. In summary, their method is composed of detecting lines and corners, labeling corners based on shadows, tracing object boundaries and verifying hypotheses using shadow information. In [6] similar to [5] used the same concepts and add a method called FUSION in order to reduce the miss detections. The method is based on taking the results of detection results of BABE (builtup area building extraction based on edges and corners), SHADE (shadow detection), SHAVE (shadow verificaiton) and

GROUPER (BABE and back projection of building/shadow edges) and fusing the results of them. Noranha and Nevatia [7] used multiple views in a similar method and increased the detection rate. Krishnamachari and Chellappa [8] used Markov Random Field (MRF) models. They propose a method to extract straight lines from edge map of an image and then MRF model is used to group these lines.

In [9], researchers used snake-based approach to extract 2D building outlines from high-resolution IKONOS satellite images and height data captured by airborne laser scanning system. A semi-automated approach is used in [10] based on active contour model (snakes) and the dynamic programming optimization technique. The method requires a digital surface model and an orthonormal image. This approach can be more effective if applied after a human operator has manually determined seed points near the boundary of a desired feature. Peng et al. [11] extract the principal contours of buildings in dense urban areas according to the radiometric behavior of buildings.

In [12] Baltsavias et al. used digital surface model (DSM) and three different approaches namely, edge operator, mathematical morphology and height bins for building detection. Brunn and Weidner [13] used normalized DSM with a threshold first in order to segment higher areas and used area thresholding for discarding single trees. In order to discriminate large areas of trees, they used geometric information of the DSM like curvature and gradient.

With the advance of imaging systems and the availability of high spatial and spectral resolution satellite images, most of the studies focus on the use of spectral reflectance values. In [14], Sha and Lee classified the IKONOS multispectral images into several classes, including water, road, roof, tree, marsh, grass and sand based on the selected training sites by using a Maximum Likelihood (ML) classifier. The resultant class maps are then vectorized to feature classes. Each vectorized building object will be used to define the searching area on the corresponding panchromatic image to delineate the building boundaries.

Ünsalan and Boyer [15] introduce measures on multispectral to detect regions of possible human activity. On these measures, they introduce a variation of the k-means clustering (KMC) algorithm to extract possible houses and street networks by combining both spatial and spectral features. This combination of information improves the final clustering results. From clustering, they obtain a binary image containing possible street network fragments and houses then decompose this binary image using a balloon algorithm based on binary mathematical morphology and represent this decomposition in a graph for which balloons serve as vertices, while their neighborhood relationships are encoded as edges.

In [16], Zhang merged multispectral classification with texture filtering. This is because objects in urban areas are very complicated with respect to both their spectral and spatial characteristics. Multispectral classification detects object classes only according to the spectral information of the individual pixels, while a large amount of spatial information is neglected. In this study, a technique is described which attempts to detect urban buildings in two stages. The first stage is a conventional multispectral classification. In the second stage, the classification of buildings is improved by means of their spatial information through a modified co-occurrence matrix based filtering.

Stassopouloua and Caelli [17] proposed a new system, which differs from standard building detection algorithms in various ways. It uses performance optimization to train the system to fit how humans identify corners, and combine diverse data, raw or partly processed, in a probabilistic framework to assign probabilities to hypotheses. Information sources relevant to the detection of a building include geometric (anything relating to its shape), radiometric (using its spectral properties) and contextual (using the strong road building relationship).

Kim et al. [18] used moments and Fourier descriptors to recognize buildings from satellite images. The Fourier descriptors and moment features (Zernike moments and Hu's moment invariants) are used as input vectors to the neural network classifier. Gilmore and Boyd [19] also utilized Hu’s well-known seven moment invariants to identify building and bridge targets with infrared imagery.

Wavelet transformation and multi-scale analysis have also been attempted in extracting building edges. Levitt and Aghdasi [20] applied wavelets and used scale-space for the extraction of buildings from monocular greyscale aerial photography. The wavelet transform extracts building edges corresponding to high frequencies while scaling provides abstraction by eliminating such frequency components. An attractive feature of the techniques investigated in [20] is that they generate a family of images at different scales and resolutions based on the original source image.

Qu et al. [21] proposed a salient building detection system using a single nature image via wavelet decomposition. In their work, they use Haar wavelet decomposition to obtain the enhanced image and then they separate the candidates of building from the background based on projection profile. Finally, they discriminate building regions by Principle Component Analysis (PCA) in RGB color space.

Bellman and Shortis [22] used Support Vector Machines to classify wavelet coefficients as a building or non-building object. The images are characterized using wavelet analysis. Selvarajan and Tat [23] used wavelet filter banks in two levels. In the first level, they constructed an edge map. In the second level, they constructed a region of interest map by using local intensity variations.

Sırmaçek and Ünsalan [24] proposed an automated approach for building detection based on Gabor filters and spatial voting. They extract features (representing buildings) using Gabor filter responses. Using these features, they form a spatial voting matrix to detect buildings. Lacroix et al. [25], extracted edges of man-made structures (buildings and roads) using Gabor filters together with the NDVI (Normalized Difference Vegetation Index) in SPOT5 images. Comparing their edges of two image sequences taken from same region, they detected changes.

An alternative solution for building detection is the analysis of texture pattern. Zimmermann [27] used well-known definitions of texture from Haralick at al. [26]

and derived six texture parameters: contrast, correlation, direction, entropy, homogeneity and uniformity. According to Zimmermann, pure texture segmentation gives a only a coarse segmentation, so they used texture segmentation only as auxiliary tool to check color segmentation and get texture parameters for the segmented regions.

Both unsupervised classification and supervised classification techniques were applied to building detection problems. In many cases, better results were obtained using supervised classification techniques [28 and 29]. Brunn [30] (2001) devised a statistical approach to building detection in range and image data using Bayesian nets. Bartels and Wei (2006) [31] performed a pixel based supervised classification algorithm based on Bayesian maximum likelihood approach using high resolution first, last echo and intensity LIDAR data and co-registered line scanner bands such as aerial photos and near infra-red photos. The Bayesian method was also employed by Maas (1999) [28] to fuse various height texture measures extracted from laser range data for the detection of buildings and trees.

Khoshelham et al. (2005) [32] developed a method to fit planar surfaces to height data within regions of a segmented aerial image for the detection of building roofs. This method is based on dividing image regions whose related height points do not fall in a single plane, and merging coplanar neighboring regions. A plane-fitting method is used to fit planar surfaces to height points.

Walter (2004) [33] applied a Bayesian maximum likelihood method to object-based classification of multi-spectral aerial data. The multispectral bands grouped by objects and very different measures that can be derived from multispectral bands represent the n-dimensional feature space for the classification. Different input channels for the classification are defined and discussed.

Rottensteiner et al. (2004) [34] and Lu et al. (2006) [35] developed methods to extract buildings from aerial imagery and laser range data based on the Dempster-Shafer theory of evidence (Shafer, 1976) [36].The Dempster-Shafer data fusion technique is applied to detect buildings from the combination of multiple data sources.

Elaksher and Bethel [37] used Neural Network to distinguish roof regions from non-roof regions. Two attributes is implemented for the classification process. The first one measures the linearity, and the second one measures the percentage of the points in the region that are higher than a threshold.

Neural networks are capable of scale and rotation invariant matching of predefined neuron graphs to images [38]. The whole operation is done in two phases. First, the neurons are trained in specifically selected data and then they are ready to detect what they were trained for in other data. Another neural network based building detection system that also Lari and Ebadi [39] proposed works in two different phases. In the first phase, the presented neural network in the system is trained with the aid of test data, and in the second phase, the system will be used for detection and extraction of buildings from satellite images. The neural network used in this system is a three-layer perceptron. A specific weight is determined for each of input values. This network functions well if all weight coefficients are truly selected. In training process of neural network, this coefficient will be modified. In this system, initial image processing stage is implemented at first and followed by image segmentation procedure. Then suggested features are calculated for each region and a three-layer perceptron neural network is trained for detection of buildings in satellite images.

Information about the shape and location of buildings is also used to improve mobile robot outdoor mapping. Persson et. al. [40] describes a system for automatic detection of buildings in aerial images taken from a nadir view. The system builds two types of independent hypotheses based on the image contents. A segmentation process implemented as an ensemble of SOMs (Self Organizing Maps) which is trained and used to create a segmented image showing different types of roofs, vegetation and sea. A line extraction process uses the edge image as input and extracts lines from it. From these edges, corners and rectangles that represent buildings are constructed. A classification process uses the information from both hypotheses to determine whether the rectangles are buildings, unsure buildings or unknown objects. The SOM is an unsupervised neural network that clusters similar data into similar categories. Lee and Lathrop [41] also used self-organizing map neural networks for urban land cover characterization.

Bellman and Shortis [42] presents the results of an investigation into the use of machine learning in the form of a support vector machine. The images are characterized using wavelet analysis to provide multi-resolution data for the machine-learning phase. A public domain Support Vector Machine [44] was used to classify the image patches into building or non-building categories. Bruzzone and Carlin [43] proposed a pixel based system to classify very high-resolution satellite images. They used support vector machines fed with al feature extractor.

Brunner and Burkhardt [45] proposed a geometric feature extraction method based on a special weighted hierarchical cluster analysis. The proposed features capture the intrinsic interrelation-ships of line segments, containing a high discriminative power verified by support vector machines with different kernel functions. They compared their proposed features with the well-established edge-orientation histogram feature. The results proof that our features possess higher discrimination ability for the class of buildings.

Although SVMs have been widely used in building detection, Adaboost has not. However, as Adaboost can select informative features from a potentially very large feature pool, it is likely to offer advantages in automatically finding good features for classification. This can greatly reduce, or eliminate the need for experts to choose informative features based on knowledge of every classification problem. Instead, one just needs to define a list of possibly informative features, and Adaboost will choose those that are actually informative.

Zingaretti et al. (2007) [46] employed an adaptive boosting algorithm (Adaboost) [52] for the automated identification of classification rules. Adaboost algorithm is used for the identification of rules for the classification of raw LIDAR data mainly as buildings, ground and vegetation. First raw data are filtered, interpolated over a grid and segmented. Then geometric and topological relationships among regions resulting from segmentation constitute the input to the tree-structured classification algorithm.

In a previous study, Maloof et al. [47] evaluated a variety of machine learning methods on the rooftop detection task. Khoshelham et al. (2010) [29] presents a comparative analysis of different methods for automated building detection in aerial images and laser data at different spatial resolutions. Three classification methods: Bayesian (both maximum likelihood and minimum distance classifiers), Dempster-Shafer and Adaboost are compared with the normalized DSM method. These five methods are tested in two study areas using features extracted at both pixel level and object level. The results showed a better performance of the Dempster-Shafer method followed by the Adaboost in both study areas. The Dempster-Shafer method reached an overall accuracy of about 97% in the Mannheim study area. Both the Dempster-Shafer and the Adaboost method also yielded higher rates of unclassified pixels. The method of thresholding a normalized DSM (nDSM) reached a detection rate of 94.5% in the less vegetated Mannheim study area, but also yielded a high false positive error rate of 10.7%. The Bayesian methods perform reasonably well (with an overall accuracy above 94%) in the Memmingen area where buildings have more or less the same heights. In both study areas, most of the errors were found at building boundaries and in areas where dense trees were present.

When it comes to the robotic area, many real time systems use building detection. Persson et al. [48] propose a new system to process an aerial photo. The aerial photo is a color photograph taken from the air, for example by an UAV, from a nadir view. The building detection process is done in real time on board. They have promising results for detection of buildings and they plan to connect the output of the system to navigation sensors in order to make modeling with position information.

Ioannidis et al. [49] mention the projects of fully automatic 3D map generation of a small area in Delhi, India. The automated system monitors the construction activities using high-resolution satellite imaging and a special multimedia mapping to build a 3D map with live cameras.

Kontitsis et al. [50] propose a machine vision system for aerial surveillance that can interpret and process data acquired by a UAV on-board infrared camera. System components include noise reduction, feature extraction, classification and decisionmaking. Decision-making is performed in terms of an alarm signal for fire detection.

Fitzgerald at al. [51] present the evolution and status of a number of research programs focused on developing an automated fixed wing UAV landing system. An emergency or forced landing (in the case of an unpowered landing), is where the aircraft is required to perform an unplanned landing due to the occurrence of some onboard emergency (eg: an engine failure). Therefore, the objective of this research is to develop an onboard capability that allows the UAV to select a suitable landing site then maneuver the UAV to land at this location autonomously. If this functionality is realized, it will bring UAVs one-step closer to flying in civilian airspace above populated areas.

## 1.2 Motivation

Object detection from satellite images has been an important research topic in computer vision for many years. Satellite images give valuable information to monitor urbanization and building construction. Land planners and government agencies need to update their maps, but doing this manually is very time consuming. Although more details are visible with the improved resolution of satellite images, the building detection is still difficult. Main reasons are the denseness and the complexness of the scene. Therefore, many researchers focused on developing robust automatic building detection algorithms in satellite images. This is the basic motivation that leaded us to study on building detection in satellite images.

Some useful applications of this subject are; updating of geographic information system (GIS) databases, urban city planning and land use analysis. With the availability of high-resolution satellite imagery, classification and detection of small-scale manmade structures therefore has been drawing great interest. The fundamental challenges that drive much of the research in this field are the edge or line extraction problems and segmentation problem.

Pixel-based methods like using local descriptors are popular tools in object recognition. They are robust to occlusion and global geometric deformations. There is a large number of features that can be used for pixel-based local image descriptor. The most important issue in this field is to select the most valuable features from a set of candidates to keep the classification efficient and reliable.

When the building detection algorithms are examined, it can be seen that several features and classifiers are utilized. It is not possible to distinguish which features have better separation capacity and which methods for classification are better since different studies use different data sets and performance criteria. For example in some studies, performance evaluation is done only checking correct detection a few number of predefined pixels, while some uses object base performance evaluation such as the detection of buildings in a given ratio and evaluate the performance over all buildings.

Subsequently, another motivation beyond this study is to select most convenient features and use only them in classification. Curse of dimensionality is a major problem in classification methods based on machine learning. If the feature vector size is large, classification performance decreases. In order to reduce the feature vector size subspace transformations like principal component analysis (PCA) are widely used. In order to use PCA all the features must be extracted which is very expensive in terms of computation time. If valuable features can be predetermined, the rest of features can be eliminated easily which saves computation time. Adaboost is a machine learning based classification algorithm which became very popular in recent years in pattern recognition area due to its feature selection property. Adaboost algorithm can select valuable features over a large feature set and based on these features it can make classification by the help of weak classifiers based on selected features.

When Building Detection literature is examined, it is observed that support vector machine (SVM) algorithm is widely used recently in building detection problems because of its superior performance; however, the usage of Adaboost algorithms for solving this problem is quite rare. Only two studies could be found in the literature. One makes classification of urban areas and rural areas, which is not directly related to building detection. The other one is a very recent study published in 2010 [29], which uses LIDAR information for detection of buildings. This type of satellite image data is different from the type of satellite images that we aimed to use. Google Earth™ is a virtual globe, map and geographic information program that display satellite images of varying resolution of the Earth's surface. For large parts of the surface of the Earth only 2D RGB images are available, from almost vertical photography.

Despite buying a multi-spectral, stereo satellite image or LIDAR data is expensive, images in Google Earth Application, are permitted to be used free for research purposes. It is also available under different licenses when to be used commercially. Already several applications based on Google Earth images have been emerged on Internet. This is the motivation, leading us to use only RGB color band in building detection, which is compatible with Google Earth images.

In this thesis, the answers for which intensity textural features and HSV domain features have more separation capacity, the effect of using these features combined or single, the effect of selection of features, and the success of two classifiers SVM and Adaboost are investigated.

Adaboost learning algorithm is employed for both classification and determining the beneficial feature subset, due to its feature selector nature. Also, SVM performance with and without feature selection methods are examined.

## 1.3 Objectives and Goals

In this thesis, it is aimed to develop a robust, reliable building detection algorithm, which is capable of detecting buildings from a single satellite image using only RGB color band. For this purpose, an automated building extraction algorithm based on local feature extraction, feature selection, classification and post processing is to be developed.

The performance of the building classifier is directly related to the extracted features. To get a robust and reliable building classifier, we need good features to distinguish building pixels from non-building pixels. Therefore, feature selection is one of the most important issues for a good performance.

In literature quite diverse type of features are used for building detection, however to compare their performance is problematic because of differences in data and also performance criteria used. An approach could be to use all the features, or to reduce the feature dimension using subspace transformations for an efficient classification. However, such an approach is time consuming since it requires all the features to be extracted, before dimension reduction. In contrary, we are aiming to increase classification performance while also decreasing the time required for feature extraction. Therefore, it becomes necessary to decide on a feature selection scheme to select valuable features from a set of candidates to keep the classification efficient and reliable while reducing the time required for feature extraction.

Adaboost algorithm, due to its feature selection property, is a potent candidate for selecting valuable features. Beside its feature selection property, Adaboost is also a classifier that can be used for building detection. For being able to compare its performance another feature selection criteria, which is histogram based errors, are used for selecting most valuable features from a large set of local image features. As classifier beside Adaboost, SVM, which is shown to be quite powerful in many pattern recognition applications and also used efficiently in building detection, is considered in this study for comparison purposes.

Consequently, in this thesis it is also aimed to see the effect of window size, window type, and using intensity textural and/or HSV domain features extracted from these windows. In addition, the effect of feature selection by Adaboost and by histogram errors is examined. Investigation of the effect of feature count on the performance and comparison of the performance of SVM kernels and Adaboost are done through meaningful performance criteria. Therefore, we will try to form the best classifier with determining the optimal parameters and comparing four SVM kernels and Adaboost.

## 1.4 Scope of the Thesis

In this thesis, satellite nadir RGB images are used as input data for both classification and learning phases. The expected outputs of the system are a set of best features selected for building classification and the segmented regions in the image that represents buildings in the image.

As mentioned in the objectives section, an important objective of this work is to present a feature selection scheme where the features are calculated locally for each pixel. Therefore, we do not consider any global image features and concentrate on local features. In addition to that, one can also extract additional information from a multi-spectral image like vegetation. Vegetation information can be extracted by using Normalized Difference Vegetation Index (NDVI). However, in this work we design a building classification that can be applied to any RGB satellite image. We also considered color image features but the system can be easily modified to grayscale mode by ignoring these color-based features.

As described in literature survey part, many researchers use DEM information or extract the height information using stereo images. However, in this thesis, we focus on building detection from a single colored image. Therefore, using a single image for building detection is another constraint that should be considered. In addition to that, the input images are selected such that they are widely used and easily obtained.

Local descriptors are popular tools in object recognition. They are robust to occlusion and global geometric deformations. For each window, a set of local feature descriptors are used in order to detect building pixels. Whereas pixel-based evaluation gives estimates of the area that is correctly classified, the results are distorted by errors at the building outlines. On the other hand, object-based evaluation techniques are less affected by such errors. However, object-based techniques need a segmentation step, which is finding a desired object and separating it from the background in the presence of distortions caused by other features such as surface markings, vegetation, shadows, and highlights. Therefore, the performance of the detection system is directly related to accuracy of the segmentation results. Image segmentation is an important research topic in computer vision and researchers has been working to find robust and reliable image segmentation methods for years. Another method that is commonly used is edge or line extraction based methods. Since edge based methods are more sensitive to these distortions, edge based methods are discarded in this work. As a result, we select to work on pixel-based evaluation techniques for building detection using local features. These local features are input to two classifiers which are Adaboost and SVM. Adaboost selects its own features and uses them for classification. For SVM kernels, the features selected by Adaboost and by histogram errors are used. No dimension reduction methods such as principal component analysis (PCA) are implemented since these methods use all feature set and do the dimension reduction. We want to get rid of the cost of extracting all features.

## 1.5 Contribution of the thesis

The main contribution of this work is to present a feature selection scheme that is connected with the classification framework of Adaboost and SVM with selected valuable features. We want to select the most valuable features from a set of candidates to keep the classification efficient and reliable. The performance of the Adaboost feature selection method is compared with histogram based error method to clarify the contribution of the proposed method. We also examined the distribution of the considered local image features for building and non-building images, which will be a useful tool for researchers working on local descriptors for building detection. By examining the distribution of the local features and Adaboost results (assigned weights and thresholds) one can select the most valuable features from a set of candidates according to his/her own needs.

Although feeding SVM kernels by the features selected by Adaboost was studied previously for a different problem, which is face detection, it was not used for building detection problem. We used this approach and also compare the classification performance of the Adaboost algorithm with SVM classifier using the features selected by both Adaboost and histogram-errors.

In this thesis a detailed sensitivity analysis including window size, window type, feature type, feature count and training set are done. The sensitivity analysis gives information about selecting critical parameters for both training and evaluation of a building classifier. Also, performance of the proposed system is evaluated using object and pixel based measures including visual outputs.

## 1.6 Organization of the Thesis

The rest of the thesis chapters are organized as follows; Chapter 2 is devoted to background information on the buildings detection subject and the algorithms used throughout this study. In Chapter 3, the method used in this study is explained in detail. Chapter 4 covers the sensitivity analysis and experimental results obtained from test cases. Finally, Chapter 5 concludes the study.

# CHAPTER 2

# BACKGROUND

This chapter is aspired to present the algorithms and clarify the means used throughout the thesis. In Section 2.1, color models are given. In Section 2.2 features used through the work is introduced. In Section 2.3, Adaboost algorithm is described. In Section 2.4, support vector machines are introduced. In Section, 2.5 morphological operations are covered. In Section 2.6, performance criteria used in this work is described.

## 2.1 Color Models

### 2.1.1 RGB Color Model

The RGB color model employs a Cartesian coordinate system and forms a unit cube shown in Figure 2.1.

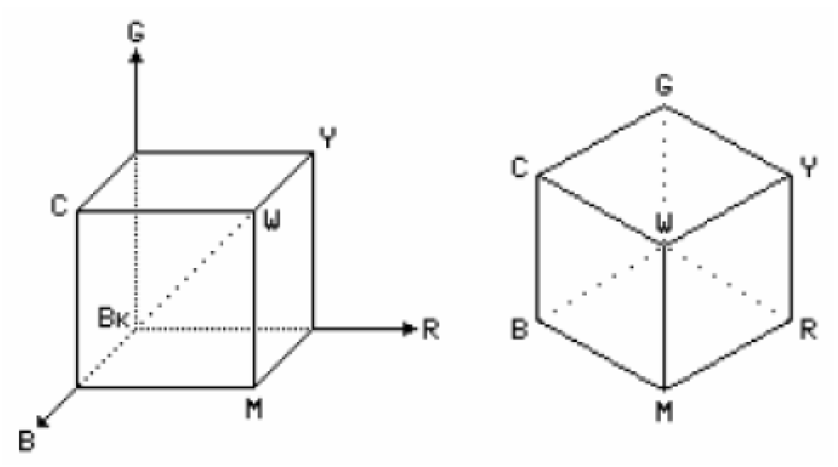


**Figure 2.1: Rgb Color Cube**

The dotted main diagonal of the cube, with equal amounts of Red, Green and Blue, corresponds to the gray levels. This diagonal is also referred to as the gray diagonal. The RGB color model is hardware oriented and is utilized in numerous image capturing, processing and rendering devices.

### 2.1.2 HSV Color Model

HSV which symbolizes hue, saturation and value is an associated symbol of points in an RGB color model which try to illustrate perceptual color correlations more precisely than RGB, while being computationally simpler. The HSV color model is shown in Figure 2.2.

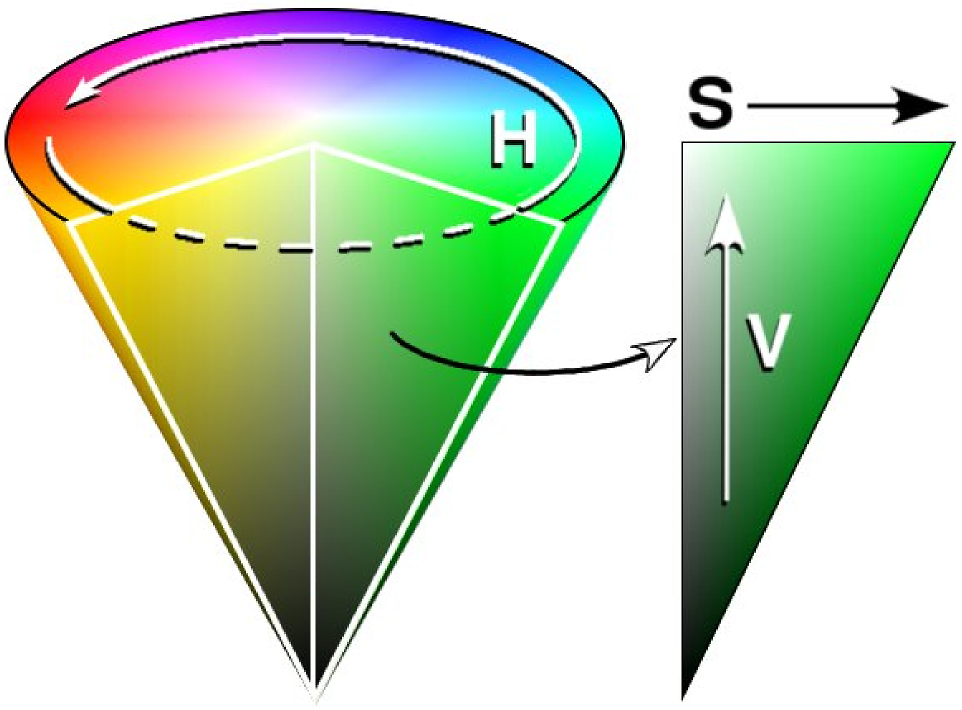


**Figure 2.2: The HSV Color Model.**

HSV is a design to depict colors as points in a cylinder (called a color solid). Central axis of that cylinder is from black at the bottom to white at the top, neutral colors are between them. The angle around the axis accounts for “hue”, the distance from the axis accounts for “saturation”, and the distance along the axis accounts for “lightness”, “value” or “brightness”.

## 2.2 Features

The features of this study are clarified below. During this part, the related image is symbolized as $f(x, y)$ which is assumed to be $N \times N$ in size.

### 2.2.1 Basic Features

Basic statistical texture features used are explained below.

**Mean** of intensity image, which stands for brightness of the pixels in a block on average, is defined as:

$$mean_f = \frac{1}{N^2} \sum_x \sum_y f(x, y) \tag{2.1}$$

**Standard deviation** of intensity image, which represents how dispersed the gray values are, i.e. contrast, is given:

$$var_f = \sqrt{\frac{1}{N^2} \sum_x \sum_y (f(x, y) - mean_f)^2} \tag{2.2}$$

**Gradient magnitude** approximation for the calculation of basic features is defined in the Equation 2.3:

$$\|\nabla f(x, y)\| = |f(x, y) - f(x-1, y) + j(f(x, y) - f(x, y-1))| \tag{2.3}$$

**Skewness** shows the asymmetrical feature of a group of pixel values. If the skewness is equal to 0, it means the data is symmetrical rigorously and the farther from 0, the more asymmetrical. For a gray scale image ($x_1$, $x_{2...}$ xn) represents a set for all pixel values in a moving window shown in Equation 2.4:

$$\text{Skewness}_\text{f} = \frac{1}{NxN} \frac{\sum_x \sum_y (f(x,y) - mean_f)^3}{\text{var}_f^{3/2}} \tag{2.4}$$

**Kurtosis** is a measure of peakedness or flatness of an image histogram, which has a form:

$$Kurtosis_f = \frac{1}{NxN} \frac{\sum_x \sum_y (f(x,y) - mean_f)^4}{\text{var}_f^{2}} \tag{2.5}$$

**Entropy** is a statistical measure of randomness calculated from the intensity histogram of the given window. The entropy equation is calculated as:

$$Entropy_f = \sum_i h_i x \log_2(h_i) \tag{2.6}$$

Where $h_i$ denotes for $i^{th}$ element of histogram.

**Energy** measures are intrinsically capable of making classification, which is given in Equation 2.7:

$$Energy_f = \frac{1}{NxN} \sum_x \sum_y f(x,y)^2 \tag{2.7}$$

**The variogram** of an image is a function, which expresses the spatial correlation of regionalized variables of the image. In probabilistic notation, the *image variogram*, denoted as *γ(h),* can be defined as the expected value of the image intensities spatially distributed apart with a distance *h*:

$$\gamma(h) = \frac{1}{2} E\{(f(x_i, y_j) - f(x_k, y_l))\} \tag{2.8}$$

**Mean of gradient magnitude**, which presents information concerning average rate of gray level change between neighboring pixels, is given in Equation 2.9.

$$mean_{\|\nabla f\|} = \frac{1}{N^2} \sum_x \sum_y \|\nabla f(x,y)\| \tag{2.9}$$

**Standard deviation of gradient magnitude**, which demonstrates how variable this rate of change can be defined as:

$$var_{\|\nabla f\|} = \sqrt{\frac{1}{N^2} \sum_x \sum_y (\|\nabla f(x,y)\| - mean_{\|\nabla f\|})^2} \tag{2.10}$$

### 2.2.2 Zernike Moments

Zernike moments are image moments, which are employed in rotation invariant recognition of images [53]. In contrast to regular image moments, they utilize a set of orthogonal basis functions, which constitute a complete orthogonal set inside the unit circle. Mapping of image function, onto Zernike polynomials, causes no redundancy between different Zernike moments, unlike regular image moments. The basis functions are in the form of Equation 2.11, i.e. Zernike Polynomials.

$$V_{n,m}(x,y) = V_{n,m}(r,\theta) = R_{n,m}(r)e^{jm\theta} \tag{2.11}$$

In this equation $j$ represents the imaginary unit, $\sqrt{-1}$, $n$ is a non-negative integer, $m$ is an integer, $n-|m|$ is even, $|m| \leq n$ and lastly $r$ and $\theta$ are the magnitude and the angle of the vector from origin to $(x,y)$ point correspondingly where $x^2 + y^2 \leq 1$. In the same equation, $R_{n,m}(r)$ represents the radial polynomial, which is described as:

$$R_{n,m}(r) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!} r^{n-2s} \tag{2.12}$$

Given these polynomials, Zernike Moments are defined as :

$$Z_{n,m}(f) = \frac{n+1}{\pi} \int_0^{2\pi} \int_0^1 f(r,\theta) V^*_{n,m}(r,\theta) r dr d\theta \tag{2.13}$$

In this equation, $*$ represents complex conjugate. Since $f(x, y)$ is a real signal and $R_{n,m}(r) = R_{n,-m}(r)$, complex conjugate of $Z_{n,m,f}$, (i.e. $Z*_{n,m}(f)$), is equal to $Z_{n,-m}(f)$, . For a digital image, the expression above turns out to be:

$$Z_{n,m}(f) = \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V^*_{n,m}(r,\theta) \tag{2.14}$$

For the computation, the origin is assumed to be the center of the image $f$ , and pixel coordinates are mapped into the range of unit circle. The pixels outside this range are omitted. In Table 2.1, Zernike Moments and their respective orders are given.

**Table 2.1: Zernike Moments and Their Respective Orders**

| Order | Moments | Number of Moments |
|---|---|---|
| 0 | $Z_{0,0}$ | 1 |
| 1 | $Z_{1,1}$ | 1 |
| 2 | $Z_{2,0}$, $Z_{2,2}$ | 2 |
| 3 | $Z_{3,1}$, $Z_{3,3}$ | 2 |
| 4 | $Z_{4,0}$, $Z_{4,2}$, $Z_{4,4}$ | 3 |

Zernike Moments are complex numbers, where rotation of the image $f$, causes a shift in the phase of these numbers. In the meantime, their magnitude does not change as demonstrated by [53]. As a result acquiring rotation invariant features is achievable by using the magnitudes of Zernike Moments. It is also revealed in the same study that under moderate noise, Zernike Features has a good performance in a wide range of classification tasks.

### 2.2.3 Circular-Mellin Features

Circular-Mellin Features are textural features, which are discrete Fourier coefficients of the spatial image represented in the polar-log coordinate system [54]. These features represent by the spectral decomposition of image in the polar-log coordinate transformation.
An image *f (x,* y) in Cartesian coordinates *(x,* y), can be represented in the polar-log coordinate space *(λ, θ)* where $e^{\lambda} = r = \sqrt{x^2 + y^2}$ and $\theta = \arctan(y, x)$; $\theta \in (-\pi, \pi)$. Correlation response of an image, $f(\lambda, \theta)$ with a filter $h(\lambda, \theta)$ can be defined as:

$$\int_x \int_y f(x,y) h(x,y) dx dy \tag{2.15}$$

Equation (2.15) can be expressed using the polar-log coordinate represantation as:

$$\int_\lambda \int_\theta f(\lambda,\theta) h^*(\lambda,\theta) e^{2\lambda} d\lambda d\theta \tag{2.16}$$

In order to be the filter invariant, Circular Harmonic Functions (CHF) can be used. Then the filter is:

$$h(\lambda,\theta) = h_q(\lambda) e^{jq\theta} \tag{2.17}$$

where $q$ is defined as the order of CHF or annular frequency.

Likewise scale invariance can be obtained by using a filter using Mellin Harmonic Functions (MHF) where $p$ is defined as the order of MHF or radial frequency. For various scales of $f(x,y)$, the correlator function generates different magnitude values, but the ratio between these values remains same. Thus scale invariance is obtained by comparing the ratios of outputs for distinct $p$ values. As a result, the filter can be defined as:

$$h(\lambda,\theta) = e^{-\lambda} h_p(\theta) e^{j2\pi p\lambda} \tag{2.18}$$

A filter function can be written combining two above filters as:

$$h_{p,q}(\lambda,\theta) = e^{-\lambda} e^{j2\pi p\lambda} e^{jq\theta} \tag{2.19}$$

So correlator function output becomes

$$C_{p,q}(f) = \int_{\lambda}\int_{\theta} f(\lambda,\theta) h^{*}_{p,q}(\lambda,\theta) e^{2\lambda} d\lambda d\theta \tag{2.20}$$

### 2.2.4 Fourier Power Spectrum

Fourier analysis offers mathematical background for the analysis of signals based on frequency. Let $f(x,y)$ be the signal depiction of an image. Fourier transform of $f(x,y)$ is defined by

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-2\pi j(ux+vy)} dxdy \tag{2.21}$$

The Fourier power spectrum is formulized as $|F|^2 = FF^*$ represents the complex conjugate. In [55] and [56] power spectrum is analyzed by ring or wedge shaped regions, and four additional statistical features of the whole spectrum is defined. Ring and wedge shaped regions are given in Equations 2.22 and 2.23 correspondingly. Illustration of these regions is shown in Figure 2.3.

$$\phi_{r_1,r_2}(f) = \int_{r_1}^{r_2}\int_{0}^{2\pi} |F(r,\theta)|^2 \, d\theta dr \tag{2.22}$$

$$\phi_{\theta_1,\theta_2}(f) = \int_{\theta_1}^{\theta_2}\int_{0}^{\infty} |F(r,\theta)|^2 \, dr d\theta \tag{2.23}$$

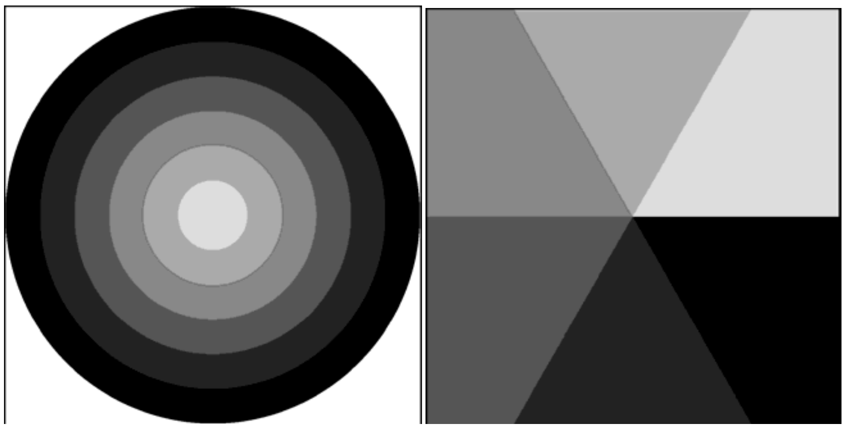


**Figure 2.3: Ring and Wedge Shaped Regions On Frequency Domain. Different Shades of Gray Represent Different Regions.**

Because of the fact that coarse textures have high values of power spectrum close to the origin and finer textures have a more spread out power spectrum [57], ring shaped regions are related with coarseness of the texture whereas wedge shape regions are related with direction. Under these circumstances, it is possible to assume that ring shaped regions are rotationally invariant and wedge shaped regions are not.

Given that discrete images are the concern for us, Discrete Fourier Transform of the $N$ by $N$ image must be utilized, it is shown in Equation 2.24.

$$F(u,v) = \frac{1}{N_x N_y} \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y) e^{-2\pi j\left(\frac{ux}{N}+\frac{vy}{N}\right)} \tag{2.24}$$

Ring shaped regions are approximated with their discrete complements. The basic features described in section 2.2.1 can be calculated for Discrete Fourier Transform of the image and also maximum magnitude of DFT can be calculated as in equation 2.25.

$$Maximum\ Magnitude(F) = \max\left\{\left|F(u,v)\right| : (u,v) \neq (0,0)\right\} \tag{2.25}$$

### 2.2.5 Gabor Filters

Gabor filters are linear filters, which involve one harmonic and one Gaussian function. These filters have optimal localization in both spatial and frequency domain by diminishing the joint uncertainty in both domains which is a striking aspect [58]. An appealing detail concerning the relationship of Gabor filters and human perception is that the characteristics of cortical cells in the human visual cortex can be approximated by Gabor filters [59]. A two-dimensional Gabor function is given in equation 2.26.

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}+2\pi jWx\right)\right]} \tag{2.26}$$

The Fourier transform of the same function is

$$G(u,v) = e^{-\frac{1}{2}\left[\frac{(u-W)^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}\right]} \tag{2.27}$$

In these equations, $W$ is in charge of the modulation and the variances have the relationship given in 2.28 and 2.29.

$$\sigma_x = \frac{1}{2\pi\sigma_u} \tag{2.28}$$

$$\sigma_y = \frac{1}{2\pi\sigma_v} \tag{2.29}$$

Gabor functions form a complete but non-orthogonal set of basis functions. With the intention of getting rid of the redundancy caused by non-orthogonality, these basis functions must be scaled and rotated through a generating function given in Equation 2.30 to obtain a Gabor filter dictionary.

$$g_{k,s}(x,y) = a^{-s} g(x', y') \tag{2.30}$$

where

$$x' = a^{-s}(x\cos\theta + y\sin\theta) \tag{2.31}$$

$$y' = a^{-s}(-x\sin\theta + y\cos\theta) \tag{2.32}$$

$$\theta = \frac{k\pi}{K} \tag{2.33}$$

In these equations $0 \le s \le (S-1)$ and $0 \le k \le (K-1)$ where $S$ and $K$ are total number of scales and orientations respectively. Given that the real part of the generating function, i.e. $\Re\{g_{k,s}(x,y)\}$, is used as filter, a symmetric frequency response is obtained for each orientation-scale pair. Half magnitudes of frequency responses of generating function and real part of the generating function is given in Figure 2.4 for an arbitrary $(k,s)$ pair.

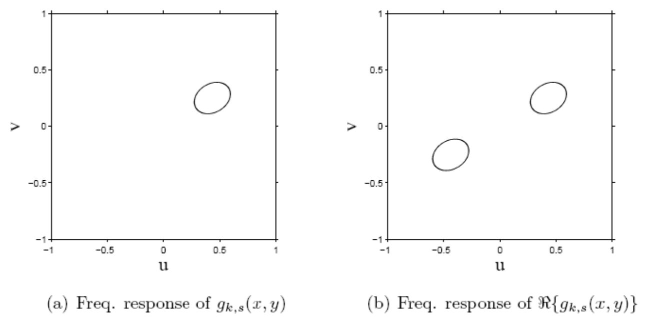


**Figure 2.4: Comparison of Complex and Real Gabor Filters. Complex Gabor Filters are not Symmetric over Frequency Domain While Their Real Counterparts are.**

### 2.2.6 Haralick Features

Two-dimensional co-occurrence (gray-level dependence) matrices (GLCM), proposed by Haralick [26] in 1973, are generally used in texture analysis because they are able to capture the spatial dependence of gray-level values within an image. A 2D co-occurrence matrix, $P$, is an $n \times n$ matrix, where $n$ is the number of gray-levels within an image. The GLCM is calculated as

$$P_{\Delta x,\Delta y}(i,j) = \sum_{x}\sum_{y}\begin{cases} 1 & if\ f(x,y) = i\ and\ f(x+\Delta x, y+\Delta y) = j \\ 0 & otherwise \end{cases} \tag{2.34}$$

In this equation, $f(x,y)$ is the image function, $(\Delta x, \Delta y)$ is the offset vector, and lastly, as GLCM matrices are symmetric in [26], $i$ and $j$ are interchangeably row or column indices. The matrix acts as an accumulator so that $P_{\Delta x,\Delta y}(i,j)$ counts the number of pixel pairs having the intensities $i$ and $j$. Pixel pairs are defined by a distance and direction which can be represented by a displacement vector $\delta = (\Delta x, \Delta y)$, where $\Delta x$ represents the number of pixels moved along the x-axis, and $\Delta y$ represents the number of pixels moved along the y-axis of an image slice. In this study six measures are computed from each matrix $P(i,j)$. These measures are given in Table 2.2.

**Table 2.2 : Haralick Feature Measures**

| | |
|---|---|
| **Inertia** | $I = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}(i-j)^2 P(i,j)$ |
| **Cluster Shade** | $A = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}(i+j-\mu_i-\mu_j)^3 P(i,j)$ |
| **Cluster Prominence** | $B = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}(i+j-\mu_i-\mu_j)^4 P(i,j)$ |
| **Local Homegenity** | $L = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\frac{P(i,j)}{1+(i-j)^2}$ |
| **Energy** | $E = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\left[P(i,j)\right]^2$ |
| **Entropy** | $H = -\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}P(i,j)\log(P(i,j))$ |

In what follows it is frequently convenient to consider $\delta = (\Delta x, \Delta y)$ not in a Cartesian form but rather in a apolar form $\delta = (d, \theta)$, where $d = \max(\Delta x, \Delta y)$ and $\theta = \arctan(\Delta y / \Delta x)$. In polar form $d$ is called the intersample spacing distance and $\theta$ is called the angular orientation. In this work Haralick features are calculated using four different orientations and three different spacing distance which correspond to adjacent pixels at $0^\circ$, $45^\circ$, $90^\circ$ and $135^\circ$ and spacing distance as 1, 2, 3 and 4 pixels, respectively.

### 2.2.7 Wavelet Features

Wavelet features are extracted by means of Discrete Wavelet Transform (DWT). DWT involves filtering and down-sampling as shown in Figure 2.5. A multi-resolution decomposition is achieved by applying the single-level wavelet decomposition recursively to low-frequency component (Figure 2.6). Multi-resolution decomposition offers a basic hierarchical system in order to interpret frequency and location based information included in the image. Given that utilized wavelet functions are orthogonal, each phase of decomposition observes different periodical aspects of the image.

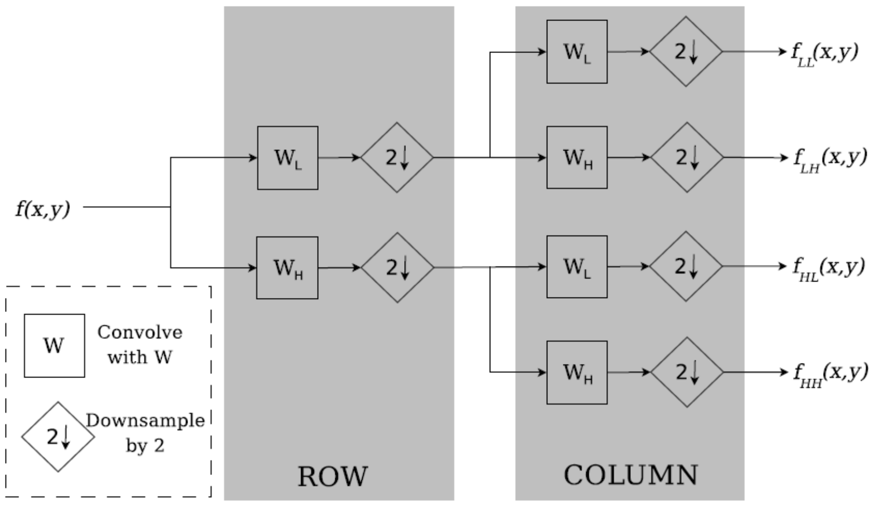


**Figure 2.5: Single-Level Wavelet Decomposition**

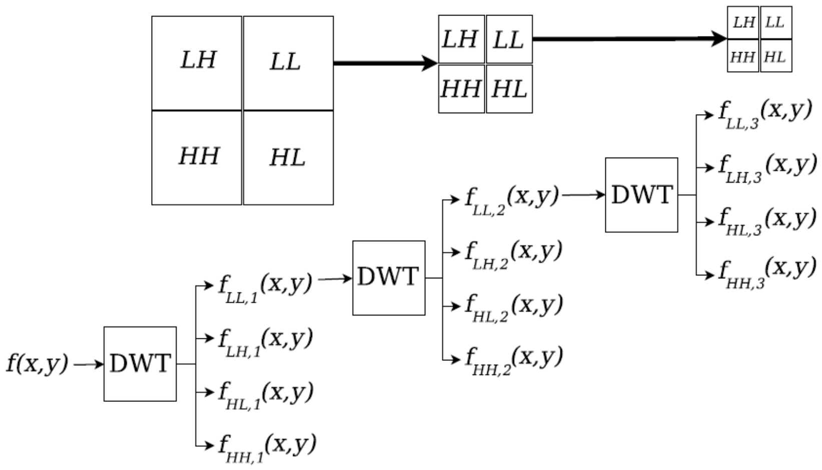


**Figure 2.6: Multi-Level Wavelet Decomposition**

### 2.2.8 HSV Color Space

Features obtained from HSV color space are employed in this thesis. The central idea of using features of HSV color space is to benefit from the success of this color space in representing perceptual color relationships. To describe these features, it is necessary to convert the images that are intrinsically in the RGB color space to HSV color space. This task is carried out with the equations given in Equations 2.35 - 2.37 where $max = \max\{r(x,y), g(x,y), b(x,y)\}$, $min = \min\{r(x,y), g(x,y), b(x,y)\}$, and finally $r(x,y)$, $g(x,y)$, $b(x,y)$ and $h(x,y)$, $s(x,y)$, $v(x,y)$ functions stand for channels of RGB and HSV respectively.

$$h(x,y) = \begin{cases} 0^\circ, & \text{if } max = min \\ \left(\dfrac{g(x,y)-b(x,y)}{max-min}60^\circ + 360^\circ\right) \mathit{mod}\ 360^\circ, & \text{if } max = r(x,y) \\ \dfrac{b(x,y)-r(x,y)}{max-min}60^\circ + 120^\circ, & \text{if } max = g(x,y) \\ \dfrac{r(x,y)-g(x,y)}{max-min}60^\circ + 240^\circ, & \text{if } max = b(x,y) \end{cases} \tag{2.35}$$

$$s(x,y) = \begin{cases} 0, & \text{if } max = 0 \\ 1 - \dfrac{min}{max}, & \text{otherwise} \end{cases} \tag{2.36}$$

$$v(x,y) = \max \tag{2.37}$$

## 2.3 Adaboost Learning Algorithm

Boosting is a common method to develop the performance of a learning algorithm. Adaboost (Adaptive Boosting) [60] is a boosting algorithm, which constitutes a linear combination out of a set of weak learners in order to produce a strong classifier. A weak learner is a classifier, which provides weak hypotheses that are lacking ability to solve the problem on its own. These weak learners are generally picked as threshold classifiers, which decide the output by judging the result of a comparison between input and a threshold. A sample threshold classifier, $h_j(x)$, is given in Equation 2.38.

$$h_j(x_j) = \begin{cases} +1 & if\ p_j x_j < p_j \theta_j \\ -1 & otherwise \end{cases} \tag{2.38}$$

In this equation $x_j$ is the feature, $\theta_j$ is the threshold, $p_j$ is the parity which decides the direction of inequality and $1 \le j \le K$ where $K$ is the number of features. Every weak learner makes its decision based on the examination of just one feature, as a result, every classifier match up with a feature. Training of a weak learner, $j$, is given in Equation 2.39, and it means determining $\theta_j$ and $p_j$ values that minimizes the classification error on the iteration $t$.

$$(\theta_j, p_j) = argmin_{(\theta_j, p_j)}\{\varepsilon_{t,j}\} \tag{2.39}$$

This operation can be accomplished by searching in intervals $\min(x) \le \theta_j \le \max(x)$ and $p_j = \{+1, -1\}$. The definition of $\varepsilon_{t,j}$ is given in Equation 2.40 where $y_i$ is the aspired output label.

$$\varepsilon_{t,j} = \sum_{i:h_j(x_i) \ne y_i} D_t(i) \tag{2.40}$$

In this equation, $D_t(i)$ is the distribution function over training samples on the $t^{th}$ iteration. This distribution is used to put emphasis on the misclassified samples, forcing the algorithm to focus on the hard examples in the training set. $D_t(i)$ is initialized to be uniform, and on every iteration it is updated in a way that the true classified samples' values are reduced and false classified samples' values are increased. Complete algorithm of the Adaboost is given in Figure 2.7.

```
**Input:** Training data are $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)$ where
  • input feature vectors are $\mathbf{x}_i \in X_1 \times X_2 \times \ldots \times X_K$
  • the $j^{th}$ element of $\mathbf{x}_i$ is represented as $x_{i,j} \in X_j$
  • the training data labels are $y_i \in Y = \{-1, +1\}$
**Initialize:** $D_1(i) = \frac{1}{m}$, for $1 \le i \le m$
**Algorithm**:
For $t = 1, \ldots, T$:
  • Train weak classifiers finding $(\theta_j, p_j)$ pairs for $i = 1, \ldots, K$
  • Get weak hypotheses $h_j(x_{i,j}) : X_j \to \{-1, +1\}$, for $j = 1, \ldots, K$
  • Select the classifier with minimum error
    - $j^*(t) = argmin_{(\theta_j, p_j)}\{\varepsilon_{t,j}\} = argmin_j \sum_{i: h_j(x_{i,j}) \neq y_i} D_t(i)$
    - Set $h_t^* = h_{\xi(t)}$
    - Set $\varepsilon_t = \varepsilon_{t,j^*(t)} = \sum_{i: h_t^*(x_{i,j^*(t)}) \neq y_i} D_t(i)$
  • Choose $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
  • Update $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if\ h_t^*(x_{i,j^*(t)}) = y_i \\ e^{\alpha_t} & if\ h_t^*(x_{i,j^*(t)}) \neq y_i \end{cases} = \frac{D_t(i) e^{-\alpha_t y_i j^*(x_{i,\xi(t)})}}{Z_t}$
      where $Z_t = \left[\sum_i D_t(i) e^{-\alpha_t y_i h_t^*(x_{i,j^*(t)})}\right]^{-1}$
**Output**:
    $H(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t^*(\mathbf{x})\right)$
```

**Figure 2.7: Adaboost Algorithm**

Adaboost is fast, simple and easy to program alongside its many advantages. At the same time, it is a nice aspect that it does not require the parameters to be tuned, except the iteration count $T$. Adaboost offers a provable method to produce an accurate prediction rule by means of combining rough and moderately inaccurate weak learners. Then again, it should be kept in mind that boosting can fail with overly complex or too weak hypotheses, yet, it is consistent with the theory.

## 2.4 Support Vector Machines

Support vector machines (SVM) are a set of related supervised learning methods used for classification and regression [60]. In another terms, it is a prediction tool that uses machine-learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. Viewing input data as two sets of vectors in an *n*-dimensional space, an SVM will form a separating hyperplane in that space, one which maximizes the margin between the two data sets. To calculate the margin, two parallel hyperplanes are constructed. Good separation is achieved by the hyperplane that has the largest distance to the neighboring data points of both classes.

Consider the linearly separable training set $x_i=f(x_{1i},x_{2i})$ $i=1\ldots l$ that is, there is a linear discriminant function of the form

$$x \rightarrow w^T x + b \qquad (2.41)$$

where w and b are learned from the training set.

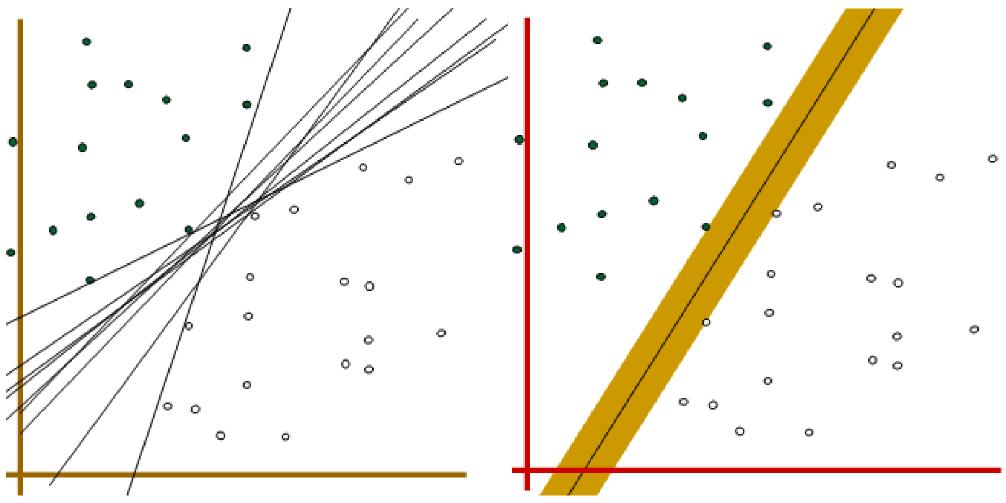

(a) (b)

**Figure 2.8: (a) Linear Classifiers and (b) Best Linear Classifier**

Of course, there can be infinitely many linear classifiers that separate the training set as shown in Figure 1. The purpose of SVM is to find the hyperplane that best classifies the training set while maximizing the minimal margin of the linear discriminant function with respect to the training set Z.

The aim of SVM is to maximize the distance between the two parallel lines which can be expressed as the following equations.

$$x \cdot w + b = 1 \quad \text{and} \quad x \cdot w + b = -1 \tag{2.42}$$

$$x_i \cdot w + b \geq 1 \quad \text{for } y_i = +1 \quad \text{and} \quad x_i \cdot w + b \leq -1 \quad \text{for } y_i = -1 \tag{2.43}$$

These two equations can be put together as:

$$y_i(x_i \cdot w + b) \geq 1 \tag{2.44}$$

$x_i$ is a feature vector of the ith training document represented by an n dimensional vector and $y_i$ is the class (positive (+1) or negative (-1)) label of the ith training document. All vectors lying on one side of the hyperplane are labeled as -1, and all vectors lying on the other side are labeled as 1. The training documents which lie on either of two dashed lines are called support vectors.

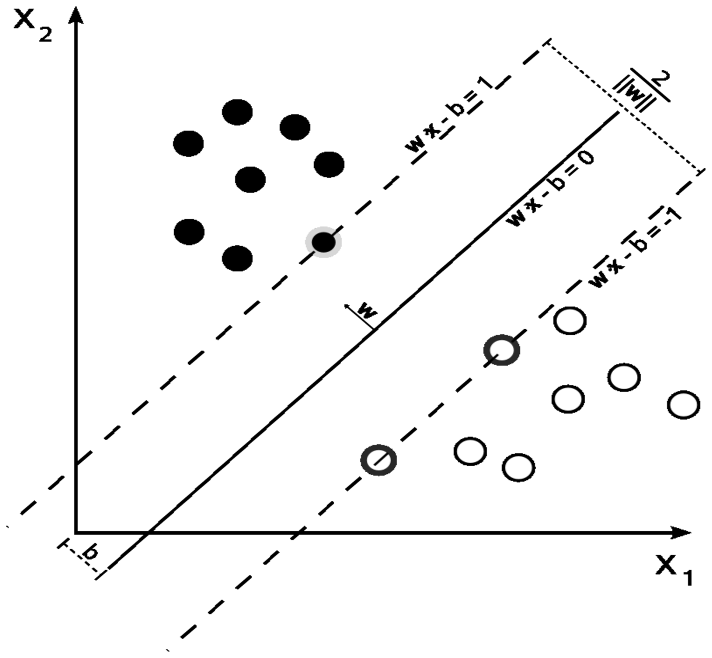


**Figure 2.9: SVM with Maximum Margin**

The margin is given by:

$$M = \frac{2}{\| w \|} \tag{2.45}$$

Hence the hyperplane that optimally separates the data is the one that minimizes:

$$L = \frac{1}{2} \| w \|^2 \tag{2.46}$$

The solution to the optimization problem of Equation 2.46 under the constraints of Equation 2.44 is given by the saddle point of the Lagrange functional (Lagrangian)

$$L_p = \frac{1}{2} \| w \|^2 - \sum_{i=1}^{l} \alpha_i y_i < x_i, w > +b) + \sum_{i=1}^{l} \alpha_i \tag{2.47}$$

where α are the Lagrange multipliers. The Lagrangian has to be minimized with respect to w, b and maximized with respect to α ≥ 0.
Solution to the problem is given by,

$$\alpha^* = \arg\min_{\alpha} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j < x_i, x_j > - \sum_{k=1}^{l} \alpha_k \tag{2.48}$$

with constraints,

$$\sum_{j=1}^{l} \alpha_j y_j = 0 \qquad \alpha_i \geq 0 \qquad i=1,2,\ldots\ldots,l \tag{2.49}$$

Solving Equation 2.48 with constraints Equation 2.49 determines the Lagrange multipliers, and the optimal separating hyperplane is given by,

$$w^* = \sum_{i=1}^{l} \alpha_i y_i x_i \qquad b^* = -\frac{1}{2} < w^*, x_r + x_s > \tag{2.50}$$

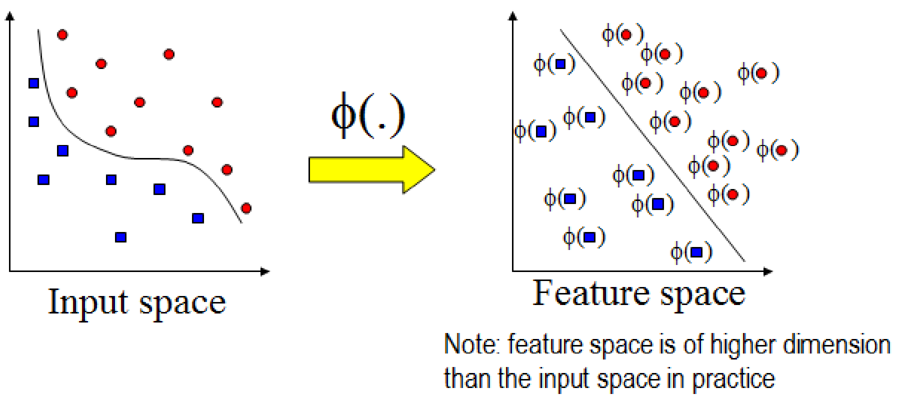
where $x_r$ and $x_s$ are any support vector from each class satisfying,

$$\alpha_r, \alpha_s > 0 \quad y_r = -1, y_s = 1 \tag{2.51}$$

The hard classifier is then,

$$f(x) = \operatorname{sgn}(< w^*, x > + b) \tag{2.52}$$

In the case where a linear boundary is inappropriate the SVM can map the input vector, x, into a high dimensional feature space, z. By choosing a non-linear mapping a priori, the SVM constructs an optimal separating hyperplane in this higher dimensional space,



Input space
$\phi(.)$
Feature space
Note: feature space is of higher dimension than the input space in practice

**Figure 2.10: Kernel Function Mapping**

This mapping is done by using polynomials, radial basis functions and certain sigmoid functions.

The optimization problem of Equation 2.41 becomes,

$$\alpha^{*} = \arg\min_{\alpha} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{k=1}^{l} \alpha_k \qquad (2.53)$$

where $K(x_i, x_j)$ is the kernel function performing the non-linear mapping into feature space, and the constraints are unchanged,

$$\sum_{j=1}^{l} \alpha_j y_j = 0 \qquad 0 \le \alpha_i \le C \qquad i = 1,\ldots,l \qquad (2.54)$$

Solving Equation 2.53 with constraints Equation 2.54 determines the Lagrange multipliers, and a classifier implementing the optimal separating hyperplane in the feature space is given by,

$$f(x) = \operatorname{sgn}\Big(\sum_{i \ni SVs} \alpha_i K(x_i, x_j)\Big) \qquad (2.55)$$

### 2.4.1 Kernel Functions

Define the kernel function $K$ by

$$K(x,x') = <\Phi(x),\Phi(x')> \tag{2.56}$$

An inner product in feature space has an equivalent kernel in input space. $K$ should be positive semi definite function. Most common kernel functions are polynomial, radial basis and quadratic functions.

#### 2.4.1.1 Polynomial

A polynomial mapping is a popular method for non-linear modeling, which has the form,

$$K(x,x') = <x,x'>^d \tag{2.57}$$

where $d$ denotes the degree of the polynomial. *Linear* (degree 1), *Quadratic* (degree 2) and *Cubic* (degree 3) kernels are special cases of the polynomial kernel and obtained using the polynomial kernels with the degrees 1, 2 and 3, respectively. The polynomial kernel is usually preferred as shown below in order to avoid problems with the hessian becoming zero.

$$K(x,x') = (<x,x'>+1)^d \tag{2.58}$$

#### 2.4.1.2 Gaussian Radial Basis Function

A radial basis function of the form,

$$K(x,x') = \exp(-\frac{\|x-x'\|^2}{2\sigma^2}) \tag{2.59}$$

which produces a piecewise linear solution which can be attractive when discontinuities are acceptable.

## 2.5 Morphological Operations

Morphology is a technique of image processing based on shapes. The value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors. By choosing the size and shape of the neighborhood, you can construct a morphological operation that is sensitive to specific shapes in the input image. Morphologic operations are especially suited to the processing of binary images and grayscale images.

### 2.5.1 Erosion and Dilation

Dilation and erosion are two fundamental morphological operations. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as a dilation or an erosion. The rule of dilation can be defined as the value of the output pixel is the maximum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1. The rule of dilation can be defined as the value of the output pixel is the minimum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0. An example of erosion and dilation is illustrated in Figure 2.11.

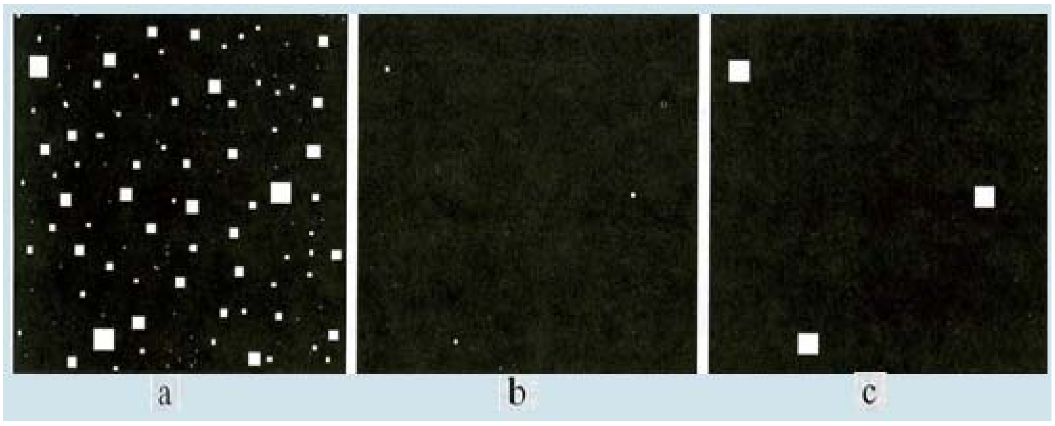


**Figure 2.11: (a) Images of Squares of Size 1,3,5,7,9,15 Pixels on the Side. (b) Erosion of (a) with a Structuring Element of 1's,13 Pixels on the Side. (c) Dilation of (b) with the Same Structuring Element.**

## 2.6 Performance Criteria

As to the accuracy assessment of the proposed methodology, pixel-based and object-based evaluation metrics were applied.In pixel-based evaluation [61], the ground truth data is compared to the output image obtained by the methodology. The accuracy assessment involves computation of True Positive (TP), False Positive (FP) and False Negative (FN) pixel counts. TP refers to the regions determined correctly as building. FP refers to the false alarm determined as buildings. FN refers to the regions, which could not be determined as buildings although they exist in the ground truth. Based on these components the split factor, (SF=FP/(TP+FP)), missing factor (MF=FN/(TP+FP)), percent of building determination (PBD=100*TP/(TP+FN) and quality percent (QP=100*TP/(TP+FP+FN)) are calculated.

In object-based error measure, the overlapping area matrix (OAM) [62 and 63], is used to measure the performance of the algorithm. Let the *i'th* ground truth object be shown as $GT_i$ while the *j'th* output object be denoted as $O_j$. The set of objects in the ground truth are denoted as $GT_r = \{GT_0, GT_1, \ldots\ldots, GT_{N_r}\}$ and the output objects are denoted as $O_o = \{O_0, O_1, \ldots\ldots, O_{N_o}\}$, where $GT_0$ is background in the ground truth, $O_0$ is background in the output, $N_r$ is the number of objects in the ground truth and $N_o$ is the number of objects in the output maps.

The sizes of the objects $GT_i$ and $O_j$ and the whole image $I$ are calculated from the OAM as

$$n(GT_i) = \sum_{j=0}^{N_0} C_{ij} \tag{2.60}$$

$$n(O_j) = \sum_{i=0}^{N_r} C_{ij} \tag{2.61}$$

$$n(I) = \sum_{i=0}^{N_r} n(GT_i) = \sum_{j=0}^{N_0} n(O_j) \tag{2.62}$$

where $C_{ij}$ is the number of pixels in the i'th object in the ground truth map that overlap with the j'th object in an output map produced by the algorithm.
By adoption of OAM on each pair of ground truth $GT_i$ and model output $O_j$, for a specified threshold *T* objects are classified as follows:

**Correct detection:** A pair of objects $GT_i$ and $O_j$ is classified as an instance of correct detection if

$$C_{ij} \geq T \times n(O_j) \tag{2.63}$$

$$C_{ij} \geq T \times n(GT_i)\,. \tag{2.64}$$

**Over detection:** An object $GT_i$ and a set of objects $O_{j_1},\ldots,O_{j_k}$, $2 \leq k \leq N_o$, are classified as an instance of over-detection if

$$C_{ij_t} \geq T \times n\left(O_{j_t}\right),\ \forall t \in \{1,\ldots,k\} \tag{2.65}$$

$$\textstyle\sum_{t=1}^{k} C_{ij_t} \geq T \times n\left(GT_i\right) \tag{2.66}$$

**Under detection:** A set of objects $GT_{i_1},\ldots,GT_{i_k}, 2 \le k \le N_r$, and an object $O_j$ are classified as an instance of under-detection if

$$\textstyle\sum_{t=1}^{k} C_{i_t,j} \ge T \times n(O_j), \tag{2.67}$$

$$C_{i_t,j} \ge T \times n\left(GT_{i_t}\right),\ \forall t \in \{1,\ldots,k\}. \tag{2.68}$$

**Missed detection:** A ground truth object $GT_i$ is classified as a missed detection if it does not participate in any instance of correct detection, over-detection or under-detection.

**False Alarm:** An output object $O_j$ is classified as a false alarm if it does not participate in any instance of correct detection, over-detection or under-detection.

# CHAPTER 3

# THE PROPOSED APPROACH FOR BUILDING DETECTION

The proposed building detection method consists of four main stages; these are feature extraction, feature selection, classification and post processing. The basic block schema of the proposed approach is given in Figure 3.1. As input data, only mono RGB images are used. The proposed method is applied on the satellite images of the cities Eskisehir and Ankara in Turkey. The set of selected satellite images have different types of buildings (i.e. different colors and shapes).

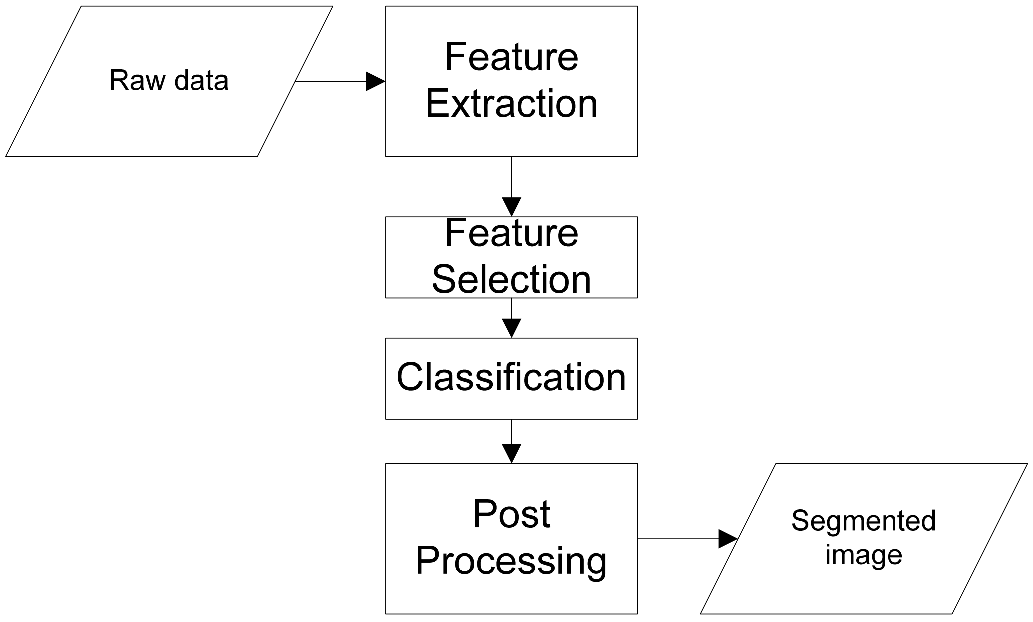


**Figure 3.1: The Basic Block Schema Of The Proposed Building Detection Approach**

The detailed block schema of the proposed approach is given in Figure 3.2. The feature extraction stage includes extraction of intensity textural features and HSV domain features. The classifiers may use these features' set one by one and all together i.e. using features extracted only from intensity, using features extracted from HSV domain and using both of them.  In feature selection stage, selection by Adaboost, selection using minimum histogram errors and no selection is applied. Adaboost uses the features itself selected. SVM kernels are fed by all three type of feature selection methods. After applying a simple post-process stage, resulting segmented image with building patches are obtained.

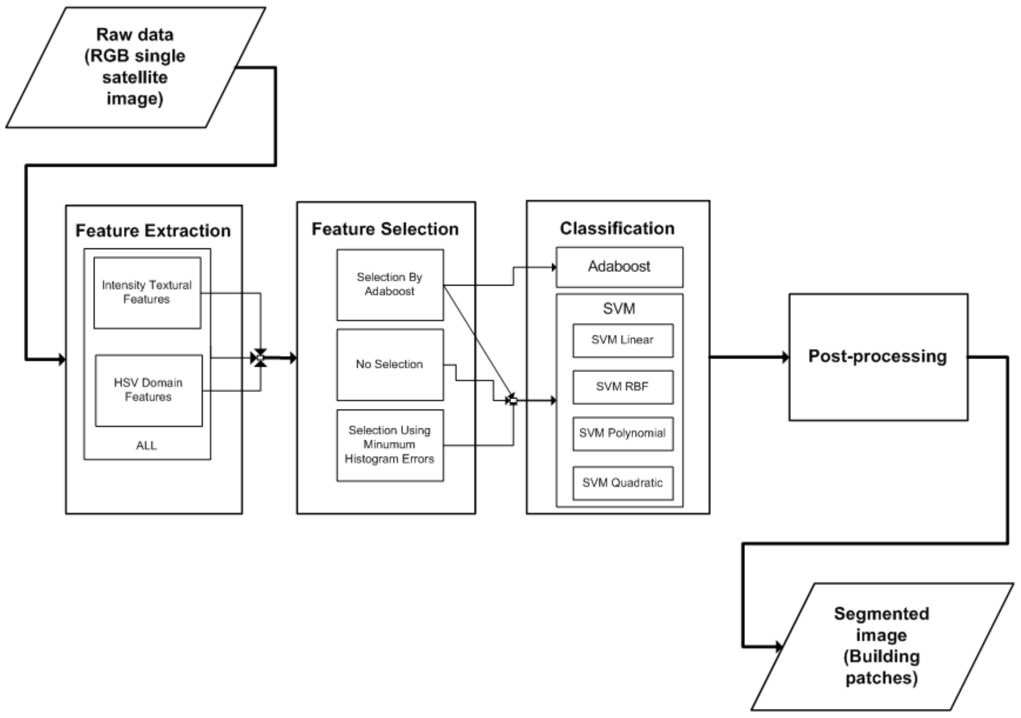


**Figure 3.2: The Detailed Block Schema of the Proposed Building Detection Approach**

## 3.1 Feature Extraction

In order to extract local features, feature extraction windows must be defined. Two different types of windows are defined in this work. In the first one, feature extraction is in non-overlapping windows and labeling (i.e. building or non-building) includes the whole window. If half of the pixels in the window belong to building, the window is in building class and vice versa. In classification stage, the window is also classified as building or non-building. Thus, the classification resolution of the image is reduced proportional to window size. In second style, features extracted are assigned to the center pixel. Feature extraction goes on an overlapping manner on the image. In training and classification, the label of the center pixel is used. Different window sizes are examined in this study for both overlapping and non-overlapping window types.

The features are examined under eight classes, which are basic features, Zernike moments, Circular Mellin features, Fourier power spectrum features, Gabor filters, Haralick features, wavelet features and HSV domain features. Except the last one that is HSV features, only intensity information is used. Here HSV is used rather than RGB since in HSV domain, (HS) which is related to color is better differentiated from illumination component, which is (V).

### 3.1.1 Basic Features

Since building regions have similar intensity characteristics, it is reasonable to use these basic features for distinguishing buildings from its surroundings. The basic feature set consists of mean and variances of intensity and gradient of intensity of the concerned image block. In addition to that, some additional features of intensity, which are entropy, energy and variogram, are also utilized in basic feature set. The variogram features are computed for three different distance (*d*) scales as 1, 2 and 3 pixels. These features are listed as given below.

$$\mathbf{F}_{basic} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \\ F_{10} \\ F_{11} \end{bmatrix} = \begin{bmatrix} mean_f \\ var_f \\ skewness_f \\ kurtosis_f \\ entropy_f \\ energy_f \\ mean\ grad_f \\ vargrad_f \\ variogram(d=1)_f \\ variogram(d=2)_f \\ variogram(d=3)_f \end{bmatrix} \tag{3.1}$$

### 3.1.2 Zernike Moments

In this study, Zernike moments are employed as features as well. Zernike moments are rotation invariant image moments. Because of the fact that the orientation of building is not known, the rotation invariance is mandatory.

$$\mathbf{F}_{zernike} = \begin{bmatrix} F_{12} \\ F_{13} \\ F_{14} \\ F_{15} \\ F_{16} \\ F_{17} \\ F_{18} \\ F_{19} \\ F_{20} \\ \end{bmatrix} = \begin{bmatrix} Z_{0,0}(\gamma) \\ Z_{1,1}(\gamma) \\ Z_{2,0}(\gamma) \\ Z_{2,2}(\gamma) \\ Z_{3,1}(\gamma) \\ Z_{3,3}(\gamma) \\ Z_{4,0}(\gamma) \\ Z_{4,2}(\gamma) \\ Z_{4,4}(\gamma) \\ \end{bmatrix} \tag{3.2}$$

### 3.1.3 Circular Mellin

Circular-Mellin features, which are orientation and scale invariant, are another set of utilized vectors used in this work. This algorithm uses two main parameters: radial frequency (*p*) and annular frequency (*q*). Some experimental results are given in [54] about the selection of these variables by a search algorithm. This search algorithm works as a feature selector and determines the best *p* and *q* combination. In reference [54] it is proposed that a single Circular-Mellin feature (*p = 1* and *q = 5*) is used for building detection example. The set of used Circular-Mellin features is given in Equation 3.3, which includes the aforementioned parameters: *p = 1, q = 5*.

$$\mathbf{F}_{circular-mellin} = \begin{bmatrix} F_{21} \\ F_{22} \\ F_{23} \\ F_{24} \\ F_{25} \\ F_{26} \\ F_{27} \\ F_{28} \\ F_{29} \\ F_{30} \\ \end{bmatrix} = \begin{bmatrix} C_{1,1}(\gamma) \\ C_{1,2}(\gamma) \\ C_{1,3}(\gamma) \\ C_{1,4}(\gamma) \\ C_{1,5}(\gamma) \\ C_{2,1}(\gamma) \\ C_{2,2}(\gamma) \\ C_{2,3}(\gamma) \\ C_{2,4}(\gamma) \\ C_{2,5}(\gamma) \\ \end{bmatrix} \tag{3.3}$$

### 3.1.4 Fourier Power Spectrum

Periodic image patterns can be extracted using Fourier Power Spectrum analysis. In this thesis, power spectrum is examined using six equal ring shaped regions. The feature vector is formed as given in Equation 3.4

$$\mathbf{F}_{fourier} = \begin{bmatrix} F_{31} \\ F_{32} \\ F_{33} \\ F_{34} \\ F_{35} \\ F_{36} \\ F_{37} \\ F_{38} \\ F_{39} \\ F_{40} \\ F_{41} \\ F_{42} \\ F_{43} \\ F_{44} \\ F_{45} \\ F_{46} \end{bmatrix} = \begin{bmatrix} \phi_{0,r_1}(\Gamma) \\ \phi_{r_1,r_2}(\Gamma) \\ \phi_{r_2,r_3}(\Gamma) \\ \phi_{r_3,r_4}(\Gamma) \\ \phi_{r_4,r_5}(\Gamma) \\ \phi_{r_5,r_6}(\Gamma) \\ \textit{Maximum of } |\Gamma| \\ \textit{Average of } |\Gamma| \\ \textit{Energy of } |\Gamma| \\ \textit{Variance of } |\Gamma| \\ \textit{Skewness of } |\Gamma| \\ \textit{Kurtosis of } |\Gamma| \\ \textit{Entropy of } |\Gamma| \\ \textit{Variogram}(d=1)\textit{ of } |\Gamma| \\ \textit{Variogram}(d=2)\textit{ of } |\Gamma| \\ \textit{Variogram}(d=3)\textit{ of } |\Gamma| \end{bmatrix} \tag{3.4}$$

where $\Gamma(u,v) = DFT\{\gamma(x,y)\}$ and $r_n$'s are equally spaced radii over frequency domain, $r_6$ being the maximum frequency.

The first rows of the feature vector represent total powers contained by each region. The remaining rows contain the maximum, average, energy and variance information of Discrete Fourier Transform magnitude of image block. The variogram of DFT magnitude is also included in the feature set for three different distance (*d*) scales as 1, 2 and 3 pixels.

### 3.1.5 Gabor filters

A dictionary of Gabor filters is computed in order to analyze directional components. Rangayyarn et al. [59] propose an approach that results in the following formulas for computing the filter parameters $\sigma_u$ and $\sigma_v$:

$$W = U_h \tag{3.5}$$

$$a = \left(\frac{U_h}{U_l}\right)^{\frac{1}{S-1}} \tag{3.6}$$

$$\sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2\ln(2)}} \tag{3.7}$$

$$\sigma_v = \frac{\tan(\frac{\pi}{2K})\left[U_h - (\frac{\sigma_u^2}{U_h})2\ln(2)\right]}{\sqrt{2\ln(2) - \frac{(2\ln(2))^2\sigma_u^2}{U_h^2}}} \tag{3.8}$$

where $U_l$ and $U_h$ denote the lower and upper center frequencies of interest. The *K* and *S* parameters are, respectively, the number of orientations and the number of scales in the desired multi-resolution decomposition procedure.

In this work, the Gabor wavelets were projected by using four scales (S = 4) and six directions (K = 6) with the lower and upper center frequencies specified as $U_l = 0.05$ and $U_h = 4.5$ cycles/pixel, respectively in accordance with [59]. The filtering process was performed in the frequency domain.

Gabor filter features are shown in Equation 3.9. The means and variances of the Gabor filtered images used as features.

$$\mathbf{F}_{gabor} = \begin{bmatrix} F_{47} \\ F_{48} \\ F_{49} \\ F_{50} \\ . \\ F_{53} \\ F_{54} \\ \vdots \\ F_{93} \\ F_{94} \\ \end{bmatrix} = \mathit{Circular\ Shift}\left( \begin{bmatrix} mean_{G_{1,1}(\gamma)} \\ var_{G_{1,1}(\gamma)} \\ mean_{G_{1,2}(\gamma)} \\ var_{G_{1,2}(\gamma)} \\ \vdots \\ mean_{G_{1,4}(\gamma)} \\ var_{G_{1,4}(\gamma)} \\ \vdots \\ mean_{G_{6,4}(\gamma)} \\ var_{G_{6,4}(\gamma)} \\ \end{bmatrix} \right) \tag{3.9}$$

where $G_{k,s}(\gamma)$ denotes the $\gamma$ signal, filtered with the Gabor filter, $g_{k,s}$.

Gabor filters typically have directionality, which is not a desired property for building detection problem. A simple method, which is proposed in [64], is used to overcome this difficulty and make Gabor filter outputs approximately rotation invariant. The resulting feature vector is circularly shifted with the intention that the scale-orientation pair having the maximum mean is located at the beginning of the vector.

### 3.1.6 Haralick Features

As described in the background section, Haralick features are calculated in the polar form *δ=(d, θ)*, where *d* is called the intersample spacing distance and *θ* is called the angular orientation. In this work Haralick features are calculated using four different orientations and three different spacing distance which correspond to adjacent pixels at $0^\circ$, $45^\circ$, $90^\circ$ and $135^\circ$ and spacing distance as 1, 2, 3 and 4 pixels, respectively. The resulting feature measures are obviously orientation dependent. In this thesis, these measures are summed up for four different orientations to decrease the orientation dependency of the measures. As a result, six measures are computed for four different spacing distances (sixteen features in total). The feature vector is given in Equation 3.10 where *d denotes* spacing distance. The computations of these measures are explained in the previous section in details.

$$\mathbf{F}_{haralick} = \begin{bmatrix} F_{95} \\ F_{96} \\ F_{97} \\ F_{98} \\ F_{99} \\ F_{100} \\ \vdots \\ \vdots \\ F_{113} \\ F_{114} \\ F_{115} \\ F_{116} \\ F_{117} \\ F_{118} \end{bmatrix} = \begin{bmatrix} \zeta_{inertia}(d=1) \\ \zeta_{cluster\ shade}(d=1) \\ \zeta_{cluster\ prominence}(d=1) \\ \zeta_{homogeneity}(d=1) \\ \zeta_{energy}(d=1) \\ \zeta_{entropy}(d=1) \\ \vdots \\ \vdots \\ \zeta_{inertia}(d=4) \\ \zeta_{cluster\ shade}(d=4) \\ \zeta_{cluster\ prominence}(d=4) \\ \zeta_{homogeneity}(d=4) \\ \zeta_{energy}(d=4) \\ \zeta_{entropy}(d=4) \end{bmatrix} \tag{3.10}$$

### 3.1.7 Wavelet Analysis

Wavelet features are extracted by means of Discrete Wavelet Transform (DWT). In this work, daubechies-4 wavelet is used for wavelet analysis. A multi-resolution decomposition is achieved by applying the single-level wavelet decomposition 3 times recursively to low-frequency component. Then, energies and standard deviations (2 measures) of four components (LL: Low-Low, LH: Low-High, HL: High-Low and HH: High-High) for three levels are employed as features (24 features in total). The consequential feature vector is given in Equation 10 where $energy_{c,s}$ and $var_{c,s}$ are the energy and variance of the wavelet filtered signals of component $c$ at stage $s$.

$$\mathbf{F}_{wavelet} = \begin{bmatrix} F_{119} \\ F_{120} \\ F_{121} \\ F_{122} \\ F_{123} \\ F_{124} \\ \vdots \\ F_{141} \\ F_{142} \end{bmatrix} = \begin{bmatrix} energy_{LL,1} \\ var_{LL,1} \\ energy_{LH,1} \\ var_{LH,1} \\ energy_{HL,1} \\ var_{HL,1} \\ \vdots \\ energy_{HH,3} \\ var_{HH,3} \end{bmatrix} \tag{3.11}$$

### 3.1.8 HSV Domain Features

HSV domain features are selected as: mean, variance, mean of gradient magnitude and variance of gradient magnitude for hue, saturation and value components. In addition to these features, center pixel values for each component are included in the feature set.

Since the saturation and value components are linear data, common mean and variance formulas, given in Section 2.2.1 can be applied. On the other hand, Hue is an angular data and directional statistics is involved in mean and variance calculations as given in Equations 3.12-3.18

$$mean_h = \angle\left[\sum_x\sum_y\left(e^{2\pi j\ h(x,y)}\right)\right] \tag{3.12}$$

$$var_h = 1 - \frac{1}{N^2}\left|\sum_x\sum_y\left(e^{2\pi j\ h(x,y)}\right)\right| \tag{3.13}$$

$$d_{h,1}(x,y) = (h(x,y) - h(x-1,y) + 360^\circ)\ mod\ 360^\circ \tag{3.14}$$

$$d_{h,2}(x,y) = (h(x,y) - h(x,y-1) + 360^\circ)\ mod\ 360^\circ \tag{3.15}$$

$$\|\nabla h(x,y)\| = \sqrt{d_{h,1}^2(x,y) + d_{h,2}^2(x,y)} \tag{3.16}$$

$$mean_{\|\nabla h\|} = \frac{1}{N^2} \sum_x \sum_y \|\nabla h(x,y)\| \tag{3.17}$$

$$var_{\|\nabla h\|} = \sqrt{\frac{1}{N^2} \sum_x \sum_y (\|\nabla h(x,y)\| - mean_{\|\nabla h\|})^2} \tag{3.18}$$

where *h*(*x, y*), *s*(*x, y*) and *v*(*x, y*) functions denote channels of HSV image and $\angle$ denotes the angle of complex number. The HSV domain features used in this work are shown in Equation 3.19.

$$\mathbf{F}_{HSV} = \begin{bmatrix} F_{143} \\ F_{144} \\ F_{145} \\ F_{146} \\ F_{147} \\ F_{148} \\ F_{149} \\ \vdots \\ F_{154} \\ F_{155} \\ F_{156} \\ \vdots \\ F_{161} \end{bmatrix} = \begin{bmatrix} mean_{hue} \\ variance_{hue} \\ mean_{\nabla hue} \\ variance_{\nabla hue} \\ center\ pixel_{hue} \\ mean_{sat} \\ variance_{sat} \\ \vdots \\ center\ pixel_{sat} \\ mean_{value} \\ variance_{value} \\ \vdots \\ center\ pixel_{value} \end{bmatrix} \tag{3.19}$$

## 3.2 Feature Selection

Among the two classifiers used in this study, Adaboost uses itself selected features so there is no need to select features for Adaboost. For SVM, this is not the case. SVM uses predefined features for both training and classification. One way is applying no feature selection and using all the features in the set. However, this is not a good approach since there may be invaluable features in the feature set. Also using more features makes optimization phase of SVM more complex and extraction of all features takes more time than extraction of only useful features.

In this study, two feature selection methods are applied. First one is take advantage of feature selector nature of Adaboost. In every iteration of Adaboost, the weights of the samples in the training set are updated. The weights of incorrectly classified samples are increased and the weights of correctly classified samples are decreased. In the following iteration these new weights are used enabling more chance for incorrectly classified samples to be correctly classified. As a result, Adaboost selects features taking into account incorrect classifications and features selected by Adaboost can be used in training and classification of SVM.

The second way used for feature selection of SVM is feature histograms. By examining the histograms of features, the separation capacity of features can be obtained. The features having more separation capacity over the train set can be chosen. Also, the errors inferred from these histograms can also be used. In Appendix A, the normalized errors of the features are given as well as six example images with normalized feature values. For the extraction of these errors, the approach explained below is used. First, the histogram of the feature is obtained with two classes. Example histogram is given in Figure 3.3.

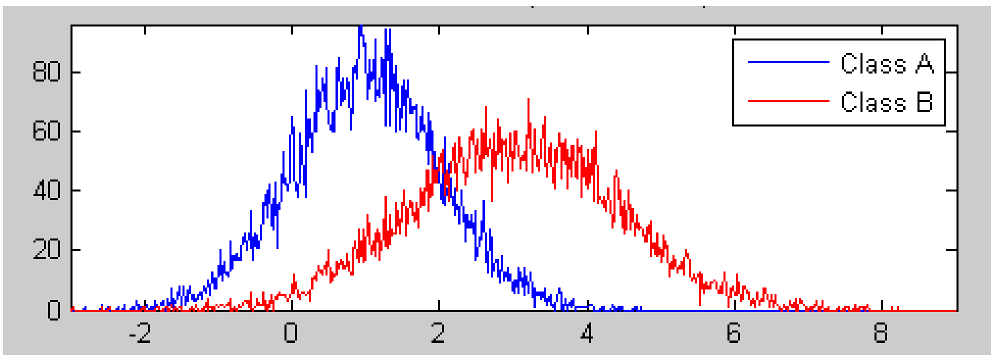


**Figure 3.3: Example Histogram for Calculating Histogram Errors**

The computation of the error is given below.

E1 corresponds to the probability a sample from class A is (mis)classified to class B. E2 corresponds to the probability a sample from class B is (mis)classified to class A. In order to estimate E1 and E2 the algorithm given below is used.

```
• Compute the histograms (h1 and h2) from the respective samples (Data1 and Data2).
• Initialize: E1 = 0 and E2 = 0.
• For each bin i in histogram h1:
    if (h1(i)<h2(i)) then E1 = E1 + h1(i)
    if (h1(i)>h2(i)) then E2 = E2 + h2(i)
• Normalize E1 and E2 (divide by the sum of the h1 and h2 respectively).
• Then, the errors are:
• E = (E1 + E2) /2
```

**Figure 3.4: Algorithm for Computation of the Errors**

All three approaches namely, no selection, Adaboost selections, selection using histogram errors are applied to SVM.

## 3.3 Classification

For the purpose of classification, Adaboost and four SVM kernels are used. These kernels are linear, polynomial, RBF and quadratic. The background information of the classifiers is given in Section 2.3 and 2.4. The detailed analyses regarding these classifiers are given in Section 4.

## 3.4 Post-processing

Two basic heuristics are used in post processing. First, one is area thresholding. Since the area of a building cannot be smaller than a threshold value, the connected regions smaller then that threshold are removed. Before this, a dilation operation is employed to break off the weak connections between regions. The second one is filling the holes, which are greater than the threshold value in connected regions.

# CHAPTER 4

# EXPERIMENTAL RESULTS

In this chapter, first, the sensitivity analyses are done, then detailed results for the selected classifiers are given, and finally, a comparison of classifiers is done.

## 4.1 Sensitivity analysis

In sensitivity analysis, we will see how the performance of building detection success is affected by changing various parameters. The analysis that we employed mainly cover the window type and size, feature types, iteration count or number of features used and training set. Adaboost and SVM with linear, quadratic, polynomial and RBF kernels are used as classifiers.

Five images are used for sensitivity analysis. These images are given in Figure 4.1 to Figure 4.5. The image in Figure 4.5 is taken from IKONOS and has a resolution of 1 meter. The other images are taken from Quickbird and their resolutions are 0.6 meters. We try to choose our images having different size and type of buildings.

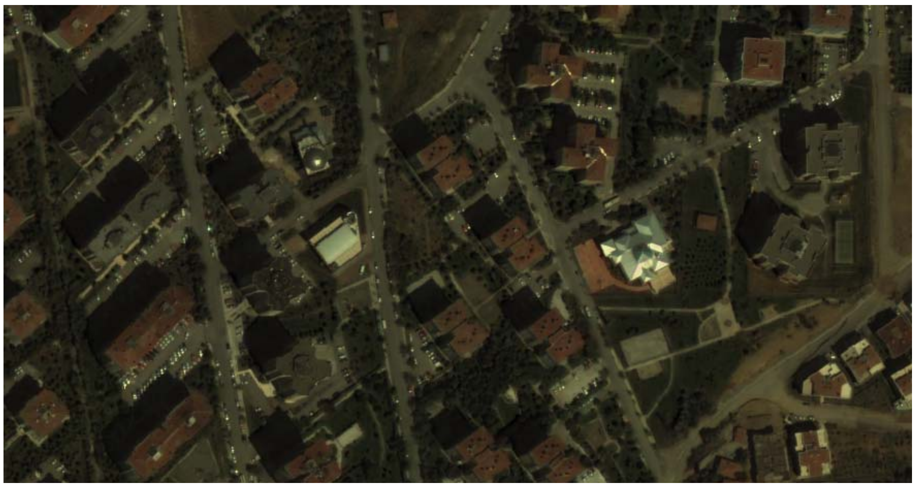


**Figure 4.1: 1. Image (Quickbird Image of Resolution 0.6 M)**

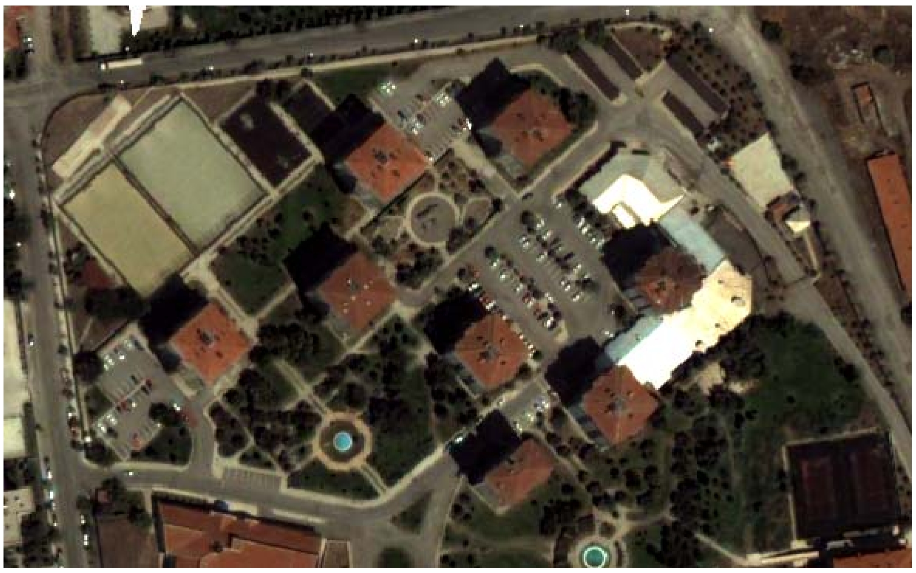


**Figure 4.2: 2. Image (Quickbird Image of Resolution 0.6 M)**

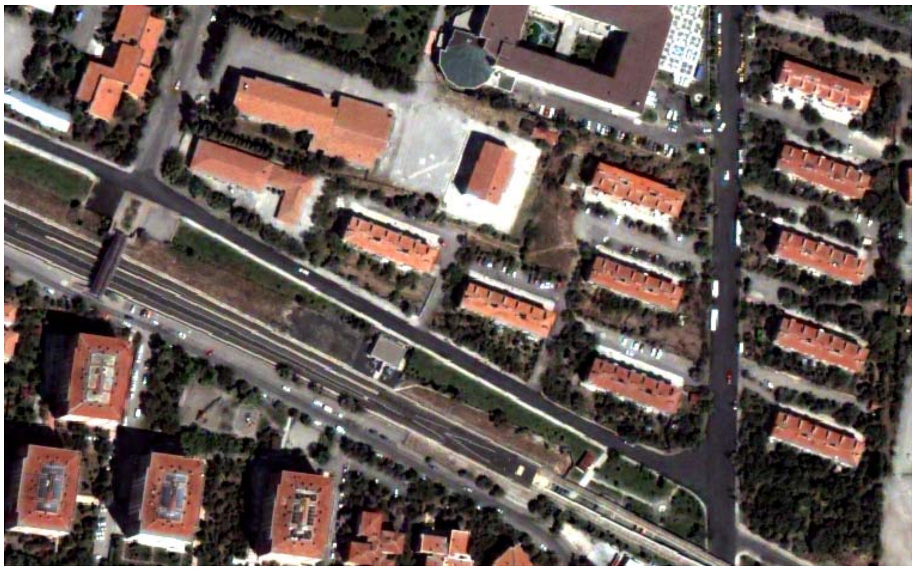


**Figure 4.3: 3. Image (Quickbird Image of Resolution 0.6 M)**

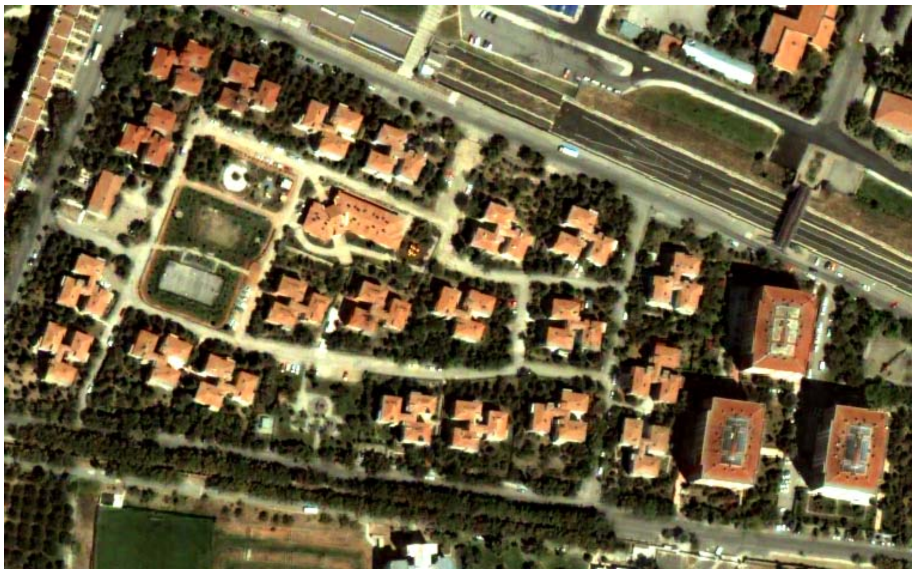


**Figure 4.4: 4. Image (Quickbird Image of Resolution 0.6 M)**

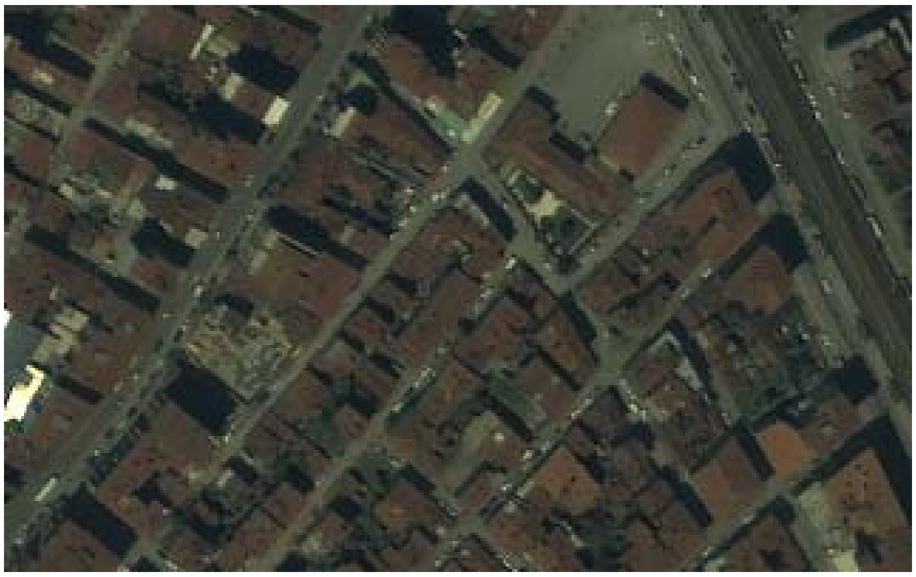


**Figure 4.5: 5. Image (IKONOS Image of Resolution 1 M)**

There are several parameters to be considered whose different combinations require an excessive number of experiments to be conducted. In order to set a reasonable number of experiments, we try to fix some of the parameters while changing others. First, we will begin with window type and size and fix a window type and size. Then we will go through the effect of changing feature set that is using intensity textural features, HSV domain features and all features. After selecting the feature set, we will see the effect of iteration count for Adaboost and number of features for SVM. We will select suitable iteration number and feature count. Also at this point, we will reduce the classifier number. Finally, we will see the effect of changing training set.

In order to do the selection of parameters, we need some performance measures. As pixel based performance, we use SF (split factor) where SF=FP/(TP+FP) that is the incorrectly detected regions that are not building over the all building output of the classifier, MF (missing factor) where MF=FN/(TP+FP) that is the regions that are building and not detected by the classifier over the all output of the classifier. PBD (percent of building detection) where PBD=100*TP/(TP+FN) is the correctly detected building regions percentage over the ground truth building regions. PBD is only a measure of building areas and it does not contain any information about incorrectly detected regions. Finally, QP (quality percentage) where QP=100*TP/(TP+FP+FN) gives a quick performance of the classifier. It uses TP count which is correctly detected building areas in the numerator and uses (TP+FP+FN) in the dominator where (TP+FN) gives all ground truth building pixels and (TP+FP) gives the classifier output i.e. dominator is formed by the union of ground truth buildings and classifier output. If correct detection of building pixels decreases that is TP decreases, QP decreases and also if incorrect detection of building pixels increases that is FP increases, again QP decreases. Table 4.1 shows an example of pixel performance values. As a result, higher PBD and QP values and lower SF and MF values are fine for us. The range for SF is 0 to 1, the range of MF is 0 to infinity and the range for PBD and QP is 0 to 100. In the ideal case which is corresponding to the first row of the Table 4.1, SF=0, MF=0, PBD=100 and QP=100. The last two rows of the Table 4.1 gives the results for giving all the pixels as building pixels and giving all the pixels as non-building pixels respectively. The building and non-building pixel numbers given in the last two rows are like our images in percentage. QP can give the overall performance by itself but the other measures are meaningful when used all together.

**Table 4.1 : Example of Pixel Performance Values**

| Example of pixel performance criteria | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| All pixels | GT Building pixels | GT non-building pixels | TP pixels | FP pixels | FN pixels | SF | MF | PBD | QP |
| 200 | 100 | 100 | 100 | 0 | 0 | 0 | 0 | 100 | 100 |
| 200 | 100 | 100 | 75 | 0 | 25 | 0 | 0,33 | 75 | 75 |
| 200 | 100 | 100 | 75 | 50 | 25 | 0.4 | 0.2 | 75 | 50 |
| 200 | 100 | 100 | 50 | 50 | 50 | 0,5 | 0,5 | 50 | 33 |
| 200 | 50 | 150 | 50 | 150 | 0 | 0.75 | 0 | 100 | 25 |
| 200 | 50 | 150 | 20 | 20 | 30 | 0.5 | 0.75 | 40 | 28.5 |
| 200 | 50 | 150 | 40 | 100 | 10 | 0.71 | 0.07 | 80 | 26 |
| 200 | 50 | 150 | 50 | 150 | 0 | 0.75 | 0 | 100 | 25 |
| 200 | 50 | 150 | 0 | 0 | 50 | 0 | nan | 0 | 0 |

Object based performance is the other measure that we use to evaluate performance. In object-based performance, performance evaluation is done based on objects. Objects are the connected pixel groups. Object based performance is evaluated using the intersection percentage of objects in the ground truth and in the output of the classifier. If this intersection percentage is higher than a threshold value “T” and there is no intersection with other objects, the object is an instance of “correct detection”. If an object in the ground truth intersects more than one object in the output and the intersection percentage is higher than the threshold value, this is an instance of “over detection”. If an object in the output intersects more than one object in the ground truth and the intersection percentage is higher than the threshold value, this is an instance of “under detection”. If an object in the ground truth has an intersection percentage less than the threshold value, this is an instance of “missed detection”. If an object in the output has an intersection percentage less than the threshold value, this is an instance of “false alarm”. Correct detections are the measure that we desired to increase, however in case some of the buildings are not correctly detected then we prefer them to be still detected by over detection and under detection. Missed detections and false alarms are the measures that we desire to decrease. The examples of these measures are given in Figure 4.6. In that figure, object detection threshold value, T = 0.4. In the figure, reds are “correct detection”, yellows are “missed detection”, blues are “under detection”, greens are “over detection” and finally, purples are “false alarm”.

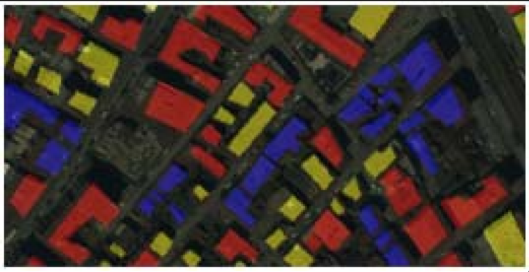
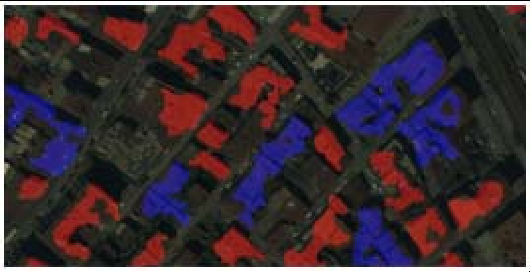
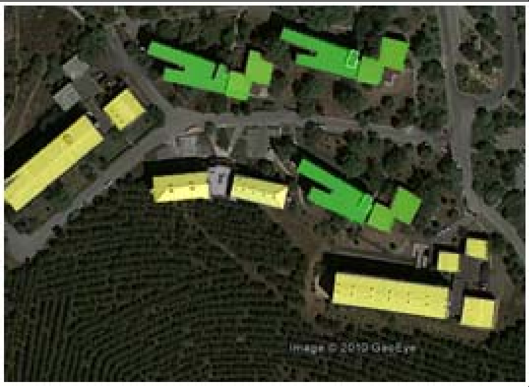
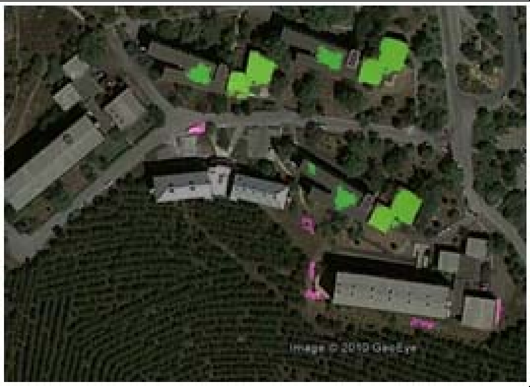
| | |
|---|---|
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

**Figure 4.6: Example of Object Detection Performance: a) Ground Truth Objects of Example Image 1, b) Classifier Output for Image 1, c) Ground Truth Objects of Example Image 2, d) Classifier Output for Image 2.**

### 4.1.1 Window Size Analysis

Our analyses start by choosing an appropriate window type and size. In order to reduce the number of experiments first we have to fix some parameters. All the features are used in this analysis and two feature selection approaches are applied. The first one is selection by Adaboost and the second one is selection according to histogram errors. The iteration count for Adaboost or the features used in SVM is fixed to 16. This number is decided by some priory experiments by observing the operation of Adaboost and SVM. Later, a further analysis will be done by considering the effect of changing iteration and feature count. Three different sets of features are formed for SVM kernels. These are features selected by Adaboost, features selected using histogram errors and all 161 features in the feature set. Adaboost also used 161 features with 16 iterations and use convenient features for training and classification. As training set %5 percent of all images are used to reflect all properties of the images and performance evaluation is done in the other %95. Similarly, a further analysis will be given considering the effect of training set. In case of a memory problem %5 percent training images are appropriately down sampled. In fact in SVM kernels that is the case and training set is reduced to 100-300 feature vectors while using all features and 150-400 feature vectors while using selected features. No object performance is given in this analysis. Only average QP values obtained through 5 images are used to evaluate the performance since QP both includes FN and FP values.

Window size analysis includes two different window types. In the first one, feature extraction is in non-overlapping windows and labeling (i.e. building or non-building) includes the whole window. If half of the pixels in the window belong to building, the window is in building class and vice versa. In classification stage, the window is also classified as building or non-building. Thus, the classification resolution of the image is reduced proportional to window size. Window sizes from 3 to 19 will be examined. In second style, features extracted are assigned to the center pixel. Feature extraction goes on an overlapping manner. In training and classification, the label of the center pixel is used.

First off all, non-overlapping window analysis is presented. In these analyses, Adaboost is used with 16 iterations. Figure 4.7 shows the QP values for different window sizes. In this figure, SVM kernels used all the features in the feature set. There was no selection of features for SVM kernels. Figure 4.8 illustrates the window size performance of SVM kernels with 16 features selected by Adaboost. Finally, Figure 4.9 presents the QP for SVM kernels with minimum histogram error features.

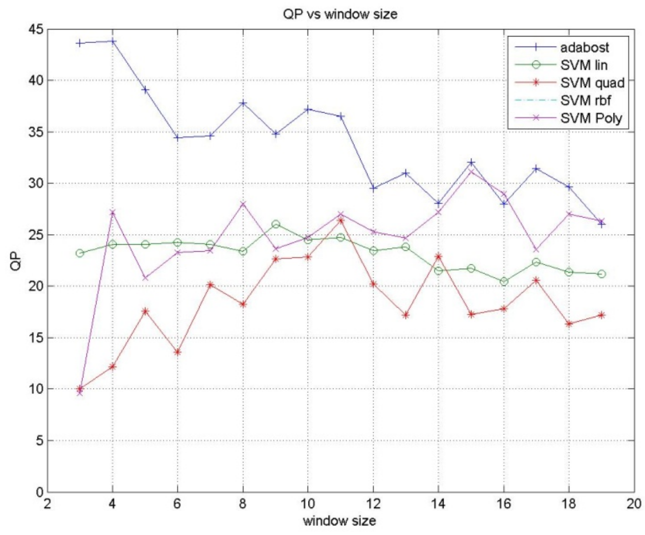


**Figure 4.7: Non-Overlapping Window; QP vs. Window Size Using All Features**

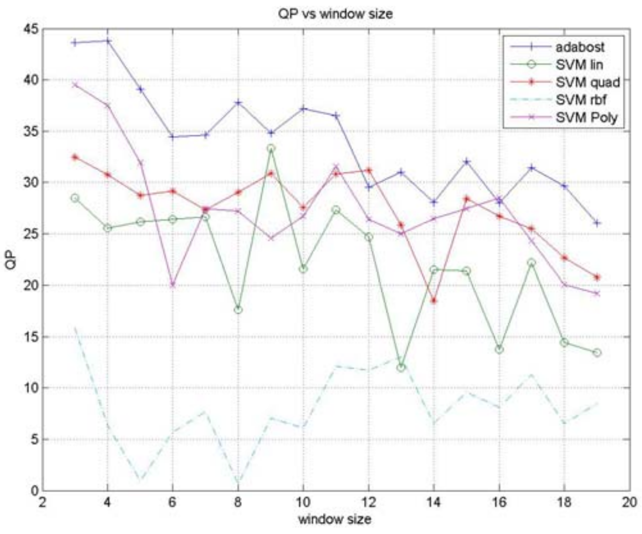


**Figure 4.8: Non-Overlapping Window; QP vs. Window Size Using Features Selected by Adaboost**

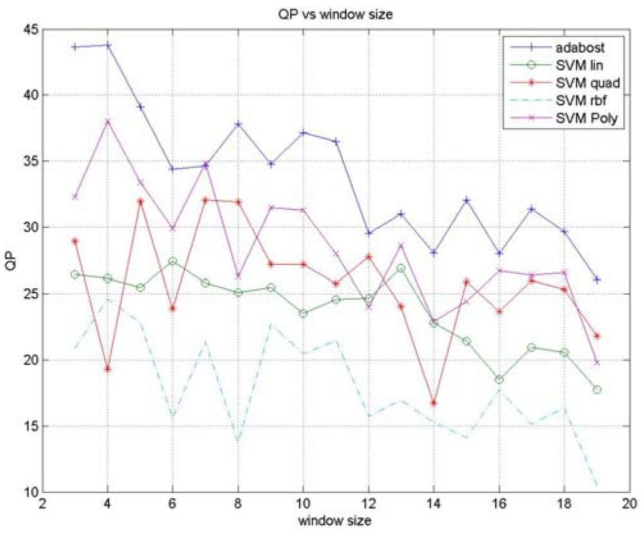


**Figure 4.9: Non-Overlapping Window; QP vs. Window Size Using Features with Minimum Histogram Errors**

There is a convergence problem for the RBF kernel while using all features. The performance of both Adaboost and SVM kernels are decreasing with increasing window size since the resolution of classification decreases with increasing window size i.e. the performance evaluation is done pixel based but classification is done with window size. An increase in the performance of SVM kernels is observed if feature selection either by Adaboost or by histogram errors is applied.

It is the second window type which features extracted are assigned to the center pixel. Feature extraction goes on an overlapping manner. In training and classification, the label of the center pixel is used.

Adaboost selects 16 features and uses them in classification in all sets. In first set, SVM kernels are fed by all the features and Figure 4.10 shows the results. Figure 4.11 illustrates the result of SVM kernels using Adaboost selected features. Figure 4.12 shows the results for SVM kernels fed by features with minimum histogram errors.

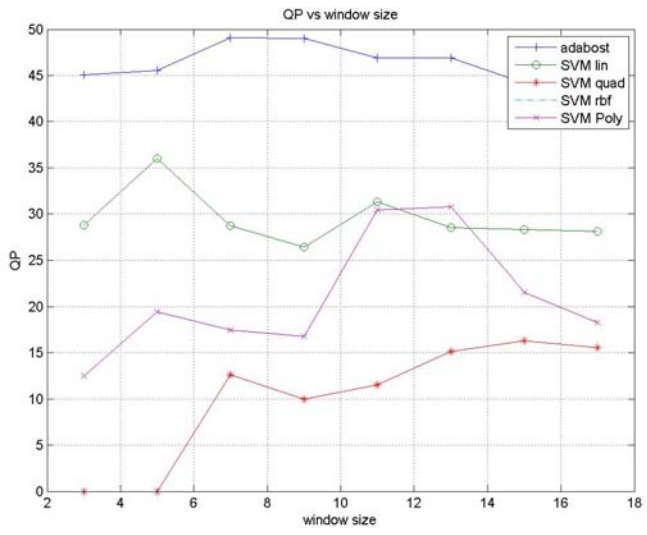


**Figure 4.10: Overlapping Window; QP vs. Window Size Using All Features**

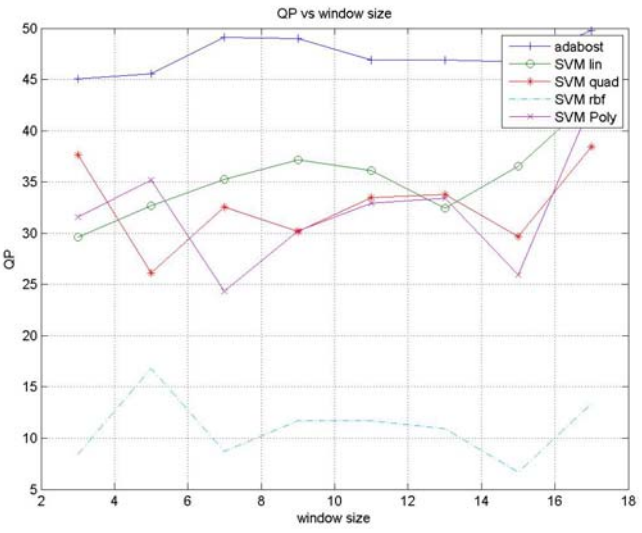


**Figure 4.11: Non-Overlapping Window; QP vs. Window Size Using Features Selected by Adaboost**

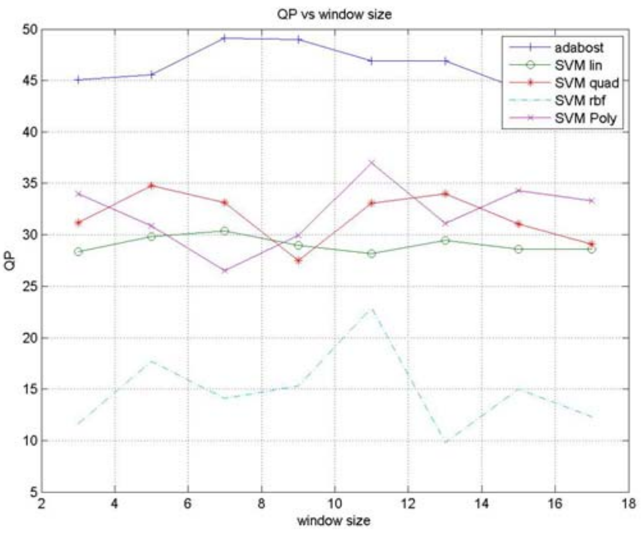


**Figure 4.12: Non-Overlapping Window; QP vs. Window Size Using Features with Minimum Histogram Errors**

At first glance, overlapping window performances are better than non-overlapping window performances. That is what we expect. Applying overlapped window increases the classification resolution on the image. In addition, there is an increase in the performance if feature selection is applied rather than using all features. After deciding to use overlapping window and feature selection, we have to choose a window size and continue the further analyses using this window size. The performance of all the methods has a tendency first to increase as the window size increases but then it decreases. This is also what we expect. If the window size is too small, it is not sufficient to extract some textural features. If it is too large, the regions belonging to both building and non-building are covered together in the window so discrimination power of the feature is reduced. Adaboost has best performance in window sizes 7 and 9. Performances in 5, 11 and 13 are also fine. SVM kernels using Adaboost selections have better results for window sizes 9 and 11. SVM kernels using minimum errors have better performances for 5, 11 and 13. Window size of 11 seems a fair and good selection for all of the classifiers.

In conclusion, overlapping window performances are better than non-overlapping window performances and feature selection increases the performance. Window size 11 is the most convenient one when overall performances of all the methods are considered. Further analyses on other parameters will be done with window size 11 and overlapping windows.

### 4.1.2 Feature Analysis

In feature analysis, the pixel performance of Adaboost and four SVM kernels are investigated. While doing experiments overlapping window with size 11 is used. The iteration count for Adaboost is 16 and feature count for SVM is 16. As training set, %5 of all images are used and performance is evaluated in other %95 part of the image. The average performance obtained through 5 images is considered.

In first set, only intensity textural features are used. Adaboost is a method which selects its own features and uses them in classification. Therefore, no additional step is needed for feature selection for Adaboost. However, SVM kernels are fed by three different set of features. The first one is all textural features in the feature set. Second one is the feature set selected by Adaboost and the third set is features with minimum histogram errors. SF, MF, PBD and QP values for 9 classifier are given in Table 4.2. The best values obtained among Adaboost and SVM kernels for each performance criteria is indicated bold in the table.

**Table 4.2: SF, MF, PBD and QP Values for 9 Classifier with Intensity Textural Features**

| INTENSITY TEXTURAL FEATURES PERFORMANCE | | | | | |
|---|---|---|---|---|---|
| Classifier/performance | | SF | MF | PBD | QP |
| ADABOOST | | 0.68 | 0.13 | 71.8 | 28.3 |
| SVM ALL TEXTURAL FEATURES | LINEAR | 0.93 | 105.5 | 0.84 | 0.82 |
| | QUADRATIC | 0.90 | 13.91 | 1.62 | 1.33 |
| | RBF | 0 | 0 | 0 | 0 |
| | POLYNOMIAL | 0.75 | 9.15 | 15.9 | 9.2 |
| SVM FEATURES SELECTED BY ADABOOST | LINEAR | 0.86 | 35.3 | 2.37 | 2.12 |
| | QUADRATIC | 0.54 | 2.36 | 23.9 | 17.6 |
| | RBF | 0.78 | 52.63 | 3.84 | 3.51 |
| | POLYNOMIAL | 0.71 | 17.6 | 9.37 | 6.52 |
| SVM FEATURES WİTH MINIMUM ERROR | LINEAR | 0.83 | 36.3 | 3,01 | 2.85 |
| | QUADRATIC | 0 | 0 | 0 | 0 |
| | RBF | 0.63 | 3.15 | 18.6 | 9,30 |
| | POLYNOMIAL | 0 | 0 | 0 | 0 |

Adaboost has a high PBD value but QP value is quite low. In addition, a high SF and low MF shows us that the classifier found most of the building areas but with a high rate of FP. Within the SVM kernels, only quadratic kernel with Adaboost selected features has remarkable performance but it also has much lower PBD value and lower QF value than Adaboost. Intensity textural features have less separation capacity compared to HSV features and here we can see that Adaboost can form a strong classifier from weak classifiers but SVM kernels have poor performances while using features with less separation capacity.

In the second set, only HSV domain features are used. Adaboost selects its own features and uses them in classification. SVM kernels are fed by three different set of features. The first one is all textural features within the HSV domain. The second one is the HSV feature set selected by Adaboost and the third set is the HSV domain features with minimum histogram errors. SF, MF, PBD and QP values for 9 classifier are given in Table 4.3. Again, the best values obtained among Adaboost and SVM kernels for each performance criteria are indicated bold in the table.

**Table 4.3: SF, MF, PBD and QP Values for 9 Classifier with HSV Domain Features**

| HSV FEATURES PERFORMANCE | | | | | |
|---|---|---|---|---|---|
| Classifier/performance | | SF | MF | PBD | QP |
| ADABOOST | | 0.42 | 0.49 | 67.61 | 41.82 |
| SVM ALL FEATURES | LINEAR | 0.23 | 1.46 | 47,6 | 40.3 |
| | QUADRATIC | 0.21 | 1.07 | 45.74 | 41.29 |
| | RBF | 0.16 | 22.28 | 4.11 | 4.07 |
| | POLYNOMIAL | 0.25 | 1.53 | 43.8 | 37.2 |
| SVM FEATURES SELECTED BY ADABOOST | LINEAR | 0.32 | 1.13 | 38.1 | 31.9 |
| | QUADRATIC | 0.27 | 14.0 | 35.1 | 31.3 |
| | RBF | 0.086 | 13.91 | 5.35 | 5.29 |
| | POLYNOMIAL | 0.27 | 2.05 | 41.7 | 34.99 |
| SVM FEATURES WİTH MINIMUM ERROR | LINEAR | 0.27 | 1.86 | 28.93 | 24.31 |
| | QUADRATIC | 0.16 | 1.12 | 38.5 | 35.52 |
| | RBF | 0.14 | 1.07 | 39.47 | 36.75 |
| | POLYNOMIAL | 0.089 | 16.09 | 39.35 | 36.43 |

Again, Adaboost has the best performance over all but this time all SVM kernels have also good results except RBF with all features and RBF with Adaboost selected features. Adaboost have acceptable SF and MF values which means that the classifier output have nearly same FN and FP values but in SVM kernels SF values are low i.e. FP values are low. The MF values of SVM kernels are high that is FN values are high. Nearly 25 percent of the images are building pixels. The size of non-building pixels is nearly three times larger than the size of building pixels so the SF values lower than 0.5 is reasonable. To conclude the results, Adaboost finds much of the building pixels but also finds non-building pixels as building pixels and SVM kernels finds less building pixels also they find less non-building pixels as building pixels.

In third set all the features are used. Adaboost selects its own features and uses them in classification. SVM kernels are fed by three different set of features. The first one is all features in the feature set. The second one is the feature set selected by Adaboost and the third set is features with minimum histogram errors. SF, MF, PBD and QP values for 9 classifier are given in

Table **4.4**.

**Table 4.4: SF, MF, PBD and QP Values for 9 Classifier with All Features**

| ALL FEATURES PERFORMANCE | | | | | |
|---|---|---|---|---|---|
| Classifier/performance | | SF | MF | PBD | QP |
| ADABOOST | | 0.35 | 0.45 | 67.8 | 46,9 |
| SVM ALL FEATURES | LINEAR | 0.23 | 6.09 | 33.5 | 31.2 |
| | QUADRATIC | 0.54 | 14.6 | 12.9 | 11.5 |
| | RBF | 0 | 0 | 0 | 0 |
| | POLYNOMIAL | 0.35 | 3.7 | 36.9 | 30,3 |
| SVM FEATURES SELECTED BY ADABOOST | LINEAR | 0.15 | 1.04 | 39.9 | 36.1 |
| | QUADRATIC | 0.29 | 1.32 | 39.5 | 33.4 |
| | RBF | 0.14 | 5.44 | 12.1 | 11.7 |
| | POLYNOMIAL | 0.21 | 2.72 | 39.9 | 32.9 |
| SVM FEATURES WİTH MINIMUM ERROR | LINEAR | 0.21 | 1.59 | 32.3 | 28.1 |
| | QUADRATIC | 0.11 | 1.34 | 35.1 | 33.0 |
| | RBF | 0.19 | 2.25 | 24.3 | 22.9 |
| | POLYNOMIAL | 0.27 | 0.85 | 43.6 | 36.9 |

Again, Adaboost classifier has the best performance and the performance is slightly increased compared to HSV domain performance. The performance of SVM kernels are slightly decreased compared to HSV domain.
The best performance is achieved is in Adaboost by using all features. SVM kernels have slightly better performance in HSV domain. Also in the absence of RGB image, Adaboost with intensity textural features may be useful. In iteration count analysis, SVM kernels will be used with all features and HSV domain features. The analysis of feature count for SVM with intensity textural features will not be done since SVM kernels are not useful with low separation capacity features.  Adaboost will be used with textural features, HSV domain features and all features.

### 4.1.3 Iteration and feature count analysis

Now that we fixed window to overlapping with size 11 and decided to use all features for all SVM kernels, intensity textural features only for Adaboost. HSV domain features also will be used for Adaboost and one of the SVM kernels that gives the best result in all features count analysis. In the past analyses %5 off all images were used in training and pixel performance in unused %95 was evaluated. In iteration count analyses, upper %20 part of the all images will be used in training and object performance with building detection threshold of 0.4 and pixel performance with PBD and QP will be evaluated in the lower %80 part of the images. These analyses will be done up to 30 features, keeping in mind that using the classifier with 161 features takes much time especially in feature extraction phase. Using more than 30 features will begin to take also much time.

One important thing to keep in mind is that Adaboost uses selected features in a weighted way but SVM kernels uses them without weighting i.e. all features in SVM kernels have the same importance. Before starting the analysis, an example table will be given in order to explain how Adaboost uses features and also in the analysis, these features will be used by SVM kernels. In addition, SVM kernels will use the features with minimum histogram errors and example of this table will also be given.

#### 4.1.3.1 Iteration and feature count analysis for all features

The performance of Adaboost using all features and the performance of four SVM kernels will be investigated in this section with the changing iteration count for Adaboost and the changing feature count for SVM kernels. For SVM kernels, two different sets of features are used. First, one is the features selected using histogram errors. Second one is the features selected by Adaboost. Note that the features multiply selected by Adaboost are removed from SVM feature set.

Before starting iteration and feature count analysis, introducing the features selected by Adaboost and the features selected using histogram errors would be better since this analysis is done using these features.
Table 4.5 illustrates the features selected by Adaboost over all features. In this table, first three columns are selection order (Adaboost iteration count), feature name for this order and the number of the feature in the feature set respectively. The forth column shows if this feature is previously selected. The fifth column is the feature count discarding the features selected more than one. The last row is the error in the current classification. Note that, after each classification the weights of training samples changes and errors are calculated according to the weights of the training set. The weight changes in training samples are done using the Beta value. The weights of incorrectly classified samples are increased and the weights of correctly classified samples are decreased. By doing so, the feature that is capable of classifying incorrectly classified samples is selected in next iteration. After each iteration, the errors are calculated using new weights of training samples for each feature (weak classifier) and the feature with minimum error is selected next. Weight of the weak classifier, which is given in the seventh column of the table, is calculated using errors and threshold is the optimal value for giving minimum error. Normalized threshold is the normalized value of the threshold to [0,1] interval over the entire set. While doing classification, if the feature value of the sample in classification is smaller than the threshold value and the parity is positive then this weight is added to the overall sum. If the value of the sample in classification is larger than the threshold value and the parity is positive then this weight is subtracted from the overall sum. If parity is negative, vice versa. In other words, positive parity shows that the feature value should be less than threshold in order to be classified as building and negative parity shows

that the feature value should be more than threshold in order to be classified as building. Then the overall sum is checked. If it is larger than the half of sum of all weights, the sample in classification is marked as building. In Table 4.5, the feature selected in first order corresponds to the red values. That is in many cases the rooftops have this color. The second feature selects the highly illuminated pixels. The third one is highly saturated colors. The fourth one corresponds to low variation in saturation gradient. Up to this point, everything seems like reflecting the buildings in the training set. The fifth one is the colors discarding red, orange and some yellow in the visible spectrum. This is selection of center pixel value of hue for the second time. This one does not reflect the properties of most of the buildings. However, this might be due to the a few buildings having different rooftops and some of the objects in the rooftops such as chimneys have different colors. The samples of incorrectly classified rooftops in the training set start to get high weights. These changes in weights forced Adaboost to select this feature. Note that most of the features are either HSV domain features or basic features. The features other than HSV domain or basic features appears after seventeenth feature and still basic features and HSV domain features appear after that point.

**Table 4.5: Features Selected by Adaboost over All Feature Set**

| Iteration count | Feature name | Feature number | Previous selection | Feature count | Parity | Weight Of the classifier | Threshold | Normalized threshold | Error | Beta |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Center pixel value of hue | 147 | N | 1 | 1 | 1.221 | 0.09 | 0.1 | 0.22 | 0.29 |
| 2 | Center pixel value of value | 161 | N | 2 | -1 | 0.799 | 0.22 | 0.22 | 0.31 | 0.44 |
| 3 | Mean of saturation | 148 | N | 3 | -1 | 0.520 | 0.44 | 0.48 | 0.37 | 0.59 |
| 4 | Variance of saturation gradient | 151 | N | 4 | 1 | 0.38 | 0.10 | 0.19 | 0.40 | 0.67 |
| 5 | Center pixel value of hue | 147 | Y | 4 | -1 | 0.537 | 0.32 | 0.33 | 0.36 | 0.58 |
| 6 | Mean of intensity | 1 | N | 5 | -1 | 0.369 | 0.13 | 0.13 | 0.40 | 0.69 |
| 7 | Center pixel value of value | 161 | Y | 5 | -1 | 0.327 | 0.67 | 0.67 | 0.41 | 0.72 |
| 8 | Entropy of intensity | 5 | N | 6 | 1 | 0.363 | 5.25 | 0.75 | 0.41 | 0.69 |
| 9 | Variance of saturation | 149 | N | 7 | -1 | 0.347 | 0.05 | 0.11 | 0.41 | 0.70 |
| 10 | Mean of hue gradient | 145 | N | 8 | 1 | 0.209 | 0.02 | 0.07 | 0.44 | 0.81 |
| 11 | Mean of hue | 143 | N | 9 | -1 | 0.242 | 0.21 | 0.21 | 0.43 | 0.78 |
| 12 | Mean of saturation gradient | 150 | N | 10 | -1 | 0.324 | 0.02 | 0.03 | 0.41 | 0.72 |

**Table 4.5 - Continued**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 13 | Mean of saturation | 148 | Y | 10 | -1 | 0.254 | 0.48 | 0.53 | 0.43 | 0.77 |
| 14 | Skewness of DFT | 41 | N | 11 | -1 | 0.286 | 8.80 | 0.79 | 0.42 | 0.75 |
| 15 | Center pixel value of hue | 147 | Y | 11 | -1 | 0.237 | 0.41 | 0.42 | 0.44 | 0.78 |
| 16 | Variance of value gradient | 158 | N | 12 | -1 | 0.212 | 0.01 | 0.04 | 0.44 | 0.80 |
| 17 | Variance of Gabor filter; fourth order | 54 | N | 13 | 1 | 0.225 | 0.00 | 0.06 | 0.44 | 0.79 |
| 18 | Wavelet variance(HH,1) | 126 | N | 14 | -1 | 0.236 | 0.01 | 0.1 | 0.44 | 0.78 |
| 19 | Mean of DFT | 38 | N | 15 | 1 | 0.180 | 0.69 | 0.22 | 0.45 | 0.83 |
| 20 | Variance of value gradient | 158 | Y | 15 | -1 | 0.223 | 0.02 | 0.08 | 0.44 | 0.79 |
| 21 | Center pixel value of sat | 154 | N | 16 | -1 | 0.179 | 0.51 | 0.51 | 0.45 | 0.83 |
| 22 | Mean of sat gradient | 150 | Y | 16 | 1 | 0.215 | 0.08 | 0.14 | 0.44 | 0.80 |
| 23 | Center pixel value of value | 161 | Y | 16 | -1 | 0.182 | 0.64 | 0.64 | 0.45 | 0.83 |
| 24 | Center pixel value of hue | 147 | Y | 16 | -1 | 0.189 | 0.09 | 0.1 | 0.45 | 0.82 |
| 25 | Center pixel value of hue | 147 | Y | 16 | 1 | 0.157 | 0.13 | 0.14 | 0.46 | 0.85 |
| 26 | Zernike of saturation | 152 | N | 17 | -1 | 0.151 | 2.06 | 0.17 | 0.46 | 0.85 |
| 27 | Mean of Gabor filter; twelfth order | 69 | N | 18 | -1 | 0.190 | 0.13 | 0.12 | 0.45 | 0.82 |

**Table 4.5 - Continued**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 28 | variance of Gabor filter; fourteenth order | 74 | N | 19 | 1 | 0.203 | 0.02 | 0.07 | 0.44 | 0.81 |
| 29 | Mean of hue | 143 | Y | 19 | -1 | 0.151 | 0.49 | 0.49 | 0.46 | 0.85 |
| 30 | Mean of saturation gradient | 150 | Y | 19 | -1 | 0.197 | 0.01 | 0.02 | 0.45 | 0.82 |

Figure 4.13 and Figure 4.14 illustrates the pixel and object performance of Adaboost vs. iteration count. Here a high performance is reached using only first feature and then performance does not change much since twenty-fifth feature and considering both pixel and object performances highest performance is achieved using 28 features. The QP value reached while using 28 features is nearly 47.

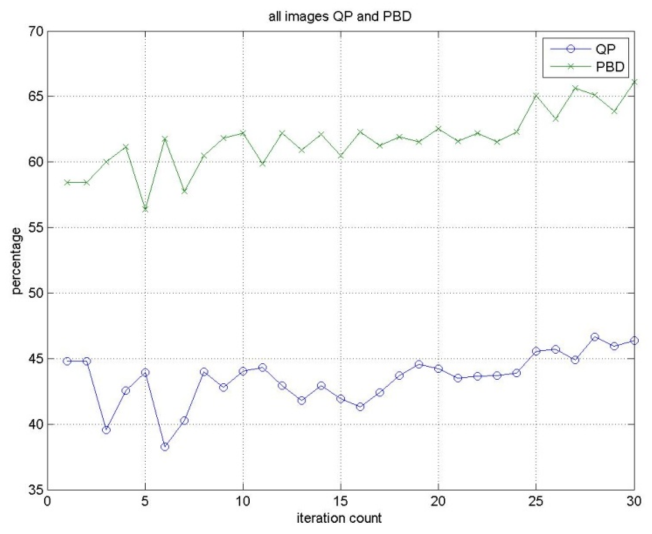


**Figure 4.13: QP and PBD Values vs. Iteration Count for Adaboost with All Features**

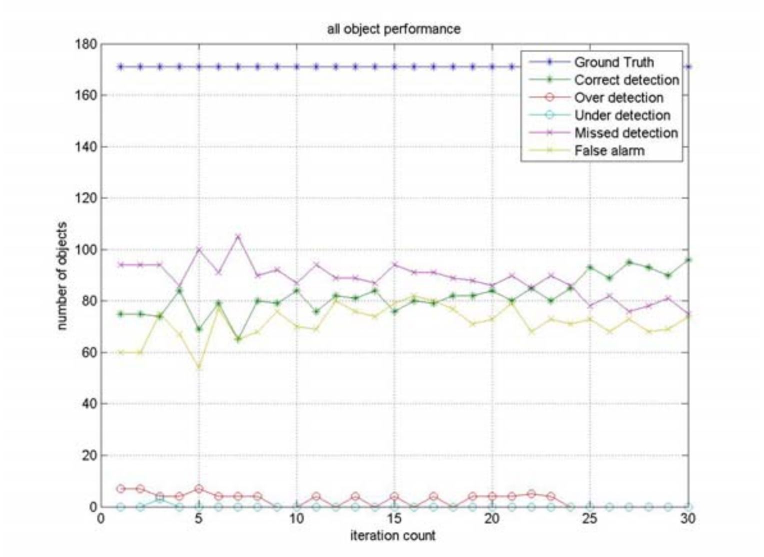


**Figure 4.14: Object Performance vs. Iteration Count for Adaboost with All Features**

The second set of SVM features is selected by using histogram errors.Table 4.6 shows the error of first 30 features. In this table no is the order of the feature considering the histogram error given in the error column. Feature name is the name of the feature. Feature number is the number of the feature in the feature set. Also in appendix A, the histogram errors of all features with example images are given.  Again, the first feature is centre pixel value of hue. The second feature is mean of hue. Mean of hue has very low error compared with the other features in the set and it has similar separation capacity as center pixel value of hue has i.e. they classify nearly the same things. Adaboost first selects the center pixel value of hue and does not select mean of hue since after the weights in training set changes with the center pixel value of hue. It is more convenient not to select mean of hue after selection of center pixel of hue since they are somehow correlated. In addition, there are many Gabor filter selections in the set, which indicates Gabor filters are more capable of discriminating the two classes compared with other intensity textural features. Note that features selected by Adaboost other than HSV domain features and basic features includes Gabor features in Table 4.6 also note that hue has very high separation capacity compared to others.

**Table 4.6: Histogram Errors of Features over All Features**

| No | Feature name | Feature number | Error |
|---|---|---|---|
| 1 | Center pixel value of hue | 147 | 0.202 |
| 2 | Mean of hue | 143 | 0.2147 |
| 3 | Center pixel value of value | 161 | 0.2755 |
| 4 | Mean of saturation | 148 | 0.2813 |
| 5 | Center pixel value of saturation | 154 | 0.2834 |
| 6 | Mean of value | 155 | 0.3033 |
| 7 | Mean of Gabor filter; second order | 49 | 0.3237 |
| 8 | Mean of Gabor filter; fourteenth order | 73 | 0.3237 |
| 9 | Mean of intensity | 1 | 0.3237 |
| 10 | Zernike (0,0) | 12 | 0.3237 |
| 11 | Maximum value of DFT | 37 | 0.3237 |
| 12 | Mean of Gabor filter; first order | 47 | 0.3237 |
| 13 | Mean of Gabor filter; third order | 51 | 0.3237 |
| 14 | Mean of Gabor filter; fourth order | 53 | 0.3237 |
| 15 | Mean of Gabor filter; thirteenth order | 71 | 0.3237 |
| 16 | Mean of Gabor filter; fifteenth order | 75 | 0.3237 |
| 17 | Mean of Gabor filter; sixteenth order | 77 | 0.3237 |
| 18 | Mean of Gabor filter; eighth order | 61 | 0.3238 |
| 19 | Mean of Gabor filter; twentieth order | 85 | 0.3238 |
| 20 | Mean of Gabor filter; twelfth order | 69 | 0.3239 |
| 21 | Mean of Gabor filter; twenty-fourth order | 93 | 0.3239 |
| 22 | Variogram of DFT with 1 distance | 44 | 0.327 |
| 23 | Ring 6 on PS | 36 | 0.3284 |
| 24 | Mean of Gabor filter; eleventh order | 67 | 0.3311 |
| 25 | Mean of Gabor filter; twenty-third order | 91 | 0.3311 |
| 26 | Variance of DFT | 40 | 0.3313 |
| 27 | Ring1 on PS | 31 | 0.3316 |
| 28 | Mean of Gabor filter; seventh order | 59 | 0.3317 |
| 29 | Mean of Gabor filter; twenty-first order | 83 | 0.3317 |
| 30 | Variogram of DFT with 2 distance | 45 | 0.3319 |

SVM often provides significantly better classification performance than other machine learning algorithms in reasonable sized datasets. They involve computationally expensive processes. SVM uses an optimization algorithm by selecting support vectors in order to maximize the margin between the classes which is hard to interpret intuitively in terms of components i.e. features of the support vectors. Although there exists some methods to choose parameters and kernels of SVM, the best way is doing experiments and choosing the best one [60].

Figure 4.15 to Figure 4.18 illustrates the pixel and object performances of SVM linear kernel with both features selected by using histogram errors and features selected by Adaboost. The maximum QP value is 42 with 8 features when using histogram errors and the saturation is at 8 features. When using Adaboost selections QP performance saturated with 9 features and maximum QP value achieved with 16 features which is 40.

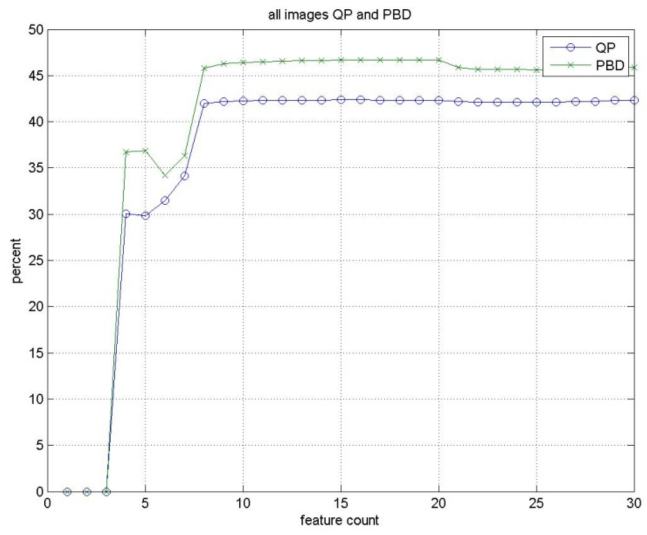


**Figure 4.15: Pixel Performance of SVM Linear Kernel Features Selected Using Histogram Errors over the Entire Set Of Features**

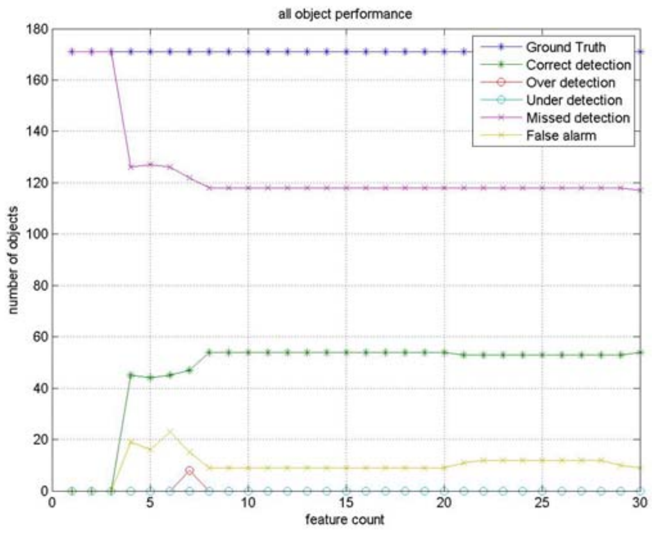


**Figure 4.16: Object Performance of SVM Linear Kernel Features Selected Using Histogram Errors over the Entire Set Of Features**

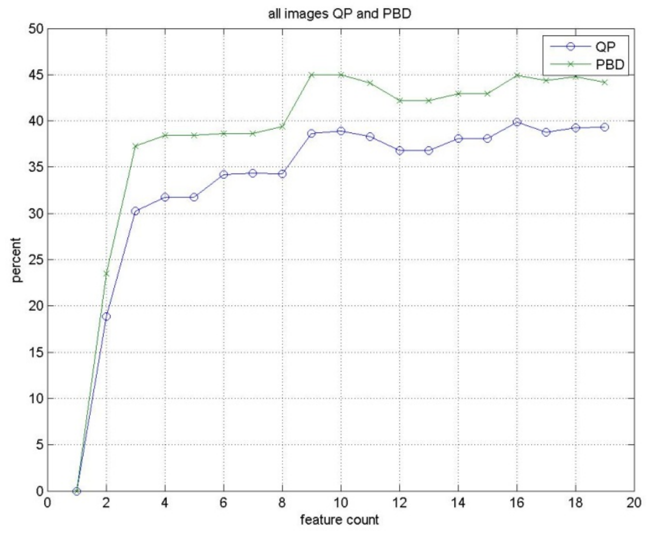


**Figure 4.17: Pixel Performance of SVM Linear Kernel Features Selected Using Adaboost Over the Entire Set of Features**

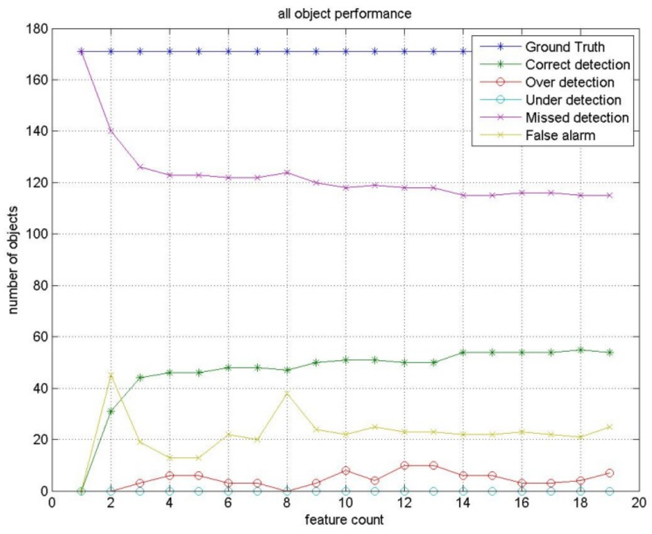


**Figure 4.18: Object Performance of SVM Linear Kernel Features Selected Using Adaboost Over the Entire Set of Features**

Figure 4.19 to Figure 4.22 illustrates the pixel and object performances of SVM polynomial kernel with both features selected by using histogram errors and features selected by Adaboost. The maximum QP value is 43 with 7 features when using histogram errors and the saturation is at 3 features. When using Adaboost selections QP performance saturated with 4 features and maximum QP value achieved with 4 features which is 44.

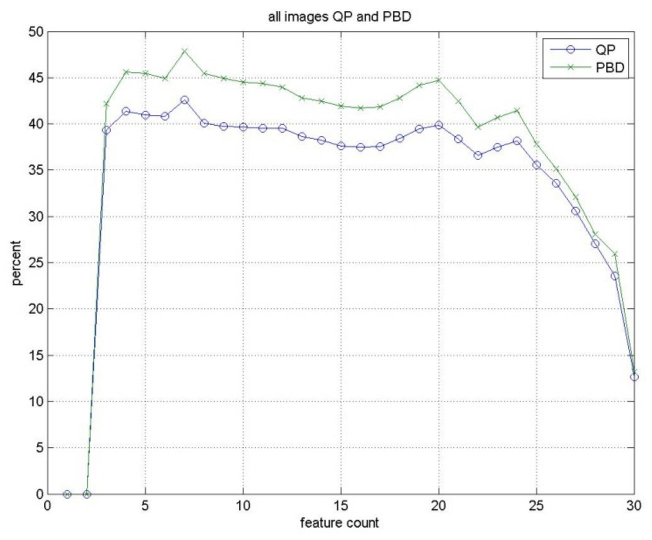


**Figure 4.19: Pixel Performance of SVM Polynomial Kernel Features Selected Using Histogram Errors over the Entire Set of Features**

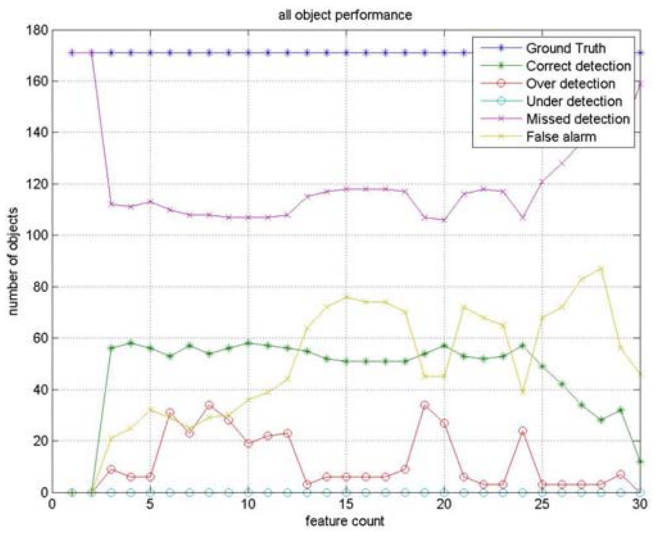


**Figure 4.20: Object Performance of SVM Polynomial Kernel Features Selected Using Histogram Errors over the Entire Set of Features**

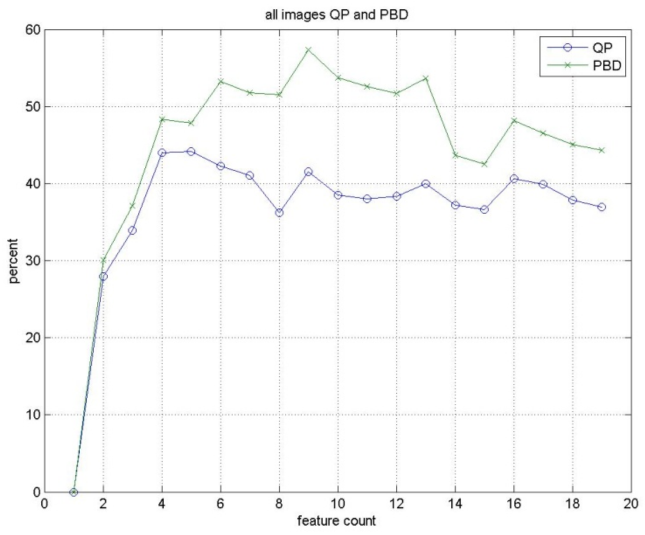


**Figure 4.21: Pixel Performance of SVM Polynomial Kernel Features Selected Using Adaboost over the Entire Set of Features**

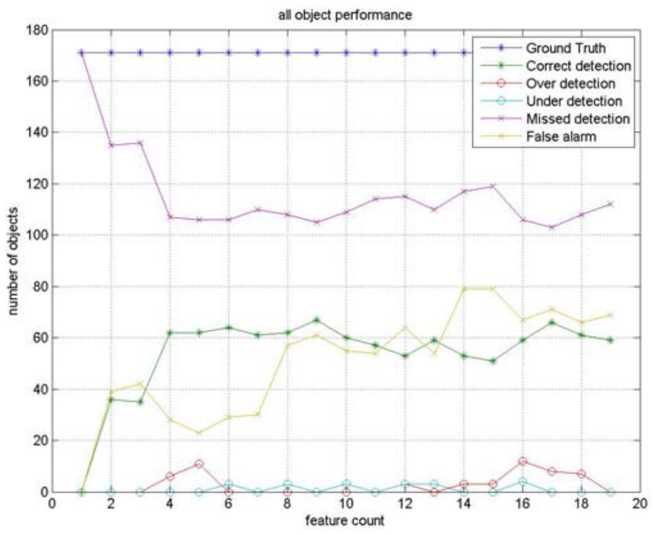


**Figure 4.22: Object Performance of SVM Polynomial Kernel Features Selected Using Adaboost over the Entire Set of Features**

Figure 4.23 to Figure 4.26 illustrates the pixel and object performances of SVM rbf kernel with both features selected by using histogram errors and features selected by Adaboost. The maximum QP value is 41 with 6 features when using histogram errors and the saturation is at 3 features. When using Adaboost selections QP performance saturated with 3 features and maximum QP value achieved with 4 features which is 42.

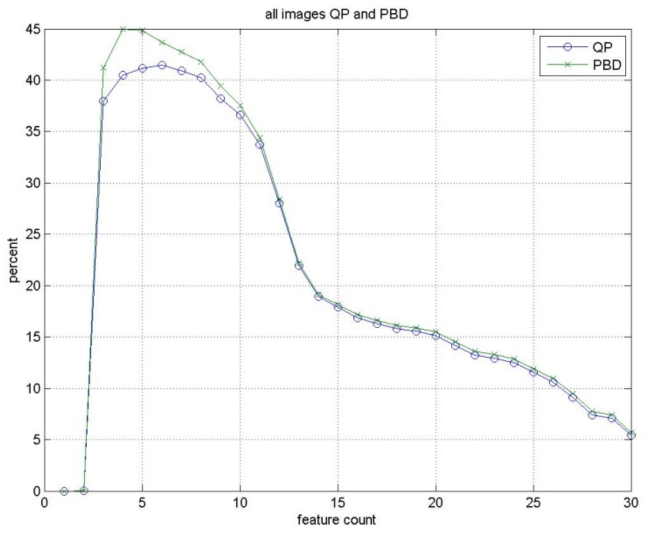


**Figure 4.23: Pixel Performance of SVM RBF Kernel Features Selected Using Histogram Errors over the Entire Set of Features**

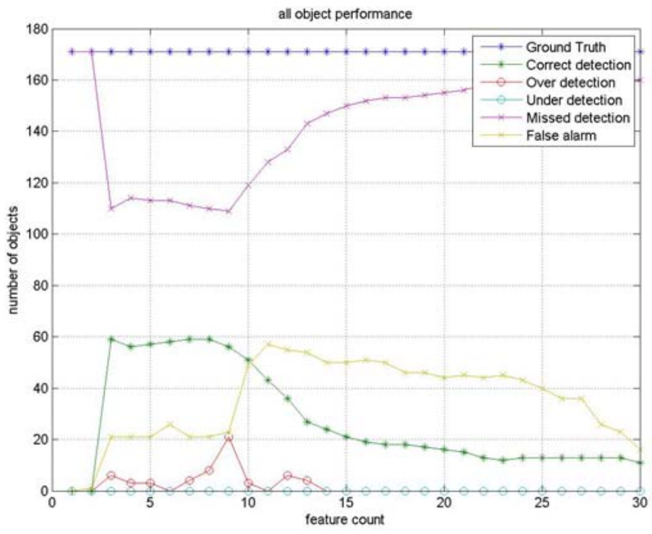


**Figure 4.24: Object Performance of SVM RBF Kernel Features Selected Using Histogram Errors over the Entire Set of Features**

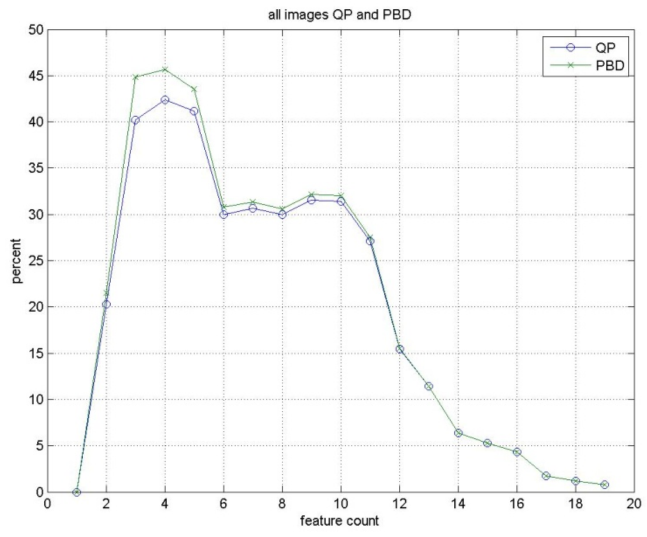


**Figure 4.25: Pixel Performance of SVM RBF Kernel Features Selected Using Adaboost over the Entire Set of Features**

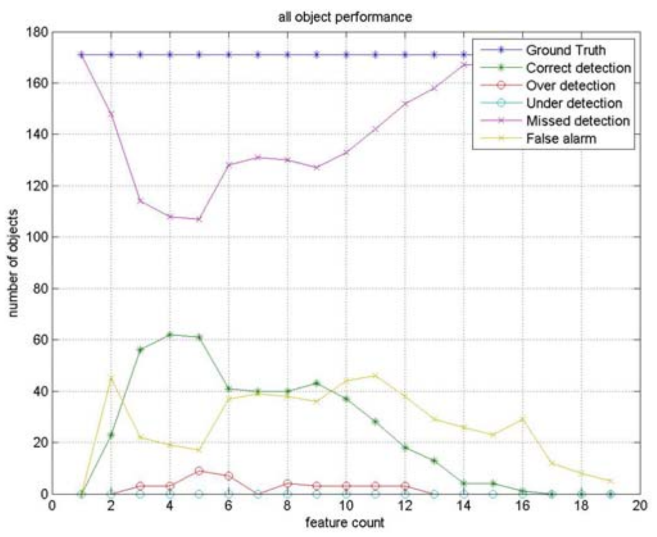


**Figure 4.26: Object Performance of SVM RBF Kernel Features Selected Using Adaboost over the Entire Set of Features**

Figure 4.27 to Figure 4.30 illustrates the pixel and object performances of SVM quadratic kernel with both features selected by using histogram errors and features selected by Adaboost. The maximum QP value is 43 with 6 features when using histogram errors and the saturation is at 4 features. When using Adaboost selections QP performance saturated with 12 features and maximum QP value achieved with 14 features, which is 51.

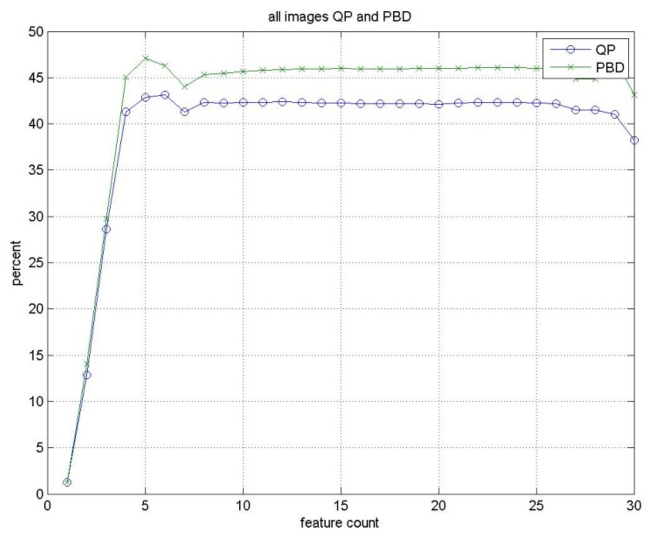


**Figure 4.27: Pixel Performance of SVM Quadratic Kernel Features Selected Using Histogram Errors over the Entire Set of Features**

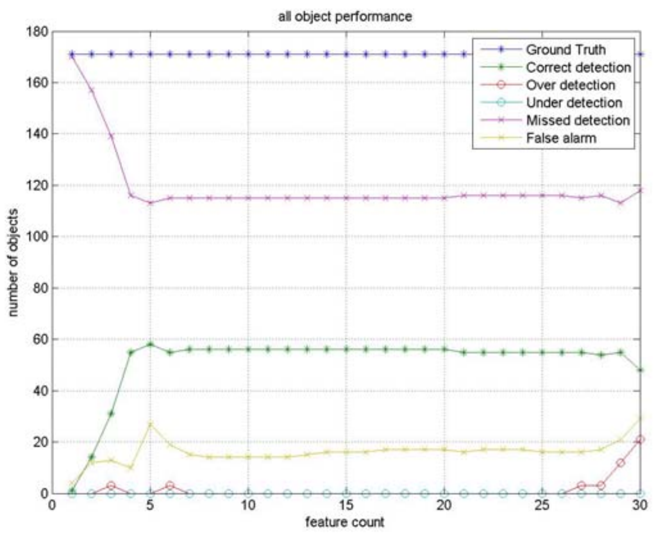


**Figure 4.28: Object Performance of SVM Quadratic Kernel Features Selected Using Histogram Errors over the Entire Set of Features**

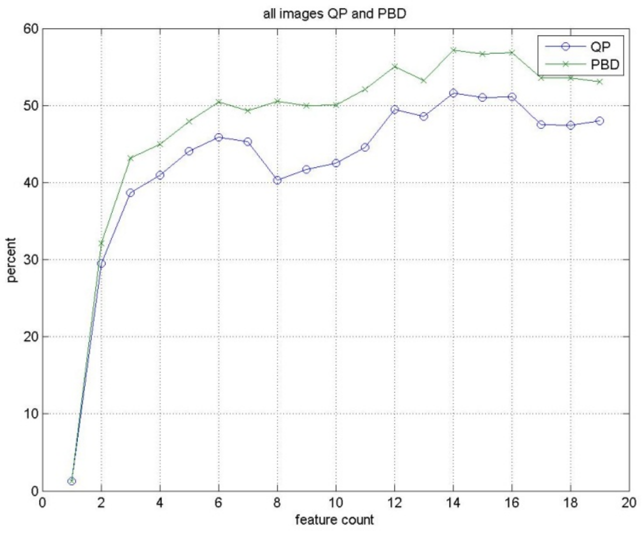


**Figure 4.29: Pixel Performance of SVM Quadratic Kernel Features Selected Using Adaboost Over the Entire Set of Features**

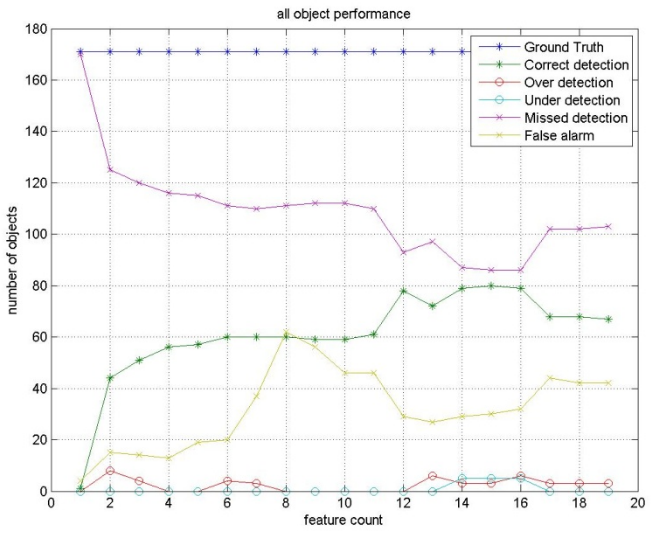


**Figure 4.30: Object Performance of SVM Quadratic Kernel Features Selected Using Adaboost Over the Entire Set of Features**

In summary, when all the features (i.e. both intensity and HSV features) are used, each kernel gave acceptable results at different feature counts. The best result is achieved when using quadratic kernel trained with Adaboost selected features. The performance of this kernel is nearly %10 better than performance of Adaboost. Most of the kernels gave better results while using Adaboost selected features. That is an expected result since Adaboost selects features taking into account the incorrectly classified samples.

#### 4.1.3.2 Iteration and feature count analysis for intensity textural features

We see in section 4.1.2 that the performance of SVM kernels by using intensity textural features was not acceptable. Adaboost can form a strong classifier by using weak classifiers obtained from features but SVM kernels unable to form a meaningful classifier by using features with low separation capacity. In this section, only iteration count analysis of Adaboost is done.Table 4.7 illustrates the intensity textural features selected by Adaboost. The first feature selected by Adaboost is variogram of DFT magnitude that is not selected by Adaboost when feature selection set was all features. Also the features are selected nearly from all classes of features.

**Table 4.7: The Features Selected By Adaboost Over Intensity Textural Features**

| Iteration count | Feature name | Feature number | Previous selection | Feature count | Parity | Weight Of the classifier | Threshold | Normalized threshold | Error | Beta |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Variogram of DFT with 1 distance | 44 | N | 1 | -1 | 0.631 | 1.90 | 0.04 | 0.34 | 0.53 |
| 2 | Skewness of intensity | 3 | N | 2 | 1 | 0.424 | 0.68 | 0.41 | 0.39 | 0.65 |
| 3 | Entropy of intensity | 5 | N | 3 | 1 | 0.288 | 5.03 | 0.71 | 0.42 | 0.74 |
| 4 | Zernike (2,0) | 14 | N | 4 | -1 | 0.368 | 6.65 | 0.32 | 0.40 | 0.69 |
| 5 | Mean of Gabor filter; twelfth order | 69 | N | 5 | -1 | 0.255 | 0.15 | 0.14 | 0.43 | 0.77 |
| 6 | Energy(LL,3) | 135 | N | 6 | 1 | 0.250 2 | 180 | 0.06 | 0.43 | 0.77 |
| 7 | Cluster shade with distance 1 | 96 | N | 7 | 1 | 0.253 | 4.57 | 0.39 | 0.43 | 0.77 |
| 8 | Zernike (2,0) | 14 | Y | 7 | -1 | 0.246 | 2.70 | 0.13 | 0.43 | 0.78 |
| 9 | Kurtosis of intensity | 4 | N | 8 | -1 | 0.216 | 8.01 | 0.08 | 0.44 | 0.80 |
| 10 | Mean of intensity | 1 | N | 9 | -1 | 0.165 | 0.13 | 0.13 | 0.45 | 0.84 |
| 11 | Variance(HL, 3) | 140 | N | 10 | 1 | 0.183 | 0.04 | 0.03 | 0.45 | 0.83 |

**Table 4.7 - Continued**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 12 | Variance(LH,1) | 122 | N | 11 | -1 | 0.1604 | 0.01 | 0.06 | 0.46 | 0.85 |
| 13 | Variance of gradient | 8 | N | 12 | 1 | 0.1606 | 0.68 | 0.08 | 0.45 | 0.85 |
| 14 | Entropy of intensity | 5 | Y | 12 | 1 | 0.1765 | 4.65 | 0.64 | 0.45 | 0.83 |
| 15 | Variance(HH,1) | 126 | N | 13 | -1 | 0.148 | 0.01 | 0.09 | 0.46 | 0.86 |
| 16 | Entropy of intensity | 5 | Y | 13 | 1 | 0.1704 | 5.35 | 0.77 | 0.45 | 0.84 |
| 17 | Variance(HL,2) | 132 | N | 14 | -1 | 0.137 | 0.13 | 0.19 | 0.46 | 0.87 |
| 18 | Mean of DFT | 38 | N | 15 | -1 | 0.1937 | 0.27 | 0.08 | 0.45 | 0.82 |
| 19 | Cluster shade with distance 1 | 96 | Y | 15 | 1 | 0.1516 | -4.57 | 0.39 | 0.46 | 0.85 |
| 20 | Energy(LL,1) | 119 | N | 16 | 1 | 0.1503 | 18.5 | 0.06 | 0.46 | 0.86 |
| 21 | Variance(LH,1) | 122 | Y | 16 | -1 | 0.1668 | 0.01 | 0.03 | 0.45 | 0.84 |
| 22 | Zernike (4,0) | 18 | N | 17 | -1 | 0.1269 | 36.7 | 0.48 | 0.46 | 0.88 |
| 23 | Mean of DFT | 38 | Y | 17 | 1 | 0.1424 | 0.66 | 0.21 | 0.46 | 0.86 |
| 24 | Ring6 on PS | 36 | N | 18 | 1 | 0.1302 | -408 | 0.97 | 0.46 | 0.87 |
| 25 | Kurtosis of intensity | 4 | Y | 18 | -1 | 0.1237 | 12.3 | 0.13 | 0.46 | 0.88 |
| 26 | Mean of Gabor filter; twelfth order | 69 | Y | 18 | -1 | 0.1205 | 0.13 | 0.12 | 0.46 | 0.88 |
| 27 | Variance of Gabor filter; first order | 48 | N | 19 | -1 | 0.124 | 0.03 | 0.22 | 0.46 | 0.88 |
| 28 | Variance(LL,2) | 128 | N | 20 | 1 | 0.0964 | 0.38 | 0.21 | 0.47 | 0.90 |
| 29 | Variance(LH,1) | 122 | Y | 20 | -1 | 0.1012 | 0.03 | 0.15 | 0.47 | 0.90 |
| 30 | Energy(LL,3) | 135 | N | 21 | 1 | 0.0976 | 150. | 0.05 | 0.47 | 0.90 |

Figure 4.31 and Figure 4.32 shows the pixel and object performance of Adaboost with intensity textural features. The QP values are near 30 percent. After iteration count of 15, false alarms in the object performance highly increases. Iteration count 14 has the best correct detection rate also with low false alarms. Taking into account this iteration count 14 with QP 28 is selected for Adaboost textural features. It is saturated with 12 iterations.

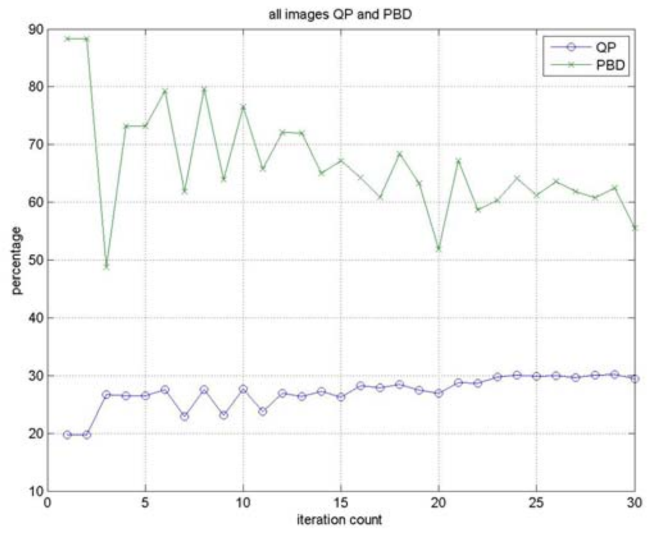


**Figure 4.31: QP and PBD Values vs. Iteration Count for Adaboost with Intensity Textural Features**

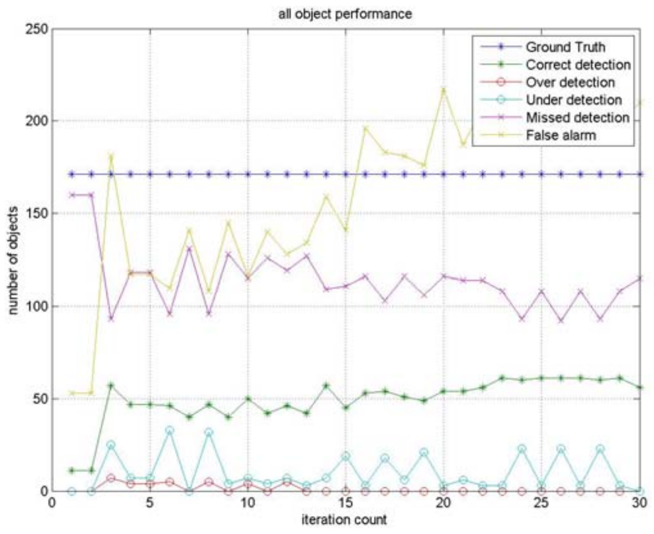


**Figure 4.32: Object Performance vs. Iteration Count for Adaboost with Intensity Textural Features**

### 4.1.3.3 Iteration and feature count analysis for HSV domain features

Previous results showed us that HSV domain features have high separation capacity. In this section, we will select a classifier using only HSV domain features. Adaboost and SVM quadratic kernel, which gave the best results previously, will be analyzed in this section.

Figure 4.33 and Figure 4.34 shows the pixel and object performances for Adaboost with HSV domain features. The maximum performance is achieved while using the first feature. The QP value for first feature is 45. That feature is the most dominant feature, which is center pixel value of hue, and the performance never passed that value again.

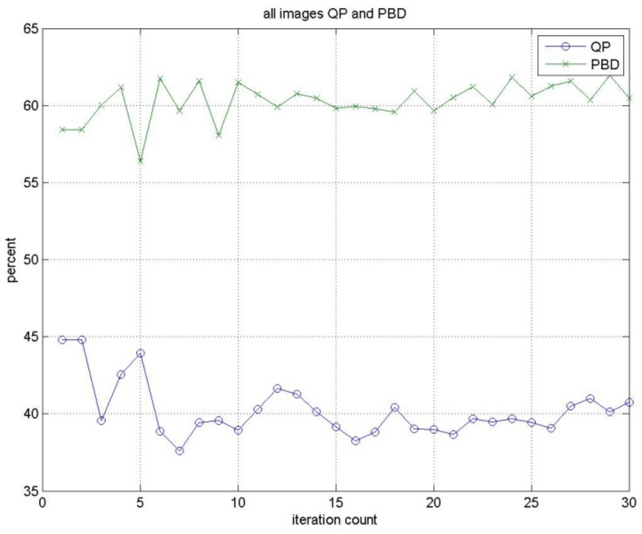


**Figure 4.33: Pixel Performance for Adaboost with HSV Domain Features**

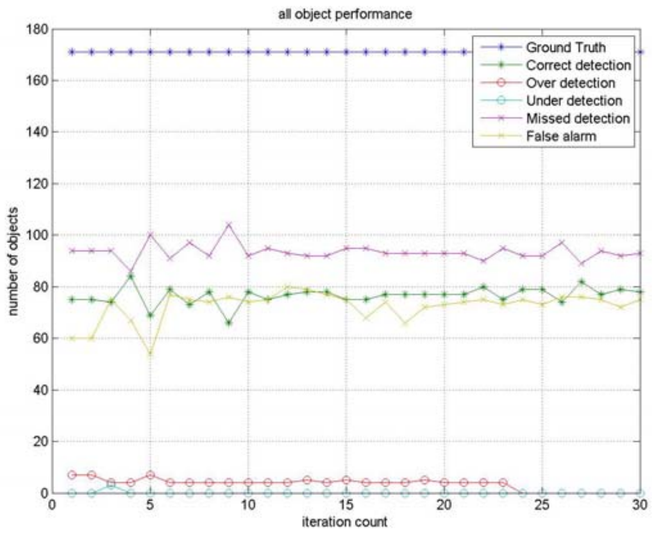


**Figure 4.34: Object Performance for Adaboost with HSV Domain Features**

Figure 4.35 and Figure 4.36 shows the pixel and object performances for SVM quadratic kernel with HSV domain features. The maximum performance is achieved while using 16 features with QP value 48. There is no need to use Adaboost selected features since the set of features is not large. Note that the performance achieved is better than the result of Adaboost with all features.

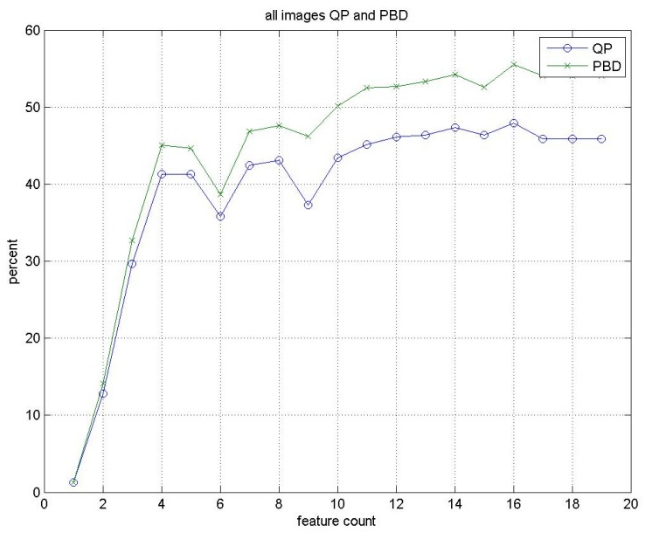


**Figure 4.35: Pixel Performance of SVM Quadratic Kernel Features Selected Using Histogram Errors over the Set Of HSV Domain Features**

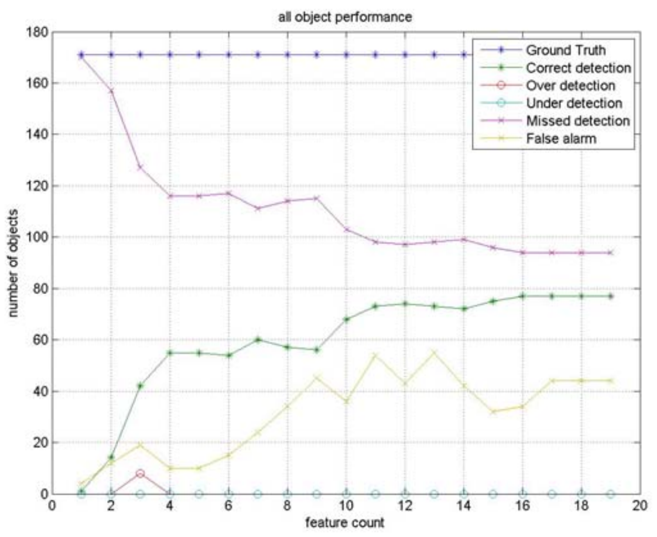


**Figure 4.36: Object Performance of SVM Quadratic Kernel Features Selected Using Histogram Errors over the Set Of HSV Domain Features**

#### 4.1.3.4 Summary of iteration and feature count analysis

In this section, a summary of the results of iteration count analysis will be given. Table 4.8 illustrates the results of the analysis. The best results for the feature sets are given bold. Also, classifiers that will be used in further detailed analysis are given bold. The best result achieved is obtained with SVM quadratic kernel. The features of the kernel are selected by Adaboost over all features set (i.e. intensity and HSV features). Adaboost gave the second best result when all features are used. In section 4.1.2, there were no acceptable results for SVM kernels using only intensity textural features so no feature count analysis is done for SVM kernels using intensity textural features. However, Adaboost gave some acceptable results while using only intensity textural features. The best results are achieved with SVM quadratic kernel while using all features and there were mainly HSV domain features in the selected features so HSV domain iteration count analysis is done using only SVM quadratic kernel and Adaboost.

**Table 4.8: Summary of Iteration Count Analysis**

| Feature set | Classifier | Histogram error selected | | | Adaboost selected | | |
|---|---|---|---|---|---|---|---|
| | | Saturation count | QP value for selected count | Feature count | Saturation for feature/iteration count | QP value for selected count | Feature/iteration count for selection |
| HSV + intensity | Adaboost | Not applicable | | | 16/25 | 47 | 19/28 |
| | SVM lin | 8 | 42 | 8 | 9 | 40 | 16 |
| | SVM poly | 3 | 43 | 7 | 4 | 44 | 4 |
| | SVM rbf | 3 | 41 | 6 | 3 | 42 | 4 |
| | SVM quad | 4 | 43 | 6 | 12 | 51 | 14 |
| Only intensity | Adaboost | Not applicable | | | 11/12 | 28 | 12/14 |
| | SVM lin | Previously eliminated | | | | | |
| | SVM poly | | | | | | |
| | SVM rbf | | | | | | |
| | SVM quad | | | | | | |
| Only HSV | Adaboost | Not applicable | | | 1/1 | 45 | 1/1 |
| | SVM lin | Previously eliminated | | | | | |
| | SVM poly | | | | | | |
| | SVM rbf | | | | | | |
| | SVM quad | 12 | 48 | 16 | Not applicable | | |

The best result for HSV domain is achieved by SVM quadratic kernel. In further analyses, we will use Adaboost with all features, SVM quadratic kernel with Adaboost selected features over all features, Adaboost with intensity textural features and SVM quadratic kernel with HSV domain features. The detailed results including visual results will be given for these classifiers and a comparison will be done using these classifiers.

### 4.1.4 Training set analyses

Upper twenty percent of images are used one by one and object and pixel performances are checked in all images. The analyses are done using Adaboost with intensity textural features, Adaboost with all features and SVM quad kernel with features selected by Adaboost. The detailed pixel and object-based performances are given. For object-based performance, building detection threshold is taken as 0.4.

Firstly, upper twenty percent of image 1 is used in training of Adaboost with intensity textural features. The pixel and object performances are given in Table 4.9 and

Table 4.10.The performance is nearly same as using all images in training set. Because, there are a few buildings, roads and green belts mainly in the upper part of image 1 and that is enough for training with intensity textural features. In all images, PBD values are high and QP values are low that is because of FP. The classifier cannot differentiate manmade objects (roads, parks, etc.) from buildings and this causes high FP values.

**Table 4.9: Pixel Performance of Adaboost with Intensity Textural Features Using Image 1 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.6889 | 0.1684 | 64.8759 | 26.6242 |
| 2 | 0.748 | 0.0399 | 86.3216 | 24.2329 |
| 3 | 0.8186 | 0.033 | 84.5984 | 17.561 |
| 4 | 0.8241 | 0.0081 | 95.5954 | 17.4453 |
| 5 | 0.6199 | 0.0505 | 88.2638 | 36.1822 |

**Table 4.10: Object Performance of Adaboost with Intensity Textural Features Using Image 1 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 14 | 0 | 0 | 21 | 48 |
| 2 | 13 | 3 | 0 | 0 | 10 | 23 |
| 3 | 24 | 6 | 0 | 0 | 18 | 34 |
| 4 | 29 | 2 | 0 | 0 | 27 | 16 |
| 5 | 70 | 1 | 0 | 0 | 69 | 5 |

**Table 4.11: Pixel Performance of Adaboost with Intensity Textural Features Using Image 2 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.7938 | 0.9163 | 18.3721 | 10.7617 |
| 2 | 0.3552 | 7.9967 | 7.4614 | 7.1667 |
| 3 | 1 | 19.2096 | 0 | 0 |
| 4 | 1 | 14.4056 | 0 | 0 |
| 5 | 0.2108 | 4.5578 | 14.7594 | 14.1995 |

**Table 4.12: Object Performance of Adaboost with Intensity Textural Features Using Image 1 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 2 | 0 | 0 | 33 | 65 |
| 2 | 13 | 0 | 0 | 0 | 13 | 10 |
| 3 | 24 | 0 | 0 | 0 | 24 | 7 |
| 4 | 29 | 0 | 0 | 0 | 29 | 3 |
| 5 | 70 | 9 | 0 | 0 | 61 | 0 |

Upper twenty percent of image 2 is used in training of Adaboost with intensity textural features. The pixel and object performances are given in 11  and Table 4.12.The performance is quite bad since there are few buildings on the training set actually only part of buildings and also there are much manmade objects which are not buildings that makes classifier to extract buildings in other images.

**Table 4.13: Pixel Performance of Adaboost with Intensity Textural Features Using Image 3 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.6837 | 7.1896 | 4.2138 | 3.862 |
| 2 | 0.7949 | 0.2875 | 41.6384 | 15.9296 |
| 3 | 0.6334 | 0.4084 | 47.2989 | 26.0269 |
| 4 | 0.6007 | 0.4949 | 44.6551 | 26.7121 |
| 5 | 0.9792 | 36.4078 | 0.057 | 0.0555 |

**Table 4.14: Object Performance of Adaboost with Intensity Textural Features Using Image 3 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 0 | 0 | 0 | 35 | 13 |
| 2 | 13 | 4 | 0 | 0 | 9 | 21 |
| 3 | 24 | 12 | 0 | 0 | 12 | 54 |
| 4 | 29 | 14 | 0 | 0 | 15 | 64 |
| 5 | 70 | 0 | 0 | 0 | 70 | 4 |

Upper twenty percent of image 3 is used in training of Adaboost with intensity textural features. The pixel and object performances are given in Table 4.13 and

Table 4.14. The textures in image 3 and 4 are close to each other and this makes an increase in the performances of these images. The buildings in image 4 and 5 are like the roads in image 3 so the performance of these images is bad.

Upper twenty percent of image 4 is used in training of Adaboost with intensity textural features. The pixel and object performances are given in Table 4.15 and Table 4.16. The performance in image 4 is good since all the buildings in the image reflects same properties. Again SF is high that is because of the other manmade objects. As in the case before, also image 3 has good performance.

**Table 4.15: Pixel Performance of Adaboost with Intensity Textural Features Using Image 4 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.0205 | 31.2394 | 3.04 | 3.0381 |
| 2 | 0.7501 | 0.8316 | 23.106 | 13.6431 |
| 3 | 0.6326 | 0.4897 | 42.8653 | 24.6633 |
| 4 | 0.5801 | 0.1727 | 70.8594 | 35.8063 |
| 5 | NaN | Inf | 0 | 0 |

**Table 4.16: Object Performance of Adaboost with Intensity Textural Features Using Image 4 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 1 | 0 | 0 | 34 | 1 |
| 2 | 13 | 2 | 0 | 0 | 11 | 26 |
| 3 | 24 | 11 | 0 | 0 | 13 | 55 |
| 4 | 29 | 19 | 0 | 0 | 10 | 41 |
| 5 | 70 | 0 | 0 | 0 | 70 | 0 |

Upper twenty percent of image 5 is used in training of Adaboost with intensity textural features. The pixel and object performances are given in Table 4.17 and Table 4.18. The performance in the training image is good since the buildings in the image are much like each other. The buildings in image 5 are much like the other manmade object in other images so SF values are high which reduces the performances.

**Table 4.17: Pixel Performance of Adaboost with Intensity Textural Features Using Image 5 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.7625 | 0.3841 | 38.2081 | 17.1579 |
| 2 | 0.7212 | 0.627 | 30.7797 | 17.136 |
| 3 | 0.9232 | 0.4873 | 13.6091 | 5.1613 |
| 4 | 0.8486 | 0.2577 | 37.0148 | 12.0402 |
| 5 | 0.4094 | 0.3967 | 59.8217 | 42.2862 |

**Table 4.18: Object Performance of Adaboost with Intensity Textural Features Using Image 5 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 6 | 0 | 0 | 29 | 47 |
| 2 | 13 | 3 | 0 | 0 | 10 | 40 |
| 3 | 24 | 0 | 0 | 0 | 24 | 74 |
| 4 | 29 | 2 | 0 | 0 | 27 | 60 |
| 5 | 70 | 14 | 0 | 5 | 52 | 10 |

Upper twenty percent of all images are used in training of Adaboost with intensity textural features. The pixel and object performances are given in Table 4.19 and Table 4.20. The overall performance is better than other cases only the performance of image 1 is not good compared to others. The performance of image 2 was not good while using image 2 in train but this time it is better since the building textures in image 1 are like in image 2.

**Table 4.19: Pixel Performance of Adaboost with Intensity Textural Features Using All Images in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.7407 | 0.5197 | 33.2886 | 17.0646 |
| 2 | 0.7025 | 0.1536 | 65.9462 | 25.7914 |
| 3 | 0.7159 | 0.0792 | 78.2031 | 26.3222 |
| 4 | 0.7359 | 0.0202 | 92.8823 | 25.8817 |
| 5 | 0.3647 | 0.5276 | 54.631 | 41.5871 |

**Table 4.20: Object Performance of Adaboost with Intensity Textural Features Using All Images in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 11 | 0 | 0 | 24 | 43 |
| 2 | 13 | 9 | 0 | 0 | 4 | 30 |
| 3 | 24 | 12 | 0 | 0 | 12 | 42 |
| 4 | 29 | 10 | 0 | 0 | 19 | 35 |
| 5 | 70 | 15 | 0 | 7 | 50 | 9 |

Upper twenty percent of image 1 is used in training of Adaboost with all features. The pixel and object performances are given in

Table 4.21 and Table 4.22. HSV domain features are more dominant while using all features. There is an increase in the performance of image 1 compared to the only intensity features case. The performance of image 5 is also high since they are more like each other compares to each other.

**Table 4.21: Pixel Performance of Adaboost with All Features Using Image 1 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0,2356 | 1,2056 | 38,8038 | 34,6593 |
| 2 | 0,8261 | 0,0006 | 99,6645 | 17,3801 |
| 3 | 0,8295 | 0,0043 | 97,5363 | 16,9807 |
| 4 | 0,7735 | 0,0013 | 99,4511 | 22,6223 |
| 5 | 0,5354 | 0,1989 | 70,0178 | 38,7484 |

**Table 4.22: Object Performance of Adaboost with All Features Using Image 1 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 18 | 0 | 0 | 17 | 9 |
| 2 | 13 | 0 | 0 | 0 | 13 | 6 |
| 3 | 24 | 3 | 0 | 0 | 21 | 14 |
| 4 | 29 | 3 | 0 | 0 | 26 | 19 |
| 5 | 70 | 5 | 0 | 0 | 65 | 10 |

Upper twenty percent of image 2 is used in training of adaboost with all features. The pixel and object performances are given in Table 4.23 and
Table 4.24. The performance of image 2 was not good while using only intensity textural features since building patches were small but this time performance is good since a small part of the building is enough for training including HSV domain features.

**Table 4.23: Pixel Performance of Adaboost with All Features Using Image 2 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0,3665 | 4,2321 | 13,0196 | 12,1075 |
| 2 | 0,0337 | 1,4487 | 40,0119 | 39,4606 |
| 3 | 0,1119 | 4,2244 | 17,3715 | 16,9995 |
| 4 | 0,3906 | 0,6578 | 48,0879 | 36,7569 |
| 5 | NaN | Inf | 0 | 0 |

**Table 4.24: Object Performance of Adaboost with All Features Using Image 2 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 6 | 0 | 0 | 29 | 11 |
| 2 | 13 | 8 | 0 | 0 | 5 | 3 |
| 3 | 24 | 4 | 0 | 0 | 20 | 13 |
| 4 | 29 | 13 | 0 | 0 | 16 | 32 |
| 5 | 70 | 0 | 0 | 0 | 70 | 0 |

Upper twenty percent of image 3 is used in training of Adaboost with all features. The pixel and object performances are given in Table 4.25 and

Table 4.26.Image 3 is quite different from other images and upper part of the image well reflects the lower part of the image so has a high performance and the performances of other images are not good.

**Table 4.25: Pixel Performance of Adaboost with All Features Using Image 3 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0,7857 | 0,2866 | 42,7846 | 16,6588 |
| 2 | 0,7867 | 0,0539 | 79,8199 | 20,2427 |
| 3 | 0,3845 | 0,0516 | 92,2705 | 58,5316 |
| 4 | 0,7807 | 0,0072 | 96,801 | 21,769 |
| 5 | 0,6043 | 7,6232 | 4,934 | 4,5883 |

**Table 4.26: Object Performance of Adaboost with All Features Using Image 3 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 3 | 0 | 0 | 10 | 9 |
| 2 | 13 | 20 | 0 | 0 | 4 | 16 |
| 3 | 24 | 10 | 0 | 0 | 19 | 23 |
| 4 | 29 | 1 | 0 | 0 | 69 | 5 |
| 5 | 70 | 3 | 0 | 0 | 10 | 9 |

Upper twenty percent of image 4 is used in training of Adaboost with all features. The pixel and object performances are given in Table 4.27 and Table 4.28. An interesting result is the performance of 3 images are quite good. In the other 2 image, there are no TP values. The results for image 1 and 5 were also quite low while using intensity textural features only. By also using HSV domain features high performances are reached in image 2, 3 and 4 which have similar buildings to the training set used.

**Table 4.27: Pixel Performance of Adaboost with All Features Using Image 4 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 1 | 48,5518 | 0 | 0 |
| 2 | 0,0412 | 0,7871 | 54,9176 | 53,6508 |
| 3 | 0,2905 | 0,0427 | 94,3277 | 68,0437 |
| 4 | 0,2486 | 0,0495 | 93,8199 | 71,5966 |
| 5 | NaN | Inf | 0 | 0 |

**Table 4.28: Object Performance of Adaboost with All Features Using Image 4 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 0 | 0 | 0 | 35 | 1 |
| 2 | 13 | 8 | 5 | 0 | 4 | 0 |
| 3 | 24 | 19 | 0 | 0 | 5 | 12 |
| 4 | 29 | 27 | 0 | 0 | 2 | 7 |
| 5 | 70 | 0 | 0 | 0 | 70 | 0 |

Upper twenty percent of image 5 is used in training of Adaboost with all features. The pixel and object performances are given in Table 4.29 and Table 4.30. The performance of image 5 is quite good but in other images, SF values are high MF values are low that means classifier output includes high FP values and high TP values.

**Table 4.29: Pixel Performance of Adaboost with All Features Using Image 5 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0,7404 | 0,1407 | 64,8605 | 22,76 |
| 2 | 0,8308 | 0,0015 | 99,1125 | 16,8929 |
| 3 | 0,7987 | 0,009 | 95,7328 | 19,9513 |
| 4 | 0,7998 | 0,0009 | 99,5456 | 19,9974 |
| 5 | 0,2562 | 0,155 | 82,7522 | 64,3955 |

**Table 4.30: Object Performance of Adaboost with All Features Using Image 5 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 11 | 0 | 0 | 24 | 42 |
| 2 | 13 | 0 | 0 | 0 | 13 | 2 |
| 3 | 24 | 3 | 0 | 0 | 21 | 20 |
| 4 | 29 | 1 | 0 | 0 | 28 | 21 |
| 5 | 70 | 27 | 0 | 20 | 28 | 2 |

Upper twenty percent of all images are used in training of Adaboost with all features. The pixel and object performances are given in Table 4.31 and Table 4.32. The overall performance is increased compared to using a single image in training. Although there were better results in some images while using only one in image in training, the classifier was not generic that is capable of classifying different images. By using different images with different types of buildings, a more robust classifier can be formed.

**Table 4.31: Pixel Performance of Adaboost with All Features Using All Images in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0,2779 | 1,5222 | 32,1742 | 28,6293 |
| 2 | 0,1648 | 0,3159 | 72,5582 | 63,4719 |
| 3 | 0,4928 | 0,0204 | 96,1421 | 49,7129 |
| 4 | 0,3764 | 0,0717 | 89,6938 | 58,1881 |
| 5 | 0,1162 | 1,6474 | 34,9162 | 33,3833 |

**Table 4.32: Object Performance of Adaboost with All Features Using All Images in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 19 | 0 | 0 | 16 | 7 |
| 2 | 13 | 10 | 0 | 0 | 3 | 9 |
| 3 | 24 | 20 | 0 | 0 | 4 | 32 |
| 4 | 29 | 27 | 0 | 0 | 2 | 15 |
| 5 | 70 | 17 | 0 | 0 | 53 | 5 |

Upper twenty percent of image 1 is used in training of SVM quadratic kernel with all features. The pixel and object performances are given in Table 4.33 and Table 4.34. When using Adaboost with all features, the performance of image 5 was fine. Now image 5 has no output. Since SVM transforms the feature set into another space, this can be the case. Commenting into this result is difficult but the result of image one which is used in training is fine that is what we expect.

**Table 4.33: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 1 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.1493 | 1.5758 | 35.0581 | 33.0255 |
| 2 | 0.0458 | 1.5907 | 37.4936 | 36.8305 |
| 3 | 0.8131 | 14.2944 | 1.2905 | 1.2219 |
| 4 | 0.2574 | 29.8882 | 2.4242 | 2.404 |
| 5 | NaN | Inf | 0 | 0 |

**Table 4.34: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 1 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 19 | 0 | 0 | 16 | 6 |
| 2 | 13 | 8 | 0 | 0 | 5 | 4 |
| 3 | 24 | 0 | 0 | 0 | 24 | 11 |
| 4 | 29 | 0 | 0 | 0 | 29 | 5 |
| 5 | 70 | 0 | 0 | 0 | 70 | 0 |

Upper twenty percent of image 2 is used in training of SVM quadratic kernel with all features. The pixel and object performances are given in Table 4.35 and Table 4.36. The kernel converged to a state that is only

**Table 4.35: Pixel Performance of SVM Quadratic with Adaboost Selected Features Using Image 2 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.3536 | 15.4232 | 4.0226 | 3.936 |
| 2 | 0.0173 | 1.9494 | 33.5145 | 33.3179 |
| 3 | 0.8844 | 2.7834 | 3.9889 | 3.0565 |
| 4 | 0.6936 | 3.2576 | 8.5964 | 7.1959 |
| 5 | NaN | Inf | 0 | 0 |

**Table 4.36: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 2 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 0 | 0 | 0 | 35 | 9 |
| 2 | 13 | 6 | 0 | 0 | 7 | 3 |
| 3 | 24 | 0 | 0 | 0 | 24 | 29 |
| 4 | 29 | 1 | 0 | 0 | 28 | 34 |
| 5 | 70 | 0 | 0 | 0 | 70 | 0 |

Upper twenty percent of image 3 is used in training of SVM kernel with Adabost selected features. The pixel and object performances are given in Table 4.37 and Table 4.38. Upper part of the image well reflects the lower part of the image so has a high performance and the performances of other images except image 4 are not good.

**Table 4.37: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 3 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.7312 | 0.379 | 41.4987 | 19.4946 |
| 2 | 0.6853 | 0.5228 | 37.5786 | 20.667 |
| 3 | 0.2791 | 0.3358 | 68.22 | 53.9647 |
| 4 | 0.5704 | 0.0361 | 92.2572 | 41.4608 |
| 5 | 0.2891 | 1.7899 | 28.4278 | 25.4826 |

**Table 4.38: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 3 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 8 | 0 | 0 | 27 | 51 |
| 2 | 13 | 4 | 0 | 0 | 9 | 35 |
| 3 | 24 | 17 | 6 | 0 | 5 | 14 |
| 4 | 29 | 23 | 0 | 0 | 6 | 30 |
| 5 | 70 | 14 | 0 | 0 | 56 | 4 |

Upper twenty percent of image 4 is used in training of kernel. The pixel and object performances are given in Table 4.39 and Table 4.40. The performance of 3 images are quite good. In the other 2 image, the results are not acceptable.

**Table 4.39: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 4 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.0487 | 5.6997 | 14.3033 | 14.1994 |
| 2 | 0.0659 | 0.7998 | 53.8729 | 51.9003 |
| 3 | 0.5023 | 0.07 | 87.6705 | 46.5129 |
| 4 | 0.0896 | 0.135 | 87.0857 | 80.2095 |
| 5 | 0.9638 | 1.3901 | 2.5383 | 1.5148 |

**Table 4.40: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 4 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 5 | 0 | 0 | 30 | 8 |
| 2 | 13 | 9 | 0 | 0 | 4 | 2 |
| 3 | 24 | 20 | 0 | 0 | 4 | 48 |
| 4 | 29 | 26 | 0 | 0 | 3 | 2 |
| 5 | 70 | 0 | 0 | 0 | 70 | 18 |

Upper twenty percent of image 5 is used in training of the kernel. The pixel and object performances are given in Table 4.41 and Table 4.42. The performance of image 5 is quite good but in other images, SF values are high MF values are low that means classifier output includes high FP values and high TP values.

**Table 4.41: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 5 in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.7928 | 0.0888 | 69.9932 | 19.0282 |
| 2 | 0.618 | 0.1613 | 70.3075 | 32.8962 |
| 3 | 0.7788 | 0.0708 | 75.7585 | 20.6547 |
| 4 | 0.5499 | 0.0872 | 83.7685 | 41.3998 |
| 5 | 0.2708 | 0.2117 | 77.4973 | 60.1761 |

**Table 4.42: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using Image 5 in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 9 | 0 | 0 | 26 | 29 |
| 2 | 13 | 5 | 0 | 3 | 6 | 26 |
| 3 | 24 | 7 | 0 | 0 | 17 | 42 |
| 4 | 29 | 21 | 0 | 4 | 5 | 44 |
| 5 | 70 | 25 | 0 | 22 | 28 | 1 |

Upper twenty percent of all images are used in training of the quadratic kernel with Adaboost selected features. The pixel and object performances are given in Table 4.43 and Table 4.44. The overall performance is increased compared to using a single image in training. Although there were better results in some images while using only one in image in training, the classifier was not generic that is capable of classifying different images. By using different images with different types of buildings, a more robust classifier can be formed.

**Table 4.43: Pixel Performance of SVM Quadratic Kernel with Adaboost Selected Features Using All Images in Training**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.2167 | 2.8313 | 21.6693 | 20.4434 |
| 2 | 0.0848 | 0.5978 | 60.4892 | 57.2784 |
| 3 | 0.0842 | 0.2356 | 79.54 | 74.1203 |
| 4 | 0.1694 | 0.1503 | 84.6772 | 72.2077 |
| 5 | 0.2979 | 1.0769 | 39.4652 | 33.8036 |

**Table 4.44: Object Performance of SVM Quadratic Kernel with Adaboost Selected Features Using All Images in Training**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 11 | 3 | 0 | 23 | 9 |
| 2 | 13 | 8 | 0 | 0 | 5 | 5 |
| 3 | 24 | 19 | 0 | 0 | 5 | 2 |
| 4 | 29 | 25 | 0 | 0 | 4 | 9 |
| 5 | 70 | 16 | 0 | 5 | 50 | 4 |

As a result, using only one image may give better results on that image or on the images similar but our aim is to form a classifier that would work on different type of images. To do so, using a large training set including different types of building and non-building areas is better than making the classifier memorize one specific image.

## 4.2 Results

In this section, detailed including visual results for four classifiers are given. The classifiers are chosen by sensitivity analyses. The first classifier is Adaboost with intensity textural features. It was the only classifier that has reasonable results when using only intensity textural features. The second classifier is Adaboost with all features. The third classifier is SVM quadratic kernel with the features selected by Adaboost over the all feature set. These two classifiers using all features are chosen since they have better performances and in order to make a comparison between them. The final classifier is SVM quadratic kernel selected using minimum histogram errors over the features extracted from HSV domain. The last classifier chosen has also good performance and extraction of its features saves time since it only uses HSV domain features.

In this section, two sets of images are used. In the first set, the lower %80 part of the images are used for performance evaluation and upper %20 part of the images are used in training. In addition, these images were used in section 4.1 for sensitivity analyses. The numbers of these images go from 1 to 5. In the second set completely new images are introduced. There are 5 images in this set. The numbers of these images are between 6 and 10. The 6. Image is taken from IKONOS. The 7. Image is taken from Quickbird and last three images are taken from Google Earth Application. In the visual results of object performance, color codes are used to differentiate the objects. In these codes; reds are “correct detection”, yellows are “missed detection”, blues are “under detection”, greens are “over detection” and finally, purples are “false alarm”. In section 4.1, these are explained in a detailed way.

### 4.2.1 Adaboost with intensity textural features

First classifier is Adaboost with textural features. The pixel performance is given in Table 4.45. In that table, high SF values shows that the detection of non-building areas as building areas is very common.

**Table 4.45: Pixel Performance of the 1. Set**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.7407 | 0.5197 | 33.2886 | 17.0646 |
| 2 | 0.7025 | 0.1536 | 65.9462 | 25.7914 |
| 3 | 0.7159 | 0.0792 | 78.2031 | 26.3222 |
| 4 | 0.7359 | 0.0202 | 92.8823 | 25.8817 |
| 5 | 0.3647 | 0.5276 | 54.631 | 41.5871 |
| All | 0.65194 | 0.2601 | 64.9902 | 27.3294 |

Figure 4.37 illustrates the results of object performance vs. object detection threshold and Table 4.46 shows the results when object detection threshold is equal to 0.4. %33 of the building found correctly and there are false alarms nearly equal to the number of buildings.

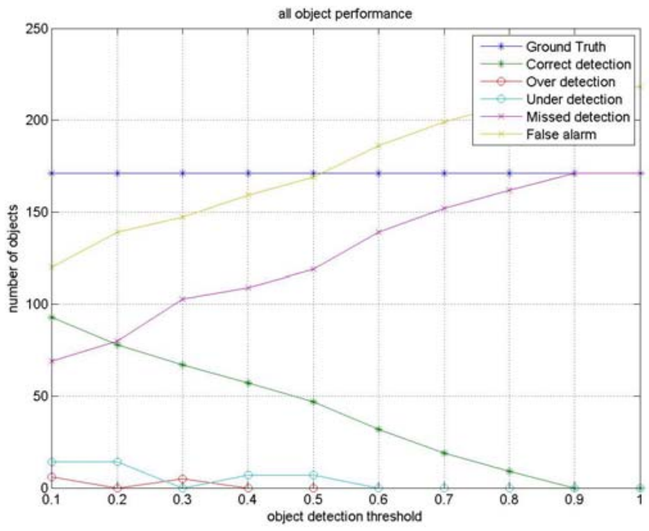


**Figure 4.37: Object Performance vs. Object Detection Threshold**

**Table 4.46: Object Performance of the 1. Set For T = 0.4**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 11 | 0 | 0 | 24 | 43 |
| 2 | 13 | 9 | 0 | 0 | 4 | 30 |
| 3 | 24 | 12 | 0 | 0 | 12 | 42 |
| 4 | 29 | 10 | 0 | 0 | 19 | 35 |
| 5 | 70 | 15 | 0 | 7 | 50 | 9 |
| All | 171 | 57 | 0 | 7 | 109 | 159 |

The visual results for image 1 to 5 are given in
Figure 4.38 to
Figure 4.42. In these figures (a) is the original image, (b) is the original image masked by the building patches, together with the borders of the buildings in the

ground truth data (c) is ground truth, (d) is the building patches extracted by the algorithm. In the results, it is seen that besides buildings, man-made objects like roads are also extracted since the texture of buildings and man-made objects are similar. Building boundaries could not be well extracted which reduces the object performance.

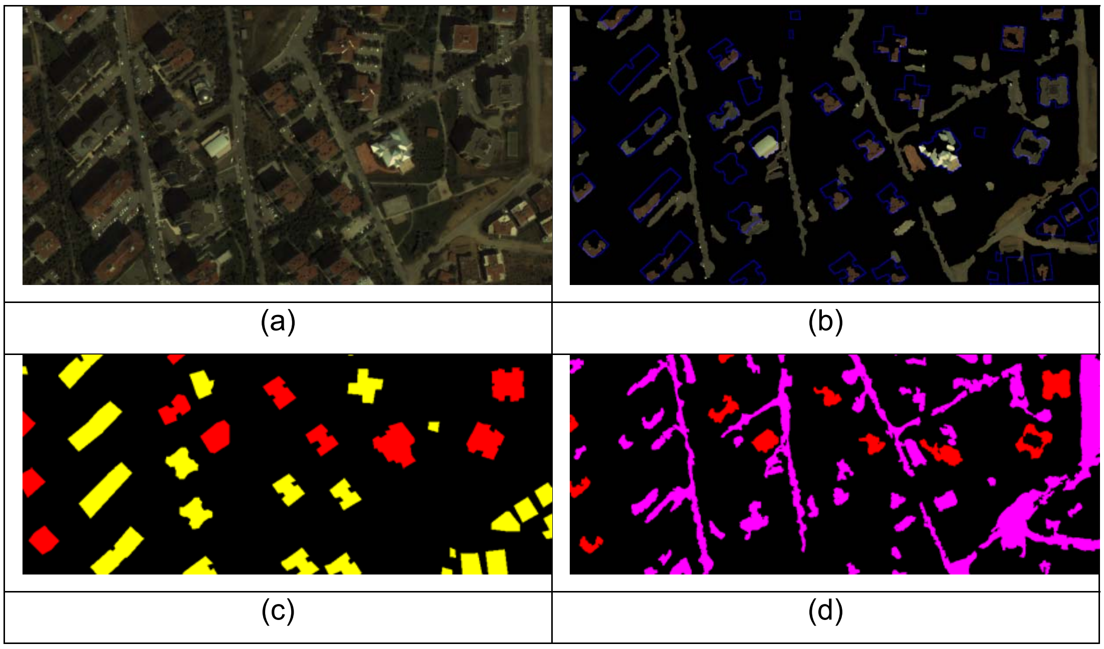


**Figure 4.38: Visual Results for Adaboost with Intensity Textural Features for 1. Image**

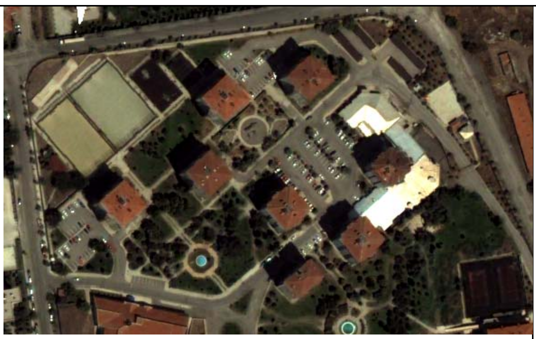
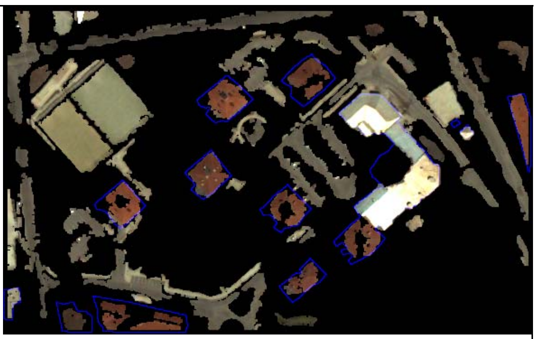
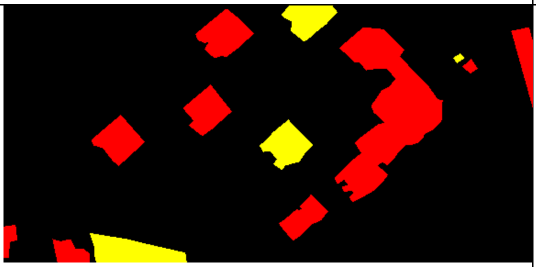
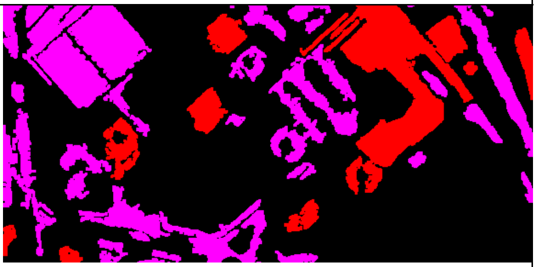
| | |
|---|---|
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

**Figure 4.39: Visual Results for Adaboost with Intensity Textural Features for 2. Image**

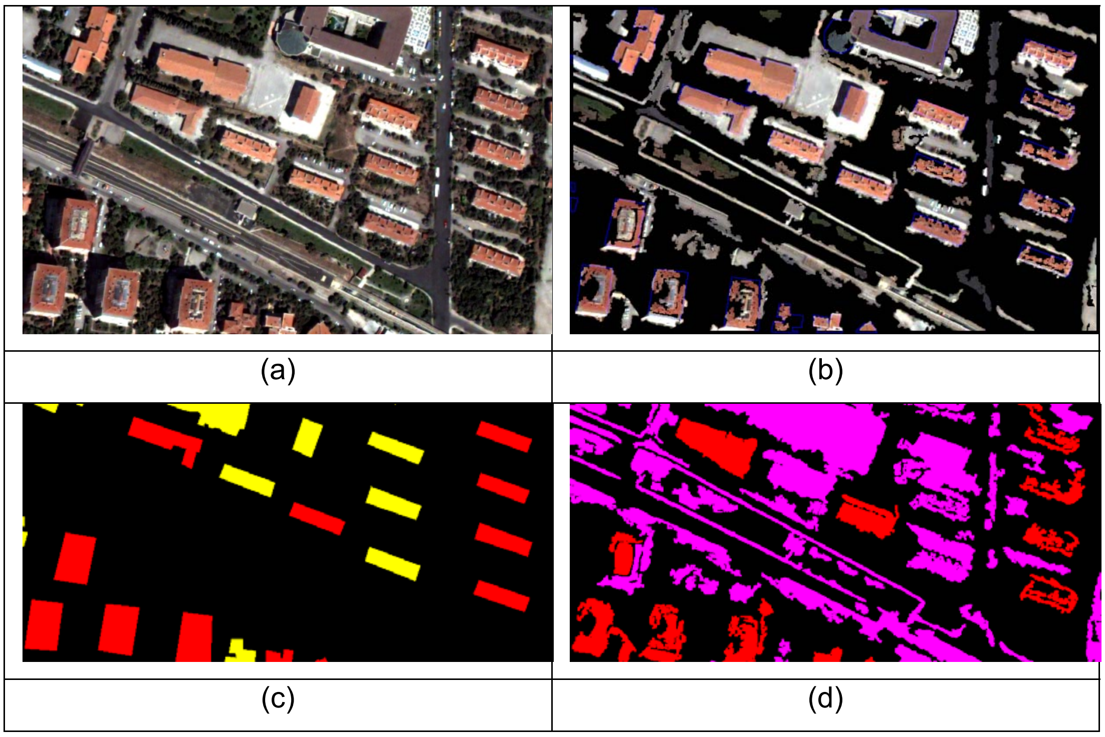


**Figure 4.40: Visual Results for Adaboost with Intensity Textural Features for 3. Image**

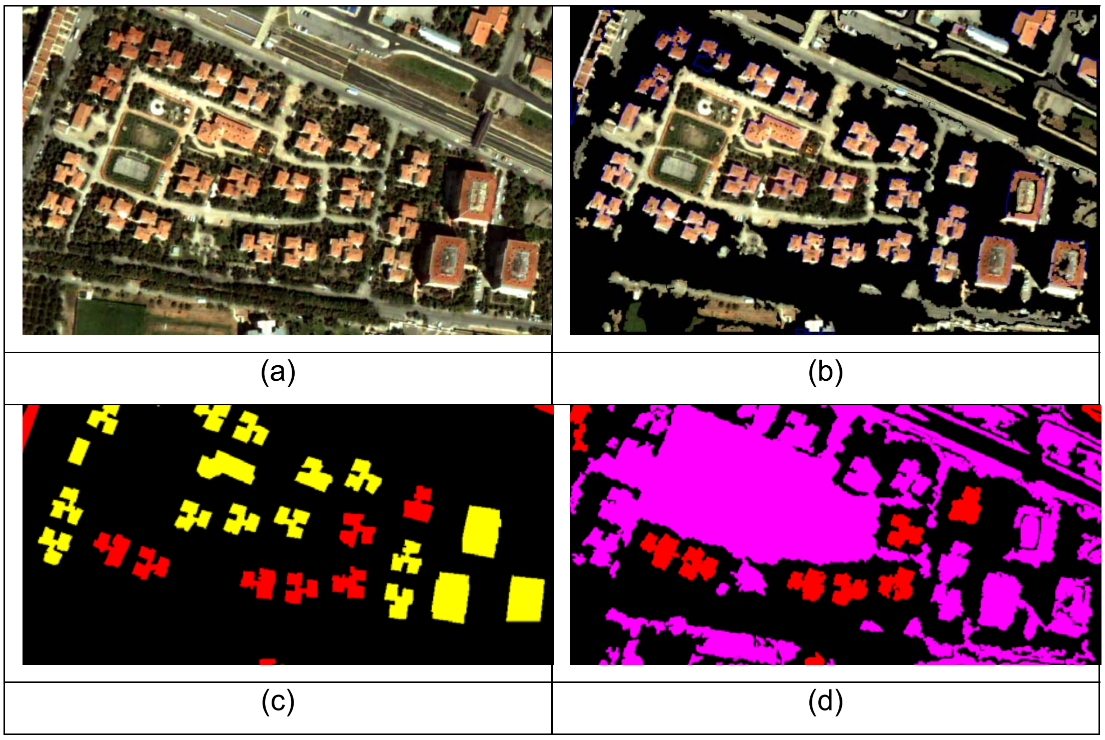


**Figure 4.41: Visual Results for Adaboost with Intensity Textural Features for 4. Image**

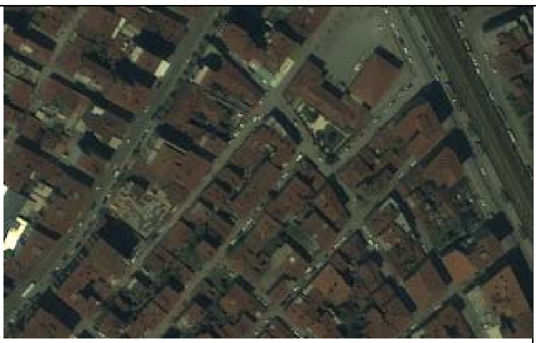
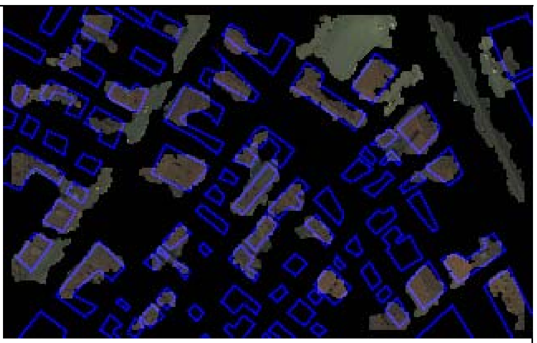
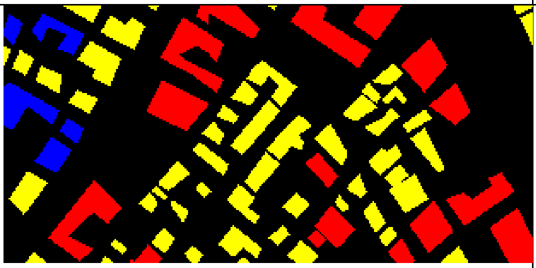
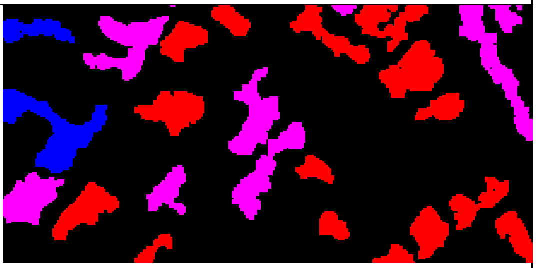
| | |
|---|---|
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

**Figure 4.42: Visual Results for Adaboost with Intensity Textural Features for 5. Image**

The pixel performance of image set 2 is given in Table 4.47. In that table high SF values shows that the detection of non-building areas as building areas is very common.

**Table 4.47: Pixel Performance of the Images in 2. Set**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 6 | 0.7474 | 0.2285 | 52.5116 | 20.5658 |
| 7 | 0.6506 | 0.3326 | 51.2284 | 26.216 |
| 8 | 0.8168 | 0.0372 | 83.1058 | 17.664 |
| 9 | 0.7492 | 0.0087 | 96.6483 | 24.8606 |
| 10 | 0.587 | 0.0314 | 92.9317 | 40.0375 |
| All | 0,7102 | 0,1277 | 75,2852 | 25,8688 |

Figure 4.43 illustrates the results of object performance vs. object detection threshold for image set 2 and Table 4.48 shows the results when object detection threshold is equal to 0.4. The correct detection rate is highly low.

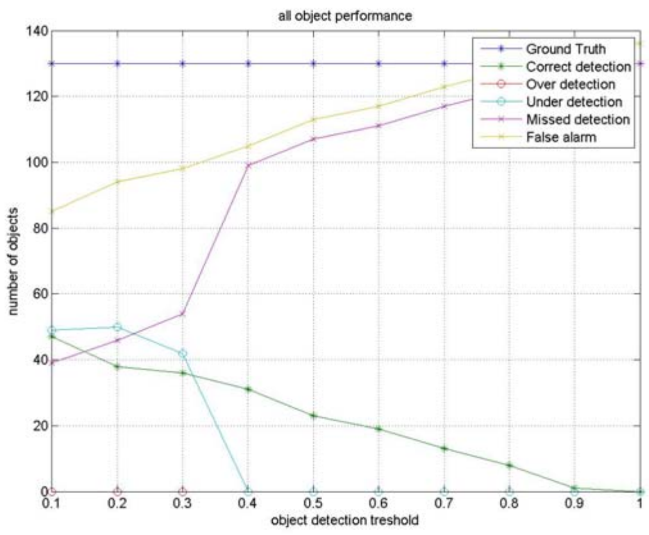


**Figure 4.43: Object Performance vs. Object Detection Threshold**

**Table 4.48: Object Performance of the 1. Set for T = 0.4**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 6 | 21 | 9 | 0 | 0 | 12 | 42 |
| 7 | 50 | 20 | 0 | 0 | 30 | 8 |
| 8 | 22 | 1 | 0 | 0 | 21 | 13 |
| 9 | 11 | 0 | 0 | 0 | 11 | 36 |
| 10 | 26 | 1 | 0 | 0 | 25 | 6 |
| All | 130 | 31 | 0 | 0 | 99 | 105 |

The visual results for image 6 to 10 are given in
Figure 4.44 to
Figure 4.48. In these figures (a) is the original image, (b) is the original image masked by the building patches, together with the borders of the buildings in the ground truth data (c) is ground truth, (d) is the building patches extracted by the algorithm. In the results, it is seen that in image 1 and image 2, some of the buildings are extracted but in image 5 which is a dense area of buildings and roads, the extraction is nearly the whole image.

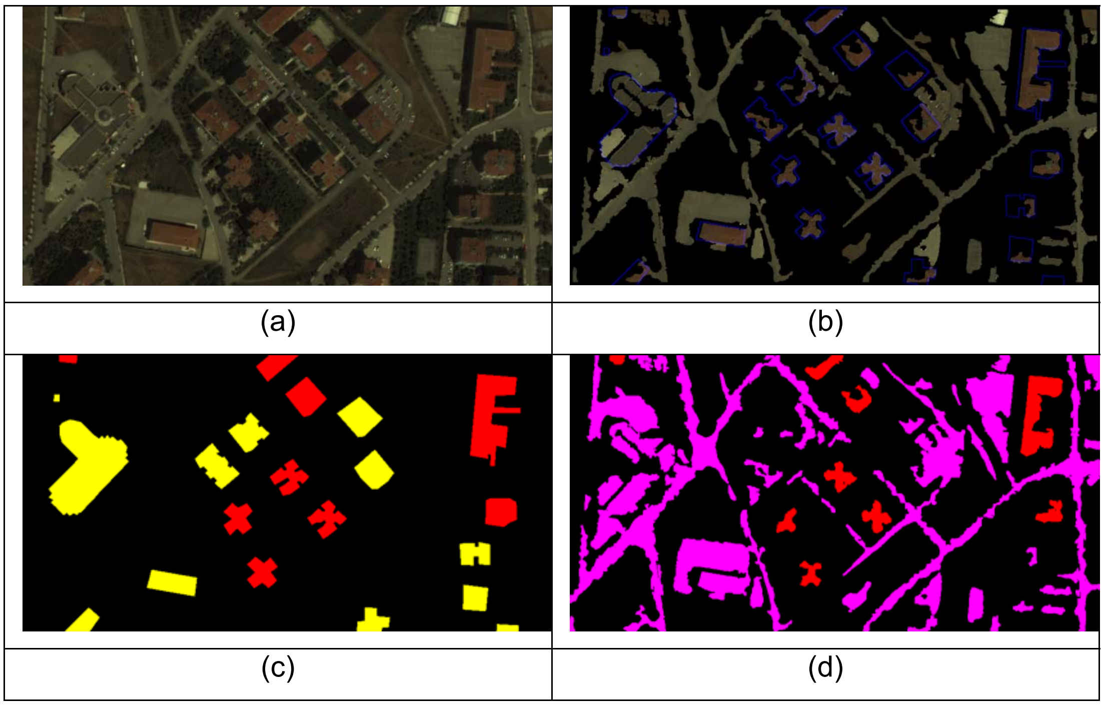
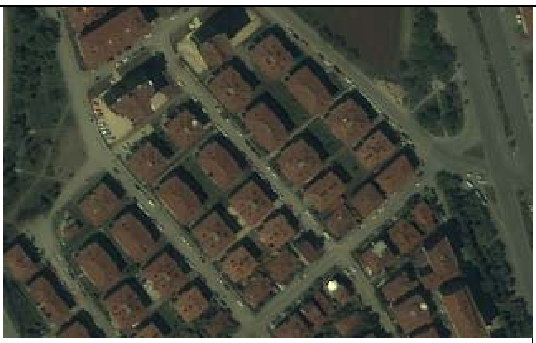
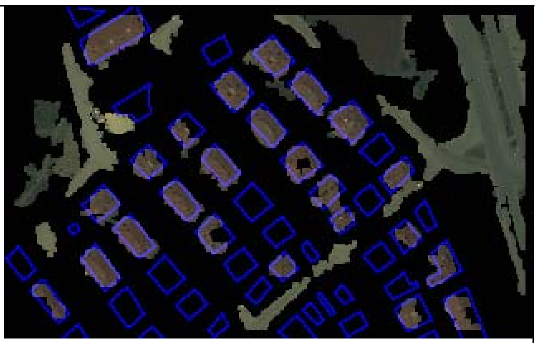


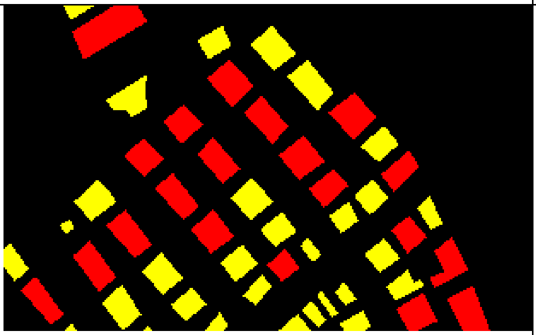
Figure 4.44: Visual Results for Adaboost with Intensity Textural Features for 6. Image



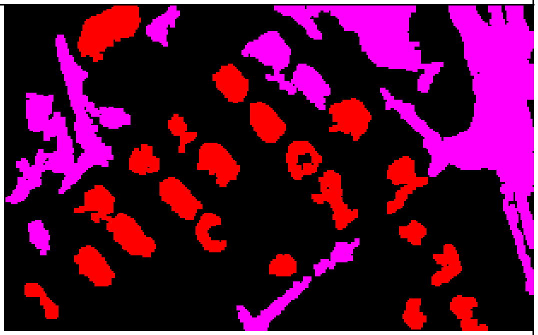
(a)



(b)



(c)



(d)

**Figure 4.45: Visual Results for Adaboost with Intensity Textural Features for 7. Image**

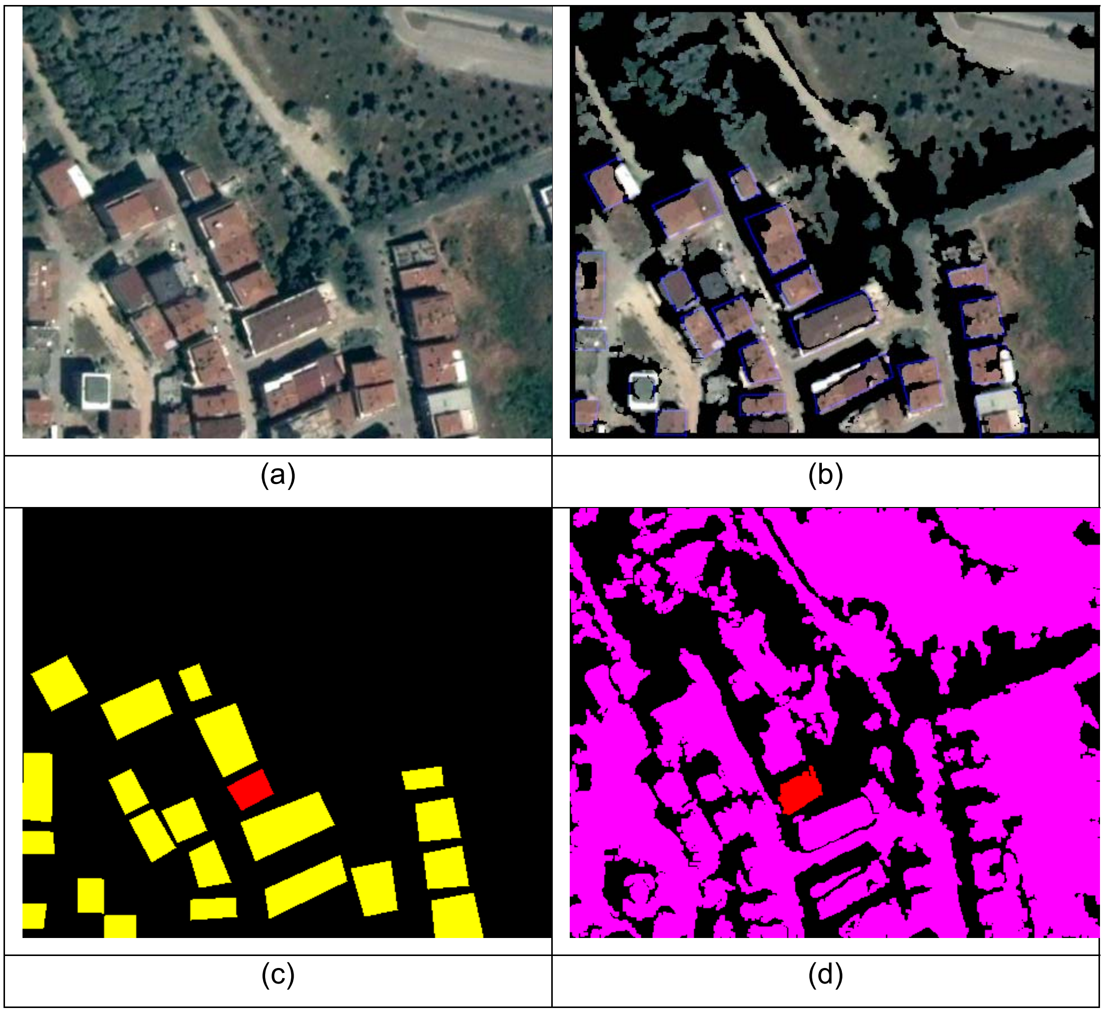


**Figure 4.46: Visual Results for Adaboost with Intensity Textural Features for 8. Image**

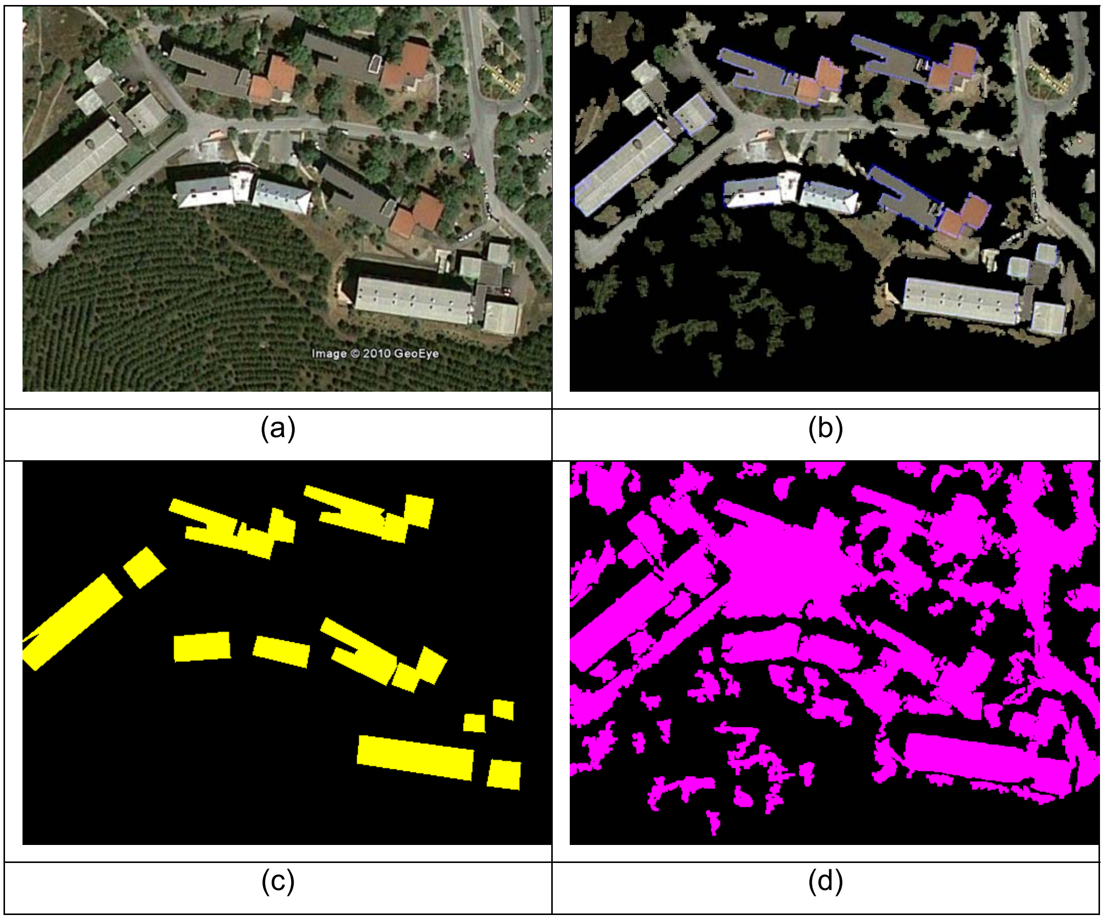


**Figure 4.47: Visual Results for Adaboost with Intensity Textural Features for 9. Image**

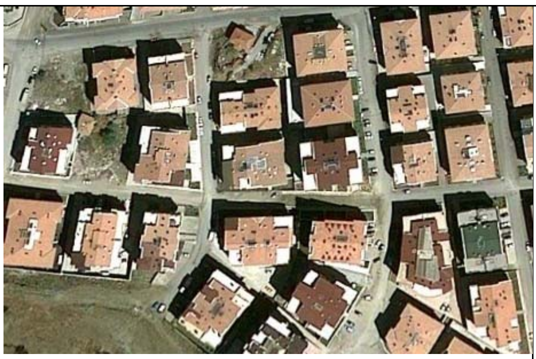
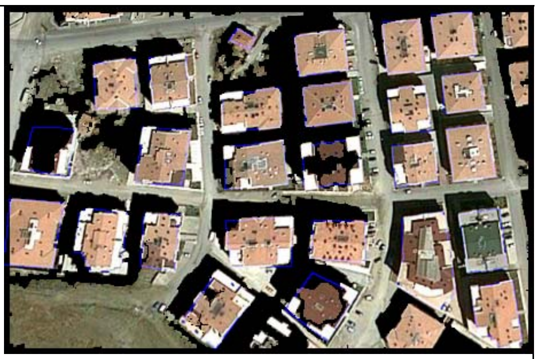
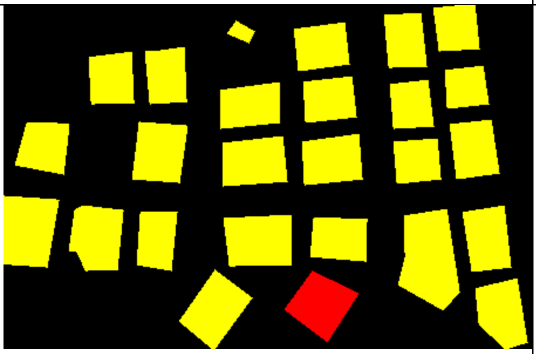
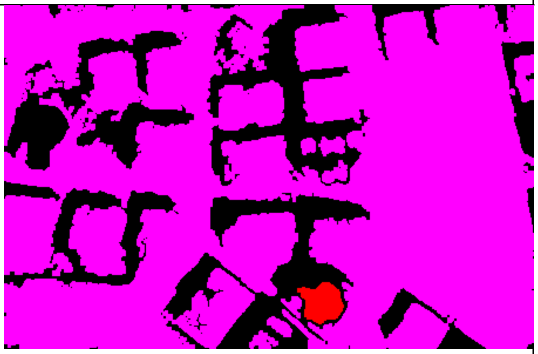
| | |
|---|---|
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

**Figure 4.48: Visual Results for Adaboost with Intensity Textural Features for 10. Image**

### 4.2.2 Adaboost with all features

Second classifier is Adaboost with all features. The pixel performance of image set 1 is given in Table 4.49.

Table 4.49: Pixel Performance of the 1. Set

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.2779 | 1.5222 | 32.1742 | 28.6293 |
| 2 | 0.1648 | 0.3159 | 72.5582 | 63.4719 |
| 3 | 0.4928 | 0.0204 | 96.1421 | 49.7129 |
| 4 | 0.3764 | 0.0717 | 89.6938 | 58.1881 |
| 5 | 0.1162 | 1.6474 | 34.9162 | 33.3833 |
| All | 0.28562 | 0.7155 | 65.0969 | 46.6771 |

Figure 4.49 illustrates the results of object performance vs. object detection threshold and Table 4.50 shows the results when object detection threshold is equal to 0.4.

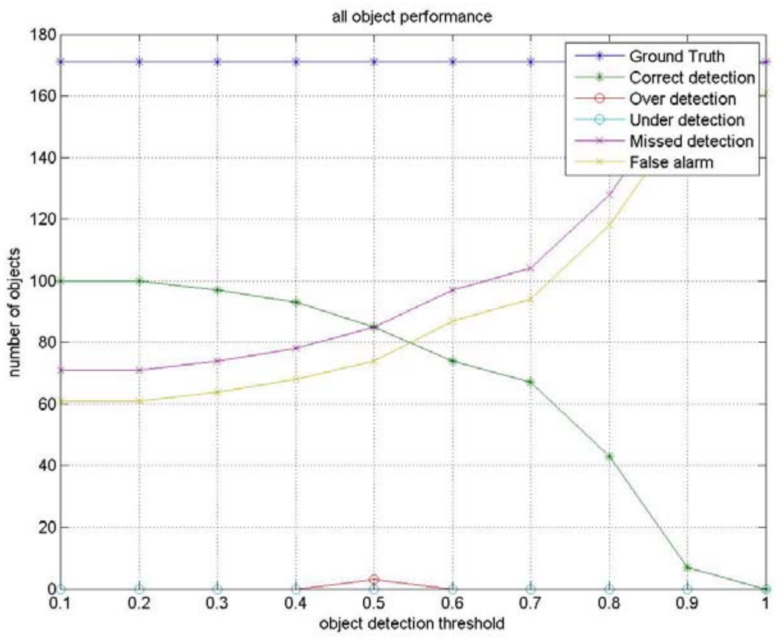


Figure 4.49: Object Performance vs. Object Detection Threshold

**Table 4.50: Object Performance of the 1. Set for T = 0.4**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 19 | 0 | 0 | 16 | 7 |
| 2 | 13 | 10 | 0 | 0 | 3 | 9 |
| 3 | 24 | 20 | 0 | 0 | 4 | 32 |
| 4 | 29 | 27 | 0 | 0 | 2 | 15 |
| 5 | 70 | 17 | 0 | 0 | 53 | 5 |
| All | 171 | 93 | 0 | 0 | 78 | 68 |

The visual results for image 1 to 5 are given in
Figure 4.50 to
Figure 4.54. In these figures (a) is the original image, (b) is the original image masked by the building patches, together with the borders of the buildings in the ground truth data (c) is ground truth, (d) is the building patches extracted by the algorithm.

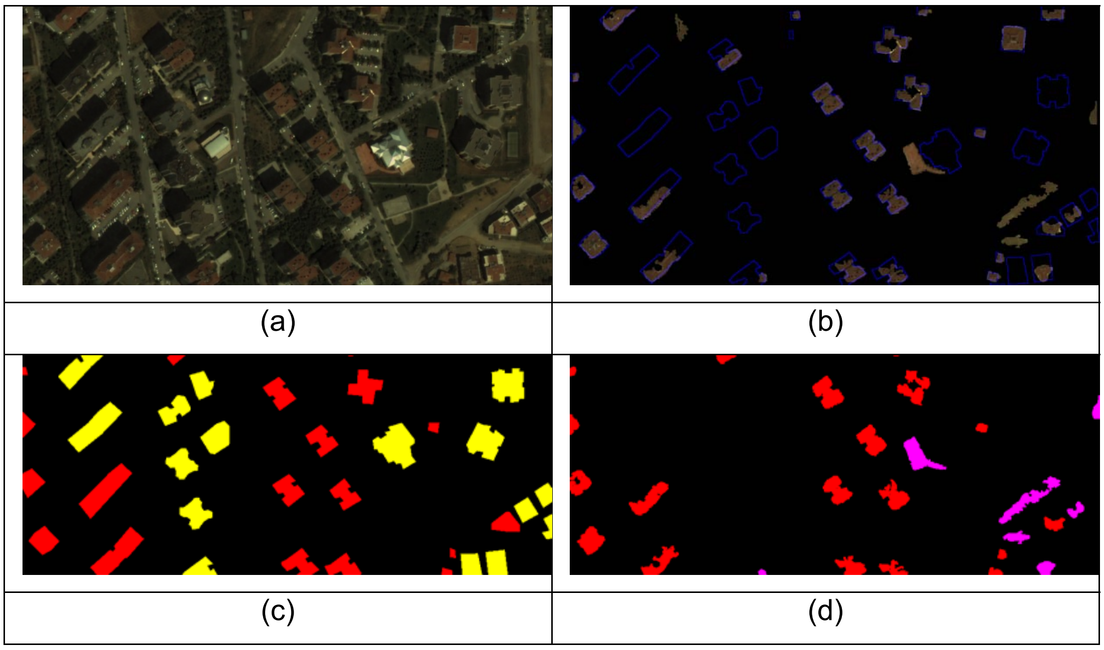


**Figure 4.50: Visual Results for Adaboost with All Features for 1. Image**

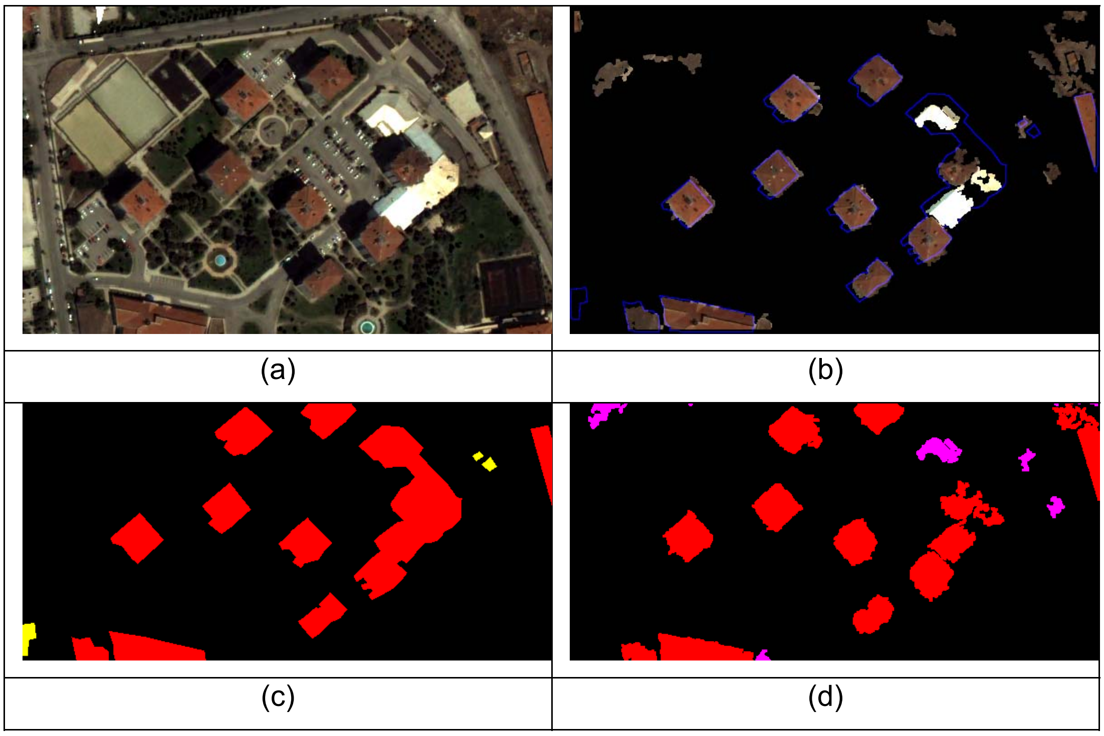


**Figure 4.51: Visual Results for Adaboost with All Features for 2. Image**

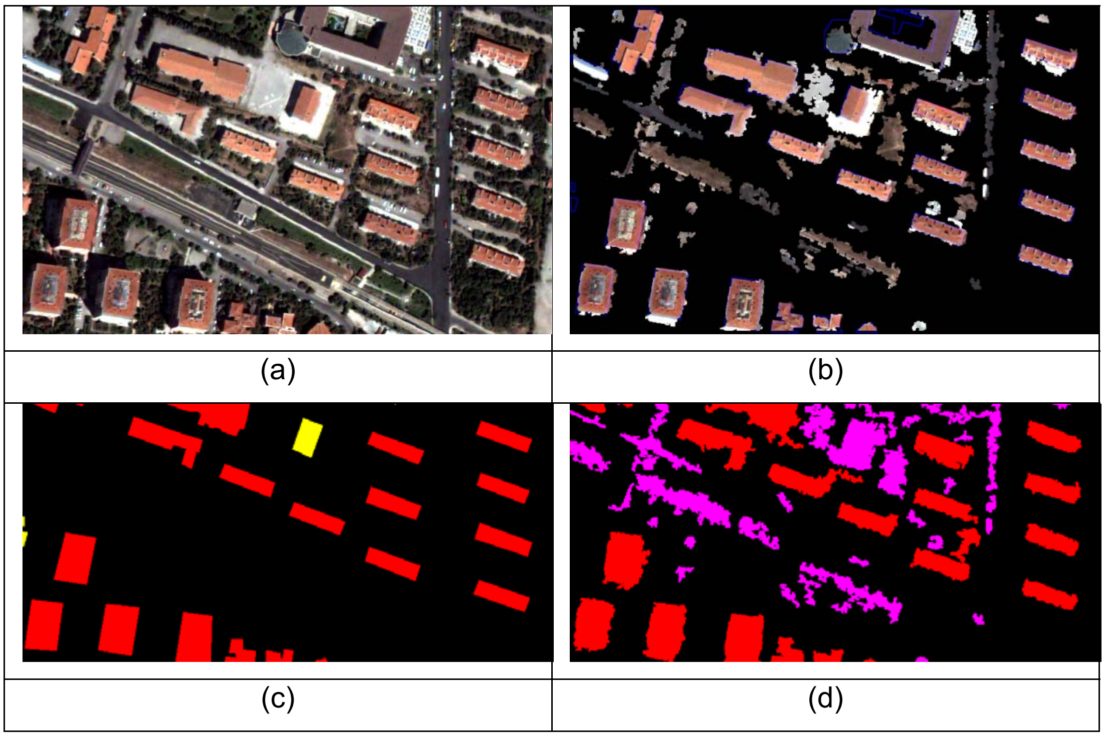


**Figure 4.52: Visual Results for Adaboost with All Features for 3. Image**

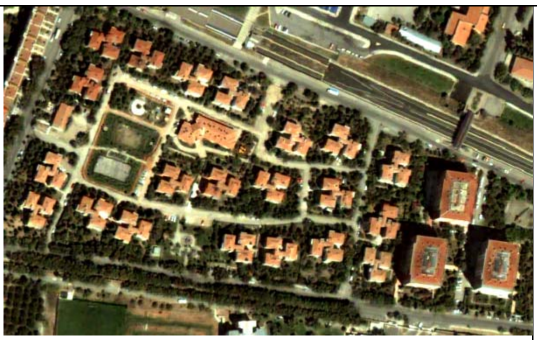
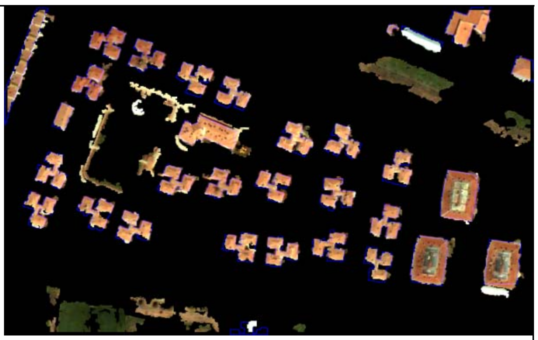
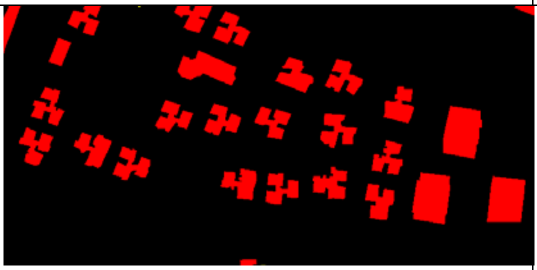
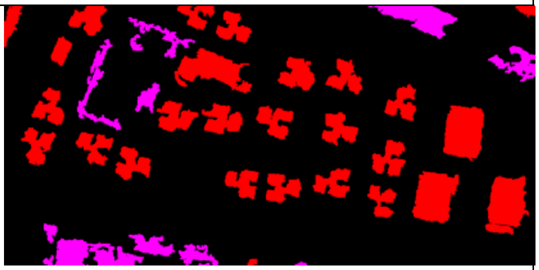
| | |
|---|---|
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

**Figure 4.53: Visual Results for Adaboost with All Features for 4. Image**

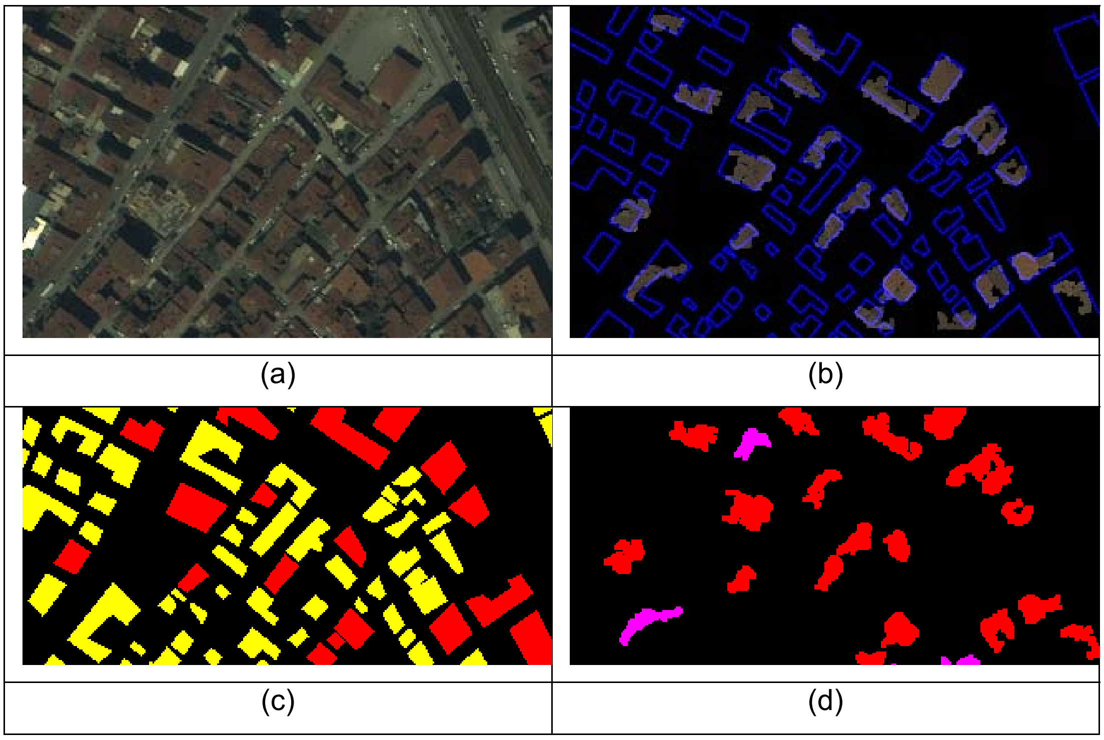


**Figure 4.54: Visual Results for Adaboost with All Features for 5. Image**

The pixel performance of image set 2 is given in Table 4.51. These are the best results achieved.

**Table 4.51: Pixel Performance of the 2. Set**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 6 | 0.1611 | 0.6137 | 57.7512 | 51.9864 |
| 7 | 0.1445 | 0.3662 | 70.0223 | 62.6156 |
| 8 | 0.7734 | 0.1006 | 69.2414 | 20.5847 |
| 9 | 0.1328 | 3.3477 | 20.5754 | 19.9471 |
| 10 | 0.1084 | 0.4707 | 65.447 | 60.6243 |
| All | 0.26404 | 0.9798 | 56.6075 | 43.1516 |

Figure 4.55 illustrates the results of object performance vs. object detection threshold for image set 2 and Table 4.52 shows the results when object detection threshold is equal to 0.4. The highest correct detection count is reached by this classifier.

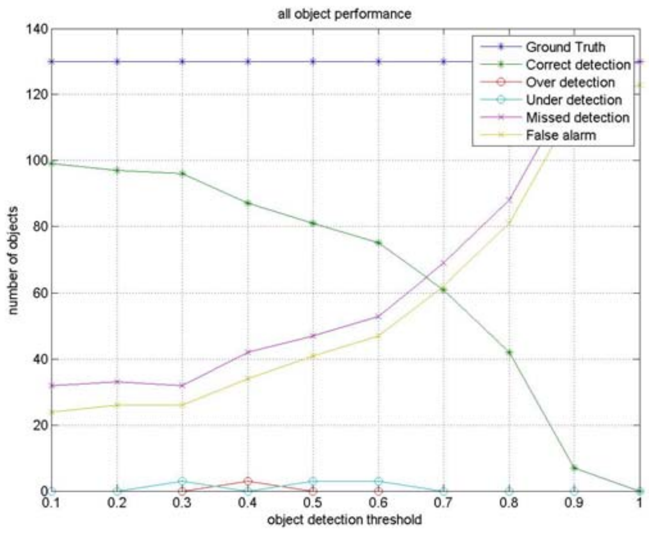


**Figure 4.55: Object Performance vs. Object Detection Threshold**

**Table 4.52: Object Performance of the Images in 2. Set**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 6 | 21 | 17 | 3 | 0 | 3 | 6 |
| 7 | 50 | 31 | 0 | 0 | 19 | 0 |
| 8 | 22 | 15 | 0 | 0 | 7 | 18 |
| 9 | 11 | 1 | 0 | 0 | 10 | 4 |
| 10 | 26 | 23 | 0 | 0 | 3 | 6 |
| All | 130 | 87 | 3 | 0 | 42 | 34 |

The visual results for image 6 to 10 are given in
Figure 4.56 to
Figure 4.60. In these figures (a) is the original image, (b) is the original image masked by the building patches, together with the borders of the buildings in the ground truth data (c) is ground truth, (d) is the building patches extracted by the algorithm.

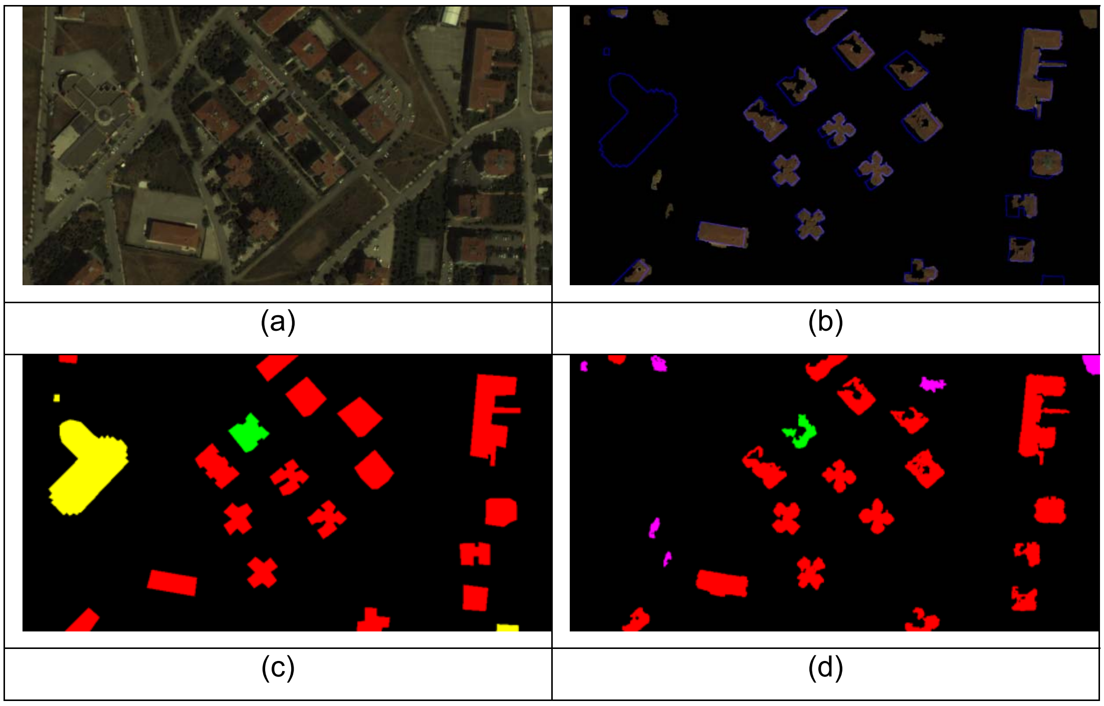


**Figure 4.56: Visual Results for Adaboost with All Features for 6. Image**

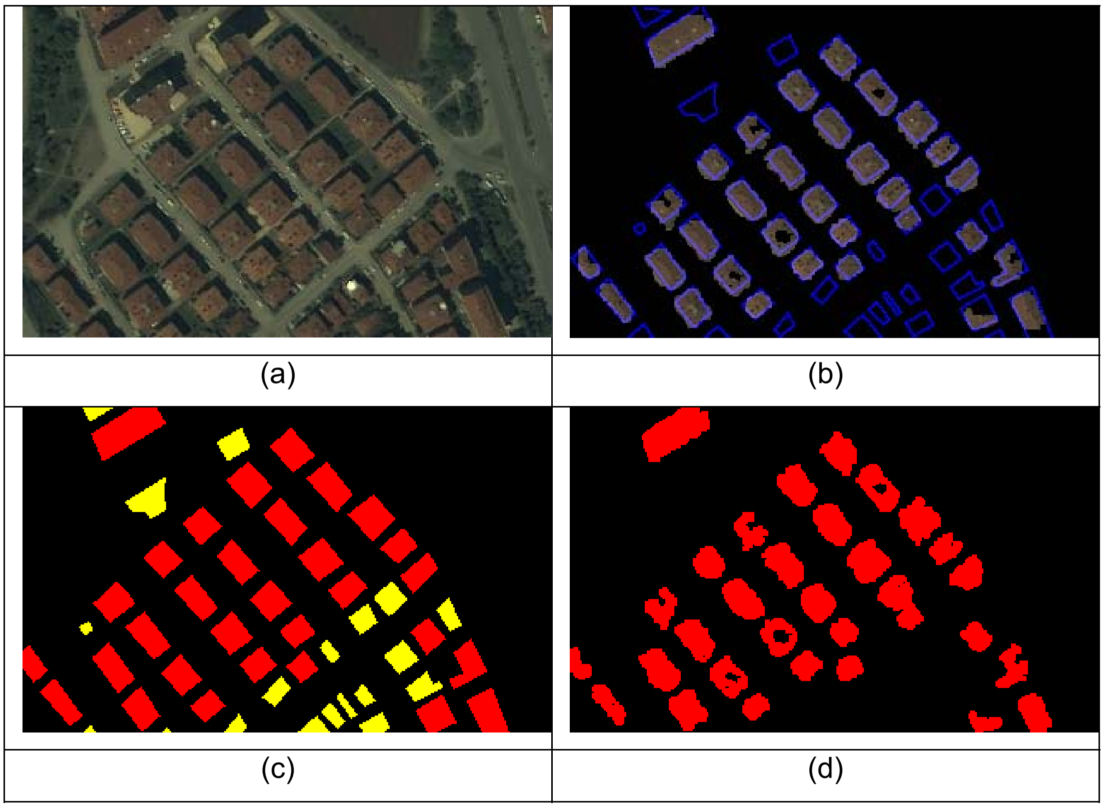

(a) (b) (c) (d)

**Figure 4.57: Visual Results for Adaboost with All Features for 7. Image**

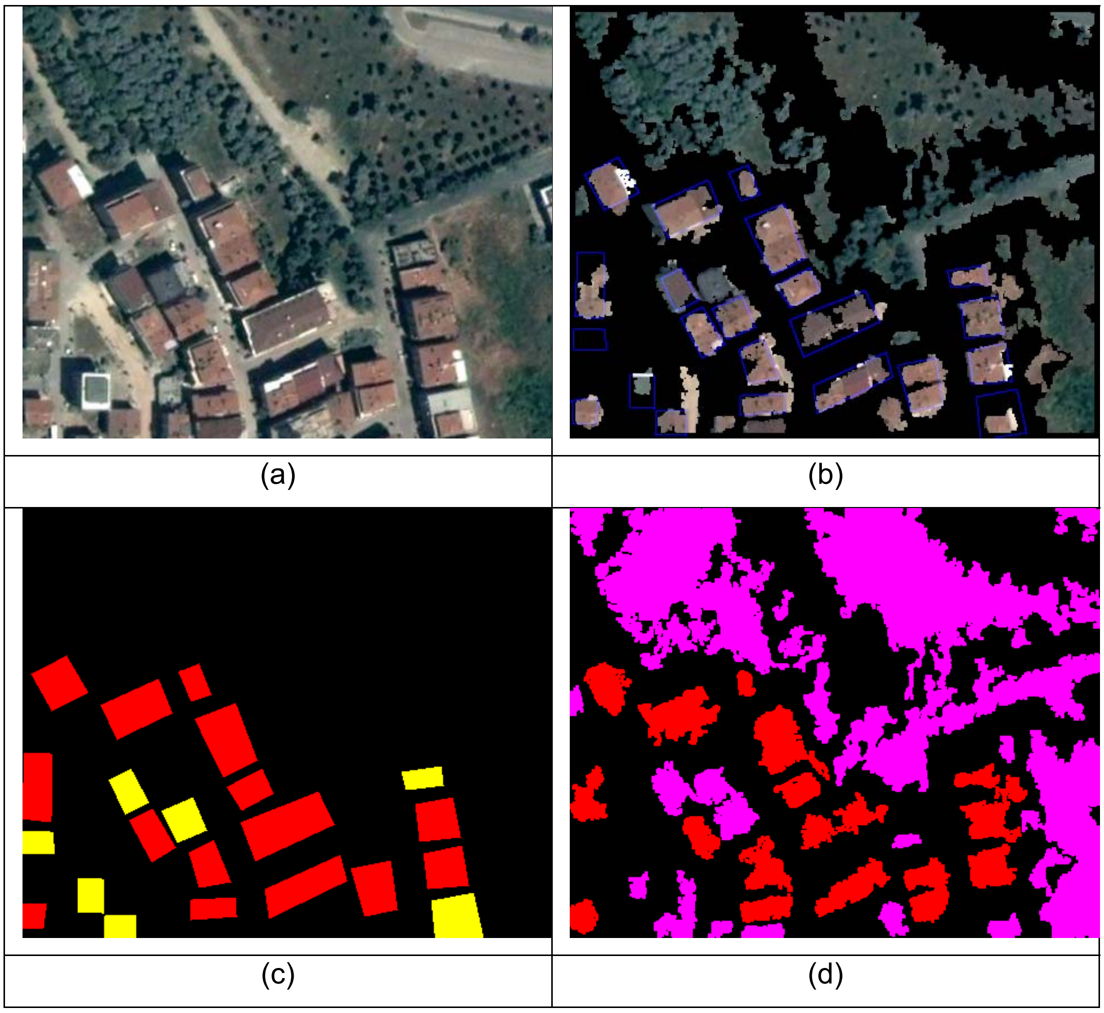


**Figure 4.58: Visual Results for Adaboost with All Features for 8. Image**

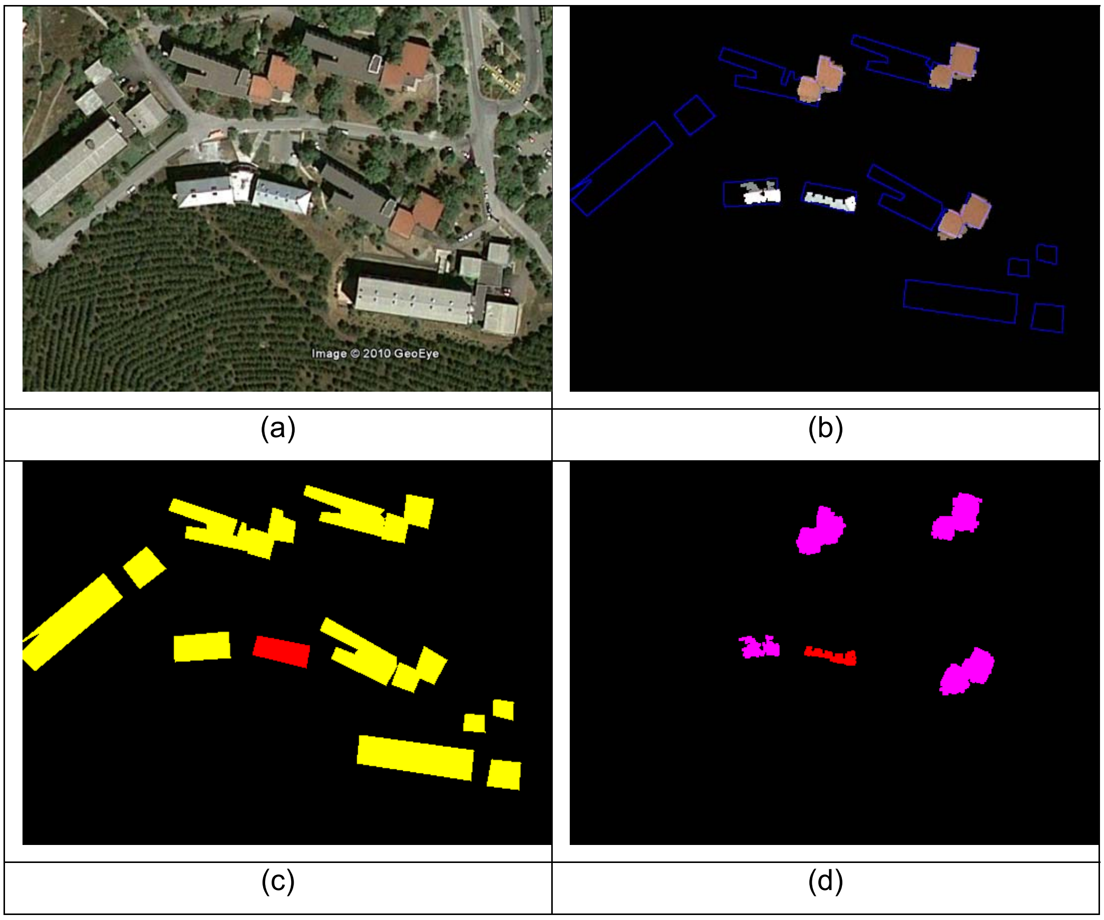


**Figure 4.59: Visual Results for Adaboost with All Features for 9. Image**

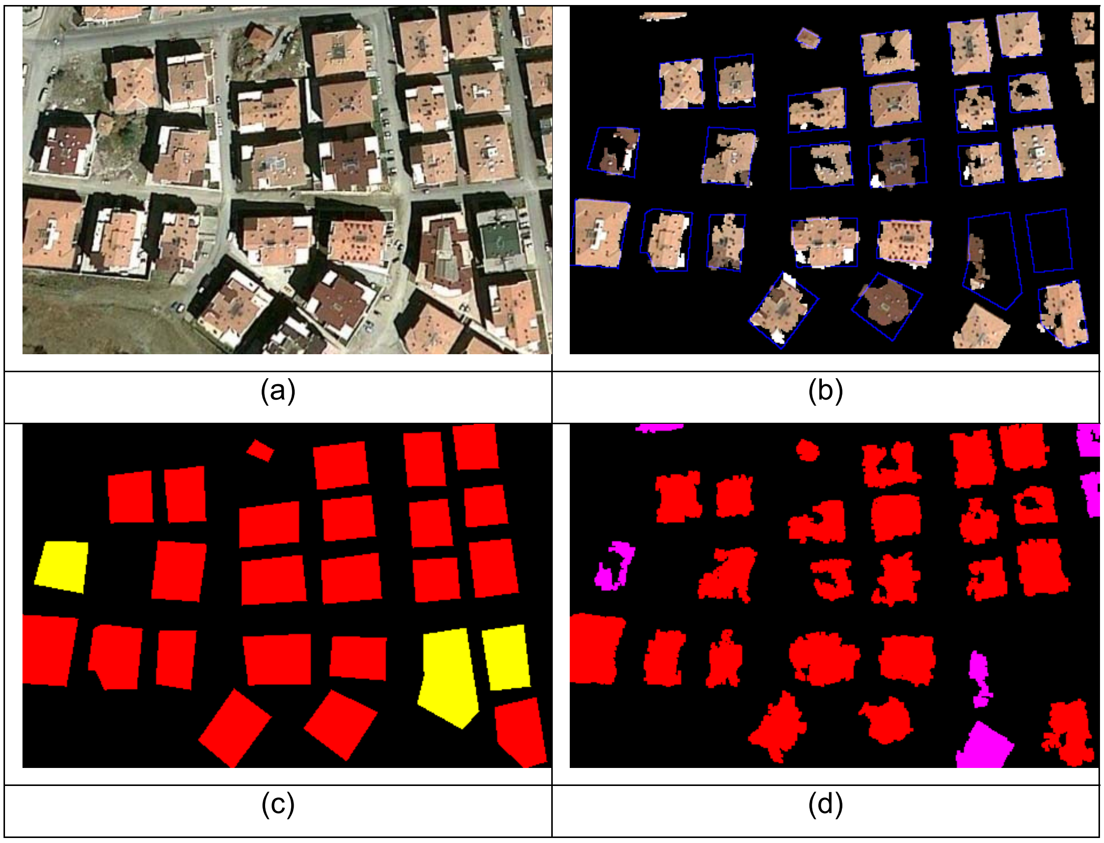


**Figure 4.60: Visual Results for Adaboost with All Features for 10. Image**

### 4.2.3 SVM quadratic kernel with Adaboost selected features

Table 4.53 gives the pixel performance of the images used in sensitivity analyses, which is image set 1. Only %20 of the images are used in training and performance is evaluated in %80 of the images. The success in image 3 is maximum and the success in image 1 is minimum.

**Table 4.53: Pixel Performance Values of Image 1 to Image 5**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.2167 | 2.8313 | 21.6693 | 20.4434 |
| 2 | 0.0848 | 0.5978 | 60.4892 | 57.2784 |
| 3 | 0.0842 | 0.2356 | 79.54 | 74.1203 |
| 4 | 0.1694 | 0.1503 | 84.6772 | 72.2077 |
| 5 | 0.2979 | 1.0769 | 39.4652 | 33.8036 |
| All | 0.1706 | 0.9784 | 57.1682 | 51.5707 |

Figure 4.61 illustrates the object performance changing with object detection threshold. After 0.9, no objects are found i.e. there are no objects overlapping in %90 in ground truth and output of the classifier but note that there may be some operator faults while extracting ground truth objects. Table 4.54 shows the results when object detection threshold is equal to 0.4.

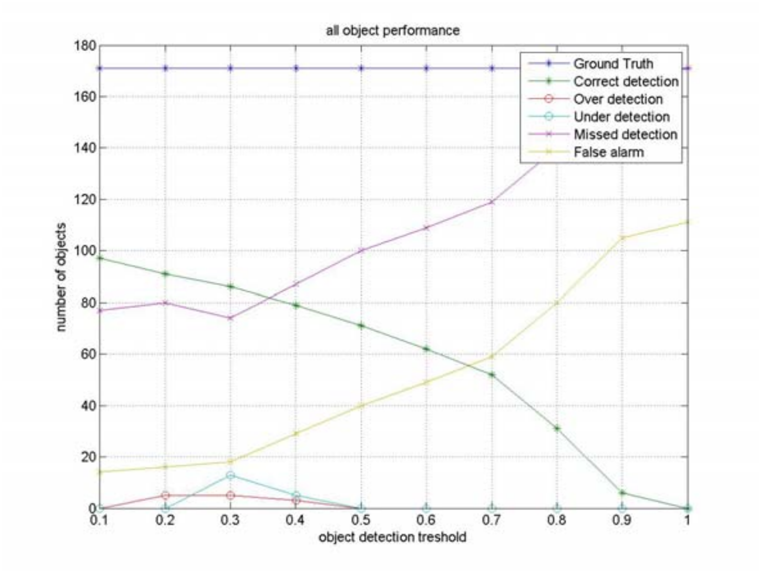


**Figure 4.61: Object performance vs. object detection threshold**

**Table 4.54: Object Performance of the 1. Set for T = 0.4**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 11 | 3 | 0 | 23 | 9 |
| 2 | 13 | 8 | 0 | 0 | 5 | 5 |
| 3 | 24 | 19 | 0 | 0 | 5 | 2 |
| 4 | 29 | 25 | 0 | 0 | 4 | 9 |
| 5 | 70 | 16 | 0 | 5 | 50 | 4 |
| All | 171 | 79 | 3 | 5 | 87 | 29 |

The visual results for image 1 to 5 are given in Figure 4.62 to Figure 4.66. In these figures (a) is the original image, (b) is the original image masked by the building patches, together with the borders of the buildings in the ground truth data (c) is ground truth, (d) is the building patches extracted by the algorithm.

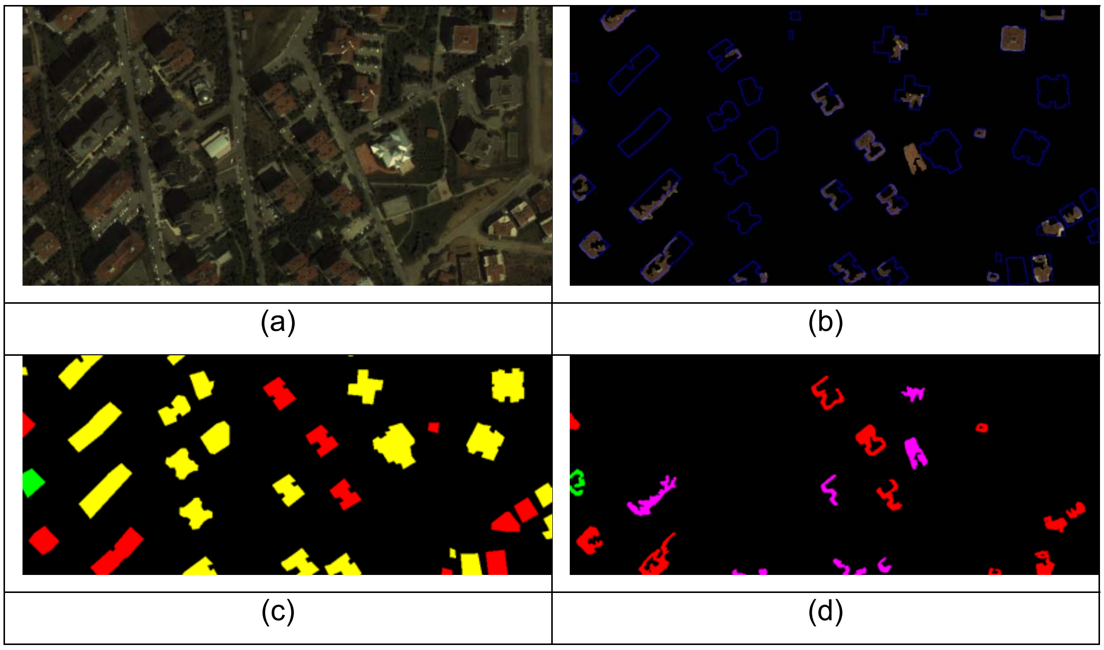


**Figure 4.62: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 1. Image**

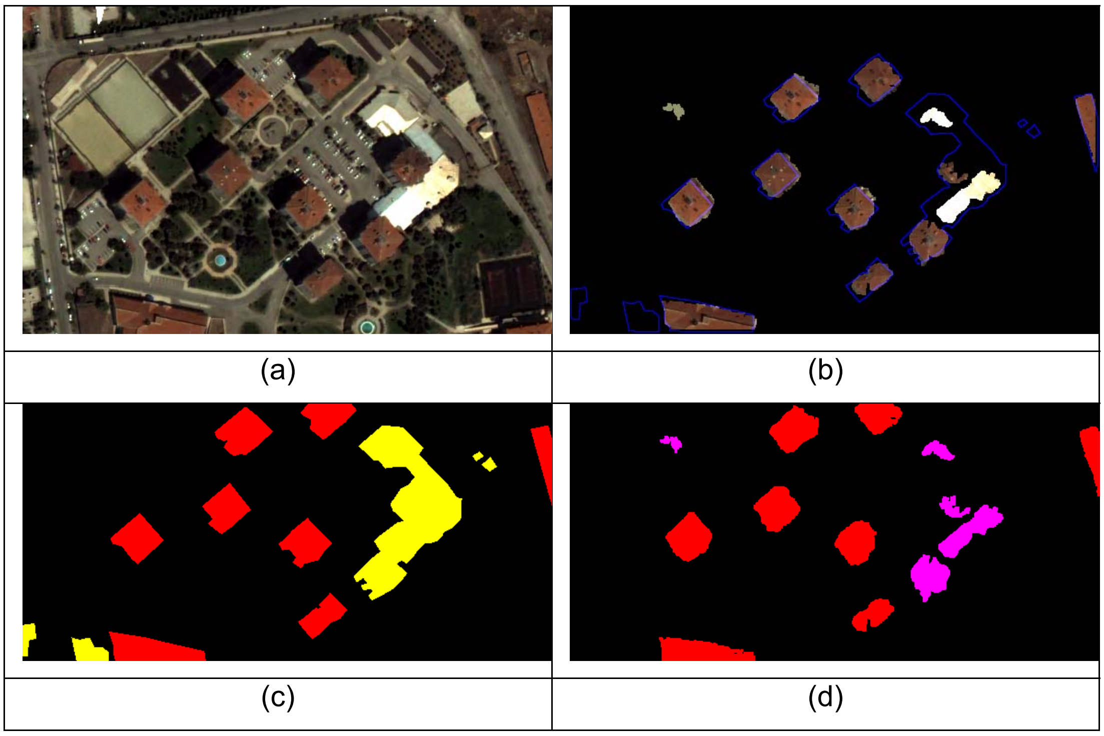


**Figure 4.63: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 2. Image**

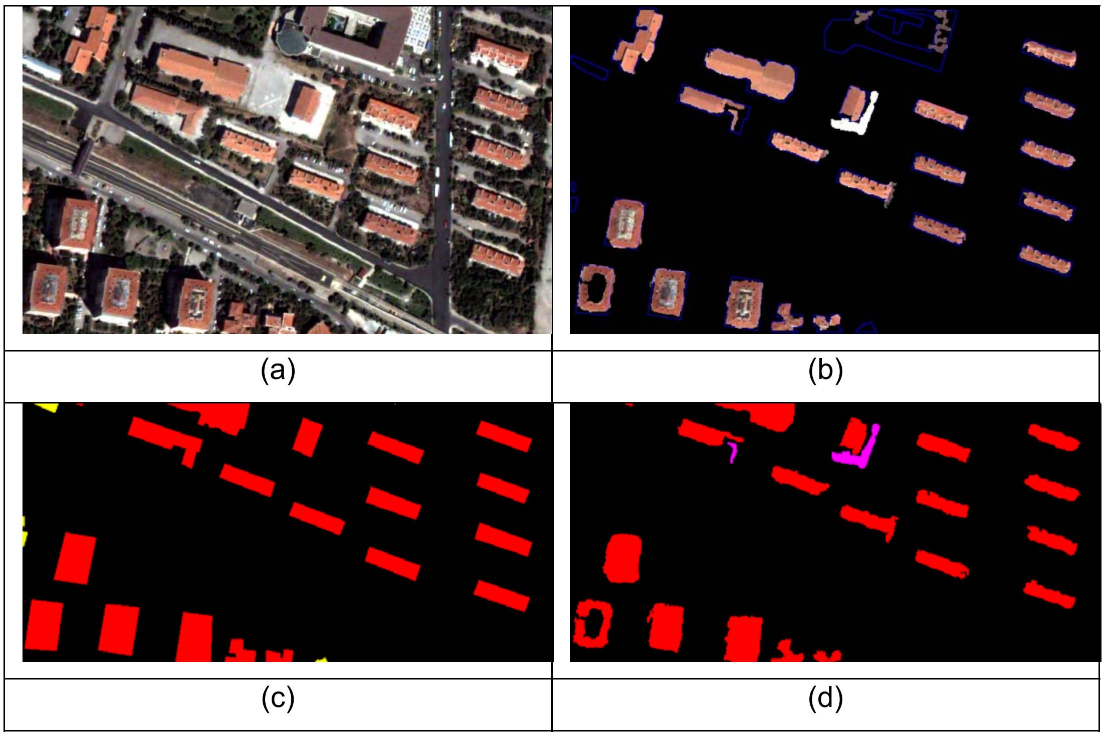


**Figure 4.64: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 3. Image**

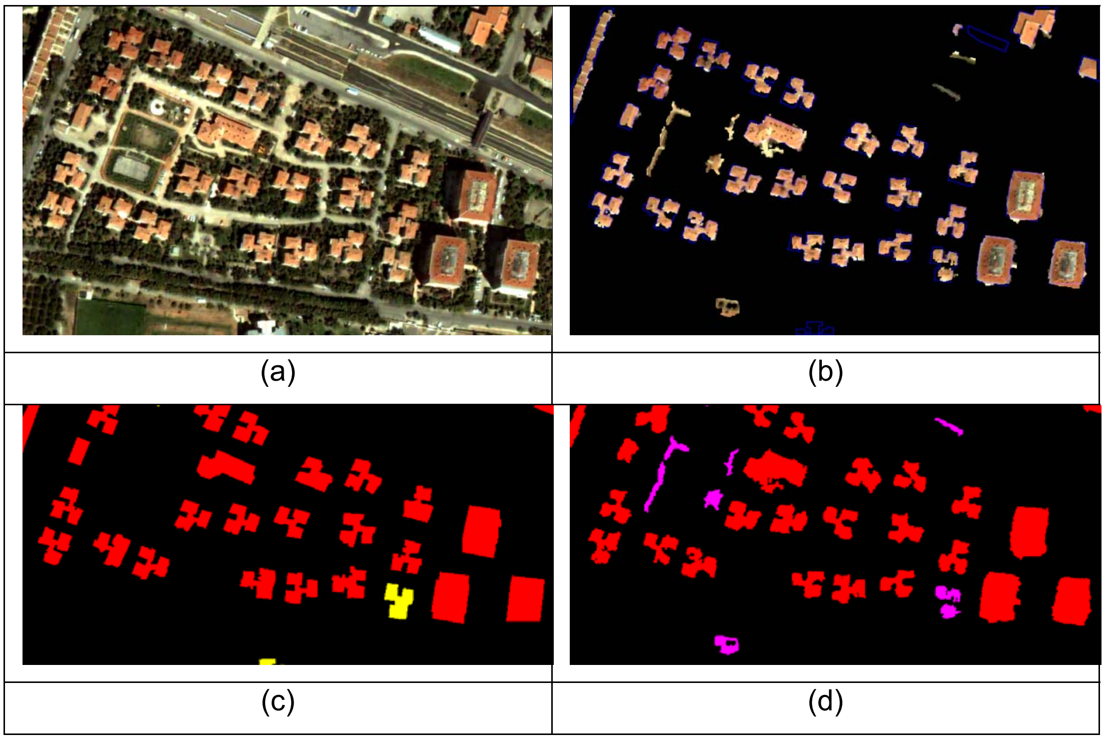

(a)
(b)
(c)
(d)

**Figure 4.65: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 4. Image**

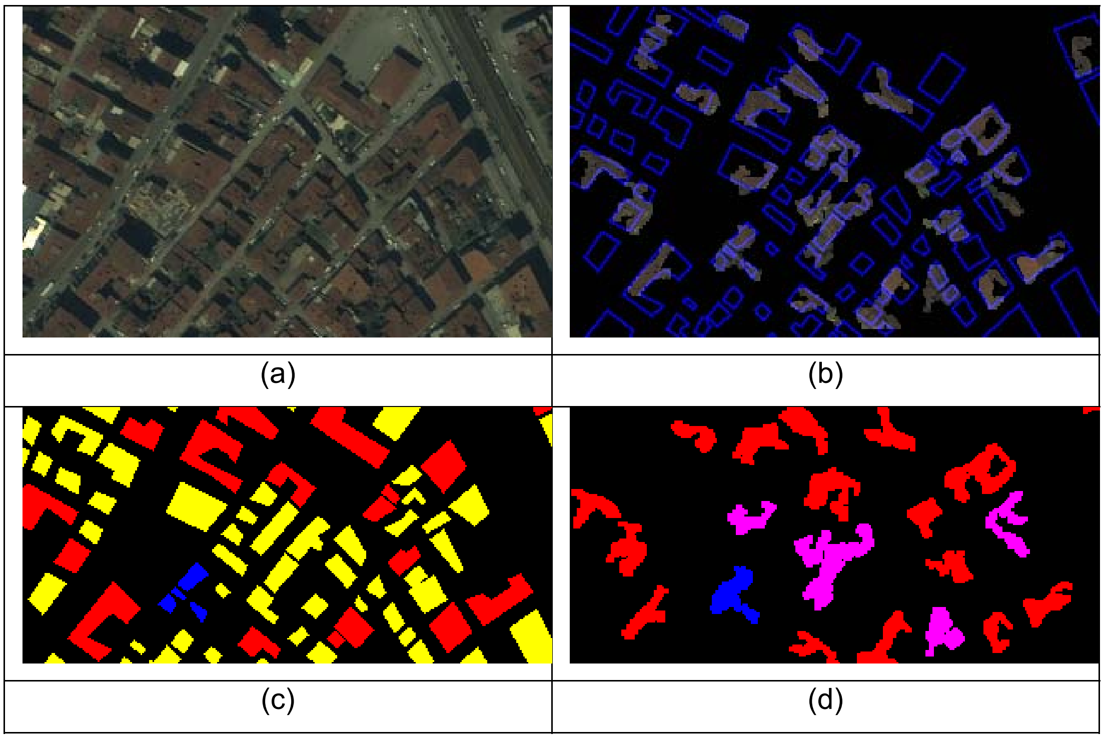


**Figure 4.66: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 5. Image**

The pixel performance of image set 2 is given in Table 4.55.

**Table 4.55: Pixel Performance of the Images in 2. Set**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 6 | 0.1564 | 0.8749 | 49.0893 | 44.9936 |
| 7 | 0.2209 | 0.2283 | 77.3348 | 63.4261 |
| 8 | 0.3797 | 1.9442 | 24.1889 | 21.0697 |
| 9 | 0.1641 | 3.1583 | 20.9279 | 20.102 |
| 10 | 0.0959 | 1.1955 | 43.0598 | 41.1781 |
| All | 0,2034 | 1,4802 | 42,9201 | 38,1539 |

Figure 4.67 illustrates the results of object performance vs. object detection threshold for image set 2 and Table 4.56 shows the results when object detection threshold is equal to 0.4.

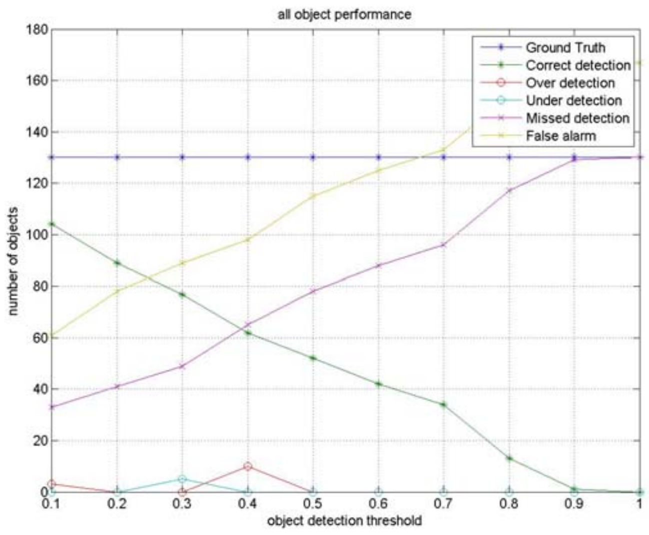


**Figure 4.67: Object Performance vs. Object Detection Threshold**

**Table 4.56: Object Performance of the 2. Set for T = 0.4**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 6 | 21 | 13 | 0 | 0 | 8 | 31 |
| 7 | 50 | 31 | 0 | 0 | 19 | 6 |
| 8 | 22 | 4 | 0 | 0 | 18 | 35 |
| 9 | 11 | 0 | 10 | 0 | 8 | 6 |
| 10 | 26 | 14 | 0 | 0 | 12 | 20 |
| All | 130 | 62 | 10 | 0 | 65 | 98 |

The visual results for image 6 to 10 are given in
Figure 4.68 to
Figure 4.72. In these figures (a) is the original image, (b) is the original image masked by the building patches, together with the borders of the buildings in the ground truth data (c) is ground truth, (d) is the building patches extracted by the algorithm.

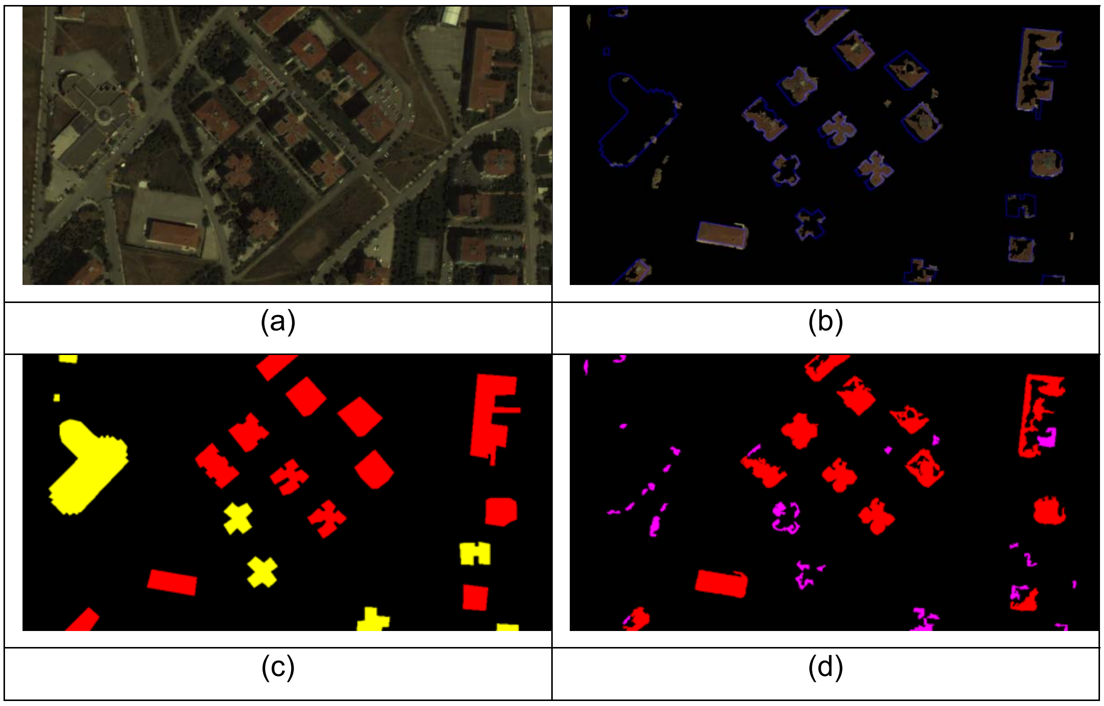


**Figure 4.68: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 6. Image**

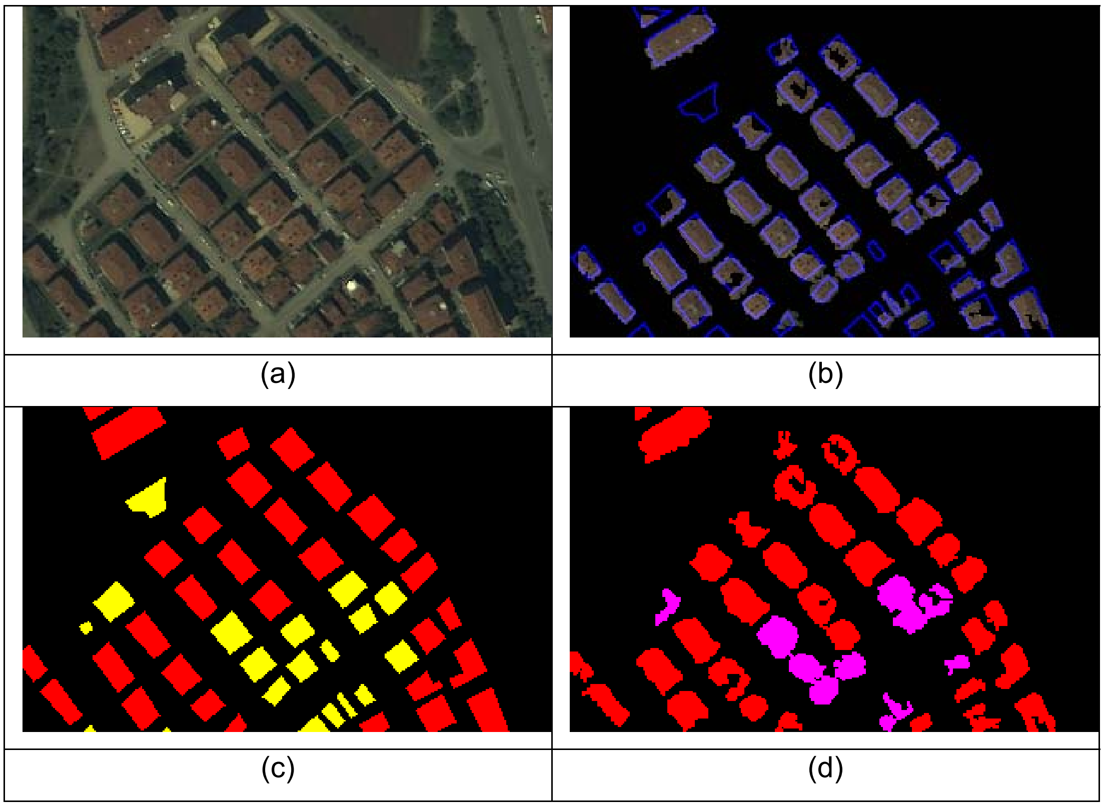


**Figure 4.69: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 7. Image**

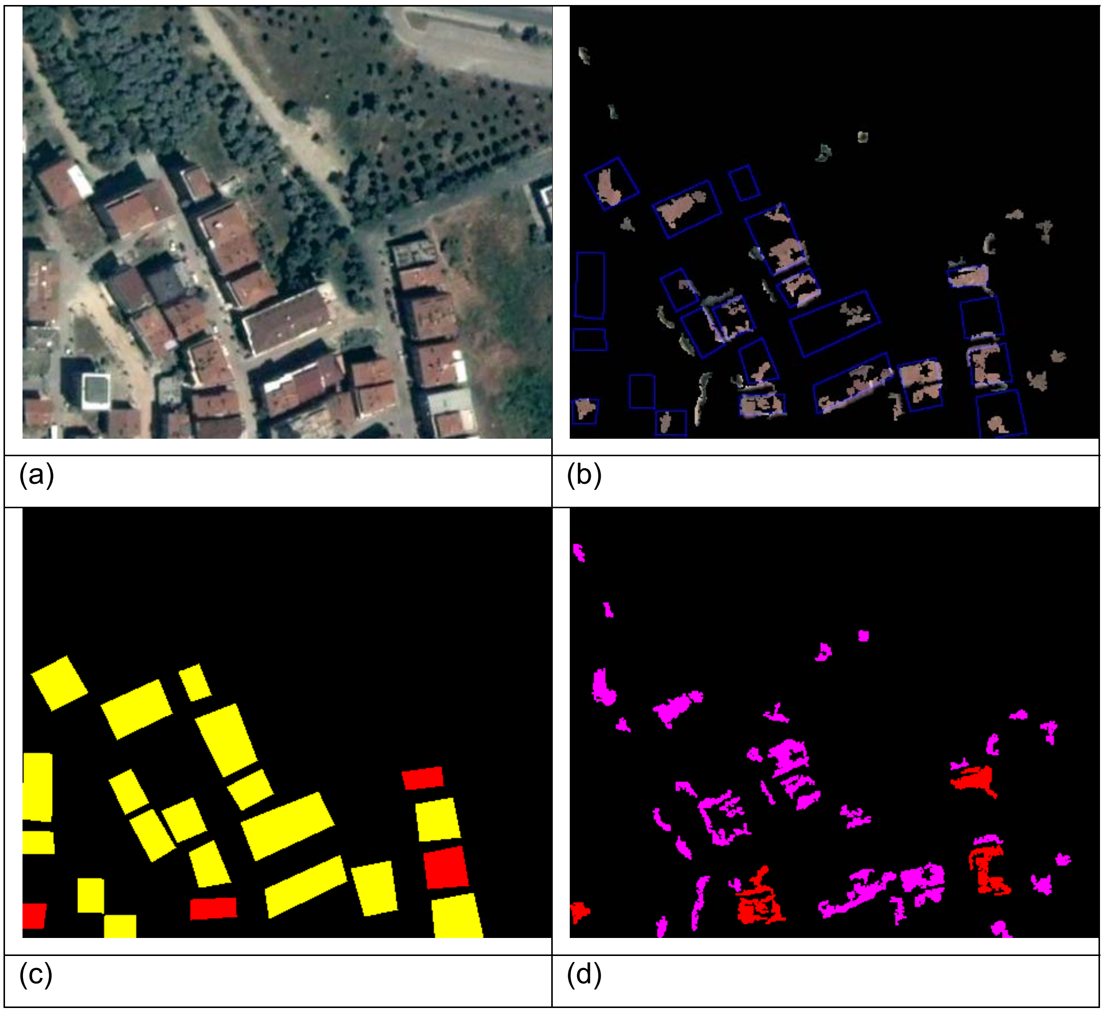


**Figure 4.70: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 8. Image**

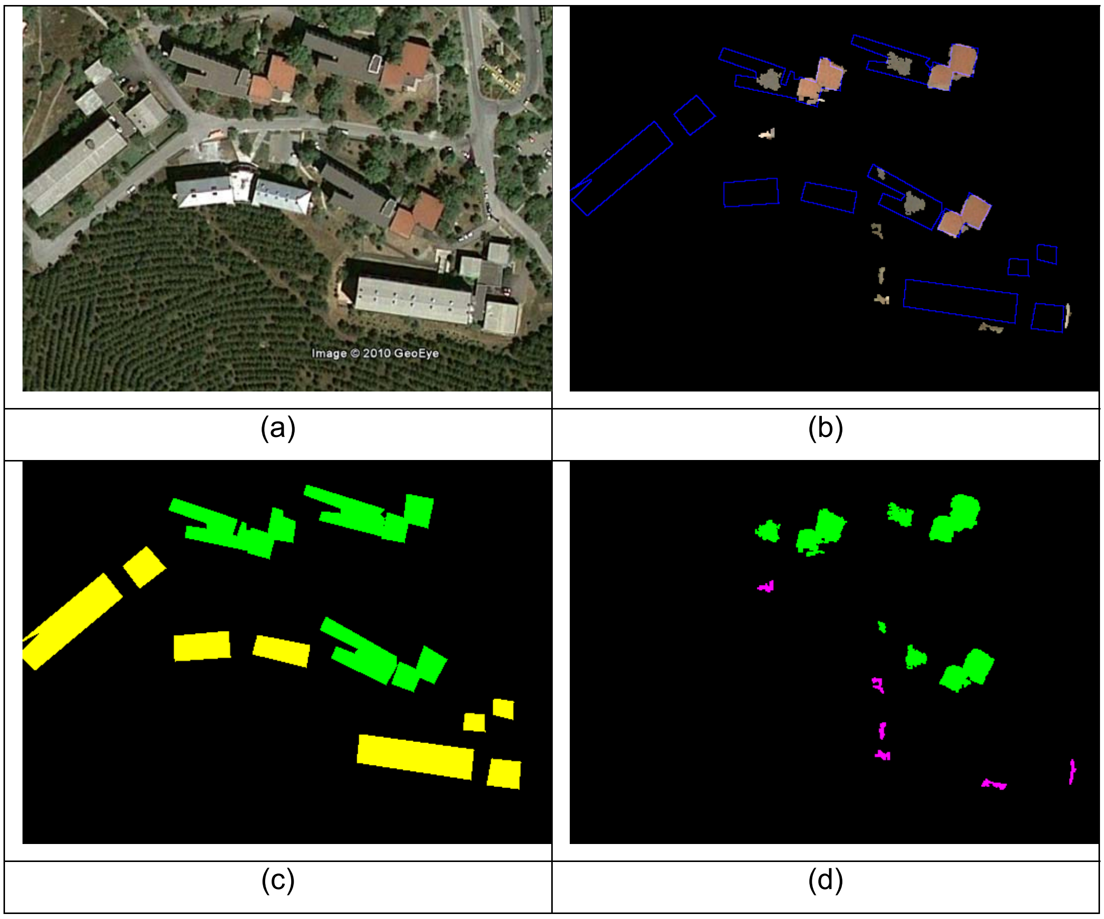


**Figure 4.71: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 9. Image**

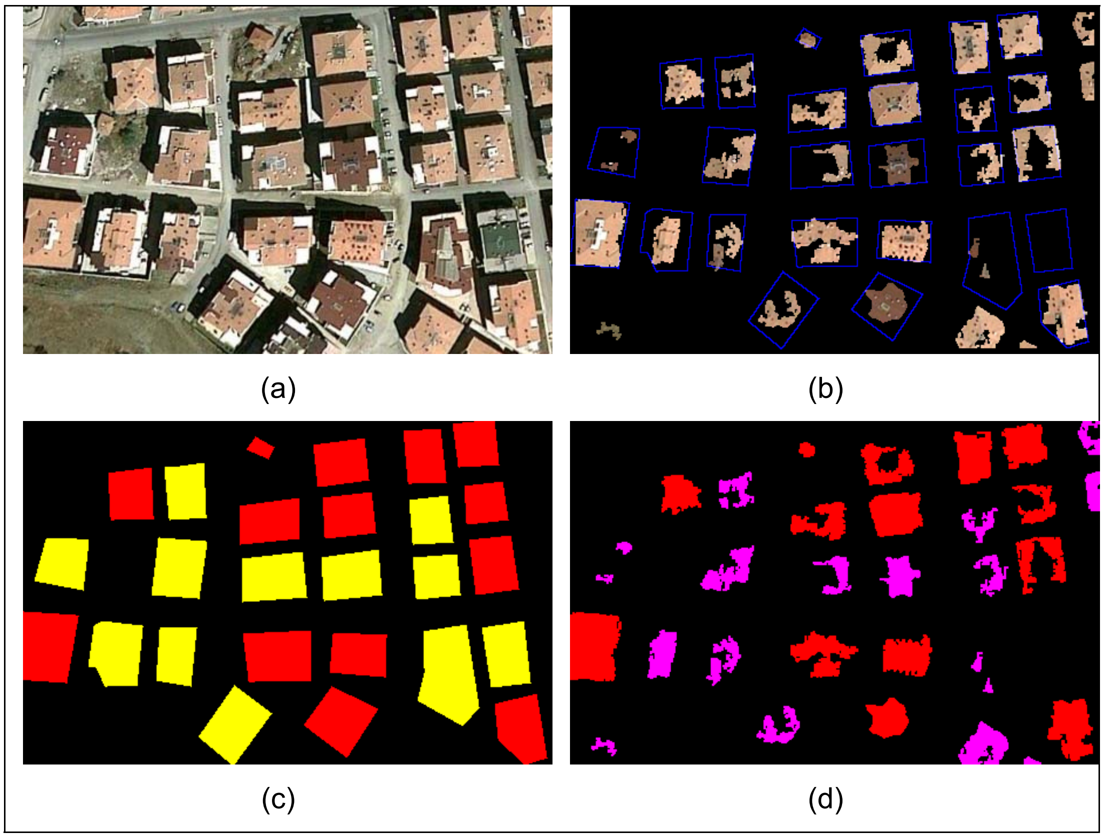


**Figure 4.72: Visual Results for SVM Quadratic Kernel with Adaboost Selected Features over All Features for 10. Image**

### 4.2.4 SVM quadratic kernel with HSV features

The pixel performance is given in Table 4.45.

**Table 4.57: Pixel Performance of the 1. Set**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 1 | 0.1672 | 1.9563 | 29.8573 | 28.1682 |
| 2 | 0.0758 | 0.7637 | 54.7562 | 52.404 |
| 3 | 0.1392 | 0.1766 | 82.9805 | 73.1633 |
| 4 | 0.2645 | 0.0563 | 92.8902 | 69.631 |
| 5 | 0.2595 | 3.603 | 17.0481 | 16.0869 |
| All | 0,18124 | 1,3112 | 55,5065 | 47,8907 |

Figure 4.73 illustrates the results of object performance vs. object detection threshold and Table 4.58 shows the results when object detection threshold is equal to 0.4.

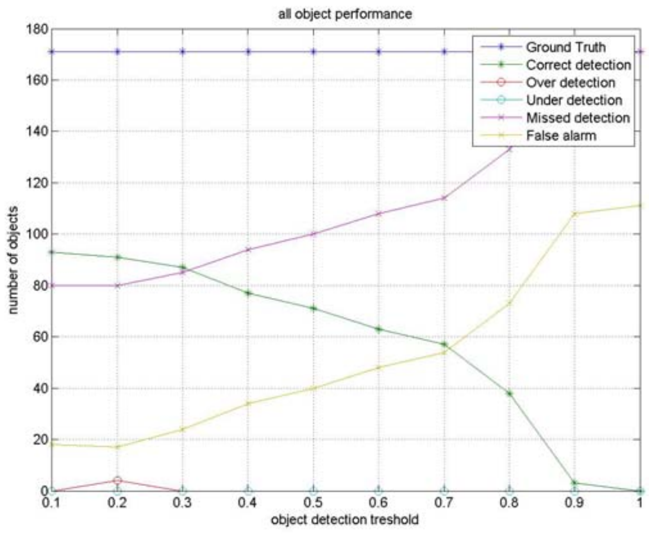


**Figure 4.73: Object Performance Vs. Object Detection Threshold**

**Table 4.58: Object Performance of the 1. Set for T = 0.4**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 1 | 35 | 17 | 0 | 0 | 18 | 7 |
| 2 | 13 | 8 | 0 | 0 | 5 | 3 |
| 3 | 24 | 18 | 0 | 0 | 6 | 9 |
| 4 | 29 | 26 | 0 | 0 | 3 | 9 |
| 5 | 70 | 8 | 0 | 0 | 62 | 6 |
| All | 171 | 77 | 0 | 0 | 94 | 34 |

The visual results for image 1 to 5 are given in
Figure 4.74 to
Figure 4.78. In these figures (a) is the original image, (b) is the original image masked by the building patches, together with the borders of the buildings in the

ground truth data (c) is ground truth, (d) is the building patches extracted by the algorithm.

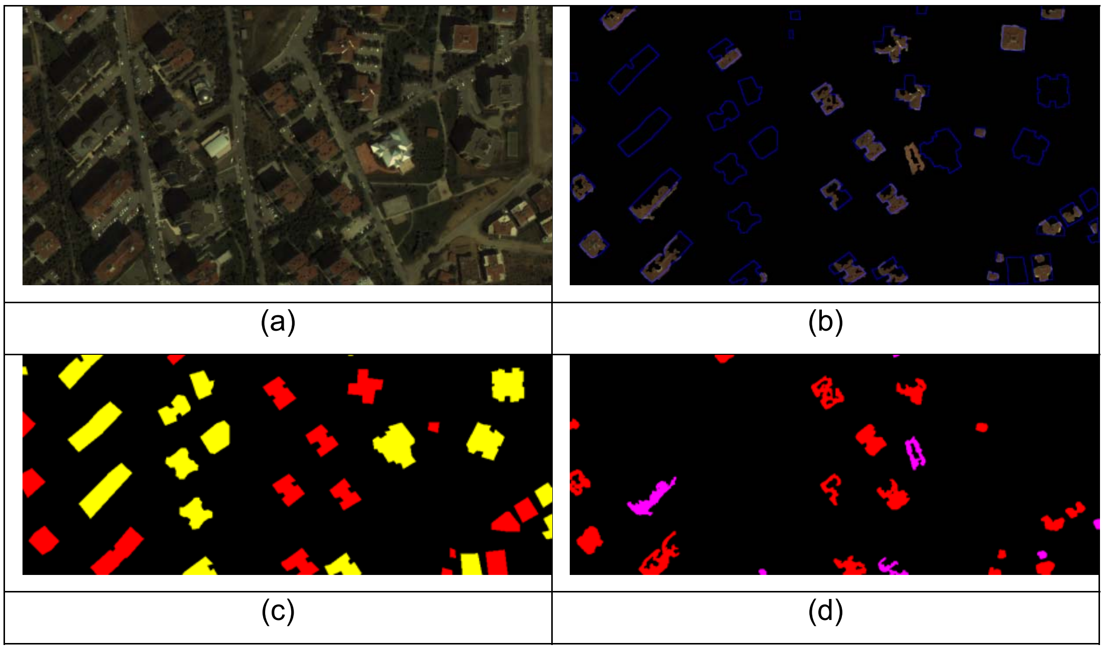


**Figure 4.74: Visual Results for SVM Quadratic Kernel with HSV Features for 1. Image**

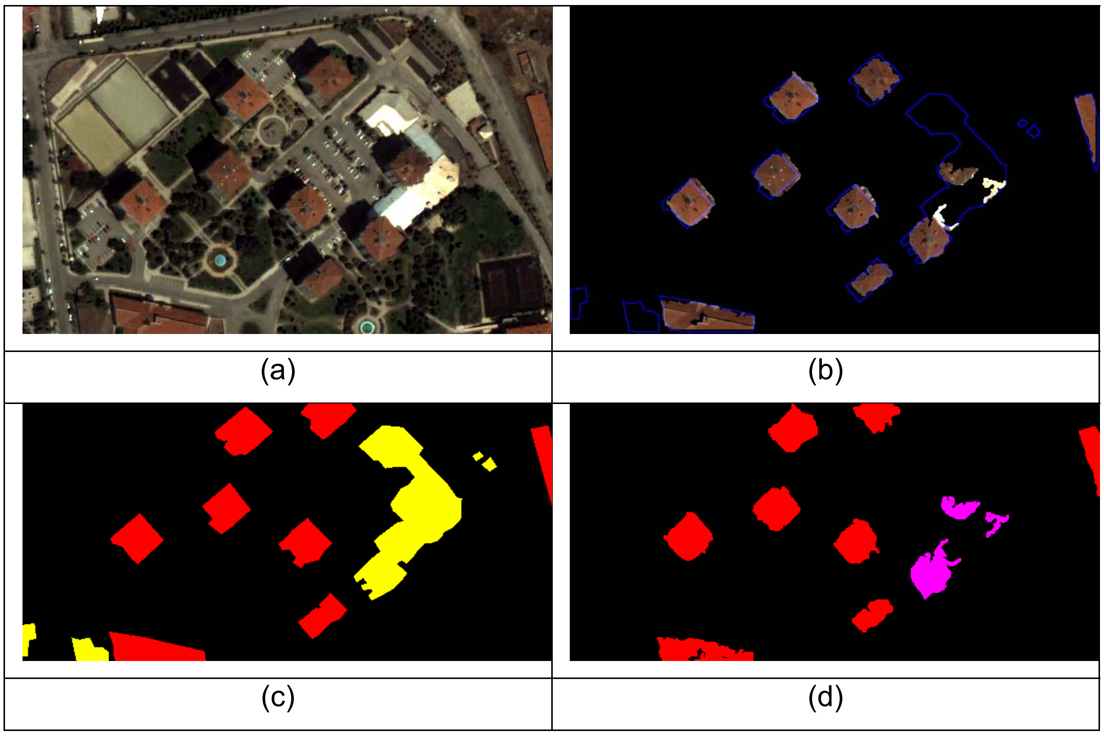
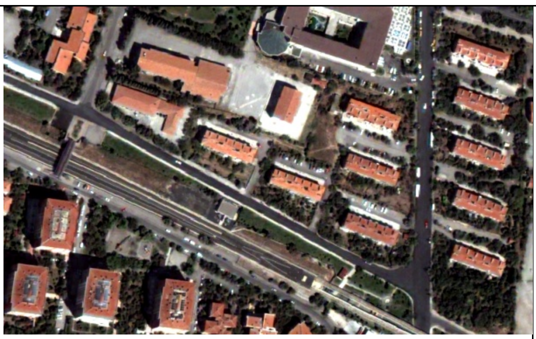
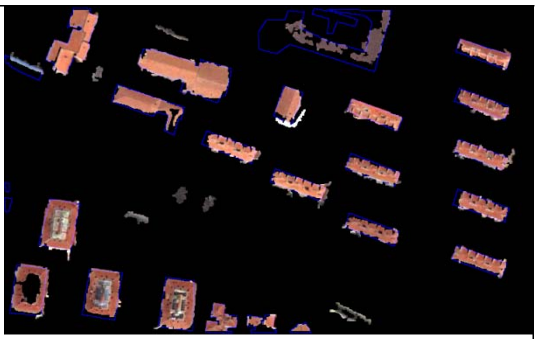
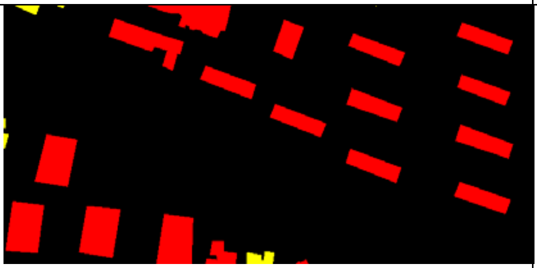


**Figure 4.75: Visual Results for SVM Quadratic Kernel with HSV Features for 2. Image**


(a)


(b)


(c)

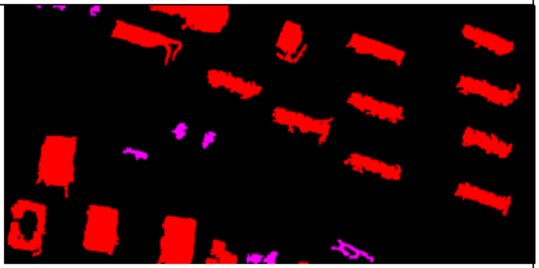

(d)

**Figure 4.76: Visual Results for SVM Quadratic Kernel with HSV Features for 3. Image**

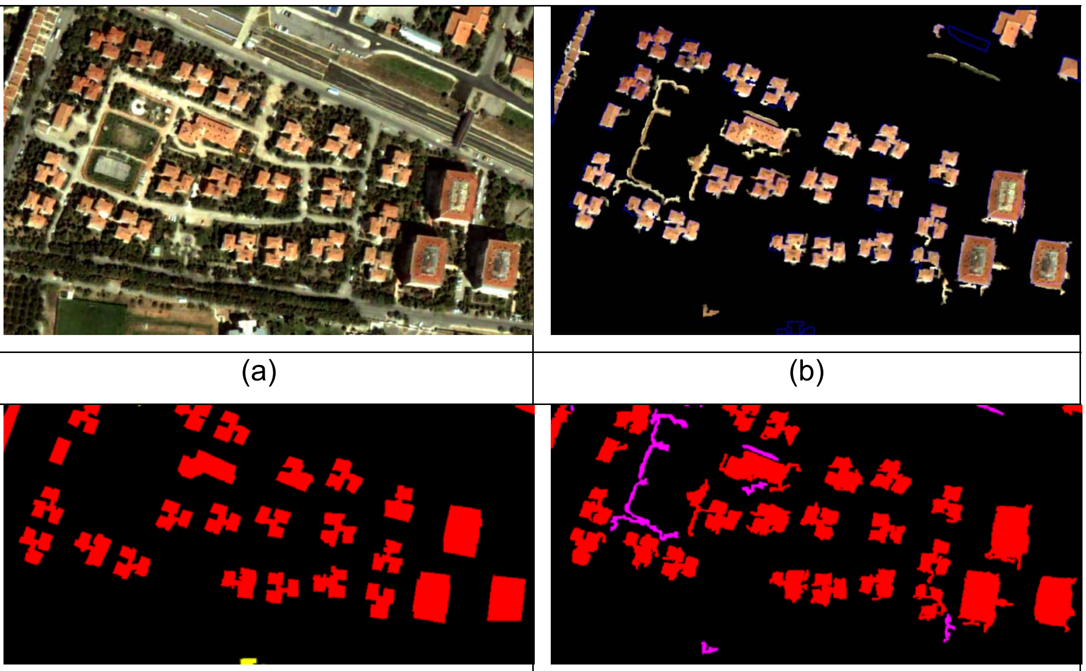

(a) (b) (c) (d)

**Figure 4.77: Visual Results for SVM Quadratic Kernel with HSV Features for 4. Image**

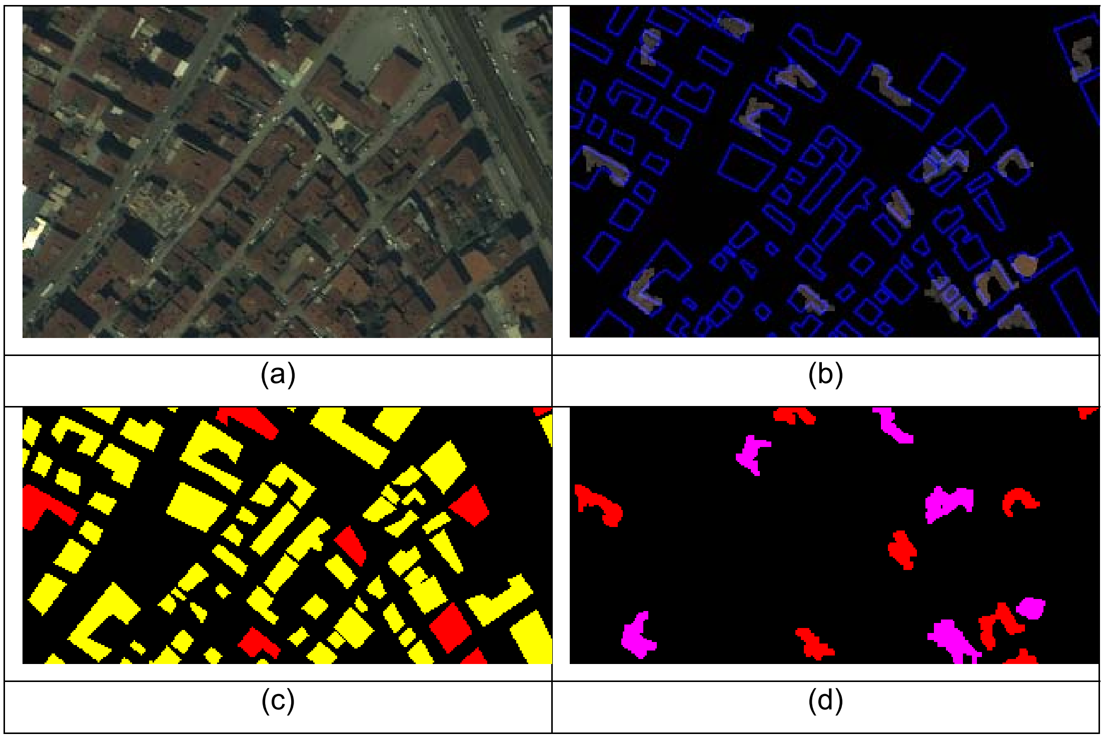


**Figure 4.78: Visual Results for SVM Quadratic Kernel with HSV Features for 5. Image**

The pixel performance of image set 2 is given in Table 4.59.

**Table 4.59: Pixel Performance of the Images in 2. Set**

| Image no | SF | MF | PBD | QP |
|---|---|---|---|---|
| 6 | 0.1352 | 0.9811 | 46.8509 | 43.654 |
| 7 | 0.1908 | 3.1069 | 20.6634 | 19.7034 |
| 8 | 0.7475 | 1.6729 | 13.1152 | 9.4474 |
| 9 | 0.1715 | 4.9918 | 14.234 | 13.8265 |
| 10 | 0.165 | 1.7219 | 32.6567 | 30.6771 |
| All | 0,282 | 2,4949 | 25,5040 | 23,4617 |

Figure 4.79 illustrates the results of object performance vs. object detection threshold for image set 2 and Table 4.60 shows the results when object detection threshold is equal to 0.4. The correct detection rate is highly low contrary to image set 1.

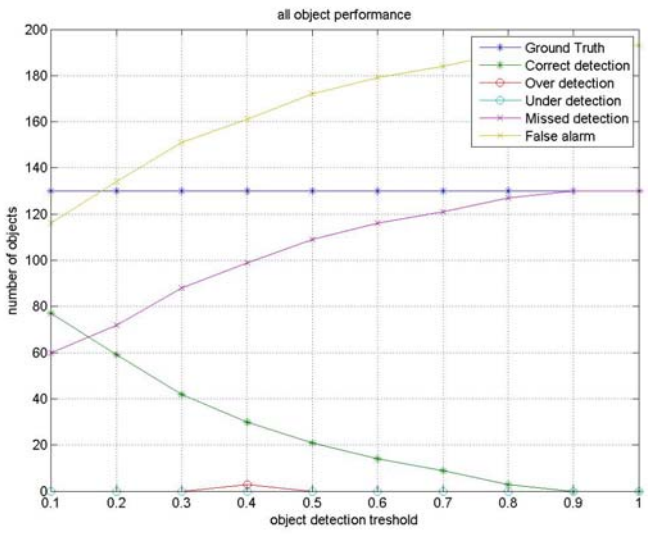


**Figure 4.79: Object Performance vs. Object Detection Threshold**

**Table 4.60: Object Performance of the 2. Set For T = 0.4**

| Image no | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| 6 | 21 | 16 | 0 | 0 | 5 | 22 |
| 7 | 50 | 5 | 3 | 0 | 44 | 17 |
| 8 | 22 | 0 | 0 | 0 | 22 | 72 |
| 9 | 11 | 0 | 0 | 0 | 11 | 10 |
| 10 | 26 | 9 | 0 | 0 | 17 | 40 |
| All | 130 | 30 | 3 | 0 | 99 | 161 |

The visual results for image 6 to 10 are given in

Figure 4.80 to

Figure 4.84. In these figures (a) is the original image, (b) is the original image masked by the building patches, together with the borders of the buildings in the ground truth data (c) is ground truth, (d) is the building patches extracted by the algorithm.

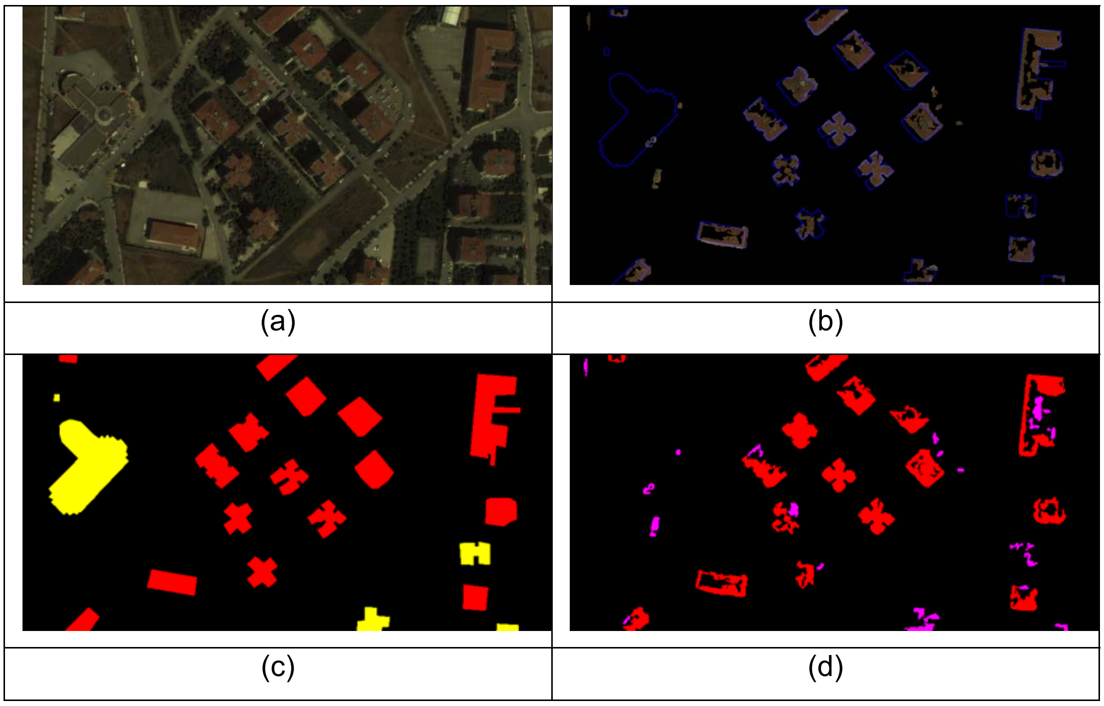


**Figure 4.80: Visual Results for SVM Quadratic Kernel with HSV Features for 6. Image**

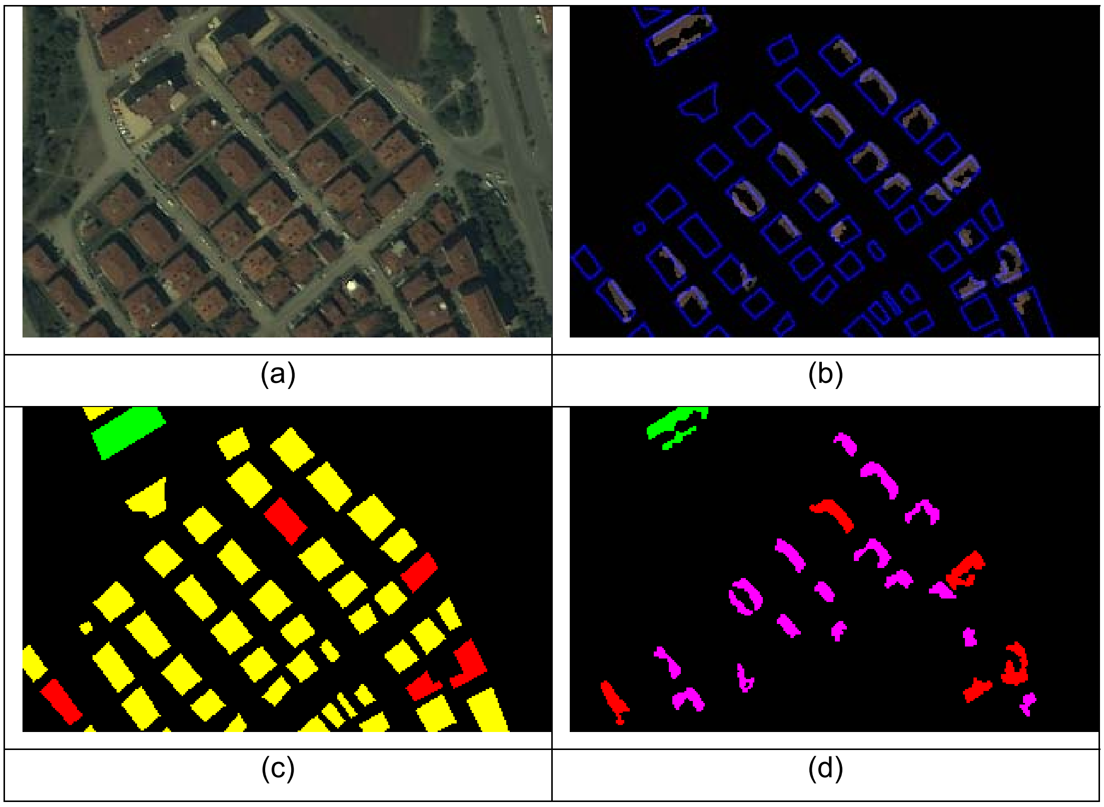
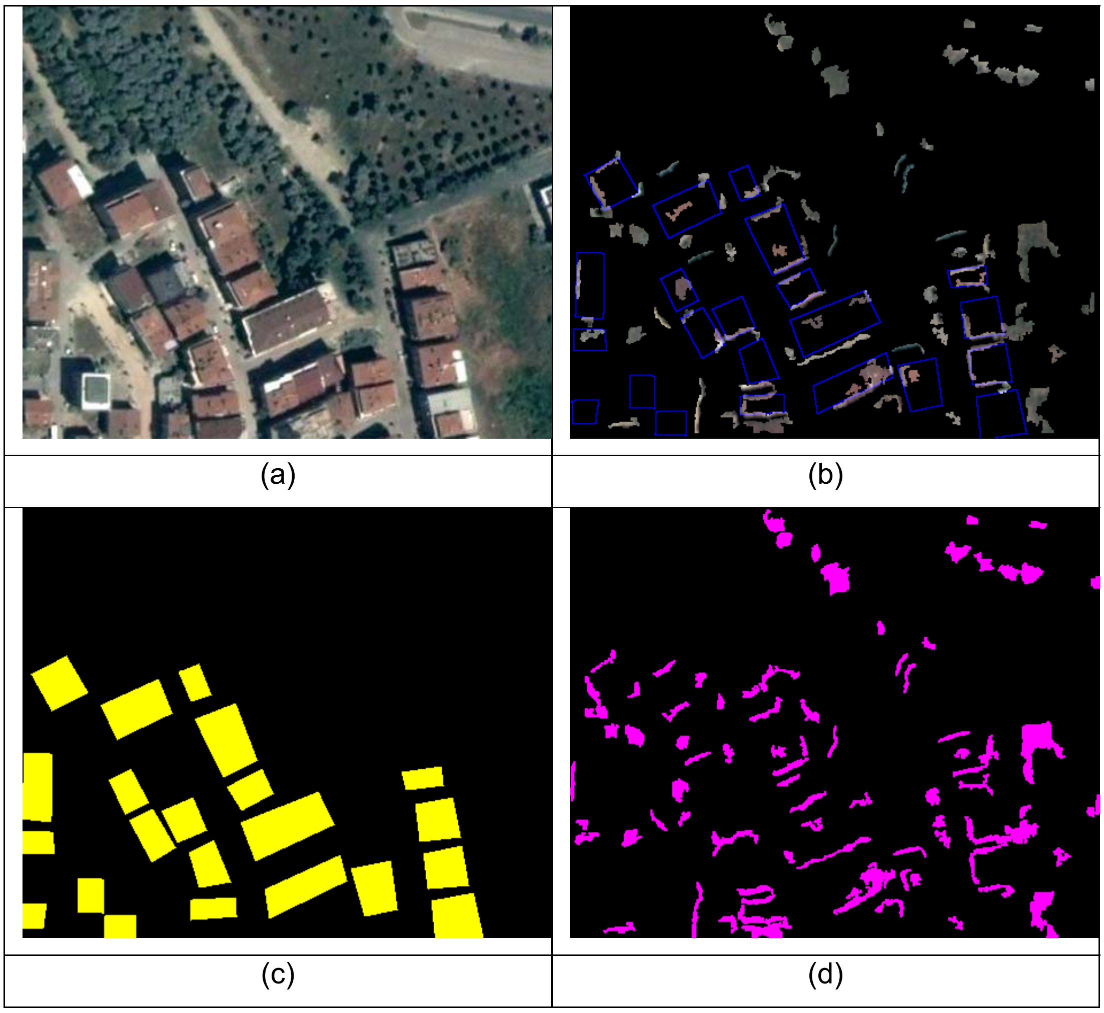
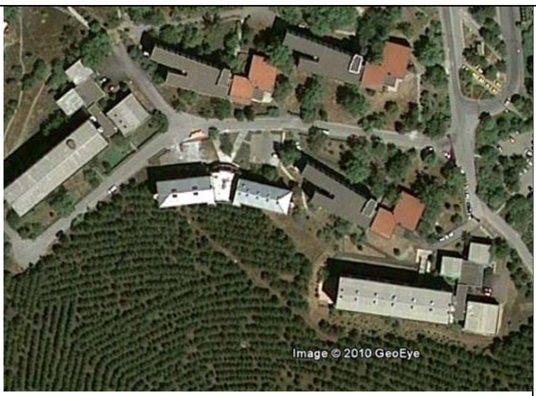


**Figure 4.81: Visual Results for SVM Quadratic Kernel with HSV Features for 7. Image**

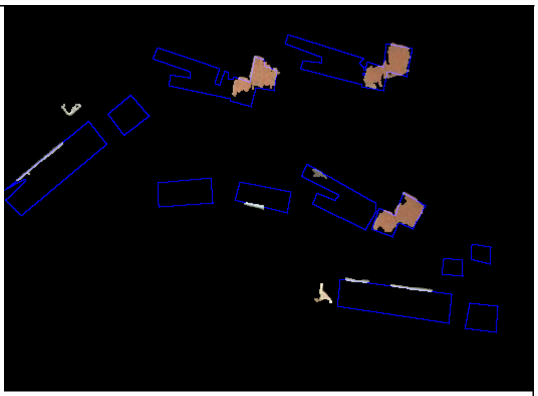


(a) (b) (c) (d)

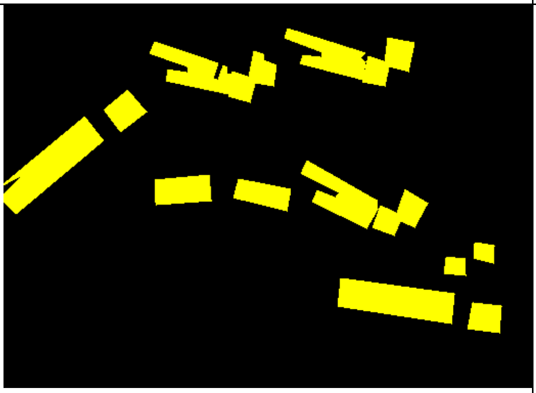
**Figure 4.82: Visual Results for SVM Quadratic with HSV Features for 8. Image**



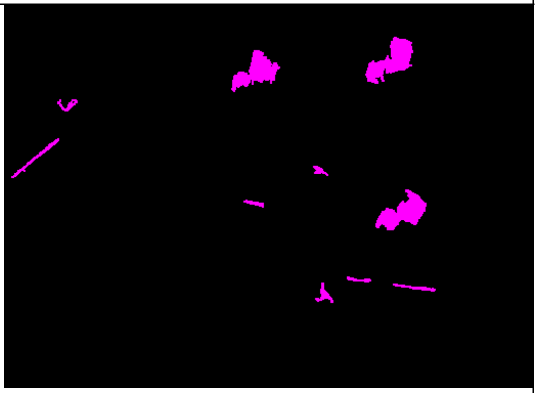
(a)



(b)



(c)



(d)

**Figure 4.83: Visual Results for SVM Quadratic with HSV Features for 9. Image**

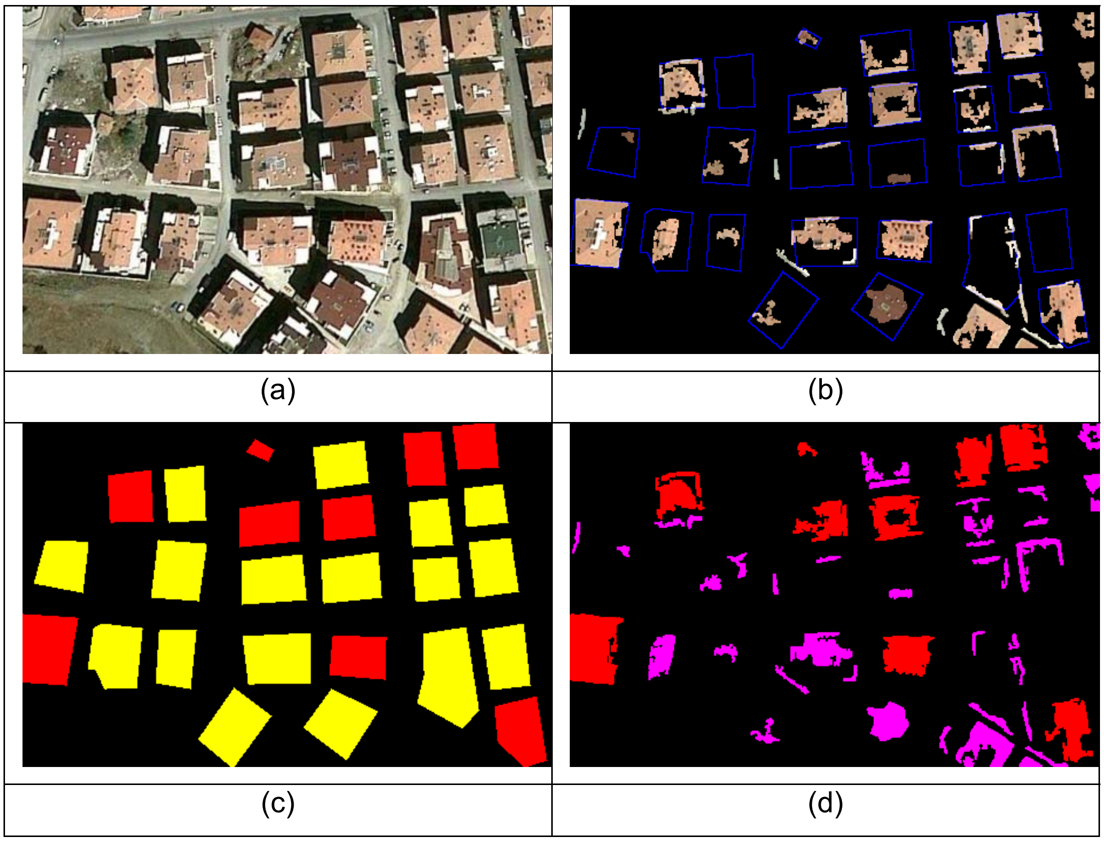


**Figure 4.84: Visual Results for SVM Quadratic Kernel with HSV Features for 10. Image**

## 4.3 Comparison of classifiers

Up to here, we have done several sensitivity analyses and by the help of these analyses, we have selected four classifiers. In addition, these classifiers are tested with a different image set then used in sensitivity analyses. We always used building detection performance as the criteria of selection. In this section, a final comparison of the classifiers will be done including pixel and object based performance criteria and time.

The pixel and object performances of image set 1 which upper %20 percent is used for training classifiers is given in Table 4.61 and Table 4.62. Note that performance evaluation is done on the lower %80 percent of the images. In this set, the best performance is achieved by SVM quadratic kernel with all features. Even SVM quadratic kernel with HSV features has better performance than both Adaboost classifiers. Adaboost with intensity textural features has low capacity to extract buildings.

**Table 4.61: Pixel Performances of the Selected Classifiers over Image Set 1**

| Classifier | SF | MF | PBD | QP |
|---|---|---|---|---|
| Adaboost with intensity textural features | 0.65194 | 0.2601 | 64.9902 | 27.3294 |
| Adaboost with all features | 0.28562 | 0.7155 | 65.0969 | 46.6771 |
| SVM quadratic kernel with features selected by Adaboost features | 0.1706 | 0.9784 | 57.1682 | 51.5707 |
| SVM quadratic kernel with HSV features selected by histogram errors | 0,18124 | 1,3112 | 55,5065 | 47,8907 |

**Table 4.62: Object Performances of the Selected Classifiers over Image Set 1 (numbers given in brackets are obtained by dividing by ground truth)**

| Classifier | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| Adaboost with intensity textural features | 171 | 57 (0.33) | 0 (0) | 7 (0.4) | 109 (0.64) | 159 (0.93) |
| Adaboost with all features | 171 | 93 (0.54) | 0 (0) | 0 (0) | 78 (0.46) | 68 (0.40) |
| SVM quadratic kernel with features selected by Adaboost | 171 | 79 (0.46) | 3 (0.2) | 5 (0.3) | 87 (0.51) | 29 (0.17) |
| SVM quadratic kernel with HSV features selected by histogram errors | 171 | 77 (0.45) | 0 (0) | 0 (0) | 94 (0.55) | 34 (0.20) |

The pixel and object based performances of the image set 2, which no parts of the images is used in training,  is given in Table 4.63 and Table 4.64. On the contrary to the 1. Image set best performances achieved while using Adaboost with all features. Adaboost with intensity textural features again has similar performance as in image set 1. However, the performance of SVM quadratic kernel with all features and with HSV features significantly decreased. The performance of SVM with HSV features even has worse results than Adaboost with intensity textural features.

**Table 4.63: Pixel Performances of the Selected Classifiers over Image Set 2**

| classifier | SF | MF | PBD | QP |
|---|---|---|---|---|
| Adaboost with intensity textural features | 0,7102 | 0,1277 | 75,2852 | 25,8688 |
| Adaboost with all features | 0.26404 | 0.9798 | 56.6075 | 43.1516 |
| SVM quadratic kernel with features selected by Adaboost features | 0,2034 | 1,4802 | 42,9201 | 38,1539 |
| SVM quadratic kernel with HSV features selected by histogram errors | 0,282 | 2,4949 | 25,5040 | 23,4617 |

**Table 4.64: Object Performances of the Selected Classifiers over Image Set 2 (numbers given in brackets are obtained by dividing by ground truth)**

| classifier | Ground Truth | Correct detection | Over detection | Under detection | Missed detection | False alarm |
|---|---|---|---|---|---|---|
| Adaboost with intensity textural features | 130 | 31 (0.24) | 0 (0) | 0 (0) | 99 (0.76) | 105 (0.81) |
| Adaboost with all features | 130 | 87 (0.67) | 3 (0.2) | 0 (0) | 42 (0.32) | 34 (0.26) |
| SVM quadratic kernel features selected by Adaboost | 130 | 62 (0.48) | 10 (0.8) | 0 (0) | 65 (0.50) | 98 (0.75) |
| SVM quadratic kernel with HSV features selected by histogram errors | 130 | 30 (0.23) | 3 (0.2) | 0 (0) | 99 (0.76) | 161 (1.24) |

The previous tables show us that, Adaboost is more robust than SVM kernels. Using only intensity textural features does not have good performance compared with using all features. HSV domain features work well only in images used in training and changing the image set causes significant decrease in the performance.

We will start time requirements first with the feature extraction. Feature extraction is the longest phase in the algorithm. For one image of size 500x500 pixels, it nearly takes 4 hours to extract all the features when using overlapping windows. Note that, there is no optimization in the source codes regarding time and the platform is Matlab 2007a which is nearly 10 times slower than C platforms. When using non-overlapping windows, this time is shortened proportional to the area of the window. If a fast algorithm is needed, non-overlapping windows can be used. However, performance will decrease. In training phase Adaboost can work with large sets and also Adaboost does the feature selection in training phase. After features are extracted, it takes nearly 1 hour for Adaboost to train 30 features that is the iteration count for Adaboost and one iteration takes nearly 2 minutes. There is no memory problem in training Adaboost if the training set is not larger than the memory capacity of PC. While training SVMs, the problem is not time but memory. SVM kernels can work on smaller training sets. The training of SVM kernels nearly takes 20 seconds after features are extracted. It takes nearly 5 seconds for Adaboost to classify one image and this time is nearly 2 seconds for SVM.

The classification times are very short compared with the feature extraction times. One way to decrease the time of the classifier is extracting only the features to be used and using large overlapping windows. Although feature selection increases the performance, using large overlapping windows decreases it as we have seen in sensitivity analyses.

# CHAPTER 5

# CONCLUSIONS

In this study, an automated building extraction system, which is capable of detecting buildings from a single satellite image using only RGB color band is implemented. The approach used in this work has four main steps: local feature extraction, feature selection, classification and building boundary extraction. First local features are extracted using a predefined window. Using these local features each pixel when the window is overlapping and all pixels in the window when the window is non-overlapping is classified as building or non-building. Finally, the image pixels are grouped that belong to building class using some morphological operations and resulting building boundaries are determined.

Several textural features are used extracted from both intensity and HSV domain. The separation capacities of the features are investigated. Different window types namely overlapping and non-overlapping are examined as well as different window sizes. Feature selection schemes based on Adaboost and histogram errors are considered. Four SVM kernels and Adaboost are used for classification with different window types and sizes, different class of features, different feature counts and different training sets. A detailed pixel and object based performance evaluation is done. Beside numerical results, visual results are also presented.

We see that non-overlapping windows have disadvantage since they reduce the classification resolution. On the contrary, overlapping windows give better results but their time of extraction is long. While using non-overlapping windows, the performance has a tendency to decrease since the classification resolution decreases. On the other hand, the performance of overlapping windows first increases since this enables extraction of textural features more precisely, and then decreases since both building and non-building areas begin to be covered in the window.

Using only intensity textural features for classification is not enough to form a building detector since other manmade object like roads are similar to buildings in means of texture. SVM kernels are not able to form an acceptable classifier using only intensity textural features but Adaboost formed a reasonable classifier. SVM kernels need features with more separation capacities. The classifier formed by using only HSV domain features have good results when trained with parts of the images but the performance significantly decreased when a new set of images introduced. As expected, using both intensity textural and HSV domain features together gave the best results. Also they are more robust to different images than using only HSV domain features.

Investigation of optimal parameters for four SVM kernels namely linear, polynomial, RBF, and quadratic and also for Adaboost is done in sensitivity analyses. Among SVM kernels, the best performance is achieved by quadratic SVM kernel.

Adaboost can form its own features over a large feature set but SVM uses predefined features. We see that selecting features for SVM rather than giving all the features supplied better performances. Both histogram and Adaboost based feature selection methods significantly increase the performance. SVM quadratic kernel operated on Adaboost selected features have higher performance (QP=51.6%) than SVM quadratic kernel operated on features considering histogram errors (QP=43.2), since Adaboost selects features by taking into account the samples incorrectly classified by the previous weak classifiers formed by using selected features. Without any feature selection, best QP=26.7 obtained for SVM by quadratic kernel.

The performance of SVM quadratic kernel with Adaboost selected features has nearly five percent better performance in terms of pixel based performance (QP=51.6%) than Adaboost (QP=46.7%) when a part of the images is used for training. In terms of object based performances regarding correct detection ratios (CDR) and false alarm ratios (FAR), SVM quadratic kernel with Adaboost selected features performed CDR=0.46 for T=0.4, CDR=0.36 for T=0.6, and FAR=0.17 for T=0.4 and FAR=0.28 for T=0.6. The same ratios for Adaboost are CDR=0.54 for

T=0.4, CDR=0.44 for T=0.6 and FAR=0.40 for T=0.4 and FAR=0.51 for T=0.6 where T is building detection threshold. This shows although Adaboost is better in detecting buildings than SVM, worse in generating false alarms.

However, if new images that had not been used in training were introduced, the performance regarding QP of Adaboost (QP=43.2%) was five percent better than SVM quadratic kernel with Adaboost selected features (QP=38.2%). In terms of object based performances regarding CDR and FAR, SVM quadratic kernel with Adaboost selected features performed CDR=0.48 for T=0.4, CDR=0.35 for T=0.6, and FAR=0.75 for T=0.4 and FAR=0.95 for T=0.6. The same ratios for Adaboost are CDR=0.67 for T=0.4, CDR=0.59 for T=0.6 and FAR=0.26 for T=0.4 and FAR=0.35 for T=0.6. In this case, Adaboost over performed the SVM quadratic kernel with Adaboost selected features both in terms of pixel based and the object based performance criteria.

As a summary, in this study the best performances achieved using Adaboost with all features. The correct detection rate is 87 over 130 (CDR=0.67) buildings when building detection threshold is 0.4 and 77 over 130 (CDR=0.59) when building detection threshold is 0.6. The false alarm rate is 34 over 130 (FAR=0.26) buildings when building detection threshold is 0.4 and 46 over 130 (FAR=0.35) when building detection threshold is 0.6. In Pattern Recognition in Remote Sensing (PRRS) competition held in 2008, the best result presented was about 850 of 3500 for correct detection (CDR=0.24) of buildings with threshold T=0.55 [63]. (The data used is copy protected so we could not be able to run our algorithm on that data). Although the data used are different, the performance criteria used are the same. This might give an idea. The results we obtained are quite promising when compared with the results in [63],

Future work involves continued evaluation of the proposed algorithm on images of different types. In this work we deal with visible band of the satellite images but the algorithms can be easily modified to work only on panchromatic band. Aerial images can also be used as input images. In addition to that, the proposed method can be extended to use all the spectral bands of the satellite image. As we mentioned before, vegetation and shadow information can be extracted by using Normalized Difference Vegetation Index (NDVI) and ratio of hue to intensity in YIQ model, respectively. This additional information will improve the building detection performance of the system.

The proposed feature selection and classification system can also be applied for objects of different types. Instead of buildings, the system can be modified to deal with road or some other special structure/area detection problems in satellite images. Some additional features will be integrated to the system to discriminate the selected object type from its surroundings.

Another important future work will be improving the performance of the further processing phase of the proposed algorithm. The algorithm proposed here can be used for hypothesis generation that is in finding candidate regions for buildings. After determining the building regions some additional processes like boundary removal and region merging can be applied together as a hypothesis verification step further eliminate false alarms. Advanced boundary extraction algorithms can be also applied to get more reliable building boundaries.

# REFERENCES

[1] Cariou, C. and Chehdi, K., “Unsupervised texture segmentation/ classification using 2-D autoregressive modeling and the stochastic expectation-maximization algorithm”, *Pattern Recognition Letters*, v.29, pp. 905–917, 2008.

[2] Mayer, H., “Automatic object extraction from aerial imagery-A survey focusing on buildings”, *Comput. Vis. Image Understand*. v.74, pp.138–149, 1999.

[3] Konecny, G. and Schiewe, J., “Mapping from digital satellite image data with special reference to MOMS-02”, ISPRS J. *Photogrammetry Remote Sens*. v. 51, pp. 173–181, 1996.

[4] Ayhan,E., Erden, Ö., and Görmüş, E. T. , “Three dimensional monitoring of urban development by means of ortho-rectified aerial photographs and high-resolution satellite images”, *Environmental Monitoring and Assessment*, v.147, pp. 413-421, 2008.

[5] Huertas, A., and Nevatia, R., “Detecting Buildings in Aerial Images”, *Computer Vision, Graphics, And Image Processing*, v. 41,pp. 131-152, 1988.

[6] Shufelt, J., and McKeown, D. M., “Fusion of monocular cues to detect man made structures in aerial imagery", *CVGIP: Image Understand*, v. 57 pp. 307–330, 1993.

[7] Noronha, S., and Nevatia, R., “Detection and description of buildings from multiple aerial images," *RADIUS: Image Understanding for Imagery Intelligence*, eds. Firschein, O. and Strat, T. M., Morgan Kaufmann Publishers, , pp. 171-183 1997.

[8] Krishnamachari, S. and Chellappa, R. “Delineating buildings by grouping lines with MRFs*”, IEEE Trans*. IP 5 (1) 164–168, 1996.

[9] Guo, T., and Yasuoka, Y. "Snake-based approach for building extraction from high-resolution satellite images and height data in urban areas", *Proceedings of 23rd Asian Conference on Remote Sensing*, 2002.

[10] Rüther, H., Martine, H. M., and Mtalo, E. G. “Application of snakes and dynamic programming optimization technique in modeling of buildings in informal 115 settlement areas” *ISPRS Journal of Photogrammetry & Remote Sensing*, v. 56, pp. 269-282, 2002.

[11] Peng, J., Zhang, D., and Liu, Y., “An improved snake model for building detection from urban aerial images” *Pattern Recognition Letters* v. 26, pp. 587-595, 2005.

[12] Baltsavias, E., Mason, S., and Stallmann, D., “Use of DTMs DSMs and orthoimages to support building extraction.” In: Gruen, A., Kubler, O., and Agouris, P. (Eds.), “*Automatic Extraction of Man-Made Objects from Aerial and Space Images*.” Birkhauser, Basel, pp. 199–210, 1995.

[13] Brunn, A., and Weidner, U., “Hierarchical Bayesian nets for building extraction using dense digital surface models”, *J. Photogrammetry Remote Sens.* v. 53, pp. 296–307, 1998.

[14] Shan, J. and Lee, D. S. , “Generalization Of Building Polygons Extracted From IKONOS Imagery “*Symposium on Geospatial Theory, Processing and Applications*, Ottawa, 2002.

[15] Ünsalan, C. and Boyer, K. L., “A system to detect houses and residential street networks in multispectral satellite images”, *Computer Vision and Image Understanding*, v. 98, pp. 423–461, 2005.

[16] Zhang, Y., “Optimisation of building detection in satellite images by combining multispectral classification and texture filtering”, *J. Photogrammetry Remote Sens*. v. 54, pp. 50–60, 1999.

[17] Stassopoulou, A. and Caelli, T., “Building detection using Bayesian networks”, Inter. J. Pattern Recognit. Artif. Intell. v.14, pp.715–733, 2000.

[18] Kim, K., Yang, Y., Park, Y., and Kim, T., “Shape discrimination by descriptors and moments using neural network”, *International Archives of Photogrammetry and Remote Sensing*. Vol. XXXI, Part B3. Vienna 1996.

[19] Gilmore, J. F., Boyd, W. W., “Building and bridge classification by invariant moments.” *Proc SPIE*., v. 292, pp.256–263, 1981.

[20] Levitt, S., and Aghdasi, F., “An investigation into the use of wavelets and scaling for the extraction of buildings in aerial images.” *Proceedings of the 1998 South African Symposium on Communications and Signal Processing*, COMSIG ’98, September 7–8, 133–138, 1998.

[21] Qu, Y., Li, C., Zheng, N., Yuan, Z., and Ye, C., “Salient Building Detection from a Single Nature Image via Wavelet Decomposition.”, *Applied and Numerical Harmonic Analysis, Wavelet Analysis and Applications* pp. 397-405, Springer, 2007.

[22] Bellman, C. J. and Shortis, M. R., “Building recognition using wavelet analysis and support vector machines,” in *Proceedings of SSC2003*, 2003.

[23] Selvarajan, S., and Tat, C., “Extraction of man-made features from remote sensing imageries by data fusion techniques,” in *The 22nd Asian Conference on Remote Sensing, Singapore*, 2001.

[24] Beril Sırmaçek and Cem Ünsalan “Building Detection Using Local Gabor Features in Very High Resolution Satellite Images “ ,2008.

[25] Idrissa, M., Lacroix, V., Hincq, A., Bruynseels, H., and Swartenbroekx, O., “SPOT5 images for urbanization detection,” in *Proceedings of Advanced Concepts for Intelligent Vision Systems*, 2004.

[26] Haralick, R. M., Shanmugam, K. and Dinstein, I. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics,* 3(6), pp.610-611, 1973.

[27] Zimmermann, P., “A New Framework for Building Detection Analysing Multiple Cue Data”. *International Archives of Photogrammetry and Remote Sensing*, Vol. XXXIII, Part B3: pp.1063-1070, 2000.

[28] Maas, H. G., “The potential of height texture measures for the segmentation of airborne laser scanner data.” In: Proc. *Fourth International Airborne Remote Sensing Conference*, Ottawa, Ontario, Canada, 21_24 June 1999, pp. 154_161. 1999.

[29] Khoshelham, K., Nardinocchi, C., Frontoni, E., Mancini, A. and Zingaretti, P., Performance evaluation of automated approaches to building detection in multi-source aerial data, *ISPRS Journal of Photogrammetry and Remote Sensing*, Volume 65, Issue 1, , pp. 123-133. January 2010

[30] Brunn, A., “Statistical interpretation of DEM and image data for building extraction.” In: Gruen, A., Baltsavias, E., Van Gool, L. (Eds.), *Automatic Extraction of Man-made Objects from Aerial and Space Images*. A.A. Balkema, Lisse, The Netherlands, pp. 171_180, 2001.

[31] Bartels, M., Wei, H., “Maximum likelihood classification of LIDAR data incorporating multiple co-registered bands.” *In: 4th International Workshop on Pattern Recognition in Remote Sensing in Conjunction with the 18th International Conference on Pattern Recognition*, Hong Kong, 20 August 2006, pp. 17_20. 2006.

[32] Khoshelham, K., Li, Z.L., King, B., A split-and-merge technique for automated reconstruction of roof planes. *Photogrammetric Engineering and Remote Sensing* 71 (7), 855_862. 2005.

[33] Walter, V., “Object-based classification of remote sensing data for change detection.” *ISPRS Journal of Photogrammetry and Remote Sensing* 58, pp. 225_238, 2004.

[34] Rottensteiner, F., Trinder, J., Clode, S., Kubik, K., Lovell, B., “Building detection by Dempster_Shafer fusion of LIDAR data and multispectral aerial imagery.” *In: Proc. 17th International Conference on Pattern Recognition*, ICPR'04, Cambridge, United Kingdom, 23_26 August 2004, pp. 339_342. 2004.

[35] Lu, Y.H., Trinder, J.C., Kubik, K., Automatic building detection using the Dempster_Shafer algorithm. *Photogrammetric Engineering and Remote Sensing* 72 (4), 395_403, 2006.

[36] Shafer, G., A Mathematical Theory of Evidence. Princeton *University Press*, Princeton. 1976.

[37] Elaksher, A.F., and Bethel, J.S., “Automatic Generation of High Quality 3D Urban Buildings from Aerial Images”, 2007.

[38] Barsi, A., “Object detection using neural self-organisation”. *In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Istanbul, Turkey, Vol. XXXV, Part B3, pp. 366-371. 2004.

[39] Lari, Z. and Ebadi, H., “Automatic extraction of building features from high resolution satellite images using artificial neural networks”, *ISPRS*, 2007.

[40] Persson, M., Sandvall, M., and Duckett, T., “Automatic Building Detection from Aerial Images for Mobile Robot Mapping”, 2008.

[41] Lee, S., and Lathrop, R.G., “Subpixel analysis of Landsat ETM+ using self-organizing map (SOM) neural networks for urban land cover characterization,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 6, pp. 1642–1654, 2006.

[42] Bellman, C.J., Shortis, M.R., “A Machine Learning Approach To Building Recognition In Aerial Photographs”, 2003. .

[43] Bruzzone, L., and Carlin, L., “A multilevel context-based system for classification of very high spatial resolution images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 9, pp. 2587–2600, 2006.

[44] Joachims, T., Making Large-Scale SVM Learning Practical. In: Advances in Kernel Methods - *Support Vector Learning*. Eds. B. Scholkopf, Burges, C.J. & Smola, A,J. Cambridge, USA, MIT Press, 1998.

[45] Brunner, G., and Burkhardt, H., Building classification of terrestrial images by generic geometric hierarchial cluster analysis features, *IAPR Workshop on Machine Vision Applications*, 2005

[46] Zingaretti, P., Frontoni, E., Forlani, G., Nardinocchi, C.,. Automatic extraction of LIDAR data classification rules. *In: Proc. 14th International Conference on Image Analysis and Processing*, ICIAP, Modena, 10_13 September. IEEE Computer Society, pp. 273_278, 2007.

[47] Maloof, M. A., Langley, P., Sage, S., and Binford, T., “Learning to detect rooftops in aerial images”, in: Proc. 1997 *Image Understanding Workshop* DARPA97, pp. 835–845, 1997.

[48] M. Persson, M. Sandvall, and T. Duckett. Automatic building detection from aerial images for mobile robot mapping. *In Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings*. 2005 IEEE International Symposium on, pages 273--278, 2005.

[49] Ioannidis, C., Psaltis, C., and Potsiou, C., “Towards a strategy for suburban informal building control through automatic change detection”, *Computers, Environment and Urban Systems* 33 (1), pp. 64–74, 2009.

[50] Kontitsis, M., Valavanis, K. P., and Tsoweloudis, N., “A UAV Vision System for Airborne Surveillance”, Procedings of the 2004 *IEEE Internetional Conference on Robotics & Automation*, New Orleans. LA April 2004.

[51] Fitzgerald, D.L., Mejias, L., Eng, P., and Liu, X., “Towards Flight Trials for an Autonomous UAV Emergency Landing Using Machine Vision”. *Australasian Conference on Robotics and Automation*, 2007.

[52] Burges, C., “A tutorial on support vector machines for pattern recognition,” *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[53] Khotanzad, A. and Hong, Y.H. Invariant image recognition by Zernike moments. Pattern Analysis and Machine Intelligence, *IEEE Transactions* on, v. 12:489-497, 1990.

[54] Ravichandran, G. and Trivedi, M. M., “Circular-Mellin Features For Texture Segmentation” *IEEE Transactions On Image Processing*, v. 4,no. 12, pp:1629-1640, 1995

[55] Augusteijn, M.F. and Clemens, L.E. and Shaw, K.A. Performance evaluation of texture measures for ground cover identification in satellite images by means of a neural network classifier. *Geoscience and Remote Sensing, IEEE Transactions on*, v.33, pp:616-626, 1995.

[56] Newsam, S. D. and Kamath, C. Comparing shape and texture features for pattern recognition in simulation data. In Dougherty, E. R. and Astola, J. T. and Egiazarian, K. O., editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, pp:106-117, 2005.

[57] Weszka, J. S. and Rosenfeld, A. A comparative study of texture measures for terrain classification. *NASA STI/Recon Technical Report N*, 76, 1975.

[58] Manjunath, B.S. and Ma, W.Y. Texture features for browsing and retrieval of image data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(8):837-842, 1996.

[59] Rangayyan, R.M. and Ferrari, R.J. and Desautels, J.E.L. and Frere, A.F. Directional analysis of images with Gabor wavelets. *Computer Graphics and Image Processing, 2000. Proceedings XIII Brazilian Symposium on*, pages 170-177, 2000.

[60] Marsland, S., *Machine Learning: An Algorithm Perspective.* CRC Press, 2008.

[61] Shufelt, A. A., and Mckeown, D.M., “Fusion of Monocular Cues to Detect Man-Made Structures in Aerial Imagery”, 1992.

[62] Beauchemin, M., and Thomson, K. P. B. “The evaluation of segmentation results and the overlapping area matrix.” *Int. Journal of Remote Sensing*, 18:3895–3899, December 1997.

[63] Aksoy, S. et al. “Performance Evaluation of Building Detection and Digital Surface Model Extraction Algorithms: Outcomes of the PRRS 2008 Algorithm Performance Contest”, *Pattern Recognition in Remote Sensing Proc of IAPR Workshop*, 1-12, 2008

[64] Newsam, S. D. and Kamath, C., “Retrieval using texture features in high resolution multi-spectral satellite imagery.” *In SPIE Defense and Security Symposium, Data Mining and Knowledge Discovery: Theory, Tools, and Technology* VI, 2004

# APPENDIX A

# EXAMPLE FEATURE RESULTS

**Table A.1: Basic Features**

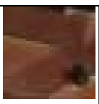
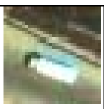
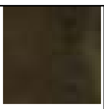
| Basic Features | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Feature Nunber | Feature Name | Building |  Building | Building | Non-building |  Non-building |  Non-building | Normali-zed histogram Error |
| 1 | Mean | 0.7498 | 0.397 | 0.2088 | 0 | 1 | 0.1942 | 0.3237 |
| 2 | Variance | 0.539 | 0.0973 | 0.0495 | 0.0882 | 1 | 0 | 0.4619 |
| 3 | Skewness | 0 | 0.2519 | 0.4697 | 0.3238 | 0.3766 | 1 | 0.398 |
| 4 | Kurtosis | 0.1853 | 0.2409 | 1 | 0 | 0.0217 | 0.505 | 0.4248 |
| 5 | Entropy | 0.9803 | 0.58 | 0.2371 | 0.5588 | 1 | 0 | 0.4428 |
| 6 | Energy | 0.6056 | 0.2047 | 0.0782 | 0 | 1 | 0.0634 | 0.3374 |
| 7 | Mean of gradient | 0.932 | 0.3006 | 0.132 | 0.3036 | 1 | 0 | 0.4515 |
| 8 | Variance of gradient | 0.165 | 0 | 0.2927 | 0.0891 | 0.124 | 1 | 0.4483 |
| 9 | Variogram with 1 distance | 0.8571 | 0.1665 | 0.0564 | 0.1124 | 1 | 0 | 0.4456 |
| 10 | Variogram with 2 distance | 0.9415 | 0.1749 | 0.0733 | 0.1361 | 1 | 0 | 0.4562 |
| 11 | Variogram with 3 distance | 0.7962 | 0.1492 | 0.0591 | 0.1256 | 1 | 0 | 0.4583 |

**Table A.2: Zernike Moment**

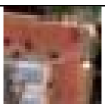
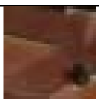
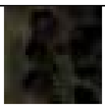
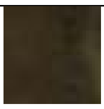
| Zernike Moments | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Featur e Nunbe r | Featur e Name |  Buildin g |  Buildin g | Buildin g |  Non-building | Non-building |  Non-building | Normali-zed histogra m Error |
| 12 | Zernik e (0,0) | 0.7498 | 0.397 | 0.2088 | 0 | 1 | 0.1942 | 0.3237 |
| 13 | Zernik e (1,1) | 0.5729 | 0.2923 | 0.1382 | 0.3063 | 1 | 0 | 0.4637 |
| 14 | Zernik e (2,0) | 0.3465 | 0.3253 | 0.128 | 0 | 1 | 0.1396 | 0.3328 |
| 15 | Zernik e (2,2) | 0.5264 | 0.0831 | 0 | 0.1015 | 1 | 0.1146 | 0.4668 |
| 16 | Zernik e (3,1) | 0.481 | 0.688 | 0 | 1 | 0.924 | 0.1723 | 0.4673 |
| 17 | Zernik e (3,3) | 1 | 0.9995 | 0 | 0.3307 | 0.425 | 0.2377 | 0.4669 |
| 18 | Zernik e (4,0) | 0.4722 | 0.4854 | 0.1383 | 0 | 1 | 0.1691 | 0.3399 |
| 19 | Zernik e (4,2) | 0.5546 | 0.1294 | 0.5384 | 0.1238 | 1 | 0 | 0.4748 |
| 20 | Zernik e (4,4) | 0.8514 | 0.1756 | 0.0879 | 0.0384 | 1 | 0 | 0.3968 |

**Table A.3: Circular Mellin Features**

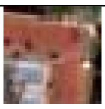
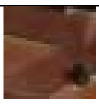
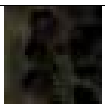
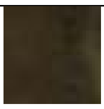
| Circular Mellin Features | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Featur e Nunbe r | Featur e Name |  Buildin g |  Buildin g | Buildin g |  Non-building | Non-building |  Non-building | Normali-zed histogra m Error |
| 21 | C(1,1) | 0.8172 | 0.0565 | 0.436 | 0.3696 | 1 | 0 | 0.4658 |
| 22 | C(1,2) | 1 | 0.1809 | 0.3519 | 0 | 0.934 | 0.2193 | 0.4713 |
| 23 | C(1,3) | 1 | 0 | 0.3581 | 0.4988 | 0.8763 | 0.1323 | 0.4606 |
| 24 | C(1,4) | 0.5468 | 0.3379 | 0.1462 | 0 | 1 | 0.1363 | 0.357 |
| 25 | C(1,5) | 0.9025 | 0.1534 | 0.389 | 0.102 | 1 | 0 | 0.4657 |
| 26 | C(2,1) | 1 | 0 | 0.4058 | 0.3759 | 0.7375 | 0.0648 | 0.4679 |
| 27 | C(2,2) | 0.1913 | 0.2268 | 0.4268 | 0 | 1 | 0.1606 | 0.4626 |
| 28 | C(2,3) | 0.7066 | 0.0995 | 0.241 | 0.2731 | 1 | 0 | 0.4622 |
| 29 | C(2,4) | 0.1972 | 0.4163 | 0.2087 | 0 | 1 | 0.1785 | 0.3375 |
| 30 | C(2,5) | 1 | 0.209 | 0.2379 | 0.113 | 0.6544 | 0 | 0.4661 |

**Table A.4: Fourier Power Spectrum Features**

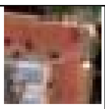
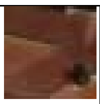
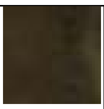
| Fourier Power Spectrum Features | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Featur e Nunbe r | Feature Name |  Buildin g |  Buildin g | Buildin g | Non-buildin g | Non-buildin g |  Non-buildin g | Normali-zed histogra m Error |
| 31 | Ring1 on PS | 0.5903 | 0.2092 | 0.0829 | 0 | 1 | 0.0709 | 0.3316 |
| 32 | Ring2 on PS | 1 | 0.153 | 0.0855 | 0.1262 | 0.8291 | 0 | 0.4573 |
| 33 | Ring3 on PS | 0.8889 | 0.1871 | 0.0723 | 0.1068 | 1 | 0 | 0.4485 |
| 34 | Ring4 on PS | 0.7114 | 0.2706 | 0.047 | 0.0872 | 1 | 0 | 0.4484 |
| 35 | Ring5 on PS | 0.832 | 0.211 | 0.0454 | 0.1117 | 1 | 0 | 0.4503 |
| 36 | Ring6 on PS | 1 | 0.2042 | 0.0395 | 0.1181 | 0.9556 | 0 | 0.3284 |
| 37 | Max of DFT | 0.7498 | 0.397 | 0.2088 | 0 | 1 | 0.1942 | 0.3237 |
| 38 | Mean of DFT | 0.9057 | 0.3649 | 0.1815 | 0.2063 | 1 | 0 | 0.4005 |
| 39 | Energy of DFT | 0.6056 | 0.2047 | 0.0782 | 0 | 1 | 0.0634 | 0.3374 |
| 40 | Variance of DFT | 0.5981 | 0.2076 | 0.0814 | 0 | 1 | 0.0687 | 0.3313 |
| 41 | Skewnes s of DFT | 0.6256 | 0.848 | 0.8017 | 0 | 0.5236 | 1 | 0.4136 |
| 42 | Kurtosis of DFT | 0.6299 | 0.8494 | 0.8026 | 0 | 0.5125 | 1 | 0.4133 |
| 43 | Entropy of DFT | 0.099 | 0.6522 | 0.8869 | 0.9048 | 0 | 1 | 0.459 |
| 44 | Variogra m of DFT 1 distance | 0.5811 | 0.2243 | 0.0858 | 0 | 1 | 0.0799 | 0.327 |
| 45 | Variogra m of DFT 2 distance | 0.5772 | 0.2111 | 0.0807 | 0 | 1 | 0.0747 | 0.3319 |
| 46 | Variogra m of DFT 3 distance | 0.5833 | 0.2065 | 0.0801 | 0 | 1 | 0.0696 | 0.3337 |

**Table A.5: Gabor Filters**

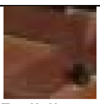
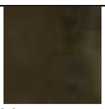
| **Gabor Filters** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Feature Nunber | Feature Name | Building |  Building | Building | Non-building | Non-building |  Non-building | Normali-zed histogram Error |
| 47 | Mean and variance pairs of gabor filters with 6 orienta-tions and 4 scales circularly shifted as pair with maximum mean located at the beginning | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3237 |
| 48 | | 1 | 0.592 | 0.481 | 0.251 | 0.894 | 0 | 0.452 |
| 49 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3237 |
| 50 | | 0.837 | 0.510 | 0.517 | 0.454 | 1 | 0 | 0.4528 |
| 51 | | 0.250 | 0.603 | 0.791 | 1 | 0 | 0.805 | 0.3237 |
| 52 | | 0.587 | 0.326 | 0.399 | 0.138 | 1 | 0 | 0.4572 |
| 53 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3237 |
| 54 | | 0.355 | 0.130 | 0.101 | 0.052 | 1 | 0 | 0.4561 |
| 55 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3497 |
| 56 | | 1 | 0.148 | 0.189 | 0.150 | 0.855 | 0 | 0.4632 |
| 57 | | 0.749 | 0.444 | 0.242 | 0 | 1 | 0.226 | 0.3689 |
| 58 | | 0.989 | 0.141 | 0.144 | 0.177 | 1 | 0 | 0.4651 |
| 59 | | 0.245 | 0.899 | 0.992 | 0.981 | 0 | 1 | 0.3317 |
| 60 | | 0.672 | 0.263 | 0.309 | 0.051 | 1 | 0 | 0.4677 |
| 61 | | 0.749 | 0.997 | 0.631 | 0 | 1 | 0.603 | 0.3238 |
| 62 | | 0.202 | 0.083 | 0.032 | 0 | 1 | 0.000 | 0.4669 |
| 63 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3594 |
| 64 | | 0.403 | 0.105 | 0.107 | 0.092 | 1 | 0 | 0.4629 |
| 65 | | 0.749 | 0.352 | 0.177 | 0 | 1 | 0.164 | 0.3463 |
| 66 | | 0.878 | 0.187 | 0.279 | 0.232 | 1 | 0 | 0.4733 |
| 67 | | 0.260 | 0 | 0.365 | 1 | 0.014 | 0.393 | 0.3311 |
| 68 | | 0.323 | 0.048 | 0.049 | 0 | 1 | 0.036 | 0.4743 |
| 69 | | 0.753 | 0.102 | 0.007 | 0.015 | 1 | 0 | 0.3239 |
| 70 | | 0.297 | 0.093 | 0.070 | 0 | 1 | 0.042 | 0.476 |
| 71 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3237 |
| 72 | | 1 | 0.422 | 0.142 | 0.346 | 0.62 | 0 | 0.4463 |
| 73 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3237 |
| 74 | | 0.941 | 0.507 | 0.132 | 0.320 | 1 | 0 | 0.4639 |
| 75 | | 0.250 | 0.603 | 0.791 | 1 | 0 | 0.805 | 0.3237 |
| 76 | | 0.683 | 0.151 | 0.065 | 0.087 | 1 | 0 | 0.4622 |
| 77 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3237 |
| 78 | | 0.539 | 0.150 | 0.030 | 0.147 | 1 | 0 | 0.4612 |
| 79 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3497 |
| 80 | | 1 | 0.288 | 0.118 | 0.506 | 0.741 | 0 | 0.4618 |
| 81 | | 0.749 | 0.444 | 0.242 | 0 | 1 | 0.226 | 0.3689 |
| 82 | | 1 | 0.294 | 0.102 | 0.329 | 0.468 | 0 | 0.4571 |
| 83 | | 0.245 | 0.899 | 0.992 | 0.981 | 0 | 1 | 0.3317 |
| 84 | | 1 | 0.210 | 0.067 | 0.714 | 0.417 | 0 | 0.4571 |
| 85 | | 0.74 | 0.997 | 0.631 | 0 | 1 | 0.603 | 0.3238 |
| 86 | | 1 | 0.210 | 0.155 | 0.528 | 0.438 | 0 | 0.4569 |
| 87 | | 0.749 | 0.397 | 0.208 | 0 | 1 | 0.194 | 0.3594 |
| 88 | | 0.978 | 1 | 0.321 | 0.288 | 0.697 | 0 | 0.4529 |
| 89 | | 0.749 | 0.352 | 0.177 | 0 | 1 | 0.164 | 0.3463 |
| 90 | | 1 | 0.821 | 0.479 | 0.311 | 0.731 | 0 | 0.4572 |
| 91 | | 0.260 | 0 | 0.365 | 1 | 0.014 | 0.393 | 0.3311 |
| 92 | | 1 | 0.748 | 0.346 | 0.354 | 0.981 | 0 | 0.4603 |
| 93 | | 0.753 | 0.102 | 0.007 | 0.015 | 1 | 0 | 0.3239 |
| 94 | | | | 0.365 | 0.330 | 0.952 | 0.006 | 0.46 |

**Table A.6: Haralick Features**

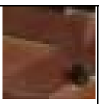
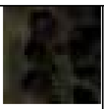
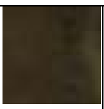
| Haralick Features | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Featur e Nunbe r | Feature Name | Buildin g |  Buildin g | Buildin g |  Non-buildin g | Non-buildin g |  Non-buildin g | Normali-zed histogra m Error |
| 95 | Inertia with distance 1 | 0.664 | 0.288 | 0.104 | 0 | 1 | 0.064 | 0.3563 |
| 96 | Cluster shade with distance 1 | 0 | 0.254 | 0.320 | 0.289 | 1 | 0.295 | 0.4253 |
| 97 | Cluster prominan ce with distance 1 | 0.440 | 0.014 | 0.023 | 0.014 | 1 | 0 | 0.4585 |
| 98 | Local homegeni ty with distance 1 | 0.066 | 0.617 | 0.614 | 0.500 | 0 | 1 | 0.4574 |
| 99 | Energy with distance 1 | 0.134 | 0.571 | 0.453 | 0.278 | 0 | 1 | 0.4542 |
| 100 | Entropy with distance 1 | 0.840 | 0.341 | 0.358 | 0.448 | 1 | 0 | 0.4556 |
| 101 | Inertia with distance 2 | 0.669 | 0.288 | 0.106 | 0 | 1 | 0.067 | 0.3536 |
| 102 | Cluster shade with distance 2 | 0 | 0.201 | 0.256 | 0.246 | 1 | 0.246 | 0.4256 |
| 103 | Cluster prominan ce with distance 2 | 0.329 | 0.011 | 0.014 | 0.013 | 1 | 0 | 0.4636 |
| 104 | Local homegeni ty with distance 2 | 0.100 | 0.652 | 0.623 | 0.461 | 0 | 1 | 0.4608 |
| 105 | Energy with distance 2 | 0.127 | 0.560 | 0.421 | 0.230 | 0 | 1 | 0.4547 |
| 106 | Entropy with distance 2 | 0.829 | 0.330 | 0.350 | 0.458 | 1 | 0 | 0.456 |

**Table A.6 - Continued**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 107 | Inertia with distance 3 | 0.670 | 0.286 | 0.106 | 0 | 1 | 0.068 | 0.3522 |
| 108 | Cluster shade with distance 3 | 0 | 0.168 | 0.231 | 0.236 | 1 | 0.226 | 0.4257 |
| 109 | Cluster prominance with distance 3 | 0.288 | 0.010 | 0.011 | 0.014 | 1 | 0 | 0.4639 |
| 110 | Local homegenity with distance 3 | 0.146 | 0.680 | 0.661 | 0.463 | 0 | 1 | 0.4578 |
| 111 | Energy with distance 3 | 0.125 | 0.537 | 0.420 | 0.215 | 0 | 1 | 0.4572 |
| 112 | Entropy with distance 3 | 0.813 | 0.330 | 0.335 | 0.460 | 1 | 0 | 0.4565 |
| 113 | Inertia with distance 4 | 0.665 | 0.285 | 0.105 | 0 | 1 | 0.068 | 0.3517 |
| 114 | Cluster shade with distance 4 | 0 | 0.224 | 0.298 | 0.318 | 1 | 0.206 | 0.4256 |
| 115 | Cluster prominance with distance 4 | 0.2881 | 0.0094 | 0.0087 | 0.0162 | 1 | 0 | 0.4631 |
| 116 | Local homegenity with distance 4 | 0.205 | 0.701 | 0.684 | 0.487 | 0 | 1 | 0.4592 |
| 117 | Energy with distance 4 | 0.121 | 0.520 | 0.420 | 0.212 | 0 | 1 | 0.4594 |
| 118 | Entropy with distance 4 | 0.806 | 0.334 | 0.330 | 0.462 | 1 | 0 | 0.459 |

**Table A.7: Wavelet Features**

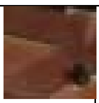
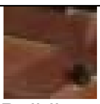
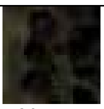
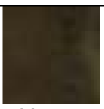
| Wavelet Features | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Feature Nunber | Feature Name | Building |  Building | Building | Non-building | Non-building | Non-building | Normali-zed histogram Error |
| 119 | Energy(LL,1) | 0.2179 | 0.068 | 0 | 1 | 0.0603 | 0.6151 | 0.3425 |
| 120 | Variance(LL,1) | 0.2509 | 0.201 | 0.2559 | 1 | 0 | 0.7218 | 0.4428 |
| 121 | Energy(LH,1) | 0.1715 | 0.011 | 0.0354 | 1 | 0 | 0.6288 | 0.4476 |
| 122 | Variance(LH,1) | 0.3795 | 0.0637 | 0.1449 | 1 | 0 | 0.7802 | 0.441 |
| 123 | Energy(HL,1) | 0.1502 | 0.037 | 0.0823 | 0.2941 | 0 | 1 | 0.4656 |
| 124 | Variance(HL,1) | 0.3254 | 0.1231 | 0.216 | 0.4934 | 0 | 1 | 0.4417 |
| 125 | Energy(HH,1) | 0.1928 | 0.057 | 0.0825 | 1 | 0 | 0.6423 | 0.4459 |
| 126 | Variance(HH,1) | 0.3759 | 0.1653 | 0.2133 | 1 | 0 | 0.7784 | 0.4302 |
| 127 | Energy(LL,2) | 0.2365 | 0.054 | 0 | 1 | 0.0599 | 0.59 | 0.3463 |
| 128 | Variance(LL,2) | 0.2244 | 0.25 | 0.2666 | 1 | 0 | 0.7106 | 0.4482 |
| 129 | Energy(LH,2) | 0.3766 | 0.0294 | 0.0991 | 0.6546 | 0 | 1 | 0.4645 |
| 130 | Variance(LH,2) | 0.593 | 0.1281 | 0.2767 | 0.8033 | 0 | 1 | 0.4591 |
| 131 | Energy(HL,2) | 0.0791 | 0.5301 | 0.1643 | 0.2798 | 0 | 1 | 0.4607 |
| 132 | Variance(HL,2) | 0.200 | 0.692 | 0.33 | 0.469 | 0 | 1 | 0.4401 |
| 133 | Energy(HH,2) | 0.202 | 0.151 | 0.120 | 0.587 | 0 | 1 | 0.4683 |
| 134 | Variance(HH,2) | 0.392 | 0.327 | 0.282 | 0.740 | 0 | 1 | 0.4634 |
| 135 | Energy(LL,3) | 0.244 | 0.035 | 0 | 1 | 0.059 | 0.527 | 0.3498 |
| 136 | Variance(LL,3) | 0.176 | 0.329 | 0.270 | 1 | 0 | 0.518 | 0.4634 |
| 137 | Energy(LH,3) | 0.140 | 0.016 | 0.045 | 0.548 | 0 | 1 | 0.4781 |
| 138 | Variance(LH,3) | 0.348 | 0.093 | 0.181 | 0.747 | 0 | 1 | 0.4724 |
| 139 | Energy(HL,3) | 0.199 | 0.086 | 0.351 | 0.486 | 0 | 1 | 0.4713 |
| 140 | Variance(HL,3) | 0.4004 | 0.2397 | 0.5455 | 0.6535 | 0 | 1 | 0.4516 |
| 141 | Energy(HH,3) | 0.148 | 0.079 | 0.223 | 1 | 0 | 0.976 | 0.467 |
| 142 | Variance(HH,3) | 0.321 | 0.210 | 0.416 | 1 | 0 | 0.986 | 0.4592 |

**Table A.8: HSV Domain Features**

| HSV domain features | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Feature Nunber | Feature Name | Building |  Building | Building |  Non-building | Non-building |  Non-building | Normalized histogram Error |
| 143 | Mean of hue | 0 | 0.0989 | 0.941 | 0.4979 | 1 | 0.566 | 0.2147 |
| 144 | Variance of hue | 0.8708 | 0.0062 | 0.0991 | 1 | 0.9533 | 0 | 0.4291 |
| 145 | Mean of hue gradient | 0.5212 | 0.0333 | 0.1582 | 1 | 0.6707 | 0 | 0.3855 |
| 146 | Variance of hue gradient | 0.8294 | 0.0829 | 0.1309 | 0.9603 | 1 | 0 | 0.4111 |
| 147 | Center pixel value of hue | 0 | 0.0324 | 0.91 | 0.2853 | 1 | 0.3521 | 0.202 |
| 148 | Mean of sat | 0.4465 | 1 | 0.1546 | 0.3159 | 0 | 0.6916 | 0.2813 |
| 149 | Variance of sat | 1 | 0.2328 | 0 | 0.971 | 0.32 | 0.0039 | 0.386 |
| 150 | Mean of sat gradient | 0.3968 | 0.158 | 0.0456 | 1 | 0.2135 | 0 | 0.4137 |
| 151 | Variance of sat gradient | 0.3929 | 0.1076 | 0.0243 | 1 | 0.1874 | 0 | 0.411 |
| 152 | Zernike of sat | 1 | 0.1519 | 0 | 0.1926 | 0.5122 | 0.2778 | 0.4385 |
| 153 | Circular mellin of sat | 0.6993 | 0 | 0.4901 | 0.4966 | 1 | 0.7997 | 0.4713 |
| 154 | Center pixel value of sat | 0.5278 | 0.3758 | 0.1302 | 1 | 0 | 0.3928 | 0.2834 |
| 155 | Mean of value | 0.9573 | 0.6226 | 0.1959 | 0 | 1 | 0.2321 | 0.3033 |
| 156 | Variance of value | 0.824 | 0.3209 | 0.1599 | 0.2469 | 1 | 0 | 0.4434 |
| 157 | Mean of value gradient | 1 | 0.3275 | 0.1302 | 0.3167 | 0.98 | 0 | 0.4459 |
| 158 | Variance of value gradient | 0.8822 | 0.4755 | 0.2132 | 0.2066 | 1 | 0 | 0.4091 |
| 159 | Zernike of value | 0.5006 | 0.4013 | 0.1358 | 0.3405 | 1 | 0 | 0.4636 |
| 160 | Circular mellin of value | 1 | 0.1223 | 0.3901 | 0.3072 | 0.8207 | 0 | 0.467 |
| 161 | Center pixel value of value | 1 | 0.7376 | 0.3191 | 0 | 0.922 | 0.3262 | 0.2755 |