An AdaBoost Based Approach to Automatic Classification and Detection of Buildings Footprints, Vegetation Areas and Roads from Satellite Images

A thesis Presented

by

Cansu Gonulalan

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical and Computer Engineering

Northeastern University
Boston, Massachusetts

July 2010
Abstract

In recent years, there has been an increasing demand for applications to monitor the targets related to land-use, using remote sensing images. Advances in remote sensing satellites give rise to the research in this area. Many applications ranging from urban growth planning to homeland security have already used the algorithms for automated object recognition from remote sensing imagery. However, they have still problems such as low accuracy on detection of targets, specific algorithms for a specific area etc.

In this thesis, we focus on an automatic approach to classify and detect building footprints, road networks and vegetation areas. The automatic interpretation of visual data is a comprehensive task in computer vision field. The machine learning approaches improve the capability of classification in an intelligent way.

We propose a method, which has high accuracy on detection and classification. The multi class classification is developed for detecting multiple objects. We present an AdaBoost-based approach along with the supervised learning algorithm. The combination of AdaBoost with “Attentional Cascade” is adopted from Viola and Jones [1]. This combination decreases the computation time and gives opportunity to real time applications. For the feature extraction step, our contribution is to combine Haar-like features that include corner, rectangle and Gabor. Among all features, AdaBoost selects only critical features and generates in extremely efficient cascade structured classifier.
Finally, we present and evaluate our experimental results. The overall system is tested and high performance of detection is achieved. The precision rate of the final multi-class classifier is over 98%.
Acknowledgements

This thesis wouldn’t have been possible without the help and support of several people. I would like to express my deep and sincere gratitude to my supervisor Octavia Camps. With her invaluable support, guidance and patience, she helped to make this thesis possible.

I would like to thank to the Scientific and Technological Research Council of Turkey (TUBITAK) for their fully financial support throughout my years at Northeastern University.

I want to express my gratitude to Jennifer Dy for her invaluable help with my research and Gilead Tadmor for guiding me to find my research advisor.

I am grateful to Caglayan Dicle and Teresa Mao. Their insightful comments have helped to considerably improve this thesis.

Last but not least, I would like to thank my family, my mother Sevgi Gonulalan, my father Ugur Gonulalan, my brother Umut Gonulalan and Serdar Bender for their unconditional love and support they have shown me throughout my life. To them I dedicate this thesis.

I would also dedicate this thesis to Mustafa Kemal Ataturk, the eternal leader of Turkish Nation and founder of the Republic of Turkey.
## Contents

Abstract

Acknowledgements

List of Figures

List of Tables

1 Introduction
   1.1 Introduction .................................................. 1
   1.2 Background .................................................. 3
      1.2.1 Sensors ................................................. 3
      1.2.2 Overview of Supervised Learning ......................... 4
      1.2.3 Pattern Classification ................................. 5
         1.2.3.1 Statistical Learning Algorithms ................. 6
         1.2.3.2 Perceptron Based Techniques ................. 7
         1.2.3.3 Support Vector Machines .................... 8
         1.2.3.4 Artificial Intelligence ..................... 8

2 Previous Work
   2.1 Edge-Based Techniques ...................................... 12
   2.2 Texture-Based Techniques .................................... 15
      2.2.1 Statistical Methods ................................... 16
         2.2.1.1 Co-occurrence Matrices ..................... 16
         2.2.1.2 Autocorrelation Features ................. 17
      2.2.2 Geometrical Methods .................................. 17
         2.2.2.1 Voronoi Tessellation Features ............ 17
      2.2.3 Model Based Methods .................................. 17
         2.2.3.1 Random Field Models ....................... 17
2.2.3.2 Fractals ........................................... 18
2.2.4 Signal Processing Methods ........................................... 18
  2.2.4.1 Fourier Transform ........................................... 18
  2.2.4.2 Wavelet Transform ........................................... 19
  2.2.4.3 Gabor Transform ........................................... 20
2.3 Feature Selection ........................................... 20
2.4 Classification-Based Techniques ........................................... 21

3 Problem Statement ........................................... 22
  3.1 Problem Definition ........................................... 22
  3.2 Study Area and Data ........................................... 23

4 Procedure ........................................... 25
  4.1 Data Preparation ........................................... 25
  4.2 Features ........................................... 31
    4.2.1 The Integral Images ........................................... 31
    4.2.2 Feature Types ........................................... 33
      4.2.2.1 Corner Features ........................................... 33
      4.2.2.2 Rectangle Features ........................................... 33
      4.2.2.3 Texture Features ........................................... 35
  4.3 Learning Algorithm ........................................... 39
    4.3.1 AdaBoost ........................................... 39
      4.3.1.1 Training Error Analysis ........................................... 42
      4.3.1.2 Multiclass AdaBoost ........................................... 44
    4.3.2 Cascade ........................................... 47

5 Experiments and Results ........................................... 51
  5.1 Method of Evaluation ........................................... 51
    5.1.1 Holdout Method ........................................... 51
    5.1.2 K-fold Cross Validation ........................................... 52
    5.1.3 Leave-One-Out Cross Validation ........................................... 52
  5.2 Performance Criteria ........................................... 53
  5.3 Evaluation Results ........................................... 55
    5.3.1 Two Class Classification Results ........................................... 55
      5.3.1.1 Building Detection ........................................... 55
      5.3.1.2 Vegetation Area Detection ........................................... 58
      5.3.1.3 Road Network Detection ........................................... 59
    5.3.2 Multi Class Classification Results ........................................... 61
  5.4 Learning Discussion ........................................... 64
# Table of Contents

5.4.1 Feature Analysis ........................................... 66
5.4.2 Cascade Structure Discussion ............................. 67
5.5 Experiments .................................................. 70

6 Conclusion ....................................................... 77
  6.1 Summary ..................................................... 77
  6.2 Future Work ................................................ 78

A Sensors in Remote Sensing Systems ...................... 80
  A.1 Passive Sensors ........................................... 80
  A.2 Active Sensors ............................................ 82

B User Manual for the MATLAB Code ..................... 85
  B.1 Functions .................................................. 85
    B.1.1 Functions to Read Data and Prepare Data Set .......... 85
    B.1.2 Functions to Building a Cascade Classifier ............ 85
    B.1.3 Functions to Test Classifier ........................... 86
  B.2 Sample Program .......................................... 86
    B.2.1 Training ............................................... 86
    B.2.2 Testing ................................................. 87

Bibliography ..................................................... 95
# List of Figures

1.1 A decision tree [11] .................................................. 9

2.1 (a) Line and Junctions , (b) Selected Hypothesis using (a), (c) Evidence [43] .................................................. 12

2.2 The shadow-ground transition [46] .................................. 13

2.3 An illustration of extracting task-specific line segments [36] .......... 14

2.4 Texture Analysis ....................................................... 15

2.5 Results of one- and two-level Haar wavelet transforms for a 512- by 512-pixel image in Buffalo, New York[70] ...................... 19

2.6 Image Processing chain in Remote Sensing [74] ...................... 20

3.1 Some Examples of Google Earth Satellite Images ...................... 24

4.1 A Set of Building Images ............................................. 27

4.2 A Set of Tree Images ................................................ 28

4.3 A Set of Road Images ................................................ 29

4.4 A Set of Background Images ....................................... 30

4.5 Representation of Integral Image ................................... 32

4.6 Calculation of rectangles using Integral Image ...................... 32

4.7 Corner Features ..................................................... 33

4.8 Rectangle Features .................................................. 34

4.9 Haar-Like Features for Gabor Filters ................................ 38

4.10 Reweighing Approach .............................................. 41

4.11 One per Class method for Multi Two-Class ......................... 46

4.12 A cascade consists of layers. ...................................... 48

5.1 Holdout Method ..................................................... 52

5.2 K-fold cross validation .............................................. 53

5.3 Leave-one-out cross validation ..................................... 54

5.4 ROC curves for Building Classifier-Holdout Method ................. 56

5.5 ROC curves for Building Classifier-10-fold cross validation method .. 57

5.6 ROC curves for Tree Classifier-Holdout Method ................... 59
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.7</td>
<td>ROC curves for Tree Classifier - 10-fold cross validation method</td>
<td>60</td>
</tr>
<tr>
<td>5.8</td>
<td>ROC curves for Road Classifier-Holdout Method</td>
<td>61</td>
</tr>
<tr>
<td>5.9</td>
<td>ROC curves for Road Classifier - 10-fold cross validation method</td>
<td>62</td>
</tr>
<tr>
<td>5.10</td>
<td>Precision-Recall Graph</td>
<td>64</td>
</tr>
<tr>
<td>5.11</td>
<td>Elements of Cascade Structure</td>
<td>65</td>
</tr>
<tr>
<td>5.12</td>
<td>Elements of Cascade Structure for Building Detector</td>
<td>68</td>
</tr>
<tr>
<td>5.13</td>
<td>Distribution of Feature Types in Building Detector</td>
<td>69</td>
</tr>
<tr>
<td>5.14</td>
<td>Distribution of Feature Types in Vegetation Area Detector</td>
<td>69</td>
</tr>
<tr>
<td>5.15</td>
<td>Distribution of Feature Types in Road Network Detector</td>
<td>70</td>
</tr>
<tr>
<td>5.16</td>
<td>Comparison with Detector Cascade to Monolithic Classifier-Building Detection (a) ROC curves</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>(b) Processing Times</td>
<td></td>
</tr>
<tr>
<td>5.17</td>
<td>Comparison with Detector Cascade to Monolithic Classifier-Vegetation Detection (a) ROC curves</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>(b) Processing Times</td>
<td></td>
</tr>
<tr>
<td>5.18</td>
<td>Comparison with Detector Cascade to Monolithic Classifier-Road Detection (a) ROC curves</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>(b) Processing Times</td>
<td></td>
</tr>
<tr>
<td>5.19</td>
<td>Examples of a Multi-Class AdaBoost Based Detection - 1</td>
<td>74</td>
</tr>
<tr>
<td>5.20</td>
<td>Examples of a Multi-Class AdaBoost Based Detection - 2</td>
<td>75</td>
</tr>
<tr>
<td>5.21</td>
<td>Examples of a Multi-Class AdaBoost Based Detection - 3</td>
<td>76</td>
</tr>
<tr>
<td>A.1</td>
<td>Passive Sensing</td>
<td>81</td>
</tr>
<tr>
<td>A.2</td>
<td>Active Sensing</td>
<td>83</td>
</tr>
<tr>
<td>B.1</td>
<td>The “MATLABGUI” program</td>
<td>87</td>
</tr>
<tr>
<td>B.2</td>
<td>Training Parameters</td>
<td>88</td>
</tr>
</tbody>
</table>
List of Tables

1.1 Spectral bands for Landsat MSS, TM and ETM+ sensors [9] ........... 4
4.1 Training and Validation Set .............................................. 26
4.2 The number of features of each type ............................... 38
5.1 Confusion Matrix for Two-Class ...................................... 54
5.2 Training, Test and Validation Set for Building Classifier-Holdout Method 55
5.3 The Resultant Confusion Matrix for Building Classifier-Holdout Method 55
5.4 Performance of Building Classifier - Holdout Method ............. 56
5.5 Training, Test and Validation Set for Tree Classifier-Holdout Method 58
5.6 The Resultant Confusion Matrix for Tree Classifier-Holdout Method 58
5.7 Performance of Vegetation Classifier- Holdout Method ........... 58
5.8 Training, Test and Validation Set for Road Classifier-Holdout Method 59
5.9 The Resultant Confusion Matrix for Road Classifier-Holdout Method 60
5.10 Performance of Road Classifier- Holdout Method .................. 61
5.11 Confusion Matrix for Multi Class-Holdout Method ................ 63
5.12 Types of Features .......................................................... 66
Chapter 1

Introduction

1.1 Introduction

Increasing population gives rise to problems associated with urbanization. Urban growth and economic development lead to high demands on better planning and management approaches. Thus, local governments have urged to monitor and track numerous items. Therefore, the up-to-date and accurate geospatial information of the desired area is a necessity.

The history of Geospatial Information Systems (GIS) started in 1962. Tomlinson first developed it for Canada Land Inventory (CLI). Then by the early 1980s, commercial vendors of GIS software were available. By the end of the 20th century, the fast progress in various GIS had been consolidated and standardized on relatively few platforms. Furthermore, the concept of viewing GIS data over the Internet, required data format and transfer standards. More recently, free and open source GIS packages are becoming popular and users can run and customize them in order to achieve specific tasks [2].

The aims of GIS are to map, analyze and assess real-world problems. Digital image base map, land use zoning, political boundaries, parcel maps, land cover, road network, building footprints, utility networks (water, sewage, electricity, etc.), topography, and
green space are the typical layers of GIS [3]. The information obtained from this technology can be divided into two parts: *spatial data* which gives the tabular information of actual locations and *attribute data* that provides additional information for that locations. With the help of these properties, GIS is an effective solution for real world problems. For a detailed description of GIS and its many applications see Longley et al. [4].

Aerial photography or satellite remote sensing is used to acquire the digital image base map. Road network, building footprints and vegetation areas are also other important layers in GIS [5]. The extraction of these layers was mostly done by manual digitization. It was not only time consuming but also an expensive process to obtain all the layers. Therefore automated and semi-automated extraction of urban geospatial information systems has been developed to solve these problems. However, it is still an important and open area to find the automated methods that are accurate and easy to implement properties.

Nowadays, the availability of free and commercial high-resolution remote sensing multispectral imagery from sensors such as IKONOS [3] increases the significance of this concept.

There are a huge number of example maps that provide accurate, current and consistent data for the United States and other countries. Urban growth planning, emergency response and management, and homeland security applications are only three of the potential uses of these maps. Moreover, real-time data updating is another reason to improve algorithms for automated object recognition from remote sensing imagery.

Our goal in this research is to find a more general solution to the “Satellite Image Understanding” task. To that need, we use machine learning algorithms rather than using pixel-based methods only. Knowing the limitations of previous approaches, we propose high accuracy on detection and classification of the object of interests in remote sensing images. Moreover, we extent the type of objects to buildings, roads and vegetation areas. Multi class classification is developed for detecting multiple objects. Supervised learning algorithm is employed. We have adopted the idea of AdaBoost
Chapter 1. Introduction

learning approach and cascade originally proposed by Viola and Jones[1] and applied to the problem of object detection in the satellite images.

As far as we know, this thesis is the first one in the literature that dealing with the detection of buildings, roads and tree from satellite images using AdaBoost learning algorithm. The previous works that we have reviewed in chapter 2 have the promising classification rates to detect only one objects at a time. Our work improves the accuracy and detects multiple objects.

1.2 Background

This section summarizes the required background for the research in this thesis. First of all, we clarify the type of sensor and provide a summary of the supervised classification and pattern recognition concepts briefly.

1.2.1 Sensors

Remote sensing systems use active and passive sensors according to energy usage. Appendix A explains sensor types used in remote sensing systems [6].

In this research, images from passive sensors are obtained using Google Earth\textsuperscript{TM}. Google Earth\textsuperscript{TM} is a free desktop Geographic Information System (GIS). The vast majority of satellite imagery of Google Earth\textsuperscript{TM} is provided by Landsat [7].

Landsat is first example of land observation satellites. The Landsat program began in 1972 when Landsat 1 satellite was launched [8]. Two satellites are currently providing images in the Landsat program. These satellites are Landsat 5 and Landsat 7, launched in 1984 and 1999 respectively. Landsat 5 has the MultiSpectral Scanner (MSS) sensor and Thematic Mapper (TM) sensor. Landsat 7 carries the Enhanced Thematic Mapper plus (ETM+) sensor.

The band designations for Landsat satellites are summarized in Table 1.1.
Table 1.1: Spectral bands for Landsat MSS, TM and ETM+ sensors [9]

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Landsat Satellites</th>
<th>Wavelength</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multispectral Scanner (MSS)</td>
<td>Landsat 4-5</td>
<td>Band 1</td>
<td>0.5 – 0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 2</td>
<td>0.6 – 0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 3</td>
<td>0.7 – 0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 4</td>
<td>0.8 – 0.11</td>
</tr>
<tr>
<td>Multispectral Scanner (MSS)</td>
<td>Landsat 1-3</td>
<td>Band 4</td>
<td>0.5 – 0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 5</td>
<td>0.6 – 0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 6</td>
<td>0.7 – 0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 7</td>
<td>0.8 – 0.11</td>
</tr>
<tr>
<td>Thematic Mapper (TM)</td>
<td>Landsat 4-5</td>
<td>Band 1</td>
<td>0.45 – 0.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 2</td>
<td>0.52 – 0.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 3</td>
<td>0.63 – 0.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 4</td>
<td>0.76 – 0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 5</td>
<td>1.55 – 1.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 6</td>
<td>10.40 – 12.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 7</td>
<td>2.08 – 2.35</td>
</tr>
<tr>
<td>Enhanced TM Plus (ETM+)</td>
<td>Landsat 7</td>
<td>Band 1</td>
<td>0.45 – 0.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 2</td>
<td>0.52 – 0.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 3</td>
<td>0.63 – 0.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 4</td>
<td>0.76 – 0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 5</td>
<td>1.55 – 1.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 6</td>
<td>10.40 – 12.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 7</td>
<td>2.08 – 2.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band 8</td>
<td>.52 – .90</td>
</tr>
</tbody>
</table>

* TM Band 6 was acquired at 120-meter resolution, but products resample to 60-meter pixels.

** ETM+ Band 6 was acquired at 60-meter resolution, but products resample to 30-meter pixels.

1.2.2 Overview of Supervised Learning

The art of machine learning plays a key role in a wide range of areas such as science, finance, etc. Machine learning provides significant improvements in data mining, artificial intelligence, other areas of engineering and other disciplines as well.
Chapter 1. *Introduction*

Machine learning process can be grouped in several forms, “Supervised”, “Unsupervised” and “Reinforcement” learning. In this thesis, only supervised type of learning is adopted and this section describes its usage.

Every method employs machine learning if it combines the information with training set [10]. The aim is to extract information from a set of training examples. The difference between supervised and unsupervised approach is described as follows. When the training examples have the ground truths, which are provided manually, it is called, supervised learning approach, since the presence of the ground truth label guide the learning process. On the other hand, unsupervised learning attempts to design classifier without any help from ground truth data and uses only clustering to obtain patterns. Since it is hard to compare results in the unsupervised case and this makes the algorithm ineffective, this type of learning develops much less than supervised one in the literature.

Our task is to design effective classification model, which predicts the unseen data best. In literature, features are called as predictors or experts of the system.

The output of learning process falls into two categories: quantitative and qualitative. If the classifier predicts quantitative responses, the learning problem is called *regression estimation*. For the qualitative responses, it is called *pattern recognition*.

We are dealing with the values calculated using pixel information, our work concentrates on pattern recognition.

### 1.2.3 Pattern Classification

Most problems in Pattern Recognition is generally formulated in a similar way. Let us first think about a two-class classification problem, where there are two different classes of objects of interest. The only information about the classes is provided by features, which are samples of real valued measurements. And these real valued measurements compose a feature vector. \( x \in \mathbb{R}^d \) where \( d \) is the dimension of the feature vector. There are also *priori probabilities* \( P_0, P_1 \) in order to model the uncertainty.
More formally, random feature vectors \( X \) are the observable inputs of the system. They are generated in two ways:

- a random class \( Y \in \{0, 1\} \) is first selected according to the a priori probabilities.
- the observed feature vector \( X \) is then selected according to the class-conditional distribution \( F_y \).

Given a realization of the measured feature vector \( X = x \), the classifier needs to decide whether the unknown object’s feature vector \( x \) belongs to class 0 or 1.

Thus, a classifier or decision rule in this case is simply a map \( g : \mathbb{R}^d \rightarrow \{0, 1\} \). For each class \( g(x) \), we can assign a feature vector \( X = x \). Given a classifier \( g \), the performance of \( g \) can be measured by the probability of error.

\[
L(g) = P\{g(X) \neq Y\} \tag{1.1}
\]

For the supervised learning situation, we have the training set as follows:

\[
S_{training} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \tag{1.2}
\]

where \( x_i \)’s are the samples and \( y_i \)’s are the labels of \( x_i \) provided by a supervisor. Our goal is to estimate the function \( \hat{f} : X \rightarrow Y \) that can correctly discriminate the classes.

A good number of methods have been developed in the pattern recognition field since 1950s. We will provide a brief introduction to five best known methods according to Kotsiantis and Xu in the following sections [11] , [12].

1.2.3.1 Statistical Learning Algorithms

The key idea of statistical learning algorithms is to use the probability model in an explicit way. Therefore the decision on classification uses this probability model rather than simply discrimination. In this method the probabilities \( P(Y) \), \( P(Y|X) \) and \( P(X|Y) \) are the parameters of the probability model.
Naive Bayesian Classifiers  Naive Bayesian Classifier assumes conditional independence between the features. One of major advantages of the Bayes Classifier is its fast computation. However, in practice, it is less accurate than sophisticated learning algorithms [13], [14], [15].

Linear Discriminant Analysis (LDA)  LDA is another method that models the classes as if they are multivariate Gaussian. While for continuous quantities of measurements, LDA and the related Fisher’s linear discriminant [16] perform well, the other technique Discriminant Correspondence Analysis (DCA) works better for dealing with categorical variables [17].

Instance Based Methods (IBM)  Contrary to most pattern recognition methods, where training time is greater than the testing time, IBM require less calculation time for training process and more computational effort on classification process. It is the reason why they are count as “lazy learning algorithms” [18].

The most straightforward example of IBM is nearest-neighbor algorithm such as k-Nearest Neighbour (k-NN) [19]. It tries to find k most nearest instances of the training set according to a distance metric. The resulting class is assigned by the most frequent class label of the k nearest instances.

Another method for estimating probability distributions of data is Maximum Entropy [20]. However, the most well known techniques are related to Bayesian Networks [21].

1.2.3.2 Perceptron Based Techniques

The perceptron is a binary classifier that maps the input vector \( x \) to an output value \( f(x) \).

\[
f(x) = \begin{cases} 
1 & w \cdot x + b > 0 \\
0 & \text{otherwise}
\end{cases}
\]
where $w$ is a vector of real-valued weights and $b$ is the “bias”, a constant term that does not depend on any input value.

The idea of perceptron was proposed by Rosenblatt in 1962 [22].

**Single Layer Perceptron** The weight vectors $w_1$ to $w_n$ are calculated and the perceptron computes $\sum_i x_i w_i$ and if the sum is above a certain threshold, output is 1; otherwise it is 0. Littlestone and Warmuth [23] have developed “WINNOW” and Freund and Schapire [24] have created “voted-perceptron”.

**Artificial Neural Networks** Artificial Neural Networks (ANN) are the solution of the nonlinear learning instances, also called multilayered perceptrons. Rumelhart et al. [25] and Zhang [26] provided an overview of ANN.

### 1.2.3.3 Support Vector Machines

A Support Vector Machine (SVM) performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories [27], [28]. The hyperplane is found by minimizing the squared norm of the separating hyperplane. A convex quadratic programming (QP) problem is

\[
\text{Minimize}_{w, b} \phi(w) = \frac{1}{2} \|w\|^2
\]

subject to $y_i(w^T x_i + b) \geq 1$, $i = 1, ..., l$.

The major advantage of SVM is that it reaches global minimum without being stuck in local minima. However, it only works well in binary class problems.

### 1.2.3.4 Artificial Intelligence

In this section, we will review the intuition-based algorithms, which are also called as artificial intelligence.
**Decision Trees** Decision trees are the structure in which instances are sorted based on feature values. See Figure 1.1

![Decision Tree Diagram](image)

**Figure 1.1**: A decision tree [11]

In the literature the most known algorithms to build up decision trees are C4.5 [29], Rainforest [30].

**Ensemble Methods** The combination of multiple decision rules leads us to ensemble methods. The algorithms has the following parameters.

- The output variable, $Y \in \{-1, 1\}$.
- The base classifiers, $h : X \to \{-1, 1\}$ of $H$, which is large family of classifiers.
- Decision stumps split $X$ along a hyper plane to the coordinate axes.
• $F$ is the functions of $f : X \rightarrow R$ obtained from the linear combination of base classifiers

\[ F = f(x) = \sum_{l=1}^{T} \alpha_l h_l(x) \]  \hspace{1cm} (1.4)

where $\alpha_l > 0$

• Each function $f$ defines a classifier $g_f(x)$ which is labeled as “1” if $f(x) > 0$ and “−1” otherwise.

Bagging, boosting and randomized trees are the examples of ensemble methods. They use the idea of combining many “experts opinions” and producing a powerful “committee. (See [31], [32] and [24])

Boosting builds an additive model iteratively. In this thesis, we are using adaptive boosting method: AdaBoost.
Chapter 2

Previous Work

Many approaches have been developed to extract objects from satellite and aerial images. These objects include inferring land usage, detecting man made objects and similar applications of remote sensing image processing framework cover a huge number of applications. This chapter investigates the current trends in the literature. However, it is impossible to cover all literature of classification and extraction of urban-related features such as buildings, roads, terrain etc. Therefore, we only focus on assessing the extraction methods, which are using passive sensors and mono images in order to be more specific.

In the literature, some of papers can be categorized according to the object of the interest. Road detection from satellite images were studied in [33], [34] and [35]. Moreover, in the researches [35], [36] and [37], buildings are extracted and classified. The surveys by Mayer [38] and [39] are perfect resources that cover most of the techniques in building detection and road network detection, respectively. Later, Unsalan and Boyer [40] conducted an extended version of Mayers review. Furthermore, the vegetation parts are extracted in [41].

We have reviewed more than 30 influential papers and grouped the papers according to the approach they used. The sections are Edge-Based Techniques, Texture-Based Techniques and Classification-Based Techniques. Moreover, we have discussed to qualitative selection of features to get better results in an additional section. Some papers
may belong to more than one approach. In this thesis, papers were grouped according to their major contributions.

\section*{2.1 Edge-Based Techniques}

Edge-Based Techniques consist of linear feature detection, groupings of parallelogram hypothesis and building polygon verifications. They make use of information such as geometrical shapes, shadows, and angles. Kim and Nevatia's paper \cite{Kim2005} focuses on three categories; shadow, wall and roof. They verify these categories according to corresponding evidences such as roof-cast, corner, region and standard deviation. These evidences are the major hints to prove the hypothesis. Another research by Lin and Nevatia \cite{Lin2006} follows the linear feature extraction method and verification of hypothesis procedure. They assume that buildings are rectilinear, which means the shape is either rectangle or the composition of rectangle, such as I, T and L shape. Figure 2.1 shows the general approach.

\begin{figure}[ht]
\centering
\includegraphics[width=\textwidth]{figure2.1.png}
\caption{(a) Line and Junctions, (b) Selected Hypothesis using (a), (c) Evidence \cite{Lin2006}}
\end{figure}

Structural features are extracted in the paper from Jaynes et al. \cite{Jaynes2007} in order to obtain planar polygonal rooftops. Local information such as orthogonal corners etc. is obtained from the image. Then, they are used a weighted graph which verifies the hypothesis. The graphs use cycles to show the relationship between the features.

Shadows are the key features in the edge-based methods. The relationships between the structure and the shadow are exploited by localization of structure/shadow boundaries and attribution of shadow segments \cite{Gardner2008}.
Shufelt and Liow et al. also use shadows as evidences for detection [46], [47]. All the above papers show that shadows play an important role on their assumptions. They enhance the detection reliability and decrease the probability of misclassification due to segmentation faults. On the other hand, shadow information may also mislead us when they are cast on buildings close by or occluded with each other.

![Figure 2.2: The shadow-ground transition [46]](image)

Related to edge based method, Sohn and Dowman (2001) used local Fourier transformation to analyze the dominant orientation angle in a building cluster, to extract rectilinear building outlines from Ikonos imagery based on a binary space partitioning tree and to form the rectilinear building boundaries [36]. Figure 2.3 displays an illustration of line segment extraction by the analysis of global dominant orientation. In (b), the line segments are extracted and the histogram of orientation angle is shown in (c). The last figure (d) is obtained by re-extracting and correcting the line segments using the global dominant orientation.

A modified snake model is also used for building detection tasks [48]. Illumination information reduces the constraints. And by combining illumination information with the snake model, the contours of the building can be found quickly in a complex environment.
Another example of edge-based methods is the method of Hough transform. It is used to provide incremental information for the length of linear feature. The true length of a line segment is calculated more accurately. It is also required to use a post-processing algorithm to extract the frame of buildings. Bayesian probabilistic approach is chosen for processing Hough space in [49].

When the object of interest in remote sensing data is a building, the geometric regularities can be used in advance. They are composed of mostly flat roofs with perpendicular outlines. In a similar manner, road pavements have parallel edges and this property makes the algorithms effective. The papers, [50], [46], [51], [52] and [53] exploit this idea.
2.2 Texture-Based Techniques

Texture feature extraction is a key factor for segmentation. It has been extensively used in classification of remote sensing images. For a detailed explanation of texture features, we recommend two papers by Tuceryan and Jain [54] and Wezska et al. [55]. We have categorized the texture analysis according to the research by Tuceryan and Jain [54].

![Texture Analysis Diagram]

**Figure 2.4:** Texture Analysis

Texture information can be also applied to multiple images in order to reconstruct 3D geometry. Moreover, remote sensing images have rich texture information. In the paper [56], they incorporate texture information from oblique aerial images by combining
multiple aerial images to determine the models textures. Their aim is to generate a composite texture. The results show that non-smooth and visible transitions are observed when there exists a significant color difference between images.

2.2.1 Statistical Methods

Statistical methods are the first ones used in the literature. Gray values have a spatial distribution in the images; therefore, they are the defining quantity of texture.

Based on statistical approach to texture analysis, the research by Aguera et al. used texture analysis to improve classification results [57]. Thus, the optimum texture parameter depends on the main objective of the image classification. They claim the mean of texture parameter helps to reduce number of wrongly detected pixels and the angular second moment of the parameter decreases the undetected or unclassified pixels. They use QuickBird and IKONOS images for their analysis.

2.2.1.1 Co-occurrence Matrices

Co-occurrence estimates are related to second order statistics and have been widely used in remote sensing applications. Since it is a powerful feature extraction method in texture analysis has been applied extensively in land use classification.

Conners et al. [58] were to first to apply this method in the terrain classification field. Haralick et al. also analyze gray level co-occurrence features. [59]. The gray level co-occurrence matrices are computed for each of the four directions, 0, 45, 90, 135. Using texture analysis, they obtained 80% accuracy for seven-class classification.

Applying texture feature extraction in the analysis was the main contribution of the paper by Irons and Petersen [60]. They analyze the mean, variance, skewness, kurtosis, and the mean of the maximum gray level difference as features on Landsat images.

Karathanassi et al. [61] develop algorithms based on statistical measurements of the texture. They employ the concepts of occurrence frequency or co-occurrence matrices
on binary data. They state that the results are better than the 79% that was obtained by using the maximum likelihood classifier.

2.2.1.2 Autocorrelation Features

Autocorrelation features can detect the amount of regularity in the image. It also provides information about fineness or coarseness of the texture of an image. Kang et al. [62] analyze the scale invariant texture analysis by taking advantage of the multi-scale local autocorrelation features. Their results show that this approach also works for segmentation of 2D images such as satellite images.

2.2.2 Geometrical Methods

Geometrical methods give more importance on the idea that texture is composed of texture elements or primitives.

2.2.2.1 Voronoi Tessellation Features

Voronoi Tessellations present the structure and organization of the primitives. One example of the primitives usage is in image edges by Ojala and Pietikyinen, [63].

2.2.3 Model Based Methods

In this type of methods, the texture is not only described but also synthesized [63].

2.2.3.1 Random Field Models

The most popular random field model is Markov Random Field method. The most important property of this method is the ability to capture contextual information of the image. Tree-structured Markov random field approach is used by Cicala et al. [64].
Another related research is by Gaetano et al. [64], applying the hierarchical analysis to the multi-resolution remote sensing images.

Lorette et al. [65] proposed textural analysis by modeling Markovian. They called their algorithm as “Anisotropic Texture Analysis with Eight Different Chain Based Model”.

2.2.3.2 Fractals

Fractals deal with surfaces in terms of statistical quality of roughness and self-similarity at different scales. Mandelbrot was the first to propose fractal geometry [66].

Ilow and Leung [67] apply concepts of fractals in satellite images of a sea surface. Their model is based on the two-dimensional (2-D) fractionally integrated autoregressive-moving average process.

2.2.4 Signal Processing Methods

Recently, texture analyses based on signal processing methods have become very popular in this area of research.

According to Tuceryan and Jain [54], we can categorize these approaches as Fourier domain, Gabor and Wavelet.

2.2.4.1 Fourier Transform

Frequency analysis is best done by Fourier analysis. It founds the global frequency content in an image.

Bajcsy and Lieberman [68] applied Fourier transform analysis on analyzing the shape of texture.
2.2.4.2 Wavelet Transform

On the other hand, wavelet transform is a window function whose width changes as the frequency changes [69].

With the help of wavelet framework, the texture information for each scale can be represented simultaneously. In order to represent texture information, it is stated that using small windows are preferable since the sub-window idea can detect objects of different sizes.

Bian [70] took advantage of the Haar Wavelet for extracting man made objects from aerial images. Figure 2.5 represents this approach.

![Figure 2.5: Results of one- and two-level Haar wavelet transforms for a 512- by 512- pixel image in Buffalo, New York[70]](image)

Figure 2.5: Results of one- and two-level Haar wavelet transforms for a 512- by 512- pixel image in Buffalo, New York[70]
2.2.4.3 Gabor Transform

Gabor transform is a special case of wavelet transform when the window function is Gaussian [71]. The filter banks describe the texture. Every filter has a specific frequency and orientation applied on the image. The features are extracted these filtered images.

[72] and [73] are the illustrations of Gabor transform for texture analysis.

2.3 Feature Selection

Tuia and Camps-Valls [74] have covered the recent advances in remote sensing in the image-processing framework. They summarize the framework from acquisition to final product as in figure 2.6 :

\[ \text{ACQUISITION} \rightarrow \text{CODING} \rightarrow \text{FUSION} \rightarrow \text{FEATURE EXTRACTION} \rightarrow \text{DENOISING} \rightarrow \text{UNMIXING} \rightarrow \text{REGRESSION} \rightarrow \text{CLASSIFICATION} \rightarrow \text{PRODUCT} \]

\textbf{FIGURE 2.6: Image Processing chain in Remote Sensing [74]}

In the previous sections, we have reviewed specific type of features that can distinguish the objects of interests in images.

This section is dealing with the methods to select them in order to get better classification results in a more efficient way.
Due to the curse of dimensionality, only essential features should be selected [75]. Features can be picked either by the filters such as correlation and mutual information [76] or by the wrappers.

Wrappers choose the features to minimize classification errors. SVM is one of the examples of wrappers and used in [77] and [78]. Moreover, these features are extracted with the help of PCA or the nonlinear methods such as locally linear embedding or isometric mapping [79].

### 2.4 Classification-Based Techniques

The last but not the least, image classification is a part of the framework in Figure 2.6. The purpose of classifying images is to identify features. The following table shows the recent research in the literature for the image classification.

**Classification Methods**

1. Unsupervised Classification Methods
   - Unsupervised change detection using split-based approach [80]
   - Unsupervised change detection using parcel-based approach [81]

2. Supervised Classification Methods
   - Neural Networks [82]
   - SVM [83] [84] [85] [86]

3. Semi-supervised Classification Methods
   - Graphs [87]
   - Cluster kernels [88]
Chapter 3

Problem Statement

3.1 Problem Definition

In this thesis, first, we focus on the topic of automated building detection. It can be considered as a sub-topic of image classification. The objective is to locate hundreds to tens of thousands of buildings in large-scale images automatically. They have relatively uniform appearance in terms of roofs, texture and shapes. Moreover, we extend our solution to the multi class detection problems. From satellite images, three regions can be observed: buildings, road networks and vegetation areas. These classes have some distinguished characteristics such as shape, texture and color features.

Although they have distinctive features, detection is still tricky due to density and complexity of the scene. Detection complexity changes with different factors.

Scene Types : Suburban, Urban, and Rural

Urban areas are far more difficult in detection problems than suburban and rural areas, since they refer to crowded cities, denser buildings, less vegetation areas and more complex road networks. On the other hand, suburban areas are perfect data for our task.
Scene Illumination  Illumination determines the amount of shadows captured by objects in the image, which makes the problem more challenging.

Scene Homogeneity  If the image has a wide scope of building sizes, road sizes and color of trees, then this is called non-homogeneous scene.

Image Resolution  The resolution is mostly low because satellite imagery is used.

We seek to design an efficient algorithm considering the above limitations. Our approach consists of the learning algorithm “AdaBoost”, and the idea of cascade classifiers.

3.2 Study Area and Data

Google Earth\textsuperscript{TM} is a great tool for viewing satellite image data from around the world. Although it does not match the quality of traditional high-resolution satellite images, it has the image quality that surpasses anything that is available at the free, consumer level. The other access data choices are QuickBird at 0.6m, IKONOS at 0.8m and SPOT-5 at 2.5m resolution. For our problem, it is enough to use Google Earth images as a data set.

There are many geographical locations that can be good candidate for building extraction or classification. One of the areas that we chose to get data is Ankara, Turkey because the buildings have similar properties in general and the area has normal density level. The other area is Dallas, TX. The following figure 3.1 shows some of the examples.
Figure 3.1: Some Examples of Google Earth Satellite Images
Chapter 4

Procedure

The proposed algorithm is based on Viola-Jones approach for Face Detection [1]. There are three key contributions, integral image representation, a learning algorithm Adaboost and combining classifiers in a cascade form. We have adopted their approach for our object detection problem for the medium or high resolution satellite images and also extended it to the multi-class case. The following steps outline the procedure in detail.

4.1 Data Preparation

The sample satellite images were obtained from Google Earth. There are more than 25 images with size of 256 by 256, labeled manually. Using ground truths of these images, building, road and vegetation parts are extracted. Then, the extracted parts are scaled to a base resolution of 24 by 30 pixels. Therefore, we have a total number of 350 building, 400 road, 700 vegetation and 600 background images in our complete data set. The set of images that we have mentioned can be observed in Figures 4.1, 4.3, 4.2 and 4.4. For the multi-class case, we have the total number of three classes as buildings, trees and road network. We do not take into account the background images since they are counted as negative images at each case in the algorithm. Then, using the images
corresponded to each class, one of the training and validation sets are obtained. The details of these sets are shown in Table 4.1. These numbers can be flexible and in the experiment part, we change them in a pattern to evaluate the performance of detection.

<table>
<thead>
<tr>
<th>Class</th>
<th>Label Name</th>
<th>Label No</th>
<th>Training</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>Building</td>
<td>1</td>
<td>300</td>
<td>50</td>
</tr>
<tr>
<td>Class 2</td>
<td>Vegetation</td>
<td>2</td>
<td>400</td>
<td>50</td>
</tr>
<tr>
<td>Class 3</td>
<td>Road</td>
<td>3</td>
<td>300</td>
<td>50</td>
</tr>
<tr>
<td>Class 4</td>
<td>Background</td>
<td>0</td>
<td>300</td>
<td>50</td>
</tr>
</tbody>
</table>
Figure 4.1: A Set of Building Images
Chapter 4. *Procedure*

Figure 4.2: A Set of Tree Images
A Set of Road Images (Class 3)

Figure 4.3: A Set of Road Images
Figure 4.4: A Set of Background Images
4.2 Features

The proposed object detection algorithm classifies images based on the value obtained from simple features [1]. Features provide the knowledge that is hard to obtain with pixels of finite set data [89]. Feature based systems are also computationally more efficient [1].

Three different feature types are used. The two of them are Haar-Like features with different shapes and colors. The third type is Gabor feature for texture analysis.

The Haar wavelets are natural set basis functions that encode differences in average intensities between different regions [90]. Moreover, the calculation of the Haar wavelets is made simple with the help of integral image.

4.2.1 The Integral Images

The method of using integral image is an easier way to compute rectangle features. The integral image value at point \((x, y)\) contains the sum of the pixels of top and left region including \(x\) and \(y\) [1]. (See Figure 4.5)

\[
ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')
\]  

(4.1)

where \(ii(x, y)\) stands for the integral image and \(i(x, y)\) refers the original image.

Any rectangular sum is computed using only four array references. This simplifies the computational cost. In the same manner, eight array references give the difference between two rectangular sums. Therefore the following equations show the relationships between rectangles A, B, C and D seen in Figure 4.6.
Figure 4.5: Representation of Integral Image

\[ ii(x_1, y_1) = \sum_{x', y' \in A} i(x', y') \] (4.2)

\[ ii(x_2, y_2) = \sum_{x', y' \in A \cup B} i(x', y') \] (4.3)

\[ ii(x_3, y_3) = \sum_{x', y' \in A \cup C} i(x', y') \] (4.4)

\[ ii(x_4, y_4) = \sum_{x', y' \in A \cup B \cup C \cup D} i(x', y') \] (4.5)

Figure 4.6: Calculation of rectangles using Integral Image


4.2.2 Feature Types

4.2.2.1 Corner Features

In Figure 4.7, the four different corner types are displayed. Given an integral image, the Haar-like features can be easily calculated with the area of the black and white rectangles.

\[
\begin{align*}
  f_1(S) &= |W_1| - |B_1|, \\
  f_2(S) &= |W_2| - |B_2|, \\
  f_3(S) &= |W_3| - |B_3|, \\
  f_4(S) &= |W_4| - |B_4|,
\end{align*}
\]

where \(|B_i| = \sum_{(x,y) \in B_i} |I(x,y)|\), \(|W_i| = \sum_{(x,y) \in W_i} |I(x,y)|\) and \(|S_i| = |B_i| + |W_i|\). In other words, \(|B|\) is the summation of all intensity values of pixels located in the black rectangle and \(|W|\) is the area under the white rectangle.

![Corner Features](image)

**Figure 4.7:** Corner Features

4.2.2.2 Rectangle Features

AdaBoost is an algorithm to extract the best features and learn the pattern of classes. If the algorithm is fed with better features, then the procedure would be faster. In our problem, Class 1 is composed of the rooftop images of buildings. Although there are some exceptions, their most distinguished properties are the color and their rectangle shapes. Therefore, using the rectangle shape feature is a must for our case.
In Figure 4.8, the five different rectangle features are displayed. Since color information is also one of the distinguished properties, we come up with the solution of combining color and rectangle features. The first five types of rectangle features are calculated by gray intensities. Then, the next features, \([f_{10}, f_{11}, f_{12}, f_{13}, f_{14}]\), and the last five features, \([f_{10}, f_{11}, f_{12}, f_{13}, f_{14}]\), are computed using hue and saturation components of the hsv color space respectively. Given an integral image, the Haar-like features can be easily calculated by the area of the black and white rectangles.

\[ f_i(S) = |W_i| - |B_i| \]

where

\[ |B_i| = \sum_{(x,y) \in B_i} |I(x,y)|, \quad |W_i| = \sum_{(x,y) \in W_i} |I(x,y)| \quad i \in [5...9] \]

\[ |B_i| = \sum_{(x,y) \in B_i} |Hue(x,y)|, \quad |W_i| = \sum_{(x,y) \in W_i} |Hue(x,y)| \quad i \in [10...14] \]

\[ |B_i| = \sum_{(x,y) \in B_i} |Sat(x,y)|, \quad |W_i| = \sum_{(x,y) \in W_i} |Sat(x,y)| \quad i \in [15...19] \]
4.2.2.3 Texture Features

Gabor Filter has been widely used in edge detection, invariant object recognition and compression [91]. Gabor filters present the best simultaneous localization of spatial and frequency information [92]. It is also appropriate for texture representation and discrimination. They are Gaussian kernel functions modulated by sinusoidal wave. 2D Gabor filter is defined with the assumption that $x$ and $y$ have the same standard deviations ($\sigma_x = \sigma_y = \sigma$) [91].

\[
\psi(z, \sigma_s, \theta_o) = \frac{1}{2\pi\sigma_s^2} \exp \left\{ -\frac{\|z\|^2}{2\sigma_s^2} \right\} \left( e^{i\pi \frac{K}{\sigma_s}} - e^{-\frac{K^2}{2}} \right) \quad (4.6)
\]

\[
(4.7)
\]

\[
z = (x', y'), \quad \left\{ \begin{array}{l}
x' = x\cos\theta_o + y\sin\theta_o \\
y' = -x\sin\theta_o + y\cos\theta_o
\end{array} \right.
\]

where $\sigma_s$ is the Gaussian deviation for the $s^{th}$ scale, $\theta_o$ is the filter angle for the $o^{th}$ orientation, $x, y$ are pixel positions in spatial domain, and $K$ is a filter bandwidth parameter.

Convolution between Gabor filters and images is defined as the Gabor representation of images.

\[
G_{s,o}(x, y) = I(x, y) \otimes \psi(x, y, \sigma_s, \theta_o) \quad (4.8)
\]

where $\otimes$ is the convolution symbol and $I(x, y)$ is the input image. The resultant image $G_{s,o}(x, y)$ is an array which includes $s \times o$ filtered image for each combination of scale and orientation.

Due to Gabor function’s maximum bandwidth limitation, Log-Gabor function proposed by Field is an alternative [93]. Log-Gabor filters can be constructed with arbitrary bandwidth and the bandwidth can be optimized to produce a filter with minimal spatial extent. Peter Kovesi’s Matlab code is used here [Kovesi]. The function Gabor
convolution has seven inputs and the filter bank outputs. Convolutions are done via FFT and all of the parameters are in frequency domain.

Parameters of 2D Log-Gabor Filter

The minimum and maximum frequencies: Wavelength of the smallest scale filter sets the maximum frequency. It is required to use more than the Nyquist wavelength, which is 2 pixels. In our case we set it as 3 pixels. On the other hand, minimum frequency is defined implicitly by the number of filter scales (nscale), the scaling between centre frequencies of successive filters (mult), and the maximum frequency. The relationship between them can be described as follows:

\[
\text{MaximumFrequency} = \frac{1}{\text{MinimumWavelength}}
\]

\[
\text{MinimumFrequency} = \frac{1}{\text{MaximumWavelength}}
\]

\[
\text{MaximumWavelength} = \text{MinimumWavelength} \times \text{mult}^{\text{nscale}} - 1
\]

The filter bandwidth: It is tuned by the ratio of standard deviation of the Gaussian in the log-frequency domain to the filter center frequency. The parameter name is \textit{sigmaOnf} in the original code. We set it to 0.55, results in a bandwidth of roughly 2 octaves.

The scaling between center frequencies of successive filters: It is required to compromise \textit{sigmaOnf} in order to reach desired bandwidth. Therefore we choose this scaling factor as 3.

The number of filter scales: It determines the number of scaling factor of the inputs and controlled by the parameter \textit{nscale} within the code. It is set as 4.

The number of filter orientations: It specifies the resolution of the orientation information obtained from the filters. 6 is fairly good value for \textit{norient}.

The angular spread of each filter: The angular overlap of the filter transfer functions is controlled by the ratio of the angular interval between filter orientations and the standard deviation of the angular Gaussian spreading function. Within
the code, this ratio is controlled by the parameter $d\Theta\text{OnSigma}$. A value of $d\Theta\text{OnSigma} = 0.5$ results in overlap that is needed to get even spectral coverage on our data set [Kovesi].

We have explained the definition of Gabor filter and its parameters in the previous paragraphs. Now it is time to describe the usage of integral Gabor-Haar transformation. Gabor transformation increases the size of image by $s$ scale and $o$ orientation. This means that given an image of size $N$, there is $so \times N$ dimensional feature vector. Only some parts of the feature vector are essential but the rest of these features are redundant. AdaBoost chooses the essential features, which leads to best classification results. According to Li et al. [92], the feature selection step can be fastened with the help of Haar-like features in extremely high dimensions. It is best to find the Gabor representation of the image first and then to compute the integral images and calculate the value of them using Haar-like feature representations, which are shown in Figure 4.9. The formal definition of calculations can be best described as follows:

$$
\begin{align*}
&f_{20}(S) = \frac{|A|}{|S|}, \quad f_{23}(S) = \frac{|E|}{|S|}, \\
&f_{21}(S) = \frac{|C|}{|S|}, \quad f_{24}(S) = \frac{|F|}{|S|}, \\
&f_{22}(S) = \frac{|A| + |C|}{|S|}, \quad f_{25}(S) = \frac{|G|}{|S|},
\end{align*}
$$

where $|A| = \sum_{(x,y) \in A} |G(x, y)|$ (i.e. the shaded area) and $S$ represents the whole region used for normalization to the range $[0, 1]$.

To sum up, for a given image which has size of $24 \times 30$, the total and the individual number of features are shown in table 4.2.
Figure 4.9: Haar-Like Features for Gabor Filters

Table 4.2: The number of features of each type

<table>
<thead>
<tr>
<th>Type</th>
<th>Image Space</th>
<th>Number of Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corner</td>
<td>Gray Scale</td>
<td>1919</td>
</tr>
<tr>
<td>Rectangle</td>
<td>Gray Scale</td>
<td>80</td>
</tr>
<tr>
<td>Rectangle</td>
<td>HSV Color Space-Hue</td>
<td>80</td>
</tr>
<tr>
<td>Rectangle</td>
<td>HSV Color Space-Sat</td>
<td>80</td>
</tr>
<tr>
<td>Gabor</td>
<td>CIE L,a,b-L channel</td>
<td>216</td>
</tr>
</tbody>
</table>

Total=2375


## 4.3 Learning Algorithm

For a given training data set and set of features, any machine learning algorithm can be employed in learning a classification. Boosting is one type of machine learning method, which performs supervised learning. The idea is: “strong classifier” can be created by linearly combining a number of “weak classifiers”. Weak Classifiers are acquired by response of the features from training data and have the detection rate, which is slightly better than the random guessing results. To raise the detection rate, it is required to utilize a machine learning approach.

AdaBoost, short for Adaptive Boosting, is the most popular boosting algorithm, which was introduced by Freund and Schapire in 1995 [94]. AdaBoost is also extended to multi-class classification problems and regression problems in [94] and [95]. In this thesis, binary classification is our first task and then, we advance our solution to multi class framework.

### 4.3.1 AdaBoost

The AdaBoost algorithm enlightens the many practical problems of boosting algorithms. It has two important modifications over the previous methods.

- using a weighted sample to focus learning on most difficult examples of training data.
- using a weighted vote in order to combine classifiers.

It is called AdaBoost because it is adjusted adaptively to the errors of weak hypothesis and returned by weak learn [94]. As in Viola and Jones paper [1], a variant of AdaBoost handles the two tasks; selecting the features and training the classifier. The pseudo code of the variant of AdaBoost is shown in Algorithm 4.1.

The weights of training set determine the probability of being selected for a feature and they are continuously updated in every iteration. If a training pattern is accurately
classified, then its chance of being used again in the next round is reduced. Since the wrongly classified examples’ weights are increased while we decrease the weights of correctly labeled examples. This is called reweighing approach. (See Figure 4.10) By this manner, it is not prone to over-fitting problem.

Algorithm 4.1 AdaBoost

Given example images \((x_1, y_1)(x_2, y_2)...(x_n, y_n)\) where \(y_i = 0, 1\) for negative and positive images respectively.

Initialize weights \(w_{1,i} = \frac{1}{2m} \cdot \frac{1}{2l}\) for \(y_i = 0, 1\) where \(m\) and \(l\) are the number of negatives and positives respectively \((m + l = n)\)

for \(t = 1\) to \(T\) do

1. Normalize the weights

\[
w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}
\] (4.9)

so that \(w_t\) is a probability distribution.

2. For each feature \(j\), train a single feature classifier \(h_j\). The error is evaluated with respect to \(w_t\)

\[
\epsilon_j = \sum_{i} w_{t,i} |h_j(x_i) - y_i|
\] (4.10)

3. Choose the classifier \(h_t\) with the lowest error \(\epsilon_t\).

4. Update the weights

\[
w_{t+1,i} = w_{t,i} \beta_t^{1 - \epsilon_t}
\] (4.11)

where \(e_i = 0\) if \(x_i\) is classified correctly and \(e_i = 1\), otherwise and \(\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}\)

end for

The final strong classifier is

\[
f(n) = \begin{cases} 
1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\
0 & \text{otherwise}
\end{cases}
\]

where \(\alpha_t = \log \frac{1}{\beta_t}\)
Figure 4.10: Reweighing Approach
The parameter $\alpha_t$ is the importance measure of the classifier. By rewriting the equations, the relationship between error and $\alpha_t$ can be described as follows

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$ (4.12)

It means that $\alpha_t \geq 0$ if $\epsilon_t \leq \frac{1}{2}$. This makes sense since we are dealing with the binary case which has only the labels of 0 and 1. If the classifier is trained for a feature which has $\alpha_t$ is greater than 0.5, like random guessing, then the feature is a good candidate for one of the Single Feature Classifier in the weak learning algorithm.

### 4.3.1.1 Training Error Analysis

At each iteration, the algorithm introduces a new hypothesis with error that is calculated by the equation of $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$. The error of final hypothesis is bounded by

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1 - \epsilon_t)}$$ (4.13)

where the final error is defined as $\epsilon = Pr[h_f(x) \neq y]$.

The equation 4.13 is proved by Freund and Schappire [94] in a following fashion:

$$\sum_{i}^{n} w_i^{t+1} = \sum_{i}^{n} w_i^{t} \beta_i^{1-\epsilon_t}$$ (4.14)

$$\leq \sum_{i}^{n} w_i^{t}(1 - (1 - \beta_t)(1 - \epsilon_t)))$$ (4.15)

$$= (\prod_{i=1}^{n} w_i^{t})(1 - (1 - \beta_t)(1 - \epsilon_t))$$ (4.16)
If we generalize the inequality over \( t = 1 \ldots T \), then we obtain

\[
\sum_{i}^{n} w_{i}^{T+1} = \prod_{t}^{T} (1 - (1 - \beta_{t})(1 - \epsilon_{t}))
\] (4.17)

The sum of the weights of the all samples is greater than the sum of the incorrect samples.

\[
\sum_{i}^{n} w_{i}^{T+1} \geq \sum_{i: h_{t}(x_{i}) \neq y_{i}} w_{i}^{T+1} \geq \epsilon_{t} \prod_{t}^{T} \beta_{t}^{1/2}
\] (4.18)

Combining equations 4.16 and 4.18, and substituting \( \beta_{t} = \frac{\epsilon_{t}}{1 - \epsilon_{t}} \)

\[
\epsilon \leq 2^{T} \prod_{t}^{T} \sqrt{\epsilon_{t}(1 - \epsilon_{t})}
\] (4.19)

Moreover error \( \epsilon_{t} \) can also be interpreted as \( 1/2 - \gamma_{t} \). Here \( \gamma \) compares the hypothesis with random guessing 1/2. If \( \gamma \) is greater than zero, then the hypothesis is better than the random guessing 1/2. Therefore the training error is redefined as;

\[
\prod_{t}^{T} 2\sqrt{\epsilon_{t}(1 - \epsilon_{t})} = \prod_{t}^{T} 2\sqrt{(1/2 - \gamma_{t})(1/2 + \gamma_{t})} = \prod_{t}^{T} \sqrt{(1 - 4\gamma_{t}^{2})} \leq \exp(-2 \sum_{t}^{T} \gamma_{t}^{2})
\] (4.20)

The equation 4.20 proves that for each weak hypothesis that is slightly better than random guessing (i.e. \( \gamma_{t} \geq 0 \)), the training error drops exponentially fast [24].

This completes the explanation of the binary classification algorithm. The next part is about to developing classification for more than one object.
4.3.1.2 Multiclass AdaBoost

Multi-class problems is more realistic than the binary classifications. Since the real world has mostly more than one object to differentiate. As we mentioned before, our second task is to distinguish the class of buildings, roads and trees. Boosting is a successful model for the two-class case. However, the AdaBoost algorithm can also be extended to multi class case. There are number of examples of the multi-class extensions of AdaBoost which we will explain briefly in this section. Our algorithm of multi-class extension is also described throughout this section.

Going from binary class classification to multi-class classification, multi-class algorithms and multiple two-class algorithms exist.

The most generalized algorithm for multi-class, called as AdaBoost.M1, is proposed by Freund and Schapire [94]. However, it fails if the weak classifier is not chosen appropriately. In the two-class AdaBoost, it is required that error of each classifier be less than 0.5 in order to have positive $\alpha$ and the assumption goes along with random guessing. When we make the problem $K$-class for $K > 2$, random guessing accuracy rate $(\frac{1}{K})$ is lower than the accuracy rate $0.5$ and it is harder to achieve required accuracy.

The algorithm that is called SAMME- Stage wise Additive Modeling tries to solve the previous problem with a Multiclass Exponential loss function [96]. In the previous method, $\alpha$ is calculated by the following equation;

$$\alpha_t = \frac{1 - \epsilon_t}{\epsilon_t}$$  \hfill (4.21)

SAMME added an extra term in calculating $\alpha$.

$$\alpha_t = \frac{1 - \epsilon_t}{\epsilon_t} + \log(K - 1)$$  \hfill (4.22)

Multi-class exponential loss function makes the algorithm equivalent to fitting a forward stage wise additive model. For the sake of positive $\alpha$, $(1 - \epsilon) > \frac{1}{K}$ is adequate.
There are also other multiclass AdaBoost methods that reduce the multi-class problem to multiple binary class problems. The first one, AdaBoost.MH, is proposed by Schapire and Singer [95] and it takes advantage of Hamming distance. The second algorithm, AdaBoost.M2 discriminates the correct label from the other labels.

In order to use the above algorithms, we need to modify the weak learner algorithms. Dietterich and Bakiri [97] proposed a method that can utilize any weak learner algorithms, error correcting output codes to train and then combine boosting with binary labeled data [95].

We have chosen the approach of splitting K-class problem to K two class problems. The reasons are

- It is easier to learn simple functions than complex functions.
- It is more efficient since the update of the weights is guided only by one output node. (As in binary case)
- The number of iterations is smaller in multiple two-class classifiers than the multiclass classifier, since it is hard to find a feature that gives the better discrimination, it keeps looking for a long time till it find it. Therefore processing time is very large for multi-class case.

After deciding to design multi two-class AdaBoost, we need to pick the optimal method for our case. Before delving into our results, we want to give some explanations of the multiple two class methods.

**One vs All (OA)** It can be also called as “One per Class” classification. In this method, K binary classifiers are constructed. The results are always a square matrix with size of $K \times K$. For K=3 case, the matrix is shown as

$$D= \begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix}$$

Multi-class and multi two class classifiers are contrasted in Figure 4.11.
One vs One (OO) In this method, we build \( K(K - 1)/2 \) pair wise classifiers. Then, the output matrix has a size of \( K \times K(K - 1)/2 \) which is shown in below. For \( K=3 \):

\[
D = \begin{pmatrix}
+1 & +1 & 0 \\
-1 & 0 & +1 \\
0 & -1 & -1 \\
\end{pmatrix}
\]

It has two main problems. First, the number of dichotomizers is \( O(K^2) \). Second, the dichotomizer is trained by only small set, with two classes, so the resultant variance is higher.

Error Correction Output Codes (ECOC) Machine learning problem is model after a communication problem using error correction codes [97]. Therefore misclassification over the classes is solved in a same manner as communication across noisy channel. Error correction output codes represented by the matrix, D. The length of a code equals the number of columns in D. The total number of classes is the number of row in D. A codeword is the defined by the row of the matrix, D.

For \( K=4 \);
D=
\begin{pmatrix}
+1 & +1 & +1 & +1 & +1 & +1 & +1 \\
-1 & -1 & -1 & -1 & +1 & +1 & +1 \\
-1 & -1 & +1 & +1 & -1 & -1 & +1 \\
-1 & +1 & -1 & +1 & -1 & +1 & -1 \\
\end{pmatrix}

In the error correction output code, we have up to $2^{(K-1)} - 1$ dichotomizers [98].

ECOC is successful only if the errors made in the individual bit positions are uncorrelated. Due to this reason, D matrix has the property of **Row Separation**, which means that each codeword should be separated by Hamming distance from each others. The other property is **Column Separation** in order to have neither identical nor complement columns which leads us large Hamming distance [97].

The separation of row and columns of D matrix is achieved for the class number $K \leq 5$. For $K = 3$ case, the matrix D is show below.

D=
\begin{pmatrix}
+1 & +1 & + \\
-1 & -1 & +1 \\
-1 & +1 & -1 \\
\end{pmatrix}

The total number of possible columns is $2^3 = 8$. Half of these columns are compliments of the rest and one of the columns is all zero or one. Therefore, the final matrix has only three columns, which is equivalent to what we have in one per class case. This explains why one per class method is chosen to perform our binary learning algorithm in multi-class case. It is good to note that for three-class case the number of dichotomizers is 3 for one per class, pair wise case and ECOC.

### 4.3.2 Cascade

One of major contributions of Viola and Jones in the paper “Robust Real-time Object Detection” is to combine strong classifiers in a cascade structure. The evolution is in the face detection field and is called “the cascade of classifiers”. In the paper [1], the construction of a cascade decreases computation time while increasing performance. In this thesis, we adopt their approach for our problem.
The inputs of the model are sub window images, which are test patterns needed to classify as positive or negative. Layers of the cascade examine the test data one after another. The examination process terminates in two ways; either it is rejected by one of the cascade layer or the final layer accepts it. Then, this corresponds to a decision of negative or positive respectively. (See Figure 4.12)

![Image: A cascade consists of layers.](image)

Figure 4.12: A cascade consists of layers.

In practice, in a single image, majority of sub-windows are negative. As such, cascade rejects most of the negative in the initial stages and will not continue on evaluating the rest of the stages. This decreases the computational time abruptly.

Cascade is build with a number of classification layers. AdaBoost construct the layers through training process. Each subsequent stage is trained by all of the positive images and only the false positive images of the preceding cascade. Therefore, the number of positive images of the training set is fixed while the negative images decrease in every stage to deal with harder examples.

There exist two important evaluation parameters for cascade of classifier, the detection rate and the false positive rate. Let the number of cascade layer be $N$, then the final false detection rate is calculated as;
Similarly the detection rate is described as;

\[ D = \prod_{i=1}^{N} d_i \quad (4.24) \]

The algorithm is directly taken from Viola and Jones study and given in detailed in Algorithm 4.2.
Algorithm 4.2 Building a Cascaded Classifier

**Inputs**

- $f =$ maximum acceptable false positive rate per layer
- $d =$ minimum acceptable detection rate per layer
- $F_{\text{target}} =$ overall false positive rate (desired)
- $P =$ set of positive images
- $N =$ set of negative images

**Initialize**

- $F_0 = 1.0$ and $D_0 = 1.0$
- $i = 0$

**while** $F_i > F_{\text{target}}$ **do**

- $i \leftarrow i + 1$
- $n_i = 0$
- $F_i = F_{i-1}$

**while** $F_i > f \times F_{i-1}$ **do**

- $n_i \leftarrow n_i + 1$
  - Train a classifier with $n_i$ features, $P$ and $N$ using AdaBoost
  - Evaluate the current cascade on a validation set and calculate $F_i$ and $D_i$
  - **if** $D_i < d \times D_{i-1}$ and $F_i < f \times F_{i-1}$ **then**
    - Decrease threshold for the $i^{th}$ classifier until the current cascade has desired detection rate
  **end if**
- $N \leftarrow 0$
- **if** $F_i > F_{\text{target}}$ **then**
  - Evaluate the current cascade on the set of negative images
  - Put any false detections into set $N$
**end if**

**end while**

**end while**
Chapter 5

Experiments and Results

5.1 Method of Evaluation

Since we have a finite number of data sets, cross validation is a good method to evaluate the performance of the model. The cross validation method is divided into three parts in terms of their computation times and partition of the data set.

5.1.1 Holdout Method

The holdout method is the easiest way of cross validation. The data set is separated into two sets, which are called the training set and the testing set. The learning algorithm decides the classifier using the training set, then the classifier calculates the values for the data in the testing set. It is important that the classifier should not see the test dataset while training process. The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model. (See Figure 5.1)

The holdout method is usually preferable to the other two cross validation methods since the number of repetitions to evaluate the learning algorithm is zero. However, its evaluation may have a high variance, since it strongly depends on the separation of data sets.
5.1.2 K-fold Cross Validation

K-fold cross validation is one way to avoid the high variance problem that we encounter in the holdout method. The data set is divided into $K$ subsets, and the holdout method is repeated $K$ times. (See Figure 5.2) At each round, one of the $K$ subsets is left for the test set and the other $K - 1$ subsets are put together to form a training set. Then, the average error across all $K$ trials is computed.

The average error

$$E_{avg} = \frac{1}{K} \sum_{i=1}^{K} \epsilon_i$$  \hspace{1cm} (5.1)

By this way, the matter of data set division is less important. Every point has to be tested once and trained $K - 1$ times. The variance is proportional to $K$ value. The higher $K$ values are, the less variance the estimated result has. However, processing time is getting larger and larger. If we consider that the processing time needed for the training process is relatively high, the $K$ times of the process is mostly undesirable.

5.1.3 Leave-One-Out Cross Validation

Leave-one-out cross validation is an extreme case of K-fold cross validation where $K$ is equal to $N$, the number of data points in the set. In other words, the learning algorithm is trained by $N - 1$ images; the remaining one is the test set and the classifier is evaluated by only this one image. (See Figure 5.3)
The average error

\[ E_{avg} = \frac{1}{N} \sum_{i=1}^{N} \epsilon_i \]  

(5.2)

where \( N \) is the number of data.

The evaluation given by leave-one-out cross validation error (LOO-XVE) is perfect, but at first pass it seems very expensive to compute.

After considering all the details that we described above, we decided not to apply leave-one-out cross validation method. Our data set has more than 2000 images and leave-one-out cross validation is not an optimal way to evaluate our classifier due to computation time. Therefore, we apply the holdout and k-cross validation methods in order to show our classifier’s performance. As a \( K \) value, 10 is chosen. In the section “Evaluation Results”, we will give the results of the experiments in detail.

### 5.2 Performance Criteria

The performance evaluation for classification is carried out over the image sub-windows. For this purpose the comparison of the ground truth data with the decision of the
Adaboost Algorithm for each image block is needed. As we mentioned before, ground truth data were labeled as true manually when more than half of the pixels of sub-windows belongs to that class of object.

The performance equations are the key concept to evaluation of the algorithm and defined by labeled ground truth data and the decision made by the AdaBoost algorithm. The most popular one is Confusion Matrix. The binary case of the confusion matrix is shown in Table 5.1. It contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix [99].

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negatives</td>
<td>A</td>
</tr>
<tr>
<td>Positives</td>
<td>C</td>
</tr>
</tbody>
</table>

\[ \text{Accuracy (AC)} = \frac{A+D}{A+B+C+D} \]

\[ \text{Recall or True Positive Rate (TP)} = \frac{D}{C+D} \]
\[\text{False Positive Rate (FP)} = \frac{B}{A+B}\]
\[\text{True Negative Rate (TN)} = \frac{A}{A+B}\]
\[\text{False Negative Rate (FN)} = \frac{C}{C+D}\]
\[\text{Precision (P)} = \frac{D}{B+D}\]

5.3 Evaluation Results

5.3.1 Two Class Classification Results

There are three binary classes in our problem. These are classes of buildings, trees and roads. We first evaluate each class independently and in the multi class section, we show their result together.

5.3.1.1 Building Detection

The data set is divided as in the table 5.2 using holdout method.

| Table 5.2: Training, Test and Validation Set for Building Classifier-Holdout Method |
|----------------------------------|------------------|------------------|
|                                 | Test | Training | Validation |
| Building Images                 | 150  | 150      | 50           |
| Non-building Images             | 1000 | 1000     | 150          |

The resultant classifier is a seven-stage cascade classifier, which includes a total number of 77 features. The confusion matrix is:

| Table 5.3: The Resultant Confusion Matrix for Building Classifier-Holdout Method |
|----------------------------------|------------------|------------------|
| Actual                           | Predicted        |                  |
|                                  | Negatives Positives |
| Negatives                       | 995              | 5               |
| Positives                       | 3                | 147             |
The evaluation parameters are calculated and displayed in table 5.4.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy (AC)</strong></td>
<td>0.9930</td>
</tr>
<tr>
<td><strong>Recall or True Positive Rate (TP)</strong></td>
<td>0.98</td>
</tr>
<tr>
<td><strong>False Positive Rate (FP)</strong></td>
<td>0.005</td>
</tr>
<tr>
<td><strong>Precision (P)</strong></td>
<td>0.967</td>
</tr>
</tbody>
</table>

A Receiver Operating Characteristics (ROC) curve shows the performance of our detector. To create the ROC curve, the threshold of the strong classifier of the cascade has been adjusted from -20 to 20 where the threshold is usually between 0 and 1.

As we mentioned before, the 10-fold cross validation is also used for the evaluation of the classifier. The whole data set is divided to ten fold, meaning only one of them is for testing set, and the rest is the training sets. In every round the cascade algorithm is
trained with the selected training set, we obtain a classifier. With 10 rounds, we have
total of ten classifiers. The performance is evaluated separately for each classifiers and
the result is the average of them. ROC curve is also calculated by changing threshold
from \(-10\) to 10 is displayed in Figure 5.5. The accuracies of the detector at every stage
and the average accuracy of the classifier are;

\[
\text{Accuracy} = \left( 0.945 \ 0.975 \ 1 \ 1 \ 0.995 \ 1 \ 0.99 \ 0.985 \ 0.995 \ 1 \right)
\]

\[
AC_{\text{avg}} = \frac{\sum_{i=1}^{10} AC_i}{10} = 0.9885 \quad (5.3)
\]

![Figure 5.5: ROC curves for Building Classifier-10-fold cross validation method](image)
5.3.1.2 Vegetation Area Detection

Using the holdout method, the data set is divided as in the Table 5.5.

<table>
<thead>
<tr>
<th>Vegetation Images</th>
<th>Test</th>
<th>Training</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>350</td>
<td>350</td>
<td>50</td>
</tr>
<tr>
<td>Non-vegetation Images</td>
<td>800</td>
<td>800</td>
<td>150</td>
</tr>
</tbody>
</table>

The resultant classifier is six-stage cascade, which includes a total number of 123 features. The confusion matrix is:

<table>
<thead>
<tr>
<th>Predicted Negatives</th>
<th>Predicted Positives</th>
<th>Actual Negatives</th>
<th>Actual Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negatives</td>
<td>769</td>
<td>31</td>
<td>82</td>
</tr>
<tr>
<td>Positives</td>
<td>268</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The evaluation parameters are calculated and displayed in table 5.7.

<table>
<thead>
<tr>
<th>Performance of Vegetation Classifier- Holdout Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (AC)</td>
</tr>
<tr>
<td>Recall or True Positive Rate (TP)</td>
</tr>
<tr>
<td>False Positive Rate (FP)</td>
</tr>
<tr>
<td>Precision (P)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0.9017</td>
</tr>
<tr>
<td>0.7657</td>
</tr>
<tr>
<td>0.0387</td>
</tr>
<tr>
<td>0.8963</td>
</tr>
</tbody>
</table>

Furthermore, the 10-fold cross validation results are also calculated. The accuracy is as follows:

\[
\text{Accuracy} = \left( \begin{array}{cccccccc}
1 & 0.86 & 0.785 & 0.85 & 0.655 & 1 & 0.995 & 0.9 & 0.96 & 0.99 \\
\end{array} \right)
\]

\[
AC_{\text{avg}} = \frac{\sum_{i=1}^{10} AC_{i}}{10} = 0.8995 \quad (5.4)
\]
5.3.1.3 Road Network Detection

Using holdout method, the data set is divided as in the table 5.8.

<table>
<thead>
<tr>
<th></th>
<th>Test</th>
<th>Training</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road Images</td>
<td>200</td>
<td>200</td>
<td>50</td>
</tr>
<tr>
<td>Non-Road Images</td>
<td>950</td>
<td>950</td>
<td>150</td>
</tr>
</tbody>
</table>

The resultant classifier is a nine-stage cascade, which includes a total number of 160 features. The confusion matrix is:

The evaluation parameters are calculated and displayed in table 5.10.
Figure 5.7: ROC curves for Tree Classifier - 10-fold cross validation method

Table 5.9: The Resultant Confusion Matrix for Road Classifier-Holdout Method

<table>
<thead>
<tr>
<th></th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Negatives</td>
</tr>
<tr>
<td>Actual</td>
<td>Negatives</td>
</tr>
<tr>
<td></td>
<td>Positives</td>
</tr>
</tbody>
</table>

The 10-fold cross validation results are shown as follows. The accuracy matrix describes the confusion matrix parameters while Figure 5.9 shows the Receiver Operating Characteristics curve of the algorithm. Moreover, the accuracy is calculated by averaging the accuracies of each step.

\[
\text{Accuracy} = \begin{pmatrix}
1 & 1 & 1 & 0.99 & 0.955 & 0.695 & 0.705 & 0.98 & 0.985 & 0.89
\end{pmatrix}
\]
Table 5.10: Performance of Road Classifier- Holdout Method

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (AC)</td>
<td>0.9339</td>
</tr>
<tr>
<td>Recall or True Positive Rate (TP)</td>
<td>0.825</td>
</tr>
<tr>
<td>False Positive Rate (FP)</td>
<td>0.043</td>
</tr>
<tr>
<td>Precision (P)</td>
<td>0.8009</td>
</tr>
</tbody>
</table>

\[ AC_{avg} = \frac{\sum_{i=1}^{10} AC_i}{10} = 0.92 \]

5.3.2 Multi Class Classification Results

In most pattern recognition problems, the classes are assumed to be mutually exclusive; an instance only belongs to one class at a time. The classifier is required not to find
Figure 5.9: ROC curves for Road Classifier - 10-fold cross validation method

overlapped or multi-label results for each instance. In this experiment, we combine all
the previous binary classifiers to construct a multi-class classifier. As a combination
rule, we chose one-vs-all.

The way to evaluate the performance at the multi-class classifier is different than the
binary one. There is no trade off between false positive rates and true positive rates.
Instead of ROC curves, it is common to use precision - recall curves for the multi-
class classifier performance evaluation. The precision-recall curve is plotted in Figure
5.10 and displays the performance of multi-class classifier. This graph proves that our
multi-class classifier has very high performance, since the curve is close to upper-right
corner.

The confusion matrix is calculated as in table 5.11. The bold numbers in the matrix
corresponds to true positives while the rest are the false positives. It is good to note
that the total number of samples is greater than what we have in confusion matrix. The difference is the samples that are not classified.

Our ultimate aim was to obtain good multi-class classifier, which detects three classes: buildings, trees and roads. We achieved this task by obtaining the multi-class classifier and showed its performance by plotting the precision-recall graph. In the next sections, we will discuss the classifier and its structure in detail.

<table>
<thead>
<tr>
<th>Table 5.11: Confusion Matrix for Multi Class-Holdout Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class 1</td>
</tr>
<tr>
<td>Expected Class 1</td>
</tr>
<tr>
<td>Class 2</td>
</tr>
<tr>
<td>Class 3</td>
</tr>
</tbody>
</table>

The parameter to obtain curve is defined as:

\[
Precision = \frac{\text{ObjectsCorrectlyClassified}}{\text{ObjectsCorrectlyClassified} + \text{MisclassifiedObjects}} \tag{5.6}
\]

\[
Precision = \frac{145+279+165}{145+279+165+1+1+9} = 0.98166 \tag{5.7}
\]

\[
Recall = \frac{\text{ObjectsCorrectlyClassified}}{\text{ObjectsCorrectlyClassified} + \text{UnclassifiedObjects}} \tag{5.9}
\]

\[
Recall = \frac{145+279+165}{145+279+165+(150-145)+(350-279)+(200-165)} = 0.85486 \tag{5.10}
\]
5.4 Learning Discussion

The process of obtaining cascade structure were described in “Cascade” section in detail. Now we examine the resultant cascades in every aspect. Every stage has one strong classifier, which is a combination of different weak classifiers. Moreover, it has the false positive rate and the detection rate for that stage. If the FP and TP is in the desired region, the algorithm stops, otherwise, it will continue to add more layers to the cascade. As we stated before, every strong classifier is made up of weak classifiers, which consist of different number of features, that are the best experts for our training set. In Figure 5.11, the big picture of the form of cascade can be observed.
Figure 5.11: Elements of Cascade Structure
5.4.1 Feature Analysis

AdaBoost has chosen best weak classifier for the desired true positive and false positive rates. We can examine which features are preferred and prove that they are crucial for our detection process.

Every feature is defined as a vector. The first two elements of the vector represent the information in terms of initial y and x location on a sub-window of size 24 x 30, and the third element is a number, which refers to the type of feature.

\[ F = (y \ x \ \text{type}) \]

Let us to summarize the types of features and their corresponding label numbers throughout the code.

<table>
<thead>
<tr>
<th>Types of Features</th>
<th>Label Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corner</td>
<td>1 - 4</td>
</tr>
<tr>
<td>Rectangle on Gray</td>
<td>5 - 9</td>
</tr>
<tr>
<td>Rectangle on Hue</td>
<td>10 - 14</td>
</tr>
<tr>
<td>Rectangle on Saturation</td>
<td>15 - 19</td>
</tr>
<tr>
<td>Gabor</td>
<td>20 - 33</td>
</tr>
</tbody>
</table>

There are also two other important parameters, which are the threshold and the polarity. For a sub window, we calculate these values using the feature vector and compare this with a threshold. Based on the polarity, we decide whether it belongs to a certain class or not. If the polarity is one, then we assign the image to the class; if value is less than the threshold. Algorithm 5.1 explains the decision process.

It is time to display the features selected by AdaBoost for every cascade (See Figure 5.12) and draw the histogram of the distribution for these features. (See Figure 5.13) By looking at the distribution of these features, there are a total number of 10 corner features \((\text{type} = 4)\), 14 features for the rectangle calculated by gray intensities \((\text{type} = 8)\), 16 features for hue values on rectangle \((\text{type} = 13)\), 26 features for rectangle on saturation intensity values \((\text{type} = 18)\) and 11 Gabor features \((\text{type} = 31)\). Therefore,
Algorithm 5.1 Decision of Classifier (Threshold and Polarity)

**Inputs** Integral Image(sub-window) and Feature Vector, Threshold, Polarity

Calculate the value of the image for that feature

if \( \text{polarity} > 0 \) then

  if \( \text{value} \leq \text{threshold} \) then
  
  Score is 1 (that image is a good candidate for the class)
  
  else
  
  Score is 0 (the image is not belonging to that class)
  
  end if

else

  if \( \text{value} \geq \text{threshold} \) then
  
  Score is 1 (that image is a good candidate for the class)
  
  else
  
  Score is 0 (the image is not belonging to that class)
  
  end if

end if

we can conclude that our final classifier has a cascade, which consists of the features from each type.

The distribution of feature types in the Tree and Road Detector can be observed in Figures 5.14 and 5.15 respectively.

### 5.4.2 Cascade Structure Discussion

In chapter 4, it is stated that the cascade structure speed up the detection processes. In this section, we will prove this idea. With the help of AdaBoost, the dimensions of features are reduced and 77 features are selected for the building detection. Then, we compare the cascade structure classifier to 77-feature classifier i.e. monolithic classifier. Figure 5.16 displays the ROC curves for both classifiers. Using the same number of classifier, we have obtained very similar curves. Therefore, we can conclude that both of these learning procedures are similar in terms of performance and one cannot claim that cascade structure is better than monolithic classifier. However, the processing times in Figure 5.16 shows the importance of cascade structure. When there is no object of interest in the image, cascade learner rejects it in the first or second layer
Figure 5.12: Elements of Cascade Structure for Building Detector
Chapter 5. *Experiments and Results*

**Figure 5.13**: Distribution of Feature Types in Building Detector

**Figure 5.14**: Distribution of Feature Types in Vegetation Area Detector
and will not continue to the evaluation. On the other hand, for n-feature classifier, the processing time is always the same even whether or not there is an object.

Moreover, for the tree detection, we compare cascade with 123-feature classifier. The results are shown in Figure 5.17. Furthermore, we have 160 features for the road detector and the comparison of the cascade with 160-feature classifier is in Figure 5.18.

5.5 Experiments

Until now, we have used our data set images which are the sub-images size of $24 \times 30$ containing very good examples of buildings, trees and roads. It is now time to test the classifiers with the images size of $256 \times 256$. These images may include a skew building, trees occluded on road etc. In Figures 5.19, 5.20 and 5.21, we show the detected classes on the real images. The red rectangles represent the detected building, green and blue colors are for trees and roads respectively.
Figure 5.16: Comparison with Detector Cascade to Monolithic Classifier-Building Detection (a) ROC curves (b) Processing Times
Chapter 5. *Experiments and Results*

Figure 5.17: Comparison with Detector Cascade to Monolithic Classifier-Vegetation Detection (a) ROC curves (b) Processing Times
Figure 5.18: Comparison with Detector Cascade to Monolithic Classifier-Road Detection (a) ROC curves (b) Processing Times
Figure 5.19: Examples of a Multi-Class AdaBoost Based Detection - 1
Figure 5.20: Examples of a Multi-Class AdaBoost Based Detection - 2
Figure 5.21: Examples of a Multi-Class AdaBoost Based Detection - 3
Chapter 6

Conclusion

6.1 Summary

The goal of this research is to develop an algorithm to extract the objects from the medium-resolution satellite images. We focus on detection of buildings, roads and trees on urban and suburban areas.

First of all, we have started by reviewing the literature. Features are the key properties in image processing framework. It is important to define which features are used and also how many of the features are crucial for detection of objects. The answers of these questions leads us two important concepts: the feature extraction methods and qualitative selection techniques. Therefore, we have classified extraction methods as edge-based, texture-based and classification-based. The first two groups can be combined to get better results in classification-based feature extraction methods.

Then, it is required to define the problem clearly. Our ultimate aim is to detect buildings, roads and trees from the satellite images in a completely automatic way.

There are number of commercial sources of satellite images such as IKONOS, Quickbird etc. Although they provide higher resolution than Google Earth offers, Google Earth is the free way of using satellite images. After deciding the source of images, the next step
is to choose areas of interest. Urban areas are more difficult than suburban areas since
they are crowded with buildings and roads and have less vegetation areas, which are
occluded by the buildings. In this research, we did not limit our solution to suburban
areas; we also test the algorithm in urban areas. Images from downtown Ankara and
from Dallas are used. They are all color images.

Among the several approaches, we present the method, which employs learning algo-
rrithm to learn the objects and detect them. We adopt the idea of [1] and combine the
idea with texture-based features as well. The procedure is addressed in Chapter 4. The
integral image concept provides the efficient way to extract shape and texture features.
Among all the features, AdaBoost chooses only the ones that are proved to be most
promising for classification. Then, the cascade structure is developed. Our final solution
consists of three binary cascade structured classifiers. They are combined to get multi
class classifier for remote sensing images. In Chapter 5, the classifiers are evaluated
and their performance is reported. The building classifier is the most successful one. It
has 99\% of accuracy with true positive rate (TP) of 98\% and false positive rate (FP)
of 0.5\%. The next classifier is the road detector. The accuracy of road classifier is 93\%
with 82\% (TP) and 4\% (FP). The final binary classifier is the vegetation classifier. It
has the accuracy of 90\% with 76\% (TP) and 3\% (FP).

Finally, the overall system, which is multi-class classifier, has the high performance. For
the multi class case, precision and recalls parameters are examined instead of accuracy.
The precision of it is 0.98\% while it has 0.85\% recall rate. The curve that describes
precision and recall relation proves how the classifier works well. It is very close to
the best curve for precision and recall. This system may be used to generate maps by
labeling buildings, roads and trees.

6.2 Future Work

Although an automated extraction of buildings, roads and trees have been developed
in this thesis, there are a number of areas in which future research, based on the results
presented here, can be explored.
First of all, the use of unsupervised learning can be explored. Thus, we do not need to label data and procedure can be done faster. Since the combination of AdaBoost and cascade provides fast processing during classification, the real time applications of the detection process can be easily developed. The maps of the interested areas may be updated by this way.

Moreover, data set can be enlarged by different cities. Then, our classifier learns the type of buildings and road networks in those cities as well. Furthermore, we can also add a post-processing step to the existing algorithm. This step provides information such as area of detected buildings, density of buildings, the length of the road networks etc. Information obtained by post processing step offers to the ability on commenting the type of area such as urban, suburban, farm etc.

Finally, the generation of 3D maps is another task for the remote sensing imaging framework. It is impossible to generate 3D map using single image, however by accessing to multiple images for the same area, the depth information can be extracted.
Appendix A

Sensors in Remote Sensing Systems

The explanation of sensor types and their usage in remote sensing systems are obtained from [6]. In this Appendix, we will describe them briefly. Remote sensing systems use active and passive sensors according to energy usage.

A.1 Passive Sensors

Passive sensors detect energy when the naturally occurring energy is available. This energy can be naturally emitted, reflected and transmitted. Since there is no reflected energy available from sun, reflected energy only takes place during daytime. Moreover naturally passive sensors can use emitted energy such as thermal and infrared day and night.

- **Photographic Cameras** are passive optical sensors that use a lens (or system of lenses collectively referred to as the optics) to form an image at the focal plane.

- **Multispectral scanning (MSS)** has a sensor with a narrow field of view (i.e. IFOV) that sweeps over the terrain to build up and produce a two-dimensional image of the surface.
• **Weather Satellites/Sensors** have fairly coarse spatial resolution (when compared to systems for observing land) and provide large areal coverage.

• **Land Observation Satellites/Sensors** Some examples of Land Observation Satellites include:

  - **LANDSAT** A number of sensors have been on board the Landsat series of satellites, including the Return Beam Vidicon (RBV) camera systems, the MultiSpectral Scanner (MSS) systems, and the Thematic Mapper (TM). The most popular instrument in the early days of Landsat was the MultiSpectral Scanner (MSS) and later the Thematic Mapper (TM).

  - **SPOT** (Systme Pour l’Observation de la Terre) is a series of Earth observation imaging satellites designed and launched by CNES (Centre National d’tudes Spatiales) of France, with support from Sweden and Belgium. The SPOT satellites each have twin high-resolution visible (HRV) imaging systems, which can be operated independently and simultaneously. Each HRV is
capable of sensing either in a high spatial resolution single-channel panchromatic (PLA) mode, or a coarser spatial resolution three-channel multispectral (MLA) mode.

- IRS The Indian Remote Sensing (IRS) satellite series, combines features from both the Landsat MSS/TM sensors and the SPOT HRV sensor.

- **Video** Cameras used for video recording measure radiation in the visible, near infrared, and sometimes mid-infrared portions of the EM spectrum. The image data are recorded onto cassette, and can be viewed immediately.

- **Forward Looking InfraRed (FLIR)** systems operate in a similar manner to across-track thermal imaging sensors, but provide an oblique rather than nadir perspective of the Earth’s surface.

### A.2 Active Sensors

Active sensors provide their own energy source for illumination. The sensor emits radiation, which is directed toward the target to be investigated. The radiation reflected from that target is detected and measured by the sensor. Advantages for active sensors include the ability to obtain measurements anytime, regardless of the time of day or season. Active sensors can be used for examining wavelengths that are not sufficiently provided by the sun, such as microwaves, or to better control the way a target is illuminated. However, active systems require the generation of a fairly large amount of energy to adequately illuminate targets.

- **Synthetic Aperture Radar (SAR)** It is an active microwave instrument, producing high-resolution imagery of the Earth’s surface in all weather. It generates its own illumination of the scene to be captured, like camera with flash. Both intensity and phase of the reflected light, which lead us to a high sensitivity to texture and some three-dimensional capabilities, can be measured by SAR.
**Figure A.2: Active Sensing**

- **Laser fluorosensor** Laser fluorosensors illuminate the target with a specific wavelength of radiation and are capable of detecting multiple wavelengths of fluoresced radiation. This technology has been proven for ocean applications, such as chlorophyll mapping, and pollutant detection, particularly for naturally occurring and accidental oil slicks.

- **Light Detection And Ranging (LIDAR)** LIDAR is the other active imaging technology. Pulses of laser light are emitted from the sensor and energy reflected from a target is detected. The time required for the energy to reach the target and return to the sensor determines the distance between the two. LIDAR is used effectively for measuring heights of features, such as forest canopy height relative to the ground surface, and water depth relative to the water surface (laser profilometer). LIDAR is also used in atmospheric studies to examine the particle content of various layers of the Earth’s atmosphere and acquire air density readings and monitor air currents.
• Radio Detection And Ranging (RADAR) RADAR systems provide their own source of electromagnetic energy. Active radar sensors, whether airborne or space borne, emit microwave radiation in a series of pulses from an antenna, looking obliquely at the surface perpendicular to the direction of motion. When the energy reaches the target, some of the energy is reflected back towards the sensor. This backscattered microwave radiation is detected, measured, and timed. The time required for the energy to travel to the target and return back to the sensor determines the distance or range to the target. By recording the range and magnitude of the energy reflected from all targets as the system passes by, a two-dimensional image of the surface can be produced. Because RADAR provides its own energy source, images can be acquired day or night. Also, microwave energy is able to penetrate through clouds and most rain, making it an all-weather sensor.
Appendix B

User Manual for the MATLAB Code

This manual describes the usage of MATLAB GUI program and its source code, and is organized as follows: First part speaks about all available functions. Then we give an example script to obtain building classifier and to test cascade structures performance.

B.1 Functions

B.1.1 Functions to Read Data and Prepare Data Set

ReadImages : It reads the image from text file

PrepareDataSet : It prepares data set for cascade structure and AdaBoost

B.1.2 Functions to Building a Cascade Classifier

ExtractFeatures : It extracts all features

BuildCascade : It produces the cascade of classifiers

85
AdaBoost : It implements the algorithm of two class Adaptive Boosting

EvaluateStrongClassifierSingleImage : It evaluates the strong classifier of adaboost on a single image;

EvaluateWeakClassifierSingleImage : It calculates the output of weak classifier on the single image

EvaluateCascadeSingleImage : It evaluates the decision of the cascade on a single image

B.1.3 Functions to Test Classifier

DetectionResult : It takes input sample image and gives the detection results

TargetDetect : It runs under the function “DetectionResult”. This function scans image and gives a binary matrix according to classification results

B.2 Sample Program

The “matlabgui” program has two important tasks (see Figure B.1). As an example, we first train Building Footprint Classifier and then it is tested using a sample image.

B.2.1 Training

1. Fill all the parameters. In the figure B.2, the default values are shown.

2. Press the “Build a Cascade of Classifier” button. Then, it is required to do the following four steps

   (a) Select the list of positive samples as in Figure B.4. The list should also include the path (see Figure B.3).
(b) Select the list of negative samples as in Figure B.5. The list should also include the path.

(c) Then save the workspace as in Figure B.6.

This concludes the training part. In the next section, we will describe the test part.

### B.2.2 Testing

After pressing “Target Detect” button, it is require to provide the following steps

1. Select the test image.
Figure B.2: Training Parameters

Figure B.3:
2. Load the .mat file which contains classifier and other parameters. (Figure B.8)

The final result will be shown as in Figure B.9
Figure B.5:
Figure B.6:
Figure B.7:
Figure B.8:
To Test a Cascade of Classifiers

1) Press "Target Detect" button
   a. Choose an image to detect
   b. Load mat file which includes cascade and other parameters

Figure B.9:
Bibliography


