



UNIVERSITY OF NAIROBI
SCHOOL OF COMPUTING AND INFORMATICS

A TEXT RECOGNITION SYSTEM FOR READING METERS

BY

PASCAL OUMA NYAPOTO

P58/63836/2011

SUPERVISOR

PROF. PETER WAIGANJO

September 2013

A research report submitted in partial fulfillment for the requirements of Master of Science in
Computer Science.

DECLARATION

The project presented in this report is my original work and has not been presented for any other university award.

Signature _____

Date _____

PASCAL OUMA NYAPOTO (P58/63836/2011)

This project has been submitted in partial fulfillment of the requirements of the Master of Science in Computer Science of the University of Nairobi with my approval as the University supervisor.

Signature _____

Date _____

PROF. PETER WAIGANJO WAGACHA

DEDICATION

To my wife Margaret. To my children Joseph Francis and Giovanni Maria. Thank you for your love and dedication.

To the late Dr. Geoffrey Williams Griffin. Your loving and generous heart has been for many a source of inspiration, hope and faith.

ACKNOWLEDGEMENT

I would like to acknowledge the many individuals who have supported me in one way or another during the course of this project. I would like to thank my supervisor, Prof. Peter Waiganjo, for his kindness, direction and encouragement which enabled me to complete the project. I would also like to thank all the faculty at SCI, especially the project coordinator, Mr. Evans Miriti. Finally, I would like to thank Jomo Kenyatta University of Agriculture and Technology for their generous sponsorship for the Masters course.

ABSTRACT

This study gives a solution for automatic reading of meters by taking digital images of the meters and using machine learning techniques to automatically read the meters. A dataset of positive examples, negative examples and test digital images is collected and used to train a classifier using the Adaboost learning algorithm. Positive examples are digital images of meters collected randomly from manually read meters and negative examples are any digital images without meters taken randomly using a digital camera, collected from past images or harvested on the internet. The test data is a subset of the positive and negative examples, and is not used in the algorithm training process. The positive and test images are manually labelled with the regions that contain meters on those images and a description file is created with the coordinates of meters in the images. The study uses OpenCV libraries for the classifier training and validation. The results of the the best performance test on the test data from the Adaboost training sessions is 75.4% recall with 19.4% false positive rate. This classifier is then used in a prototype that reads meters from digital images. Once the meter region is detected on an image, template matching techniques are used to locate the meter reading. The meter reading region is then binarized and presented to an OCR engine for text reading. It is this reading that can then be transmitted to utility providers for customer billing. Various integrity checks are proposed to ensure the transmitted data is accurate to avoid losses to utility providers and customers. The study presents this prototype to demonstrate the ability to use current technologies to solve common problems within society.

LIST OF FIGURES

Figure 1: A reproduction of the Cabioc, Gerardo and Byun (2011) System Architecture.....	8
Figure 2: A Greyscale Image Of A Water Meter, 160x160 Pixels	10
Figure 3: Water Meter Image After a Sobel Filter Operation.....	11
Figure 4: Training Set For The Family Car Class. (Alpaydin, 2010, P. 22).....	12
Figure 5: Hypothesis Class Example. (Alpaydin, 2010, P. 23).....	13
Figure 6: Adaboost Learning Algorithm Pseudocode. (Freund And Schapire, 1999, P. 2).....	16
Figure 7: Architecture of a Text Information Extraction System (Jung Kim and Jain, 2004).....	18
Figure 8: Positive Example Image Without Bounds (Left), With Bounds (Right).....	23
Figure 9: Positive Examples Description List.....	24
Figure 10. Rectangle Features. (Viola And Jones, 2004, P. 2)	25
Figure 11: Integral Image. (Viola And Jones, 2004, P. 3).	25
Figure 12: Cascade Of Classifiers. (Viola And Jones, 2004, P. 5)	26
Figure 13: Meter Reading System Context Diagram.....	30
Figure 14: Meter Reading System Use Case Diagram.....	31
Figure 15: Utility Provider Entity Relationship Diagram	32
Figure 16: Entity Relationship Diagram for Customer Device Database	32
Figure 17: Meter Reading System Component Diagram.....	33
Figure 18: Creating Positive Samples Vector File.....	36
Figure 19: Algorithm Training Process	37
Figure 20: Converting the Cascade File.....	38
Figure 21: The Training Data Set, Positive And Negative Examples.....	38
Figure 22: Screen Shot Showing Customers.....	39
Figure 23: Screen shot with Meter Detection	39
Figure 24: Meter Reading Detection.....	40

Figure 25: Binarized Image Of The Meter Reading For OCR Detection	40
Figure 26: OpenCV Performance Measure Tool	41
Figure 27: Results Of Opencv Performance Tool Run	42
Figure 28: Description File For Test Data With Meter Coordinates.....	43
Figure 29: ROC Curve Comparing Performance Based On Training Sample Size.....	55
Figure 30: ROC Curve Comparing Performance of Best Two Training Sessions.....	55
Figure 31: ROC Curve Comparing Performance Based On Scanning Pixel Window Size.	56
Figure 32: ROC Curve For A 10-Stage Classifier And 9-Stage Classifier.....	56
Figure 33: 1-Precision Curve For The First Three Training Sessions.	57
Figure 34: ROC Curve Comparing 24x24 and 48x48 Window Size.....	58

LIST OF TABLES

Table 1: Performance Results Of The First Training Session On Test Data.	45
Table 2: Performance Results Of The Second Training Session On Test Data.....	45
Table 3: Performance Results Of The Third Training Session On Test Data.....	47
Table 4: Performance Results Of The Fourth Training Session On Test Data.....	48
Table 5: Performance Results Of The Fifth Training Session On Test Data.....	49
Table 6: Results Of Sixth Training Session.....	50
Table 7: Results of Seventh Training Session With 24x24 Window Size.....	51
Table 8: Results For 9-Stage Classifier For The Classifier In Table 2.	52
Table 9: Results For 8-Stage Classifier For The Classifier In Table 2.	53
Table 10: False Positives for Haar Training	53
Table 11: False Positives For LBP Training.....	54

ACRONYMS

AMR	Automated Meter Reading
FP	False Positives
FPR	False Positive Rate
GPRS	General packet radio service
GPS	Global Positioning System
GSM	Global System for Mobile
LBP	Local Binary Patterns
OCR	Optical Character Recognition
OpenCV	Open Source Computer Vision Library
PDA	Personal Digital Assistant
RF	Radio Frequency
RGB	Red, Green and Blue
ROC	Receiver Operating Characteristic
SMS	Short Messaging System
TP	True Positives
TPR	True Positive Rate

TABLE OF CONTENTS

DECLARATION	ii
DEDICATION	iii
ACKNOWLEDGEMENT	iv
ABSTRACT.....	v
LIST OF FIGURES	vi
LIST OF TABLES	viii
ACRONYMS.....	ix
TABLE OF CONTENTS.....	x
CHAPTER 1: INTRODUCTION	1
1.1 Background.....	1
1.2 Problem Statement	1
1.3 Justification	2
1.4 Objectives	2
1.5 Research outcomes and their significance to key audiences.....	3
1.6 Research Questions.....	3
1.7 Scope.....	3
1.8 Assumptions of the research	3
1.9 Definition of Key Terms	4
1.10 Report Organization.....	5
CHAPTER 2: LITERATURE REVIEW	6
2.1 Overview.....	6
2.2 Types of Meters	7
2.3 Automated Meter Reading	8
2.4 Digital Image Processing	9

2.5 Machine Learning Techniques.....	11
Supervised Learning	11
The Naive Bayes Classifier.....	14
Adaboost Learning Algorithm	15
Text Detection in Digital Images	17
CHAPTER 3: RESEARCH METHODOLOGY	21
3.1 Introduction.....	21
3.2 Text Detection and Recognition Methodology	21
3.3 Data Collection	21
3.4 Data Clean-up	22
3.5 Adaboost Classifier Features Selection.....	24
3.6 Software Development Life Cycle.....	26
CHAPTER 4: ANALYSIS AND DESIGN	27
4.1 Current Techniques for Billing Customers	27
4.2 The Proposed Solution	28
Functional Requirements	29
Data Requirements.....	30
System Design	31
Types of Meters used in the study	33
Evaluation of the Learning Algorithm.....	34
CHAPTER 5: SYSTEM IMPLEMENTATION AND RESULTS	35
5.1 Adaboost Learning Algorithm Implementation.....	35
5.2 Meter Reading Prototype	39
5.3 Results.....	41
Adaboost Algorithm Performance Results.....	41
Performance Comparison based on Training DataSet	43

Comparing Haar Features and LBP Features.....	53
Visualization of Results	54
Discussion of Results.....	59
CHAPTER 6: CONCLUSION.....	61
6.1 Achievements.....	61
6.2 Research Contributions.....	62
6.3 Recommendation / Future Work.....	62
6.4 Assumptions and Limitations.....	63
REFERENCES	64
Appendix I: Code.....	66
Appendix II: Training Cascade File.....	76
Appendix III Installation Of OpenCV And Other Tools	78

CHAPTER 1: INTRODUCTION

1.1 Background

Automated reading of meters, for example, electricity, water and gas meters, is a problem area where new solutions are required as utility providers seek to reduce cost and improve services to their customers. Recent advances in wireless communication networks such as GSM has also created a great opportunity for developing applications with wide geographic coverage. Furthermore, with the ready availability of digital cameras, many people now have access to large databases of digital images leading to interest in research in applications that can manipulate these images to extract useful information. Research has focused on developing efficient machine learning algorithms to detect objects of interest in digital images. One of the objects of interest in digital images is text. Research has therefore also focussed on developing ways of reading text in digital images, an area popularly known as text information extraction (Jung, Kim and Jain, 2004). This area of research has led to the development of useful applications such as systems for assisting the blind by reading street signs, reading of vehicle number plates, mining of text in large databases of digital images such as the internet, etc. Another useful application of text information extraction is the problem of reading of meters, such as electricity and water meters, from digital images of the meters. The reading of meters from realtime digital images of the meters is the focus of this research paper.

1.2 Problem Statement

Meters are used by utility providers for recording utility consumption in many households and premises, distributed over a wide geographical area. In order to bill their clients, attendants of the utility provider are dispatched to manually read and record meter readings. Because of the large number of clients, this is a time consuming exercise. Conventional meter reading can also pose problems both for the customer and the utility provider when, for example, inadvertent reading of meters increase or decrease the customer bill. [Cabioc, Gerardo and Byun, 2011, p. 230]. In addition to possible errors in meter reading, there is also the problem of manual data entry of the readings into utility provider's systems which can also lead to more errors. There is need therefore for a solution that can automate the capturing of meter readings and send the data to the utility provider's systems. This can reduce the inefficiencies of the manual process.

1.3 Justification

Most utility providers are faced with the need to reduce operating expenses and increase profitability by adopting automated meter reading systems (AMR) (Galle, 2010). These are systems that automate the task of meter reading by using smart meters with existing network infrastructures such as GSM networks, Bluetooth, Radio Frequencies, Wireless networks, to automatically transmit meter readings from the customer site to the utility provider servers. This whole infrastructure is referred to as Advanced Metering Infrastructure (AMI) and has the key benefits of reducing costs and improving quality of service to the customer. (Galle, 2010). AMRs are becoming popular because of the inherent problems with traditional manual meter reading process. Some of the drawbacks of the manual process are: firstly, the introduction of human errors which may lead to overcharging or undercharging the customer, secondly, the practise by meter personnel of simply estimating consumption, which is not acceptable and the huge amount of time consumed by the utility provider personnel moving from house to house. (Sharef et al, 2013).

AMRs are however expensive to implement. This is because implementation of AMRs involves replacing the existing meters with smart meters, in addition to rolling out a network infrastructure for transmitting the readings. Thus, most utility providers cannot afford them. This is a motivation for exploring technologies that can achieve similar benefits, but in a cost effective manner for the utility providers. This study is aimed at contributing to the solution of the problems outlined. It proposes a system where customers, by use of readily available digital cameras on mobile phones or other handheld electronic devices, and by use of existing mobile and wireless networks can be able to point their cameras at the meters and the system reads and transmits the meter readings to the utility provider. The proposed system will not require replacement of existing meters or acquisition of a network infrastructure for transmitting the meter readings.

1.4 Objectives

The main goal of this study is to develop and test a prototype that can be used to read meters and transmit the readings to a utility provider.

The objectives of the study are to:

1. Design, develop and test a prototype that can use machine learning techniques to accurately read meters.
2. Design, develop and test a prototype that can transmit meter readings from a handheld device to the utility provider systems using a suitable communication infrastructure.
3. Evaluate various techniques that can be used to automatically read meters using machine learning algorithms.

1.5 Research outcomes and their significance to key audiences

The main output of this study will be a prototype that can be used to read meters and transmit those readings to the utility provider's systems using a suitable communication infrastructure. Such a prototype can be used to design a complete system that can be used to automatically read meters without the intervention of utility provider personnel. This can further bring cost savings to the utility provider due to the time saved and increased accuracy of meter readings. Customers of the utility provider will also benefit from the efficiency and accuracy of the system.

1.6 Research Questions

1. How can we develop a system that can automatically read meters?
2. What kind of hardware, software, and network communication infrastructure can be used to transmit the readings to a utility provider's systems?
3. What are the suitable machine learning techniques that can be used to automatically read meters from digital images of the meters?

1.7 Scope

This study will cover the problem of taking meter readings from digital images of meters using machine learning techniques. It will also cover the problem of how to transmit those readings to the utility provider's servers in a remote location. In both cases the study will develop a prototype to address the problem.

1.8 Assumptions of the research

Some of the assumptions of the study are:

1. Customers of the utility provider have handheld devices such as smartphones with cameras connected to GSM networks.
2. Customers will be willing to use the proposed system to take photos of the meters at regular intervals. The system will then use GSM network to send information to the utility provider.
3. Utility providers trust the data sent by their customers for use in billing the customers.

1.9 Definition of Key Terms

Automated Meter Reading – reading electric, gas or water meters automatically from a remote place without any human intervention.

Bluetooth - A wireless technology standard for exchanging data over short distances. (Wikipedia, 2013)

False Positive Rate – The percentage of false regions detected as true objects in an image by the classifier.

GPRS - A packet oriented mobile data service on the 2G and 3G cellular communication system's GSM (Wikipedia, 2013).

Machine Learning – field of study that gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1957 quoted in Wikipedia, 2013).

Precision – This is a measure of how much the classifier detects objects correctly without making mistakes. It is measured by FPR. A lower FPR means that the classifier is more precise, while a higher FPR means the algorithm is not as precise and will make more mistakes by classifying false objects as true objects.

Recall – This is a measure of how much the classifier detects objects correctly, without any misses. It is measured by TPR. A higher TPR means the classifier has better recall and will detect many objects correctly without missing the objects in an image.

Text Detection – the process of determining if a digital image contains text or not.

Text Localization – the process of determining the actual location of text in a digital image and generating bounding boxes to enclose the pixels in an image with the text.

Text Recognition – the actual reading of text from a digital image by a computer. This results in the read text being stored within the computer as text, separate from the digital image where the text was contained.

True Positive Rate – The percentage of regions correctly detected as true objects in an image by the trained algorithm.

WiFi - A popular technology that allows an electronic device to exchange data wirelessly using radio waves (Wikipedia, 2013).

ZigBee - A specification for a suite of high level communication protocols using small, low-power digital radios based on an IEEE 802 standard for personal area networks. (Wikipedia, 2013)

1.10 Report Organization

The rest of this report is organized as follows:

Chapter 2 gives a review of current literature on the research area. The current meter reading solutions are discussed to justify the need for other solutions. Various topics are then discussed including digital image processing and machine learning techniques. The Adaboost learning algorithm is then presented, and past work that has used the algorithm for solving problems in object detection is then given.

Chapter 3 gives the methodology followed in this study. The methodologies for text recognition, data collection and preparation, classifier feature selection and software development life cycle are presented.

Chapter 4 presents the analysis and design work done during the study. The current meter solutions are discussed at length, and a justification for a new solution is given. The proposed solution is then described using functional requirements and use case diagrams. This section ends by giving how the algorithm will be evaluated.

Chapter 5 presents the tools used in implementing the proposed solution. It shows how the algorithm training was conducted and how the results are used in a prototype to give the desired solution. The performance results of the algorithm are then presented and discussed.

Chapter 6 gives the recommendations for future work, assumptions and limitations.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

There is a push by most utility providers to adopt Automated Meter Reading (AMR) systems in the process of reading utility meters such as electricity, water and gas. This is due to the benefits that utility providers and customers reap from such systems. According to Galle (2010), the main reason for this push is the need to reduce costs and maximize profits by utility providers. Even though AMRs are more advantageous and are expected to gain more ground in the future, traditional meters are still dominant (Abdollahi, Dehghani and Zamanzadeh, 2012). In Kenya for example, there has been an attempt to install prepaid electricity meters alongside the conventional analogue meter readers. While there are prepaid meter solutions in the market, in places like Nairobi, manually read meters are commonly in use.

Utility providers can therefore benefit from research in the automation of meter reading. While there are AMR solutions coming from research in this area, new technologies are still required. This study proposes such an automated system. The system will allow customers with a suitable handheld devices such as a smartphone, connected to a GSM network, to take digital images of the meter, use machine learning techniques to take the current meter reading from the images and transmit the readings to the utility provider.

Advances in the field of Information Technology and the digital revolution in general has not only led to a drop in the cost of consumer digital devices such as smartphones, but it has also led to widespread use of communication networks such as internet and GSM which enable fast, secure and accurate information exchange (Bala and Babu, 2012). This means that customers of utility providers have access to smartphones and communication platforms to enable them transmit meter readings to utility providers.

Existing research in the area of extraction of useful information from digital images and videos can contribute to a solution to the problem of automating the reading of meters. While most research traditionally focused on analysis and processing of scanned documents, present research has extended to detecting and extracting objects in digital images (Viola and Jones, 2004). Text is one of the common object of interest in digital images. With the wide use of digital image capture devices such as digital cameras, mobile phones and PDAs, research in the extraction of text from images has been receiving more attention (Pan, Hou and Liu, 2008) .

2.2 Types of Meters

Meters can either be digital meters or analog meters. They can however, be classified in various ways, depending on how they operate and how the customer pays for the utility. The following types of meters can be identified in the literature:

Conventional Meters

Conventional metering is where the utility providers install meters at the consumer site and send monthly bills based on actual or projected consumption (Zyl, 2011). Utility providers employ staff to take meter readings at monthly or more intervals at the customer sites. This method is prone to errors and sometimes the staff may not get access to the customer site. McNabb (2011) has documented other methods used to circumvent the difficulty of accessing the customer site. One of them is to connect the meter to an electronic unit outside the customer site where meter reading staff can read the meters using handheld devices. Another method is to connect the meter to a radio frequency transmitter and the meter reading staff drive by the customer site and automatically take the meter readings using a handheld device.

Prepaid Meters

According to Zyl (2010), prepaid meters have builtin processing units that can automatically disconnect a consumer's supply. Consumers purchase utility in advance, and by use of a token or electronic signal, the amount purchased is fed into the meter. The meter automatically shuts off once the amount purchased is exhausted. There is therefore no need for employing staff to manually read meters or handle debt collection as in conventional meters. However, Zyl (2010), the main disadvantage of prepaid meters is the cost of the meters due to additional gadgets that must be incorporated for the meter to work. These meters also have more components that can fail and thus require regular maintenance and care.

Automated Meters

Automatic Meters take advantage of existing technologies such as wireless networks to deliver a solution that can allow for continuous meter readings from a remote location without any human intervention (McNabb, 2011). This system uses smart meters, which are composed of a two-way communication between the meter and utility provider, allowing the provider to obtain meter readings on demand and even command the meter remotely. The smart meters are usually components in a communication network, which is also connected to the utility providers billing system.

2.3 Automated Meter Reading

Automatic Meter Reading (AMR) is a state-of-the-art technology for taking electric, gas or water meter readings automatically from a remote place without any human intervention (Bala and Babu, 2012).

Cabioc, Gerardo and Byun (2011), propose an SMS-based automatic billing system for reading household meters. The system is composed of a remote site and a base station (Figure 1). The remote site, which is the customer site, has a customized meter reader with a GSM modem. The GSM modem can send SMS messages containing the meter identification and current reading. The base station at the utility provider contains a GSM modem, main server software for the utility provider and an SMS messaging server. The base station can request a remote site to send it the current reading. Once the reading is sent, the server at the base station can then generate customer bills which are sent back to the customer using SMS.

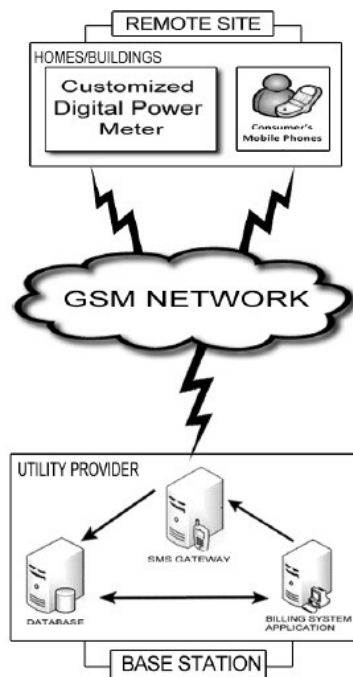


Figure 1: A reproduction of the Cabioc, Gerardo and Byun (2011) System Architecture.

A similar SMS-based system is proposed by Abdollahi, Dehghani and Zamanzadeh (2012). The system contains additional security features and sends more complex messages to the base station. The remote site must first identify itself to the base station by sending an acknowledgement SMS before it is admitted as a remote site. The system thus rejects any SMS received from unidentified sources.

Sharef et al (2013) on the other hand designed an AMR system which uses short radio frequency (RF). The system comprises an analog meter connected to a radio frequency transmitter at the customer site, and a receiver RF device which can be able to receive readings from the transmitter. The system, however, can only operate within a 300 feet radius. Such a system would therefore be semi-manual since the utility provider attendants would still have to go round with the receiver modules to take the meter readings.

Instead of using SMS, Sun and Zheng (2009) propose the use of a sensor network using ZigBee technology. Zigbee works like WiFi and Bluetooth, but costs much less and has low power consumption. Their system architecture includes meters connected to a ZigBee wireless sensor network. The ZigBee network is then connected to the centralized server using GPRS. Meter readings are sent to the centralized server via GPRS.

One common feature of these AMR systems is that they require additional devices to be installed in addition to changing some of the conventional meters. This can come at a high cost to the utility providers and therefore they might shy away from investing in the new technology. There is therefore a need for an AMR system which does not require the utility provider to make changes to the meters at the client site. This study proposes a system where the utility provider does not need to change the meters. There is need only for an investment in the infrastructure to receive the data sent by customers using their handheld devices. The utility providers will thus incur minimal costs compared to the other methods.

2.4 Digital Image Processing

Gonzalez and Woods (2004) define a digital image as a two-dimensional function, $f(x,y)$, where x and y are spatial coordinates. The value of f at any pair of coordinate (x,y) is called the intensity. The intensity values for a monochrome image is called a grayscale and is normally between 0 (black) and 255 (white). Color images are formed by a combination of individual 2-D images. For example, RGB images are represented by three individual component images for red, green and blue. An image can therefore be easily represented as a matrix of dots with corresponding discrete intensities. Each dot is referred to as a picture element (pixel).

One can perform operations on pixels individually without any regard to other pixels in the image. This is known as point operation. It is also possible to operate on a pixel based on the neighbors of the pixel in a process known as spatial filtering. The idea is to operate on a small neighborhood, for example, 3X3 neighborhood and change the pixel based on the intensity values of its neighbors.



Figure 2: A Greyscale Image Of A Water Meter, 160x160 Pixels

Point Operations on Pixels

Point operations are important for digital image processing as they can be used for pre-processing to make the images more suitable for use with other applications, such as machine learning algorithms. They are also used in image enhancement. Gonzalez and Woods (2004) represent point operations as a transformation function T , such that,

$g(x,y) = T[f(x,y)]$, where g is the new pixel value after the transformation.

Several transformations can be performed on a pixel, such as addition, subtraction, multiplication and finding the complement of the pixel. The complement of an image is the photographic negative of the image. Other point operations on images are thresholding and logarithmic functions.

Spatial Filters

When transformations on a pixel take into account the neighborhood, then the process is known as spatial filtering. According to Gonzalez and Woods (2004), spatial filtering consists of (1) defining a center point (x,y) , (2) performing an operation that involves only the pixels in a predefined neighborhood about that center point, (3) letting the result of the operation be the response at that point, and (4) repeating the process for every point in the image. The filters are linear spatial filters or non-linear spatial filters depending on whether the operations themselves are linear or not.

While there are many kinds of operations that can be performed in filters, in this study we are especially interested in filters that can be used for edge detection in an image. Such filters perform first order and second order derivative operations on the neighborhood. Examples given by Gonzalez and Woods (2004) include the Laplacian filters and Sobel filters. These can be used to enhance the edges in an image, making objects in an image easier to detect.

Figure 3, shows an example of a Sobel filter applied to the image in Figure 2. It is easy to see that the Sobel filter enhances the edges in an image, making objects in the image easier to detect. Other important edge detection techniques include Canny edge detectors and Hough transform.

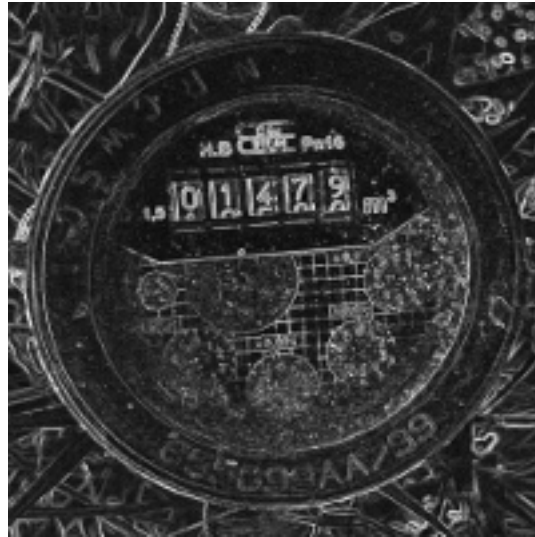


Figure 3: Water Meter Image After a Sobel Filter Operation

2.5 Machine Learning Techniques

Alpaydin (2010) defines machine learning as programming computers to optimize a performance criterion using example data or past experience.

For example, the problem of handwriting recognition can be considered a machine learning task (Mitchell, 1997). In this case the task is to recognize handwritten words in a digital image. A database of handwritten words with their given classifications can be used to provide past experience and the performance of the learning can be measured by the percentage of words correctly classified.

Machine learning can either be supervised learning or unsupervised learning. In supervised learning, labeled training examples are provided to the learning algorithm by the supervisor in addition to the input data. Instead in unsupervised learning there are no training examples but only the input data.

Supervised Learning

Alpaydin (2010) gives an example of supervised learning that can give clarity to the basis of this study. This is because the problem of reading meters from digital images of meters can also be considered a supervised learning problem.

Let us say we have a *class C* of a family car that we want to learn, a set of examples of cars and a group

of people to whom we show the cars. This group of people look at the cars and categorize them as either family cars (*positive examples*) or not (*negative examples*). Class learning therefore involves finding a description that is shared by all positive examples and none of the negative examples. From this, given a car we have not seen before, by checking with the description learned, we can classify it as a family car or not.

From all the attributes of a family car, we have to decide, with the help of experts, which attributes can describe whether a car is a family car or not. In our example, it is the price of the car and engine capacity of the car. These two attributes form the *input* to the class recognizer and once decided all other attributes are irrelevant for our example.

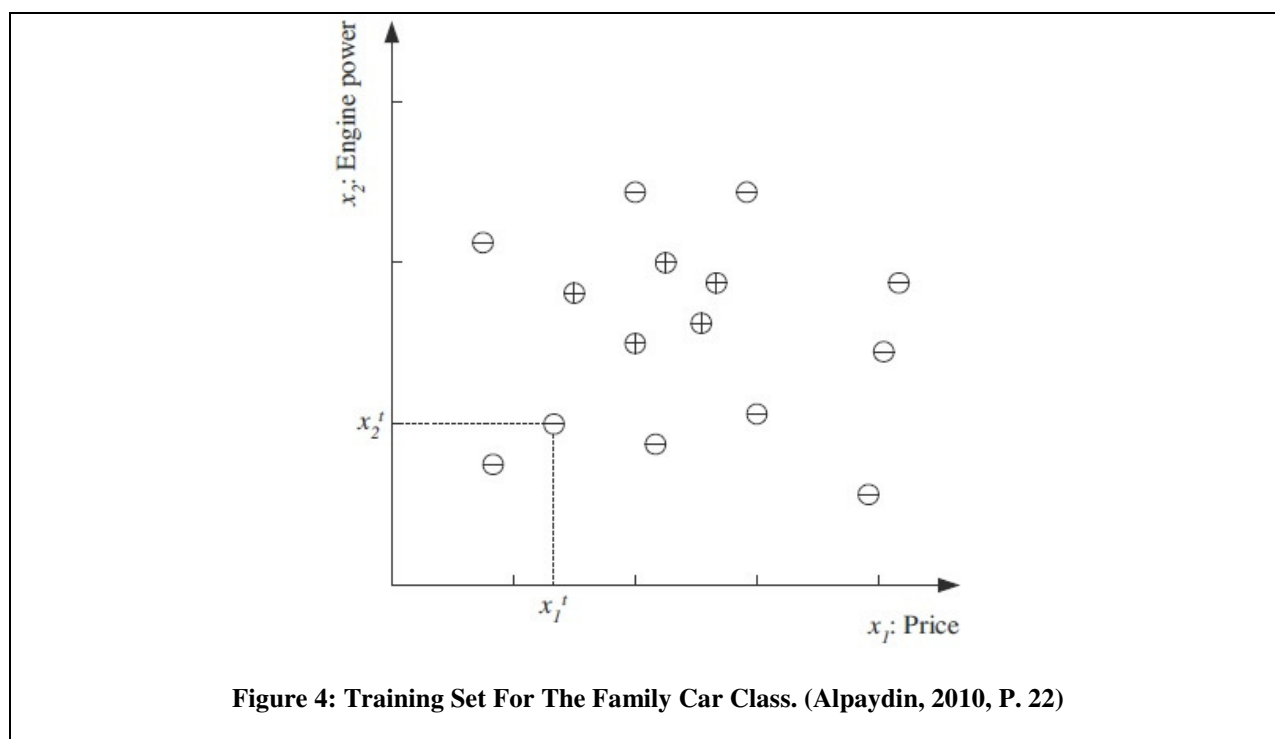


Figure 4: Training Set For The Family Car Class. (Alpaydin, 2010, P. 22)

The training set can be plotted as shown in Figure 4. Each point represents a single car, with the x attribute representing the price, y attribute the engine power and t is used to show that this is a training example. At each point, '+' denotes a positive example and '-' denotes a negative example.

From discussion with the experts, the range of or price and engine power can be: price is between p_1 and p_2 and engine power between e_1 and e_2 . This sets the *hypothesis class*, H , from which C is drawn, i.e., a rectangle in the price-engine power space as shown in Figure 5.

The goal of the learning algorithm is then to find the particular hypothesis $h \in H$ to approximate C as

closely as possible.

Since our training set X is only a small subset of the set of all possible x , there can be empirical error which is the proportion of training examples instances where predictions of h do not match the required values given in X .

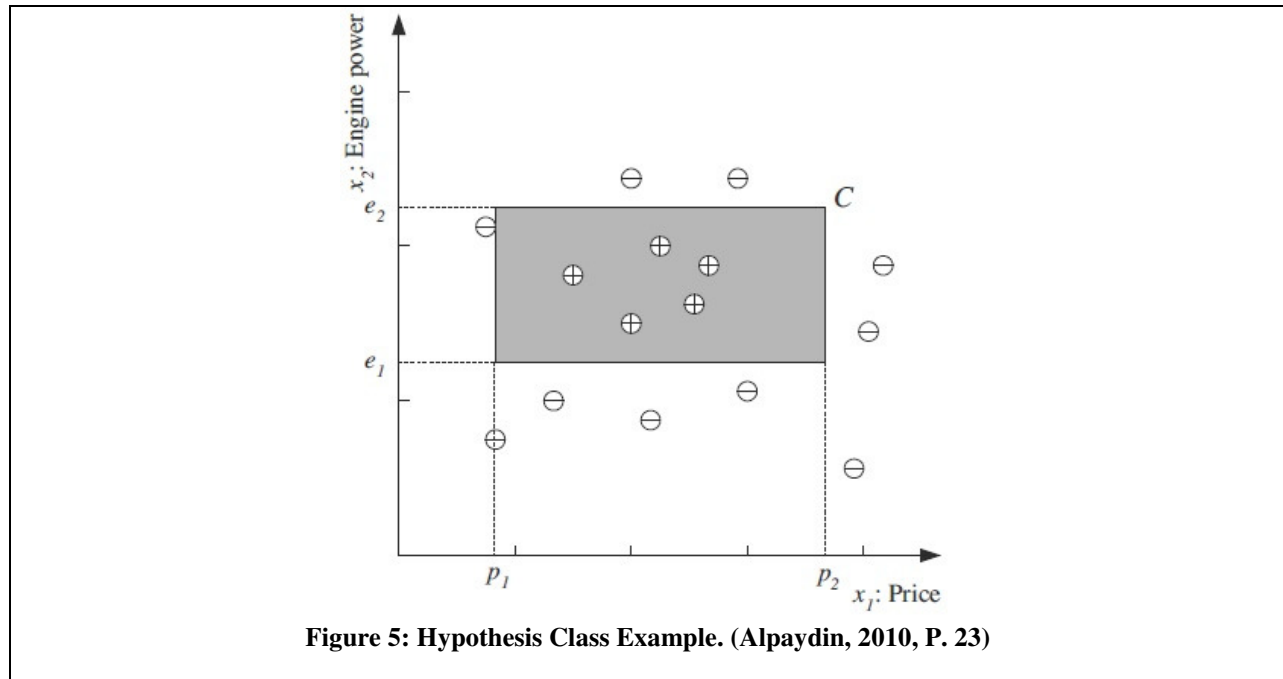


Figure 5: Hypothesis Class Example. (Alpaydin, 2010, P. 23)

We can also find the most specific hypothesis, S , which is the tightest rectangle that includes all the positive examples and none of the negative ones. The most general hypothesis, G , is the largest rectangle we can draw that includes all the positive examples and none of the negative ones. Any $h \in H$ between S and G is a valid hypothesis and is said to be consistent with the training set, and such h make up the version space.

Given another training set, S , G , the version space, the parameters and the hypothesis, h , can be different. While in this example there is only one class, we can have multiple classes. For example we can have another class of luxury cars, sports utility vehicles, etc.

A learning problem can be considered ill-posed, because the training set normally contains only a small subset of all possible instances and the data is never sufficient to find a unique solution. Because of this we always have to make some extra assumptions to have the unique solution with the data we have. The set of assumptions we make to have learning possible is called inductive bias of the learning algorithm. We have to choose the correct bias for the learning algorithm to be correct and this is known as model selection, which is choosing between possible H .

How well a model trained on the training set predicts the right output for new instances is called generalization. For best generalization we must match the complexity of the class H with the complexity of the function underlying the data. If H is less complex than the function, this is known as underfitting. If H is too complex, or there is noise in the data, we may learn not only the data, but also the noise, and end up with a bad hypothesis $h \in H$ which may lead to a bad fit. This is known as overfitting.

We can measure how well our generalization of the hypothesis is, and thus the quality of our inductive bias, by dividing the training set between the training set and the validation set. Given a large enough training set and validation set, the hypothesis that is most accurate on the validation set, given a large training and validation set is the best one.

The Naive Bayes Classifier

The Naive Bayes Classifier draws its generalizations from Bayes Theorem. The theorem can be defined as follows (Murty and Devi, 2011):

$$P(H_i|X) = P(X|H_i)P(H_i)/P(X)$$

$P(H_i|X)$ is the posterior probability of H_i , conditioned on X .

$P(H_i)$ is the prior probability of H_i , i.e., the probability of the hypothesis regardless of the value of X , obtained before observing X .

$P(X|H_i)$ is the probability of X conditioned on H_i .

In this case, X is the pattern whose class label is unknown, and H_i is a hypothesis which indicates the class to which X belongs. For example, H_i can be the hypothesis that a pattern belongs to class C_i .

A Naive Bayes Classifier is a simple probabilistic classifier based on applying Bayes theorem where every feature is assumed to be class conditionally independent (Murty and Devi, 2011).

When the objects we are classifying have a large number of classes and features, classification is difficult because a lot of data will be required to estimate the probabilities. The Naive Bayes Classifier assumption eliminates this difficulty by assuming that the effect of a variable value on a given class is independent of values of other variables (class conditional independence).

Murty and Devi (2011) give a definition of the Naive Bayes Classifier as follows:

The probability model for classifier is a conditional model $p(C|F_1, F_2, \dots, F_n)$ over a dependent class variable C with a small number of outcomes or classes, conditional on several feature variables F_1 to F_n .

Because of the class conditional independence, the conditional distribution over the class variable C can be expressed as:

$$p(C|F_1, \dots, F_n) = (1/Z)p(C) \prod_{i=1}^n p(F_i|C)$$

Where Z is a scaling factor dependent only on F_1, \dots, F_n , i.e., a constant if the value of the feature variable are known, $p(C)$ are the prior probabilities of the classes and $p(F_i|C)$ are the independent probability distributions.

If there are k classes, and if a model for $p(F_i)$ can be expressed in terms of r parameters then the corresponding naive Bayes model has $(k-1) + nrk$ parameters. In most cases, $k=2$ (binary classification) and $r=1$ are common.

The naive Bayes Classifier is one of the simplest and most popular classifiers because of its naive assumption: features are assumed to be independent given the class. Even though this is an unrealistic, hence naive, assumption, it has been found to provide very good results (Casacuberta et al, 2005). This classifier has been used successfully in pattern recognition and human language processing. For example, Casacuberta et al (2005) have given successful performance results for its use in optical character recognition (OCR), speech recognition, word disambiguation and text classification.

Adaboost Learning Algorithm

Adaboost algorithm is a method of combining a set of weak classifiers to make a strong classifier (Chen and Youlle, 2004). It was formulated in 1995 by Freund and Schapire. The algorithm is described well in the introduction paper by Freund and Schapire (1999), where the pseudocode is also given as in Figure 6.

The algorithm takes as input a training set $(x_1, y_1), \dots, (x_m, y_m)$ where each x_i belongs to some domain or instance space X , and each label y_i is in some label set Y . We assume for this case that $Y = \{-1, +1\}$.

Adaboost calls a given weak learning algorithm repeatedly in a series of rounds $t=1, \dots, T$. The algorithm maintains a distribution or set of weights over the training set, denoted by $D(t_i)$ for a training example on round t .

Initially all weights are set equally but on each round, the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set.

The weak learner must find a weak hypothesis $h_t : X \rightarrow \{-1, +1\}$ appropriate for the distribution D_t . The goodness of a weak hypothesis is measured by its error ϵ_t as per the formula in Figure 6.

Once the weak hypothesis h_t has been received, Adaboost chooses a parameter α_t , which measures the importance assigned to h_t . From the pseudocode, it is clear α_t gets larger as ϵ_t gets smaller.

D_t is then updated as shown in the rule in the pseudocode. This rule increases the weight of examples misclassified by h_t and decreases the weights of correctly classified examples. The weight thus tends to

concentrate on hard examples.

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

- **Train weak learner using distribution** D_t .
- **Get weak hypothesis** $h_t : X \rightarrow \{-1, +1\}$ **with error**

$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].$$

- **Choose** $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$.
- **Update:**

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right).$$

Figure 6: Adaboost Learning Algorithm Pseudocode. (Freund And Schapire, 1999, P. 2)

The final hypothesis H is a majority vote of the T weak hypotheses where α_t is the weight assigned to h_t .

This algorithm has been used extensively in the area of detection of objects in images and videos. The first such work was for face detection by Viola and Jones (2004), which obtained very good results. Their method inspired Chen and Youlle (2004) to apply the algorithm to detection of text in natural scene images.

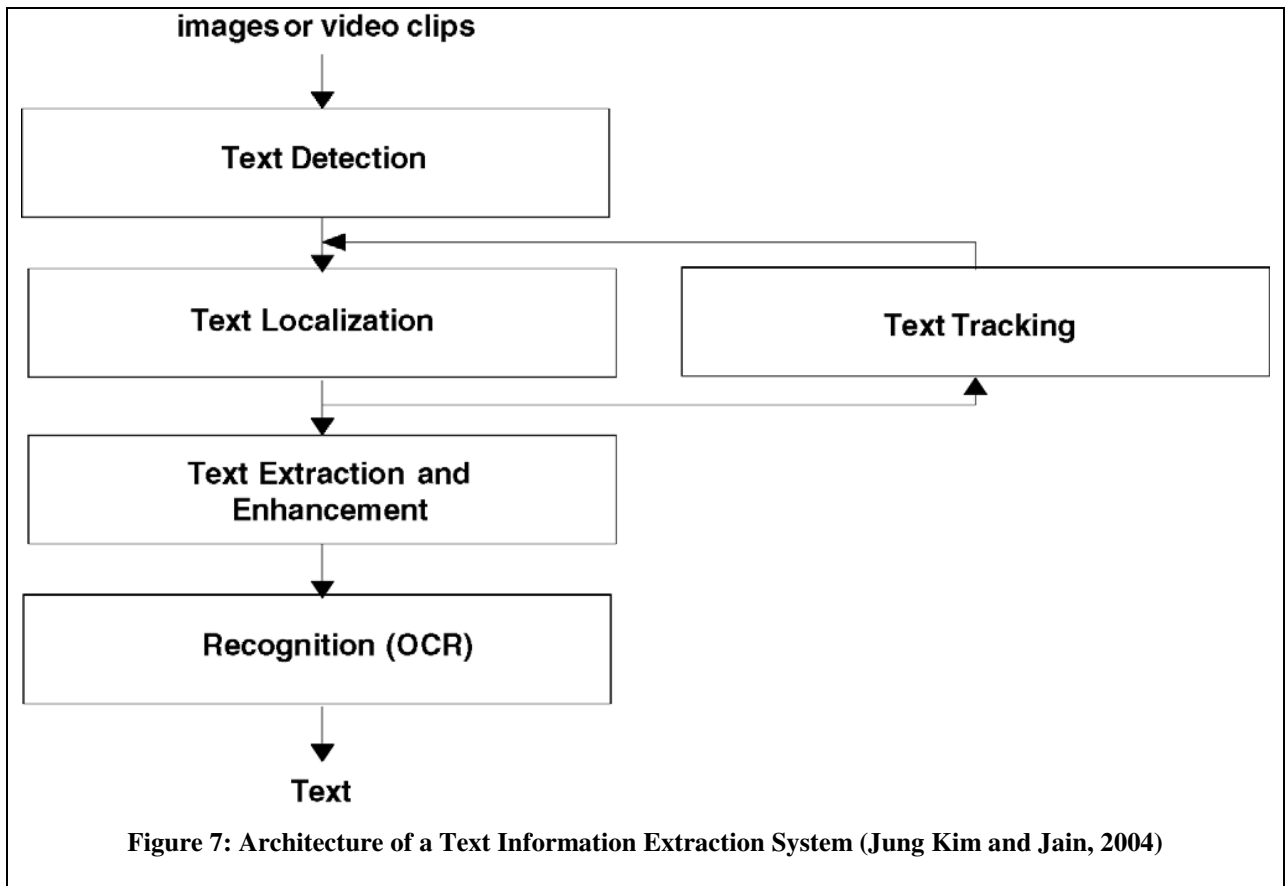
According to Chen and Youlle (2004), Adaboost is the most effective algorithm for detecting objects of interest in images. Guida, Comanduci and Colombo (2011) have also used Adaboost algorithm in an application that reads bus numbers for the visually impaired using smartphones.

Text Detection in Digital Images

Detection of objects in digital images and videos has become an interesting area of research because of the wide use of digital image capture devices such as digital cameras, mobile phones and PDAs (Pan, Hou and Liu, 2008). Text is one of the objects of interest in digital images because there are a lot of applications that can be developed that uses text in images. Some of the applications in the literature include Pan, Hou and Liu's (2008) work to detect and localize text in natural scene images such as highways, shopping centers and billboards. Another work in this area is the one for reading bus numbers for the visually impaired using mobile phones by Guida, Comanducci and Colombo (2011). Chen and Youlle's (2004) important work in this area must also be mentioned as they were the first to adapt Adaboost learning to text detection in natural scenes. Escalera et al (2009) have also given us their work on text detection in urban scenes.

A number of steps involved in the process of extracting text from digital images have been proposed. In their text information extraction survey, Jung, Kim and Jain(2004) propose four important steps for extracting text from digital images (Figure 7):

The first step in this process is text detection, which refers to the determination of the presence of text in a given image. The second step, text localization, is the the process of determining the location of text in the image and generating bounding boxes around the text. There is an intermediate step known as text tracking, which is performed to reduce the processing time for text localization and to maintain the integrity of position across adjacent frames. The third step is text extraction and enhancement which is used to segment the text from the background to facilitate its recognition. Finally in the fourth step, the extracted text image is converted into a binary image and enhanced, before being fed into an OCR engine for transformation into plain text. Text detection, text localization and text extraction are used interchangeably in the literature.



There are two classes of methods that have been proposed to solve the text detection and localization problem (Pan, Hou and Liu, 2009): connected component based methods and region based methods.

In connected component based methods, text in images are seen as sets of separating connected components, each with distinct intensity or color distributions and linked edge contours. Text detection and extraction in this case has three steps: firstly, extraction of connected components from images, secondly analysis of connected components is done to determine if they are to text components and finally, post processing to group components into text regions. However, this method has two disadvantages: the difficulty in extracting connected components, and the difficulty in designing an algorithm that can take care of the many text-like components in digital images (noise).

Region based methods on the other hand are based on the fact that text regions in images have distinct characteristics from non-text regions. Such distinctions include high density gradient distribution and distinctive texture and structure. Text detection is done by extracting features of sampled regional windows to determine if they contain text. Window grouping or clustering methods are then used to generate candidate text lines in the text localization step.

Region based methods have the same accuracy as component based methods but are less sensitive to noise in images that contain text-like features but are not text. However, computational cost is high in region based methods. The work of Chen and Youlle (2004) have proposed a method using Adaboost algorithm which has solved the problem of computational cost, making region based methods much better than component based methods.

Text recognition is done using an Optical Character Recognition (OCR) engine, which mostly accept grey scale images as input. However, in addition to binarization, the text regions identified must be extended for better performance [Chen and Youlle, 2004]. The extension is due to the fact that most text regions sometimes miss letters or digits at the start and end of the text.

In the literature different variants of these methods have been used in solving some of these problem. Here we consider a few of the works and the various methods that were used.

Firstly we consider the important work of Chen and Youlle (2004) on detection and reading of text in city scenes. This work was based on another great work by Viola and Jones (2004) on rapid object detection using Adaboost learning algorithm. Features for text detection are selected using the principle of informative features, which are good for distinguishing text from non-text. Other features used are Haar features, histogram of intensities, gradient direction and intensity gradient. Joint probability distributions are then calculated based on these feature responses on text regions and non-text regions to obtain weak classifiers as log-likelihood ratio tests. By use of Adaboost training methods with a training set consisting of positive and negative images, the weak classifiers can be used to build a strong classifier. The strong classifier can then be used to localize the text regions. The localized text regions are then binarized using adaptive binarization and connected component analysis to make them suitable for reading by OCR applicatins.

Pan, Hou and Liu (2009) have also proposed a robust method to detect and localize text in natural scene images. In their method the images are first preprocessed by transforming RGB to grayscale images. The grayscale image is then re-scaled to form an image pyramid using nearest neighbour interpolation to take care of texts of different sizes in the image. Feature selection is done using histogram of oriented gradients and multiscale local binary pattern. Adaboost learning algorithm is used to build a strong classifier using these features from the weak learning algorithm. In the text localization process, text lines are then generated from the candidate regions detected as text by grouping overlapping windows. Local binarization and connected component analysis are then used to prepare the text to be read by an OCR engine.

The work of Guida, Comanducci and Colombo (2011) on reading of bus line numbers using a smartphone for the visually impaired can also be cited in this study. The system allows for a visually impaired person to direct their smartphone at an approaching bus and acquire the bus number from images of the bus. The oncoming bus is localized inside the image to generate a region of interest which is the bus, eliminating unnecessary detail. Localization is done using the Adaboost algorithm by exploiting a cascade of weak classifiers and boosting training method. To train the classifier, several images of the bus (positive examples) and other images (negative examples) were used. The next step was to perform the line number localization to locate the bus number on the image. This was done using a template bus facade description to geometrically rectify the projective distortions arising from image projection. The template was built by selecting some distinct features of the bus. A robust geometric matching and other classic image analysis techniques like Canny edge detectors and Hough transform were used to localize the bus number. Binarization is then done to make the bus numbers easier to read using OCR software. As an alternative to OCR software, a classifier was trained to recognize the bus numbers. To make the algorithm more accurate, several consecutive images are taken, processed and the most frequent result is the one returned to the user using voice synthesis.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

Machine learning techniques are an established way of solving complex problems that require computer applications to make intelligent decisions based on past experience. Important to machine learning techniques is the data required to train and validate the classifier. Data collection and clean-up is for this reason an important part of the methodology of solving the problem of automatic reading of water meters. Since the machine learning algorithm is simply intelligent software, it is also important to consider what software development methodologies are used to build the final software prototype.

3.2 Text Detection and Recognition Methodology

The method that closely fits this study is the method proposed by Guida, Comanducci and Colombo (2011) for reading bus line numbers using a smartphone for the visually impaired. Adapting this methodology for our study, the following steps will be followed:

1. The first step after pre-processing the image will be to localize the meter region and make it our region of interest. This will be done using the Adaboost algorithm by exploiting a cascade of weak classifiers and boosting training method. To train the classifier, several photos of the meters (positive examples) and other photos (negative photos) will be used.
2. The next step will be to perform the meter reading localization. This will be done using geometric matching, template matching and classic image analysis techniques.
3. Binarization will then be done and the binarized image presented to an OCR engine.
4. The resulting number will be the meter reading that can be transmitted to the utility provider. To increase accuracy several images of the meter can be taken in succession, read by the system and the reading that is most frequent can be used as the correct reading.

3.3 Data Collection

The data required for the learning algorithm falls into the following categories:

1. Positive examples
2. Negative examples
3. Test Data

Positive Examples

Several digital photos of meters were taken with a 5 megapixel camera of a smart phone and a 10.1 megapixel digital camera to form the positive examples. These were taken under different light conditions, scales and angles. While focusing the camera on the meter, the most important thing was to ensure the whole meter face and the meter reading is visible on the camera. The whole face of the meter was captured to enable the system to detect whether a meter is present in the captured image or not.

Negative Examples

The negative examples were digital images of objects and sceneries that do not contain meters. Some of these images were harvested by different entities (Fergus, 2010) and are available on the internet. The harvesting was done using automated tools which collect images with particular keywords on internet search engines. Other images in this category were randomly taken with a digital camera to capture different scenes and objects.

Test Data

The test data is a subset of the positive examples and negative examples. The test dataset contains images used for testing the learning algorithm after training with positive and negative examples. Positive test dataset is a subset of the positive examples and contains images with meters. Negative test dataset on the other hand is a subset of the negative examples and contains images without meters. The test data is therefore used to evaluate the learning algorithm and determine its effectiveness.

3.4 Data Clean-up

Images harvested from the internet did not need any clean-up. This is because they were already used in other machine learning algorithms (Fergus, 2010) and were of reasonable pixel sizes ranging from 100 by 100 pixels to 500 by 500 pixels.

However, the dataset collected manually required some clean-up. The following data clean-up activities were carried out on the manually collected dataset:

- The images were first rotated to ensure they were upright.
- Images were resized to 600 X 400 pixels. This was to ensure that the training algorithm, which is a memory intensive program, works on smaller images. The images however did not lose their visual content and the meter readings could easily be seen on the positive images. Similarly negative images did not lose the scenery and various objects they contained.

- Positive images contained meters which was our object of interest. To prepare the positive images for training, a tool was used to draw a boundary of the meter within the image. This was to ensure the learning algorithm only concentrates on learning the meter region for detection, ignoring other objects near the meter in the digital image. This is shown in Figure (8) below.

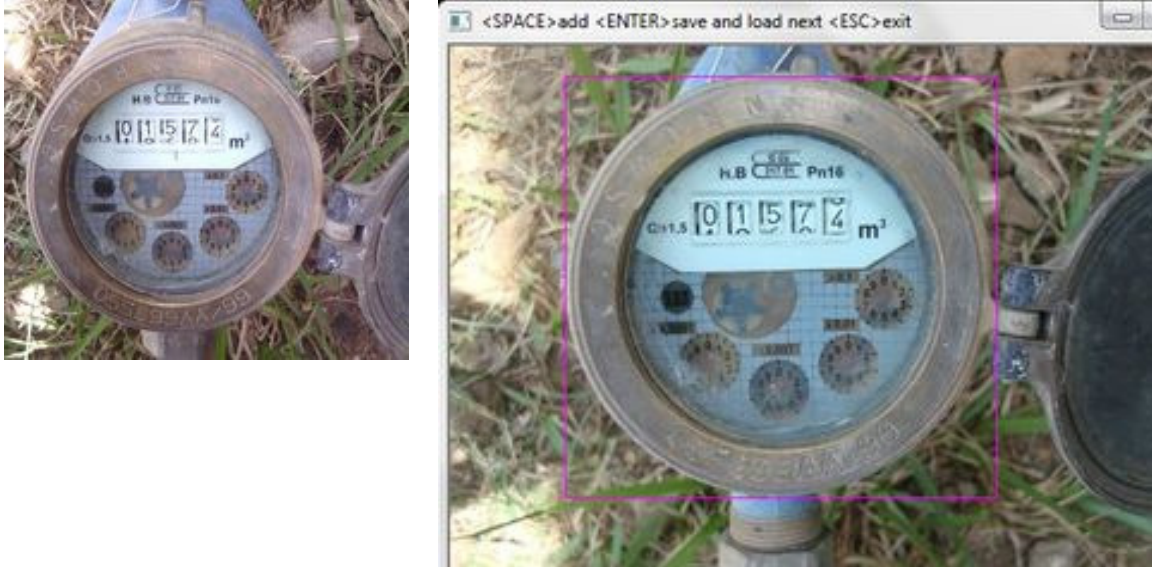
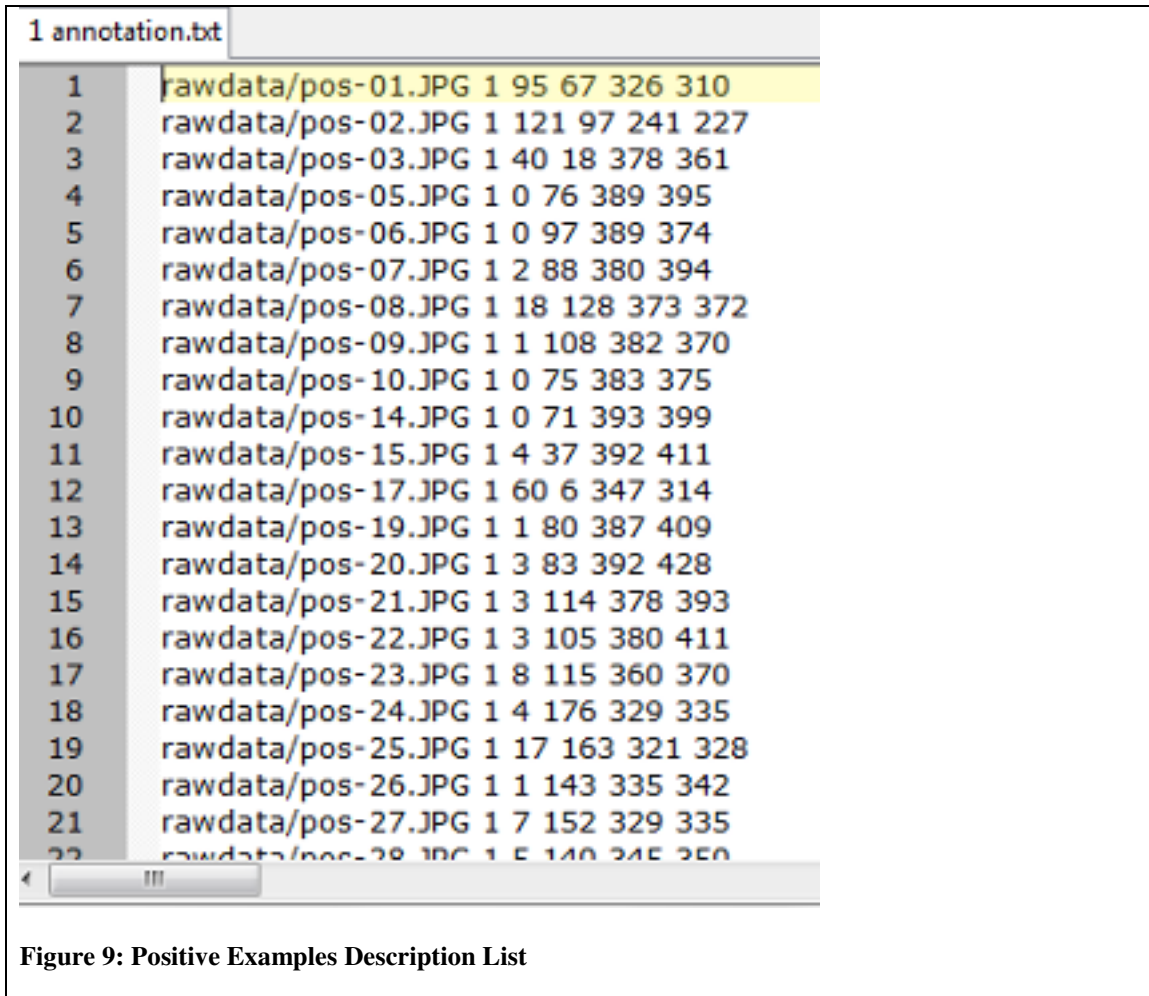


Figure 8: Positive Example Image Without Bounds (Left), With Bounds (Right)

- The tool for drawing the boundary of the meter also creates a description file containing the name of the image, the number of objects of interest present in the image, top left and top right coordinates of the bounding frame, and the width and height of the bounding frame. This is shown in Figure 9.
- The positive and negative examples were also renamed to make them more meaningful. A description file for the negative examples, similar to Figure 9, was also created.
- Similar to the description file created for positive examples as shown in Figure 9, for the test images, a description file containing the region of the meter in each test image was also created.



3.5 Adaboost Classifier Features Selection

The features used by the Adaboost training algorithm on the positive and negative examples are Haar-basis functions. These are faster and easily encode ad-hoc domain based knowledge which would be difficult to learn working directly with pixels (Viola and Jones, 2004).

Three types of features are used:

- Two-rectangle feature (Figure 10 A, B), whose value is the difference between the sum of the pixels between two rectangular regions.
- Three-rectangle feature (Figure 10 C), where the value of the feature is the difference between the sum sum of pixels at the center and the two outside reactangle regions.

- Four-rectangle feature ((Figure 10 D), whose value is the difference between the sum of diagonal pairs of rectangles.

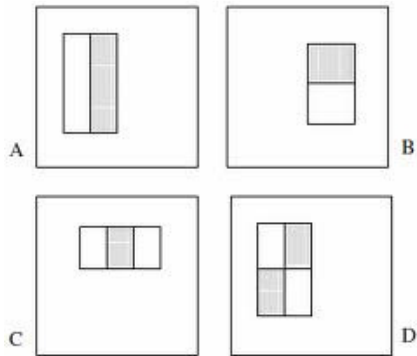


Figure 10. Rectangle Features. (Viola And Jones, 2004, P. 2)

The set of rectangle features in a positive or negative example image is quite large. For example, in a 24x24 image window, there are over 180, 000 features. This can make calculations quite slow. To solve this hurdle a different representation of the image referred to as an integral image is used. An integral image (Viola and Jones, 2004) at a location is the sum of the pixels above and to the left of that point. In Figure 11, the integral image at location 1 is the sum of pixels at A, and at location 3 is (A+C) and at 4 is (A+B+C+D). A quick way to calculate the integral image at location 4 is $((4+1) - (2+3))$.

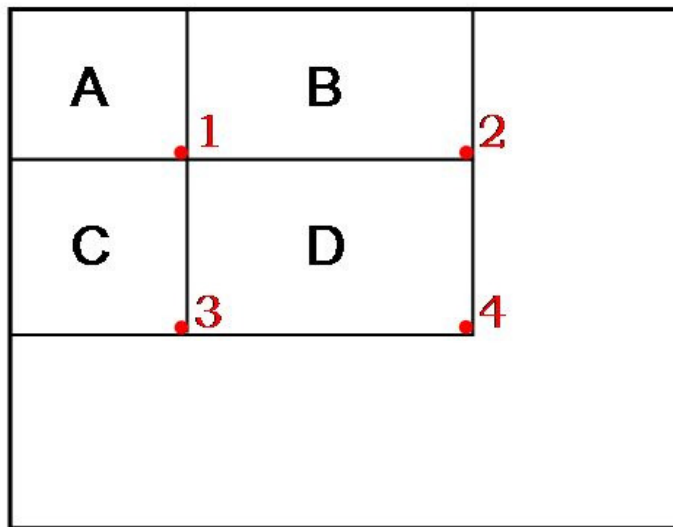


Figure 11: Integral Image. (Viola And Jones, 2004, P. 3).

Given the feature set, the Adaboost learning algorithm is used to learn a classification function, by creating a strong classifier from weak classifiers.

The weak learning algorithm selects the single rectangle feature that best separates the positive and negative examples. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples is misclassified.

According to Viola and Jones (2004), a weak classifier $h_j(x)$ consists of a feature f_j , a threshold θ_j and a parity p_j , indicating the direction of the inequality sign, i.e.:

$$h_j(x) \text{ is } 1 \text{ if } p_j f_j(x) < p_j \vartheta_j \text{ or } 0 \text{ otherwise.}$$

Where x is a 24x24 pixel sub-window of a positive or negative example image.

In order to increase performance and accuracy of the classification function, a cascade of classifiers is also used (Viola and Jones, 2004). The cascades are arranged such that simpler classifiers are used to discard majority of the image sub-windows which are negative, before more complex classifiers are called upon to achieve low false positive rates. This is shown in Figure 12.

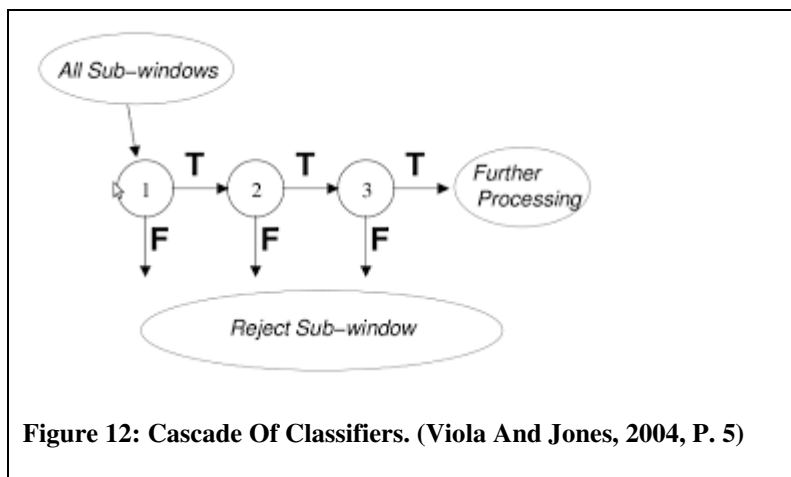


Figure 12: Cascade Of Classifiers. (Viola And Jones, 2004, P. 5)

Each stage in the cascade reduces the false positive rate and decreases the detection rate. A target is selected for the minimum reduction in false positives and the maximum decrease in detection. Each stage is trained by adding features until the target detection and false positives rates are met.

3.6 Software Development Life Cycle

The waterfall lifecycle was used as the software development methodology, since the software to be developed required a thorough understanding of the requirements before the prototype could be developed. The software development process therefore followed a simple requirements analysis, design, coding, testing and implementation.

CHAPTER 4: ANALYSIS AND DESIGN

The problem of billing customers by utility providers is a recurrent one. Utility providers will therefore continue to look for new solutions to make the process efficient and cost effective. Various methods are currently being used to provide this solution as documented in the literature review. It is important to list some of these methods in order to compare them with the solution proposed in this study.

4.1 Current Techniques for Billing Customers

Conventional Metering

One such method is the use of conventional meters, where the utility providers install meters at each customer site. Staff of the utility provider then visit each customer site at regular intervals to manually take the meter readings and submit them to the utility provider for entry into the billing system. Utility providers can also estimate the consumption rate in case the meter readings are not taken regularly. Customers are then sent their bills through snail mail and are required to pay their bills by a stipulated date. While it is still the most common method employed by utility providers, it has its disadvantages which among others include: the need to have a huge workforce to take meter readings, inability to access the customer premises, inaccuracy of manual readings, which may lead to loss of cash both to the consumer and the utility provider. This method is therefore generally inefficient.

Conventional Metering with Radio Frequency

Another method which complements the conventional metering is where the meter at the customer site is fixed with a radio frequency transmitter. Staff of the utility provider still need to visit each customer site, but they do not need to access the site. With a handheld radio frequency device, they can automatically take meter readings within a short distance from the customer sites. This data can then be downloaded to the billing system when the utility reading staff returns to the utility providers premises. This method however requires additional gadgets to be mounted on the meters. This may be expensive to implement since additional gadgets have to be installed at each customer site, and the gadgets will also require a power source. Depending on the existing meters, they may also have to be changed in order to install compatible meters.

Prepaid Metering

Prepaid metering has gained traction as an efficient alternative to conventional metering. These meters are installed at each customer site. Consumers purchase utility in advance, and by use of a token or electronic signal, the amount purchased is fed into the meter.

When the token is exhausted, the prepaid meters automatically disconnect a consumer's supply. Utility providers therefore avoid the cost of employing meter reading staff and debt collection staff, since customers pay for their utilities in advance. The meter automatically shuts off once the amount purchased is exhausted. There is therefore no need for employing staff to manually read meters or handle debt collection as in conventional meters. However, the movement from conventional meters to prepaid meters still come at an additional cost to the utility providers, in which they have to invest. The meters have additional gadgets for recognizing the tokens and shutting off supply. These meters also have more components that can fail and thus require regular maintenance. Utility providers also need to rollout an efficient mechanism for selling meter tokens.

Automated Metering

In automated metering, utility providers implement smart meters connected to wireless networks, allowing them to take meter readings on demand without any human intervention. The smart meters are equipped with wireless transmission devices with a two-way communication link with the utility provider. The utility providers implement a base station from where all the meters can be controlled. In the literature cited, there are different implementations of automated metering including the use of SMS applications to send meter readings to the utility provider and to receive bills from the utility provider. Automated metering also gives utility providers functionalities beyond the basic taking of meter readings. These include advising customers on how to cut consumption rates and detection of any leaks on the system.

Automated metering is therefore the most advanced way of implementing meters. However, it also comes at the greatest cost to the utility providers and customers. Utility providers will need to replace all the conventional meters at the customer sites with expensive smart meters. They also have to implement the wireless communication network to connect all the consumer meters to the base station.

4.2 The Proposed Solution

The challenges encountered in the various methods of metering employed by utility providers is a motivation to seek alternative solutions to overcome those challenges. One of the major challenges is the high cost of rolling out a new metering infrastructure to replace conventional meters. Since the conventional meters are still commonly in use, this solution proposes to use handheld devices to take digital images of the consumer meter, have the meter reading extracted from the digital image and sent to the utility provider servers for billing. This can eliminate the need for replacing existing meters. Utility providers can also employ fewer meter reading staff to confirm, once in a while, the meter self-readings sent by the cutomers.

The idea of self-reading of meters is not strange to utility providers. In conventional meter implementations, utility providers allow customer to perform self reading of the meters and use these readings to prepare customer bills. The utility provider staff then take meter readings only once in a while to confirm the self-readings and make adjustments to the customer accounts accordingly.

The proposed solution can ride on this arrangement, only with additional security and data integrity requirements which will be better than manual self-readings.

Functional Requirements

The following are the functional requirements of the proposed solution

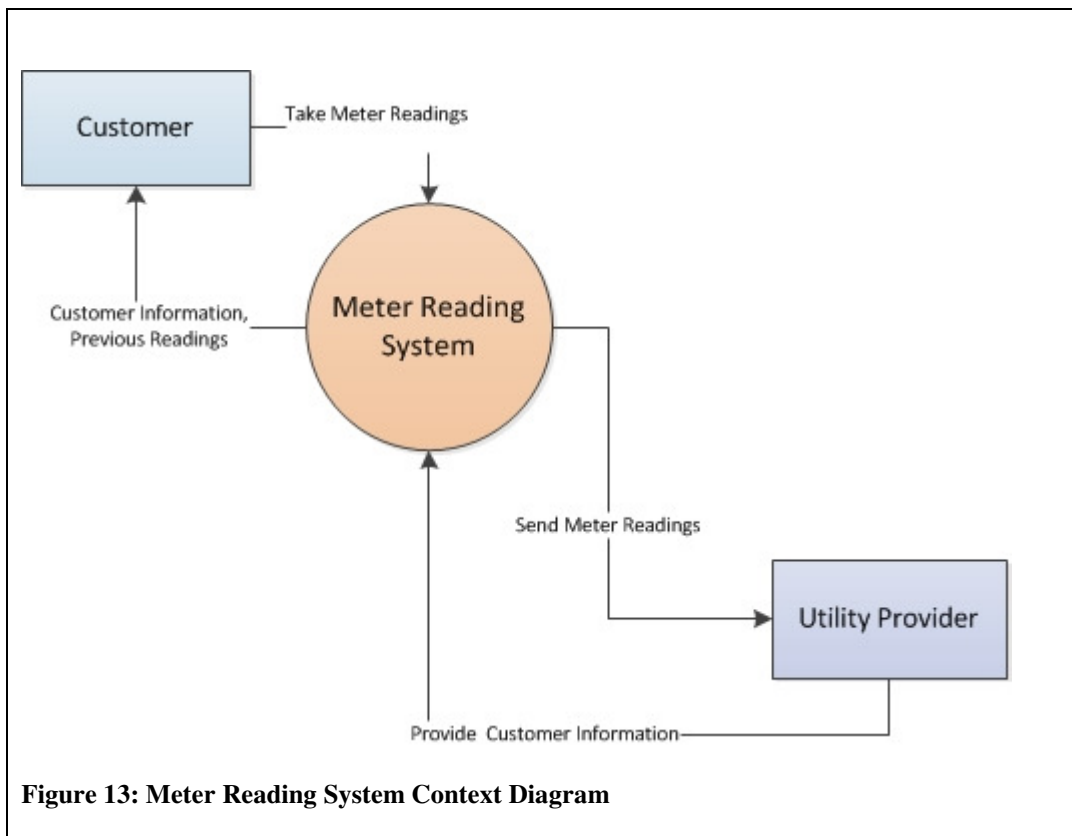
1. The system should be able to record details of the customer such as name and account number on the handheld device.
2. The system should be able to store customer details, customer location, current and previous readings at the utility provider's database.
3. Using a handheld device with a camera and wireless or GSM network, a user should be able to point their camera to the water meter and click a button.
4. When the button is clicked, the system should be able to take a digital image of the meter, process the digital image to obtain an accurate meter reading and send the reading to the utility provider database, making reference to the customer account number and name stored on the handheld device.
5. The system should be able to query the utility provider database for the last reading in order to perform data integrity checks. This will ensure the system does not send any incorrect readings to the utility provider database.
6. If the handheld device is equipped with GPS, the system should be able to use the current location information to approximately determine the location of the customer device and use it to verify that the reading has been sent by the respective customer.

Security and trust of the information sent by the customer to the utility provider is an important consideration in the proposed system. This is to ensure that utility providers can rely on the data to collect money from customers without any losses. The following requirements will ensure that the system can be trusted by the utility providers:

1. The system shall only process real-time images of the meters, and shall not allow images taken earlier to be processed.

2. The system shall communicate with the utility providers databases to compare values read from the meters and previous meter readings. Business rules can then be used to ensure data integrity.
3. The communication between the handheld device and the utility providers database will be through a secure channel and data will be encrypted.
4. The use of GPS to confirm customer location. This can later be a mandatory requirement as more and more devices are integrated with GPS.
5. The registering of the handheld devices with utility providers, to identify a device with a particular customer account.

The following is the context diagram for the proposed system, showing how external entities will interact with the system:



Data Requirements

The proposed solution has the following data requirements:

Customer Accounts Details at the Utility Provider

This is the sample data of customer accounts details held by the utility provider. This data does not have to be the actual data for the purposes of evaluating the prototype and can be generated as test data.

Data for Training and Validating the Learning Algorithm

This data will be used to train and validate the machine learning algorithm for extracting meter readings from digital images. This data will be made up of positive and negative examples for the learning algorithm. Positive examples are the digital images of meters, while negative examples are digital images that do not contain meters. Positive examples will be collected by taking real digital images of existing meters. Negative examples on the other hand can be obtained from existing images.

System Design

Use Case Diagram

Below is the use case diagram showing how various users will interact with the proposed system:

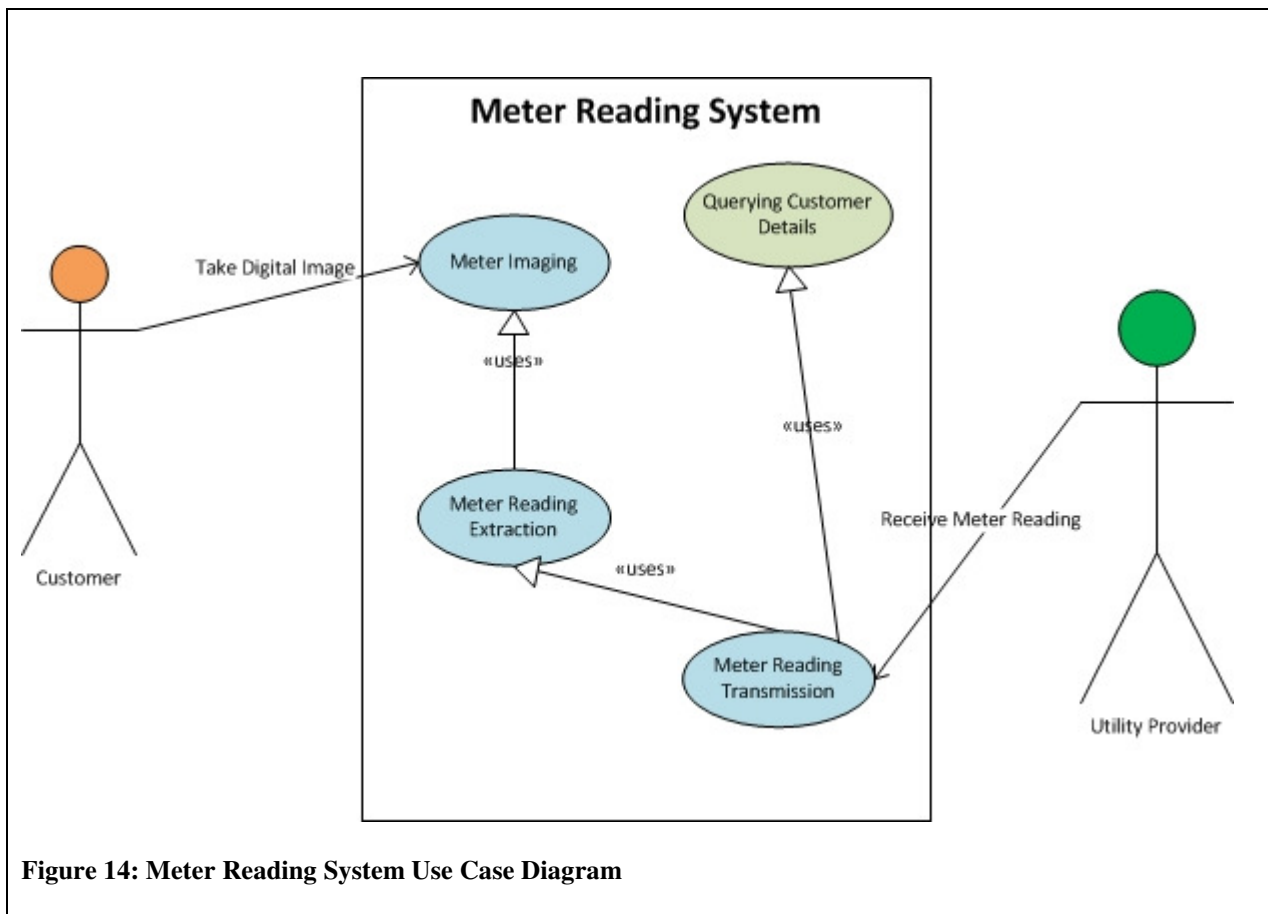


Figure 14: Meter Reading System Use Case Diagram

Entity Relationship Diagrams

The following is the entity relationship diagram for the utility provider database:

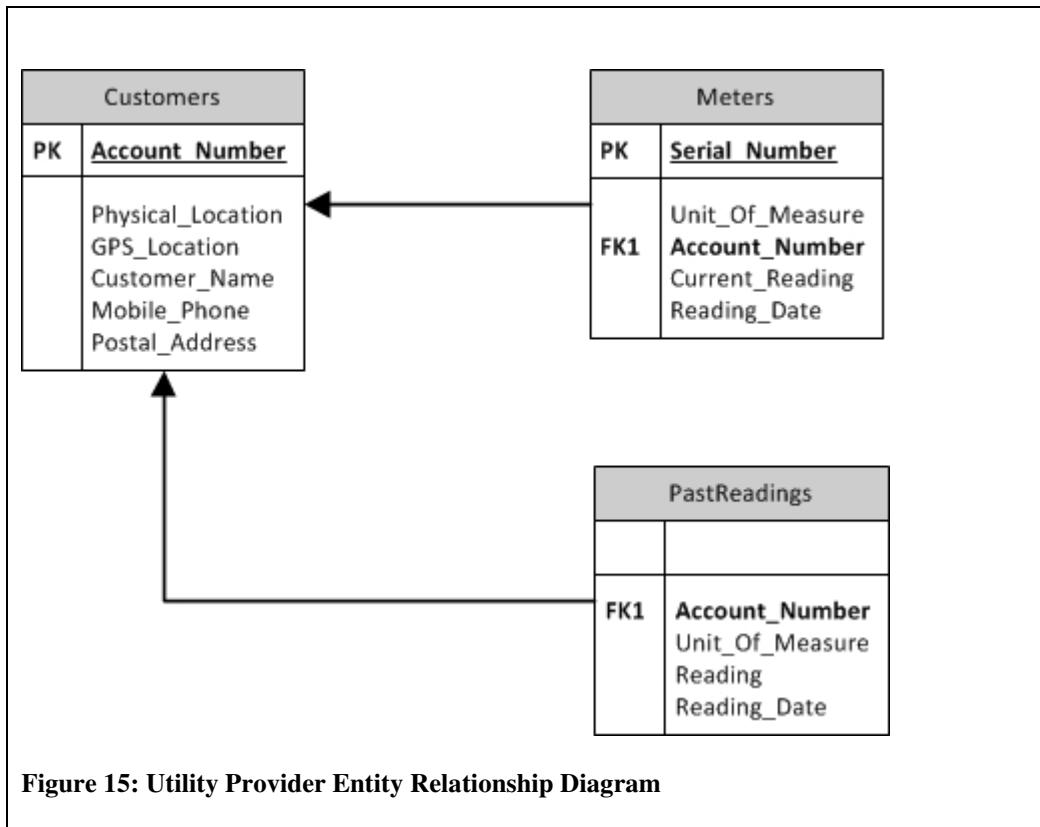


Figure 15: Utility Provider Entity Relationship Diagram

The following is the entity relationship diagram for the customer device database. This database will reside on the customer device:

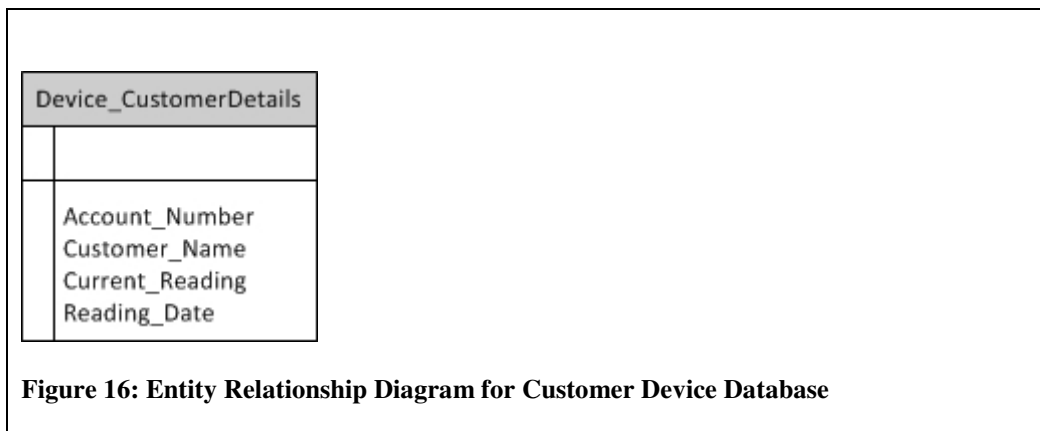
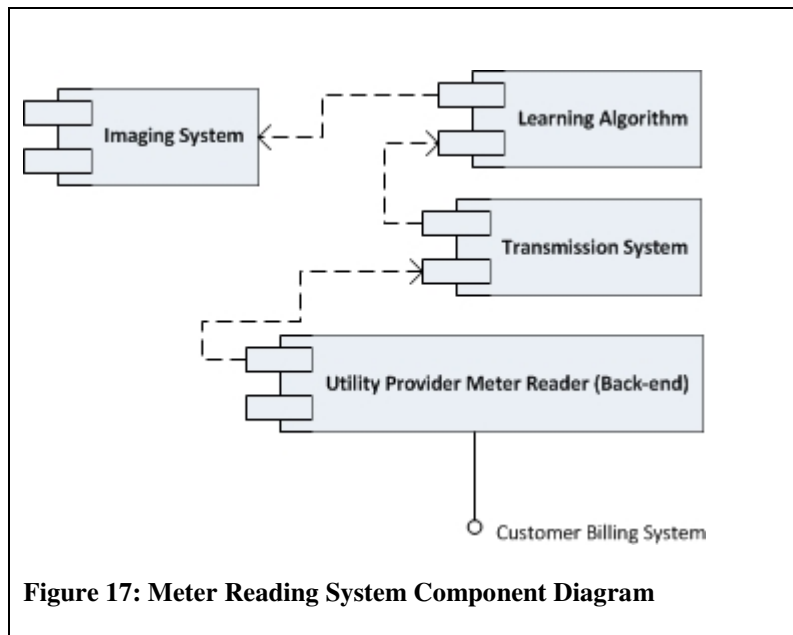


Figure 16: Entity Relationship Diagram for Customer Device Database

Component Diagram

The following is a component diagram showing the various subsystems of the proposed system. The system interfaces with the utility provider customer billing system.



Types of Meters used in the study

Meters can either be analog or digital. However, for the purpose of this study, this distinction is not critical since in most cases the meter reading is a number on the face of the meter, whether the meter is analog or digital.

Meters are also made by different manufacturers. This means that the shape of the meter and the information on the face of the meter are not standardized. This study focused on conventional meters with a round face commonly used for reading water meters. Since this study uses machine learning techniques, extending them to other meter types would involve training on those other meter types before the system can be able to read them.

Evaluation of the Learning Algorithm

For evaluating the performance of the object detection algorithm, we use the evaluation method proposed by Agarwal, Awan and Roth (2004) which suggests the use of a Receiver Operating Characteristics (ROC) curve. The ROC curve plots true positive rate (TPR) versus the false positive rate (FPR) under different threshold settings in the learning algorithm.

$$\text{True Positive Rate (TPR)} = TP / nP,$$

where TP is the number of true positives and nP is the total number of positives in the dataset.

$$\text{False Positive Rate (FPR)} = FP / nN,$$

where FP is the number of false positives and nN is the total number of negatives in the dataset.

Agarwal, Awan and Roth (2004) also suggest a variation of the recall precision curve, where the interest is in the trade-off between the number of objects detected (recall) and how often the detections made are false (precision). This is represented as:

$$\text{Recall} = TP / nP.$$

$$\text{Precision} = TP / TP + FP.$$

$$1\text{-Precision} = FP / TP + FP.$$

The proportion of objects that are detected is given by recall, which is equivalent to the TPR. The number of false detections, relative to the total number of detections by the classifiers is 1-Precision quantity.

Recall versus 1-Precision curves therefore can also be plotted to show the tradeoff between recall and precision.

CHAPTER 5: SYSTEM IMPLEMENTATION AND RESULTS

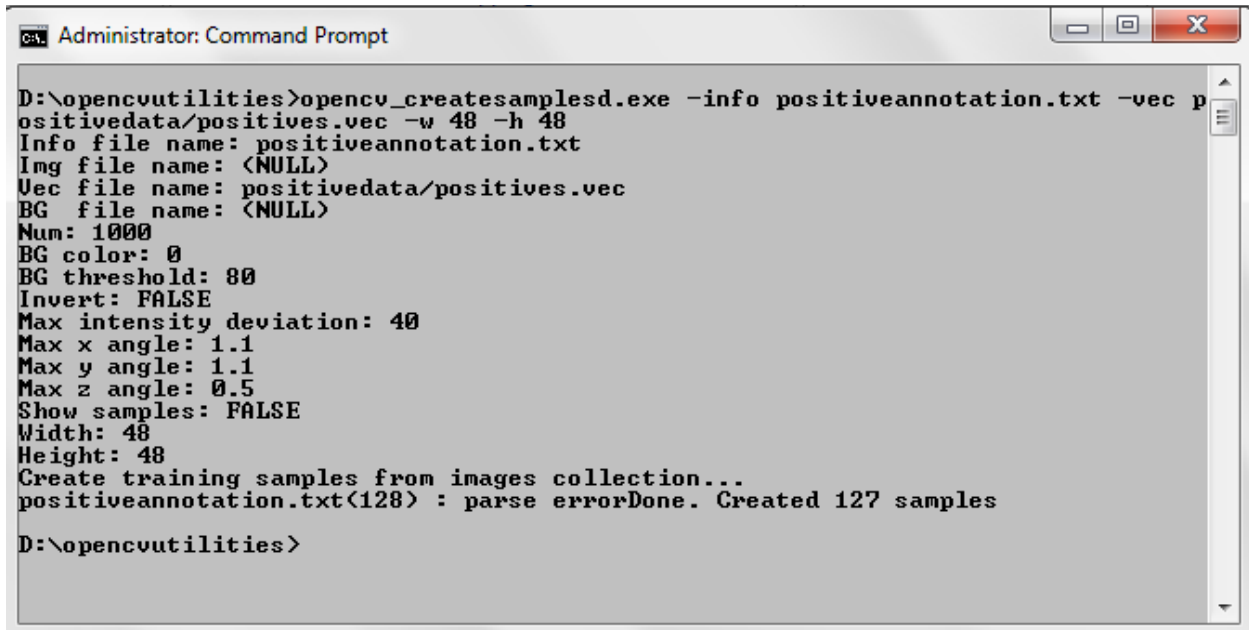
5.1 Adaboost Learning Algorithm Implementation

The learning algorithm for detecting and reading meters was implemented using the OpenCV library version 2.4.9. OpenCV is an extensive open source library developed by Intel, with over 2500 optimized machine learning and computer vision algorithms. Of importance to this study is that the library can be used to recognize and detect objects and provides libraries for classifier training in object detection. OpenCV also offers many functions for manipulation of digital images, which is also useful when detecting objects in digital images.

Another major advantage of OpenCV is that it is a cross-platform application interface. It is available in various programming languages such as Java, C++, C and Python. In addition, there are several other libraries that have been written for other programming languages such as Microsoft Visual Studio C#. The OpenCV libraries can also be ported to handheld devices such as smartphones and have versions for both Android platform and iPhone smartphone. This means that while the prototype has been created on a computer desktop, it can easily be ported to a smartphone.

After performing data cleanup and preparation on the positive and negative example images, the OpenCV tool for creating samples was used to create a vector file with all the positive images. This is shown in the Figure 18.

The image width and height of 48X48 pixels is the sub-window size of the Adaboost Haar training algorithm, which is slided across an image during the training process to calculate the image features. The image size of 48x48 is also the smallest image that the classifier can detect once training is done and its choice is therefore important. This size was chosen as a compromise between the time for training and the reasonable size of a meter head. A higher value will be slower to train and detect, but may also offer better rates of recall. Other window sizes were also used during the training sessions for performance comparison. These include 54x54 pixels and 24x24 pixels.



```
Administrator: Command Prompt
D:\opencvutilities>opencv_createsamples.exe -info positiveannotation.txt -vec p
ositedata/positives.vec -w 48 -h 48
Info file name: positiveannotation.txt
Img file name: <NULL>
Vec file name: positedata/positives.vec
BG file name: <NULL>
Num: 1000
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Width: 48
Height: 48
Create training samples from images collection...
positiveannotation.txt(128) : parse errorDone. Created 127 samples
D:\opencvutilities>
```

Figure 18: Creating Positive Samples Vector File

The OpenCV tool for Adaboost training was then used to train the learning algorithm. The training process was initiated as shown in Figure 19.

In the training instance in Figure 19, the number of positive training examples was 238. A lower figure of 215 ie, 0.9 times the number of positive examples was used in the training session. This gives room for the training algorithm to pick other positive images, if a positive image is excluded during the training session. The number of negative examples was slightly less than half of the positive examples.

The number of stages for the training is 10, corresponding to the cascades in the classifier as shown in Figure 12. We also note that the type of features are Haar basis functions and the boosting type is Gentle Adaboost. The training process will select a maximum of 100 features from the many features available in the training examples. These are the features that can correctly classify an image as containing the meter or not. When the algorithm reaches a False Positive Rate of 0.5, the training algorithm stops. For example, in Figure 19, the false positive rate (also known as false alarm rate) at stage 0 is 0.0181818. This means that the training process will proceed to stage 1. The training process in this case was exhausted at stage 4 and thus the learning algorithm will perform its detection in a cascade of 5 stages.


```

D:\opencvutilities>opencv_traincascaded.exe -data trainout/ -vec rawdata/positives.vec
-bg negative/negative.txt -numPos 215 -numNeg 110 -numStages 10 -precalcValBufSize 128
-precalcIdxBufSize 128 -baseFormatSave -featureType HAAR -w 48 -h48 -minHitRate 0.999
-maxFalseAlarmRate 0.500000 -weightTrimRate 0.95 -maxDepth1 -maxWeakCount 100 -mode ALL
PARAMETERS:
cascadeDirName: trainout/
vecFileName: rawdata/positives.vec
bgFileName: negative/negative.txt
numPos: 215
numNeg: 110
numStages: 10
precalcValBufSize[Mb] : 128
precalcIdxBufSize[Mb] : 128
stageType: BOOST
featureType: HAAR
sampleWidth: 48
sampleHeight: 48
boostType: GAB
minHitRate: 0.999
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100

===== TRAINING 0-stage =====
<BEGIN
POS count : consumed  215 : 215
NEG count : acceptanceRatio  110 : 1
Precalculation time: 14.278
+-----+-----+-----+
| N |  HR |  FA |
+-----+-----+-----+
| 1 |    1|0.0181818|
+-----+-----+-----+
END>

```

Figure 19: Algorithm Training Process

The result of the training process is a classifier configuration file with the thresholds at every stage in the training process. The configuration file had to be modified by changing one of the elements to make it usable with the application. The third level element, `<cascade>` had to be changed to make it `<output type_id="opencv-haar-classifier">`. The configuration file was then converted using the tool shown in Figure 20:

```

Administrator: Command Prompt

C:\OpenCV246\bin\Debug>c-example-convert_cascade.exe --size=48x48 d:\opencvutili
ties\trainout\cascade.xml cascade_test.xml

This sample demonstrates cascade's convertation
Usage:
./convert_cascade --size="<width>x<height>"<convertation size>
                    input_cascade_path
                    output_cascade_filename
Example:
./convert_cascade --size=640x480 ../../opencv/data/haarcascades/haarcascade_eye.
xml ../../opencv/data/haarcascades/test_cascade.xml
C:\OpenCV246\bin\Debug>

```

Figure 20: Converting the Cascade File

A sample of the file for the training process in Figure 19 is given in Appendix II, showing two of the five stages. This is the file used by the detection function to detect the presence of a meter on a new image. During the implementation, different sizes of the training dataset were used to provide performance measures. However, training the algorithm takes time. For example, training on a sample of 520 positive examples and 230 negative examples took over 24 hours on a 3GB RAM, 2.4 GHz 2 Core Intel processor.

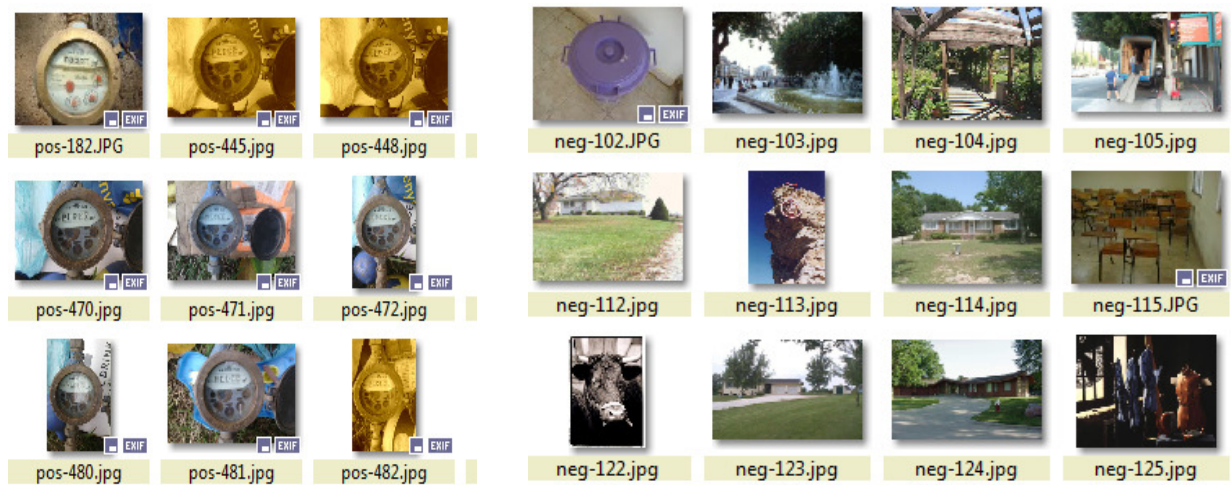


Figure 21: The Training Data Set, Positive And Negative Examples

5.2 Meter Reading Prototype

The classifier configuration file was used to detect meters in an application created in the Microsoft Visual Studio C# programming language. The database was implemented in MySQL. Because OpenCV is not originally available in C#, the Emgu project wrapper tool, version 2.4.9.187, was used. This project contains a wrapper application which uses OpenCV libraries to provide native machine learning programming in C#. A screenshot of the application is shown in Figure 22.

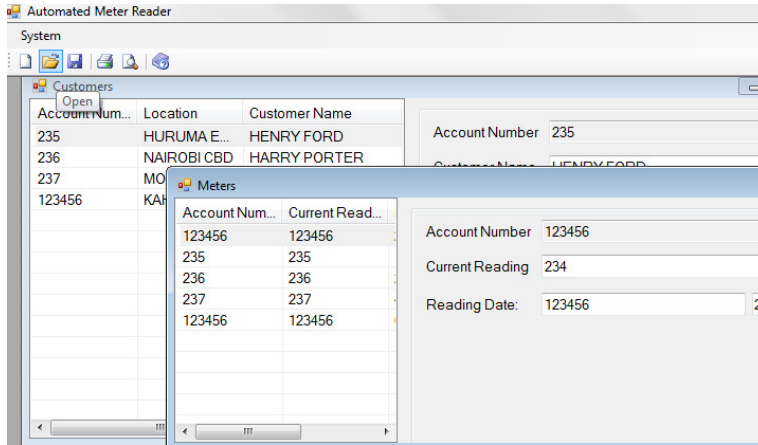


Figure 22: Screen Shot Showing Customers

Figure 23 shows a screenshot of the application with a detected meter. The first step is to open a test image by clicking the Detect button. The meter region is detected using the trained algorithm configuration file. The red bound shows a meter region that has been detected.

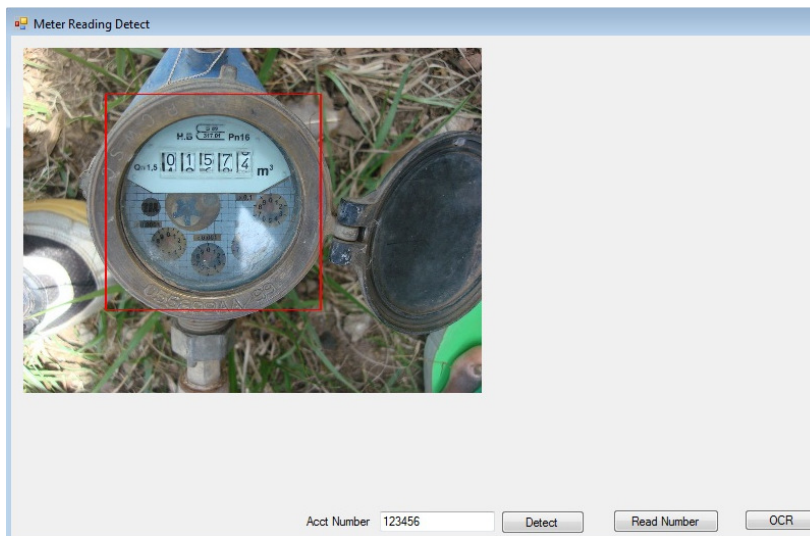


Figure 23: Screen shot with Meter Detection

The second step is to use template matching to focus on the meter reading, by clicking the Read Number button. This results in the meter reading region being detected as shown in Figure 24.

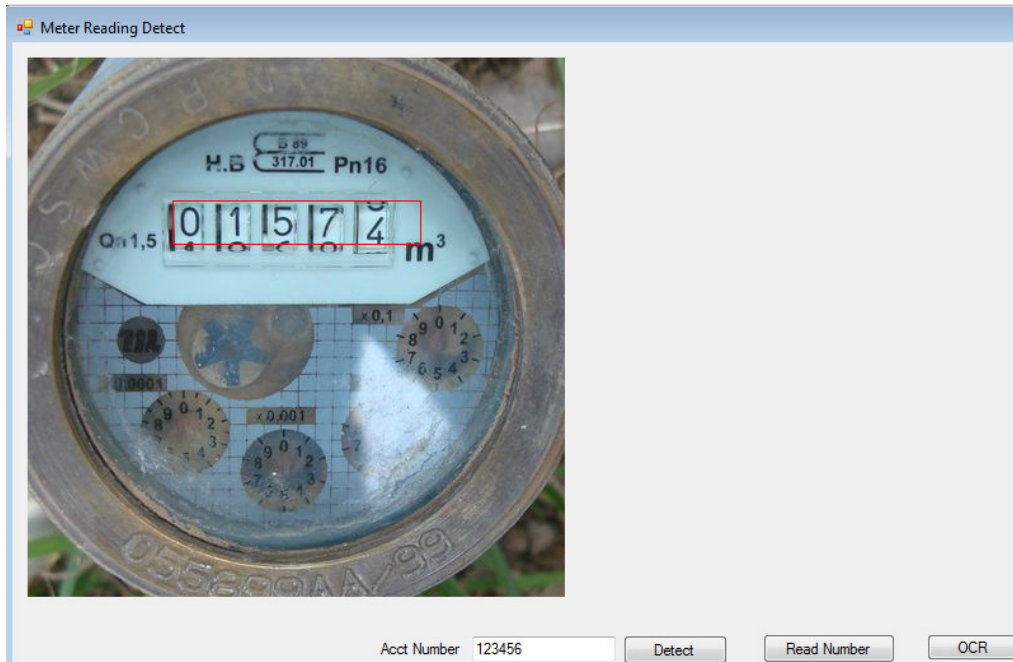


Figure 24: Meter Reading Detection

Binarization is done using a combination of image processing techniques including histogram equalization, Canny edge detection and binary thresholding. The binarized image, shown in Figure 25, was then presented to the Tesseract open source OCR engine for character recognition.

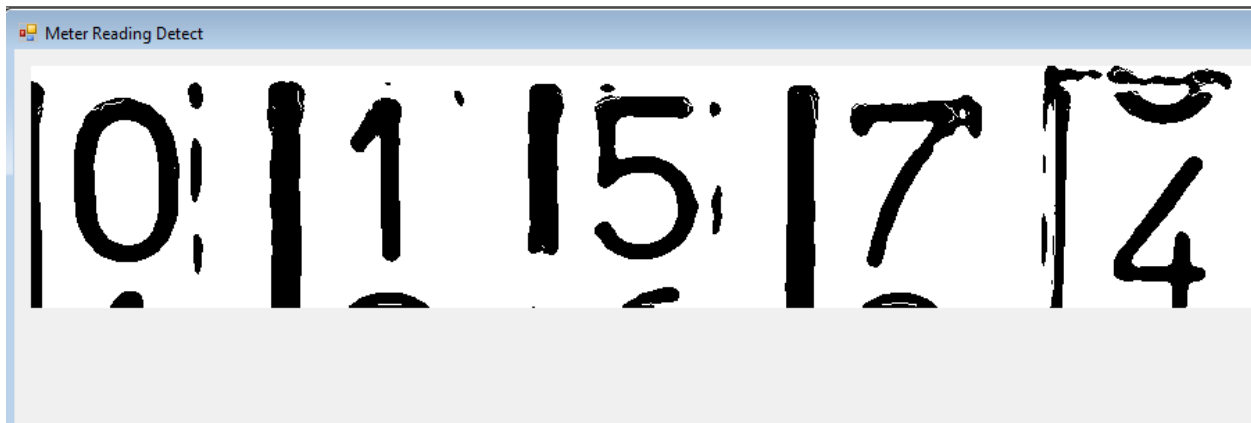


Figure 25: Binarized Image Of The Meter Reading For OCR Detection

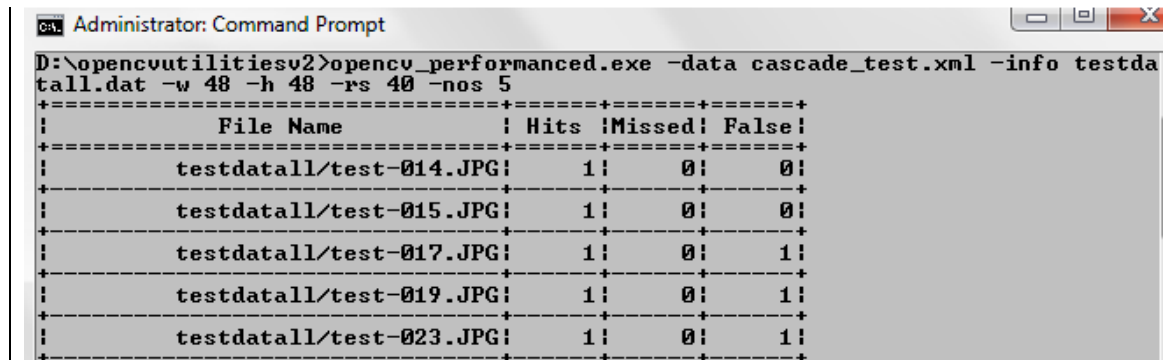
5.3 Results

The performance of the trained classifier on test data was determined using the OpenCV performance tool and can be presented, showing how the settings of the various training sessions affected the performance achieved.

Adaboost Algorithm Performance Results

Several training sessions of the Adaboost algorithm were conducted to determine its optimal performance on test data. The distinction between one training session and another was in the training dataset, i.e., the number of positive and negative examples, the scanning window size and the type of features selected, whether Haar features or local binary pattern (LBP) features.

Performance was measured using the OpenCV tool for evaluating performance on the test data. Figure 26 shows a performance test run for the resulting classifier configuration file of an Adaboost training session. The scanning window is of size 48x48 pixels, the number of ROC curve points is 40 and the number of stages is 5.



```
Administrator: Command Prompt
D:\opencvutilitiesv2>opencv_performance.exe -data cascade_test.xml -info testda
tall.dat -w 48 -h 48 -rs 40 -nos 5
=====+=====+=====+
|          File Name          | Hits | Missed | False |
|-----+-----+-----+
| testdata1/test-014.JPG |    1 |     0 |    0 |
|-----+-----+-----+
| testdata1/test-015.JPG |    1 |     0 |    0 |
|-----+-----+-----+
| testdata1/test-017.JPG |    1 |     0 |    1 |
|-----+-----+-----+
| testdata1/test-019.JPG |    1 |     0 |    1 |
|-----+-----+-----+
| testdata1/test-023.JPG |    1 |     0 |    1 |
|-----+-----+-----+
```

Figure 26: OpenCV Performance Measure Tool

This resulted in the performance results shown in Figure 27.

testdata1/pos-590.jpg		1	0	0
Total		106	12	294
Number of stages: 5				
Number of weak classifiers: 13				
Total time: 5.478000				
5				
106	294	0.898305	2.491525	
106	294	0.898305	2.491525	
106	294	0.898305	2.491525	
100	216	0.847458	1.830508	
98	168	0.830508	1.423729	
94	132	0.796610	1.118644	
91	108	0.771186	0.915254	
88	84	0.745763	0.711864	
85	67	0.720339	0.567797	
83	59	0.703390	0.500000	
82	51	0.694915	0.432203	
81	49	0.686441	0.415254	
79	46	0.669492	0.389830	
78	42	0.661017	0.355932	
78	41	0.661017	0.347458	
76	40	0.644068	0.338983	
74	37	0.627119	0.313559	
72	36	0.610169	0.305085	
70	36	0.593220	0.305085	
69	32	0.584746	0.271186	
68	30	0.576271	0.254237	

Figure 27: Results Of Opencv Performance Tool Run

The number of Haar basis features selected for this classifier as shown is 13 out of the many features in the training examples. These are the weak classifiers (features) which are combined to create a strong classifier, and that can partition the positive and negative examples to enable the classifier to detect meters and reject other objects.

The OpenCV performance tool (Figure 26) takes as input the description file with the exact location of meters in the test images and the classifier configuration file from Adaboost training session. An example description file is shown in Figure 28, and Appendix II has an example classifier configuration file.

1 testdatall.dat					
1	testdatall/test-014.JPG	1	91	28	227 224
2	testdatall/test-015.JPG	1	74	2	261 244
3	testdatall/test-017.JPG	1	18	17	273 285
4	testdatall/test-019.JPG	1	29	52	256 248
5	testdatall/test-023.JPG	1	41	39	249 265
6	testdatall/test-027.JPG	1	26	32	267 280
7	testdatall/test-029.JPG	1	7	18	53 54
8	testdatall/test-031.JPG	1	49	91	209 202
9	testdatall/test-033.JPG	1	51	68	214 200
10	testdatall/test-034.JPG	1	65	88	214 205
11	testdatall/test-037.JPG	1	43	60	222 210

Figure 28: Description File For Test Data With Meter Coordinates.

The first column in the description file in Figure 28 has the test image, the second column contains the number of meter objects within the image, which in this case is 1. The third and fourth columns are the top left X and Y coordinates of the meter region, and lastly the fifth and sixth columns are the width and height of the meter region. The description file therefore gives the exact location of the meter object within the test image.

The OpenCV performance tool measures the number of hits, misses and false detections for each test image as shown in Figure 26. The tool does this by performing an actual meter detection on the test image. It then compares the detected regions with the description file. If the detected region lies within the given meter region as in the description file, it is counted as a hit. If other regions within the test image are detected as meters, this is recorded as a false detection or false alarm. Finally, if there is a meter in the test image, but the the algorithm does not detect it, this is considered a miss. A false detection can be made at the same time as a hit and a miss.

The OpenCV performance tool then summarises the performance results as shown in Figure 27. The first column shows the true positives (hits), the second column shows the false positives(false alarms), while the third and fourth columns show the true positive rates and false positive rates respectively. It is this information that can be used to plot a ROC curve and calculate other measures such as 1-Precision. From Figure 26, the ROC curve size is 40, which means that the summarized results will have 40 points or rows as shown in Table 1 – Table 6.

Performance Comparison based on Training DataSet

The first training was conducted with 238 positive examples and 110 negative examples, 48x48 pixels scanning window size and Haar features. This resulted in a 5-stage classifier with 13 weak classifiers and the detection for all the 118 test images took 16.8 seconds. The results of the first training achieved 89.8 % true positive rate and 249.1% false positive rate as shown in the last row of Table 1:

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
51	4	0.432203	0.033898	0.072727
51	4	0.432203	0.033898	0.072727
50	4	0.423729	0.033898	0.074074
51	5	0.432203	0.042373	0.089286
53	6	0.449153	0.050847	0.101695
53	7	0.449153	0.059322	0.116667
53	8	0.449153	0.067797	0.131148
55	9	0.466102	0.076271	0.140625
55	9	0.466102	0.076271	0.140625
57	11	0.483051	0.09322	0.161765
61	13	0.516949	0.110169	0.175676
58	13	0.491525	0.110169	0.183099
61	14	0.516949	0.118644	0.186667
62	15	0.525424	0.127119	0.194805
64	16	0.542373	0.135593	0.2
65	18	0.550847	0.152542	0.216867
66	20	0.559322	0.169492	0.232558
66	23	0.559322	0.194915	0.258427
68	27	0.576271	0.228814	0.284211
68	30	0.576271	0.254237	0.306122
69	32	0.584746	0.271186	0.316832
72	36	0.610169	0.305085	0.333333
70	36	0.59322	0.305085	0.339623
74	37	0.627119	0.313559	0.333333
76	40	0.644068	0.338983	0.344828
78	41	0.661017	0.347458	0.344538
78	42	0.661017	0.355932	0.35
79	46	0.669492	0.38983	0.368
81	49	0.686441	0.415254	0.376923
82	51	0.694915	0.432203	0.383459
83	59	0.70339	0.5	0.415493
85	67	0.720339	0.567797	0.440789
88	84	0.745763	0.711864	0.488372
91	108	0.771186	0.915254	0.542714
94	132	0.79661	1.118644	0.584071
98	168	0.830508	1.423729	0.631579
100	216	0.847458	1.830508	0.683544
106	294	0.898305	2.491525	0.735

106	294	0.898305	2.491525	0.735
106	294	0.898305	2.491525	0.735

Table 1: Performance Results Of The First Training Session On Test Data.

The second training session was conducted with 513 positive examples and 230 negative examples, 48x48 pixels scanning window size and Haar features. This resulted in a 10-stage classifier with 42 weak classifiers classifiers and the detection for all the 118 test images took 19.0 seconds. This training session achieved 75.4 % true positive rate and 19.4% false positive rate as shown in Table 2:

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
0	0	0	0	0
0	0	0	0	0
1	0	0.008475	0	0
4	0	0.033898	0	0
5	0	0.042373	0	0
6	0	0.050847	0	0
9	0	0.076271	0	0
11	0	0.09322	0	0
14	0	0.118644	0	0
19	0	0.161017	0	0
26	0	0.220339	0	0
33	1	0.279661	0.008475	0.029412
39	1	0.330508	0.008475	0.025
47	1	0.398305	0.008475	0.020833
54	1	0.457627	0.008475	0.018182
61	1	0.516949	0.008475	0.016129
66	1	0.559322	0.008475	0.014925
72	1	0.610169	0.008475	0.013699
78	1	0.661017	0.008475	0.012658
79	1	0.669492	0.008475	0.0125
81	1	0.686441	0.008475	0.012195
82	1	0.694915	0.008475	0.012048
84	2	0.711864	0.016949	0.023256
84	1	0.711864	0.008475	0.011765
85	3	0.720339	0.025424	0.034091
88	8	0.745763	0.067797	0.083333
89	23	0.754237	0.194915	0.205357
89	23	0.754237	0.194915	0.205357
89	23	0.754237	0.194915	0.205357

Table 2: Performance Results Of The Second Training Session On Test Data

The third training session had 485 positive examples and 835 negative examples, 48x48 pixels scanning window size and Haar features. This resulted in a 4-stage classifier with 13 weak classifiers classifiers and the detection for all the 118 images took 19.7 seconds. A 65.2 % true positive rate and 22.8% false positive rate was achieved as shown in Table 3:

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
0	0	0	0	0
0	0	0	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
4	0	0.033898	0	0
5	0	0.042373	0	0
6	0	0.050847	0	0
7	0	0.059322	0	0
8	0	0.067797	0	0
12	0	0.101695	0	0
15	0	0.127119	0	0
18	0	0.152542	0	0
22	0	0.186441	0	0
27	0	0.228814	0	0
31	0	0.262712	0	0
34	0	0.288136	0	0
36	0	0.305085	0	0
38	0	0.322034	0	0
38	0	0.322034	0	0
42	0	0.355932	0	0
43	0	0.364407	0	0
44	0	0.372881	0	0
47	1	0.398305	0.008475	0.020833
47	1	0.398305	0.008475	0.020833
48	1	0.40678	0.008475	0.020408
48	1	0.40678	0.008475	0.020408
51	1	0.432203	0.008475	0.019231
54	1	0.457627	0.008475	0.018182
56	1	0.474576	0.008475	0.017544
59	1	0.5	0.008475	0.016667
62	2	0.525424	0.016949	0.03125
65	3	0.550847	0.025424	0.044118

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
69	7	0.584746	0.059322	0.092105
73	15	0.618644	0.127119	0.170455
77	27	0.652542	0.228814	0.259615
77	27	0.652542	0.228814	0.259615
77	27	0.652542	0.228814	0.259615

Table 3: Performance Results Of The Third Training Session On Test Data

The fourth training session had 323 positive examples and 600 negative examples, 54x54 pixels scanning window size and Haar features. This resulted in a 5-stage classifier with 12 weak classifiers and the detection for all the 118 test images took 14.4 seconds.

A 62.7 % true positive rate and 83.8% false positive rate was achieved as shown in Table 4:

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
1	0	0.008475	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
2	0	0.016949	0	0
2	0	0.016949	0	0
2	0	0.016949	0	0
3	0	0.025424	0	0
3	0	0.025424	0	0
3	0	0.025424	0	0
3	0	0.025424	0	0
4	0	0.033898	0	0
5	0	0.042373	0	0
5	0	0.042373	0	0
7	0	0.059322	0	0
8	0	0.067797	0	0
9	0	0.076271	0	0
9	0	0.076271	0	0
13	1	0.110169	0.008475	0.071429
13	1	0.110169	0.008475	0.071429

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
14	2	0.118644	0.016949	0.125
18	4	0.152542	0.033898	0.181818
20	4	0.169492	0.033898	0.166667
22	7	0.186441	0.059322	0.241379
24	10	0.20339	0.084746	0.294118
27	15	0.228814	0.127119	0.357143
32	23	0.271186	0.194915	0.418182
38	32	0.322034	0.271186	0.457143
45	37	0.381356	0.313559	0.45122
48	42	0.40678	0.355932	0.466667
55	52	0.466102	0.440678	0.485981
61	59	0.516949	0.5	0.491667
65	77	0.550847	0.652542	0.542254
74	99	0.627119	0.838983	0.572254
74	99	0.627119	0.838983	0.572254
74	99	0.627119	0.838983	0.572254

Table 4: Performance Results Of The Fourth Training Session On Test Data

The fifth training session was conducted with 530 positive examples and 944 negative examples, 54x54 pixels scanning window size and Haar features. This resulted in a 4-stage classifier with 9 weak classifiers and the detection for all the 118 test images took 14.1 seconds. This training achieved 75.4% true positive rate and 18.6% false positive rate as shown in Table 5:

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
0	0	0	0	0
0	0	0	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
1	0	0.008475	0	0
2	0	0.016949	0	0
4	0	0.033898	0	0
4	0	0.033898	0	0
5	0	0.042373	0	0
6	0	0.050847	0	0
8	0	0.067797	0	0
10	0	0.084746	0	0
14	0	0.118644	0	0

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
17	0	0.144068	0	0
19	0	0.161017	0	0
19	0	0.161017	0	0
24	0	0.20339	0	0
24	0	0.20339	0	0
27	0	0.228814	0	0
30	0	0.254237	0	0
34	0	0.288136	0	0
36	0	0.305085	0	0
39	1	0.330508	0.008475	0.025
47	1	0.398305	0.008475	0.020833
49	1	0.415254	0.008475	0.02
53	1	0.449153	0.008475	0.018519
65	1	0.550847	0.008475	0.015152
71	1	0.601695	0.008475	0.013889
79	3	0.669492	0.025424	0.036585
83	9	0.70339	0.076271	0.097826
89	22	0.754237	0.186441	0.198198
89	22	0.754237	0.186441	0.198198
89	22	0.754237	0.186441	0.198198

Table 5: Performance Results Of The Fifth Training Session On Test Data.

The sixth training session was conducted with 530 positive examples and 600 negative examples, 48x48 pixels scanning window size and Haar features. This resulted in a 6-stage classifier with 16 weak classifiers and the detection for all the 118 test images took 14.8 seconds, achieving a 72.8% true positive rate and 12.7% false positive rate as shown in Table 6:

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
0	0	0	0	0
0	0	0	0	0
1	0	0.008475	0	0
3	0	0.025424	0	0
4	0	0.033898	0	0
5	0	0.042373	0	0
7	0	0.059322	0	0
7	0	0.059322	0	0
8	0	0.067797	0	0

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)	1-Precision (FP/(TP+FP))
8	0	0.067797	0	0
8	0	0.067797	0	0
8	0	0.067797	0	0
9	1	0.076271	0	0
9	1	0.076271	0	0
9	0	0.076271	0	0
9	0	0.076271	0	0
9	0	0.076271	0.008475	0.1
9	0	0.076271	0.008475	0.1
11	1	0.09322	0.008475	0.083333
14	1	0.118644	0.008475	0.066667
20	1	0.169492	0.008475	0.047619
22	1	0.186441	0.008475	0.043478
25	1	0.211864	0.008475	0.038462
28	1	0.237288	0.008475	0.034483
32	1	0.271186	0.008475	0.030303
34	1	0.288136	0.008475	0.028571
38	1	0.322034	0.008475	0.025641
41	1	0.347458	0.008475	0.02381
45	1	0.381356	0.008475	0.021739
52	1	0.440678	0.008475	0.018868
61	2	0.516949	0.016949	0.031746
71	3	0.601695	0.025424	0.040541
77	7	0.652542	0.059322	0.083333
86	15	0.728814	0.127119	0.148515
86	15	0.728814	0.127119	0.148515
86	15	0.728814	0.127119	0.148515

Table 6: Results Of Sixth Training Session

The seventh training session was conducted with 530 positive examples and 600 negative examples, 24x24 pixels scanning window size and Haar features in order to see the effect of halving the window size. This resulted in a 5-stage classifier with 12 weak classifiers and the detection for all the 118 test images took 5.8 seconds, achieving a 57.6% true positive rate and 25.4 % false positive rate as shown in Table 7:

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
1	0	0.008475	0
3	0	0.025424	0
7	0	0.059322	0
17	0	0.144068	0
30	1	0.254237	0.008475
49	1	0.415254	0.008475
61	6	0.516949	0.050847
68	30	0.576271	0.254237
68	30	0.576271	0.254237
68	30	0.576271	0.254237

Table 7: Results of Seventh Training Session With 24x24 Window Size

Comparing Lower and higher Stages of Training.

For every training algorithm lower stages show higher recalls, but also lower precision. This is shown in the last row of Table 8, which shows the 9-stage classifier results for the 10-stage classifier in Table 2. The classifier selects 36 weak features.

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)
0	0	0	0
0	0	0	0
1	0	0.008475	0
1	0	0.008475	0
3	0	0.025424	0
4	1	0.033898	0.008475
7	1	0.059322	0.008475
10	1	0.084746	0.008475
11	1	0.09322	0.008475
11	1	0.09322	0.008475
13	2	0.110169	0.016949
16	2	0.135593	0.016949
21	2	0.177966	0.016949
25	2	0.211864	0.016949

28	2	0.237288	0.016949
39	3	0.330508	0.025424
43	5	0.364407	0.042373
48	6	0.40678	0.050847
57	6	0.483051	0.050847
63	7	0.533898	0.059322
69	8	0.584746	0.067797
72	8	0.610169	0.067797
75	9	0.635593	0.076271
77	10	0.652542	0.084746
79	10	0.669492	0.084746
81	12	0.686441	0.101695
83	12	0.70339	0.101695
83	12	0.70339	0.101695
84	13	0.711864	0.110169
85	16	0.720339	0.135593
86	20	0.728814	0.169492
88	32	0.745763	0.271186
89	89	0.754237	0.754237
89	89	0.754237	0.754237
89	89	0.754237	0.754237

Table 8: Results For 9-Stage Classifier For The Classifier In Table 2.

In Table 9, the 8-stage classifier results for the 10-stage classifier in Table 2 are shown. This selects 31 weak classifiers. Even though the training session achieves a 76.2% recall, compared to the 75.4% of both the 9-stage and 10-stage, the classifier has a very high false positive rate of 136%.

Hits (TP)	Missed (FP)	True Positives Rate (TPR)	False Positives Rate (FPR)
65	13	0.550847	0.110169
69	14	0.584746	0.118644
72	15	0.610169	0.127119
75	15	0.635593	0.127119
77	15	0.652542	0.127119
79	15	0.669492	0.127119
81	15	0.686441	0.127119
83	15	0.70339	0.127119
83	15	0.70339	0.127119
85	17	0.720339	0.144068
87	25	0.737288	0.211864
88	39	0.745763	0.330508

89	75	0.754237	0.635593
90	161	0.762712	1.364407
90	161	0.762712	1.364407
90	161	0.762712	1.364407

Table 9: Results For 8-Stage Classifier For The Classifier In Table 2.

Comparing Haar Features and LBP Features

Table 10 shows the false positives count for the performance test results shown in Table 1 for the first 10 test images. The results are for Adaboost training with Haar features.

Test Image File Name	Hits	Missed	False Positives
testdatall/test-014.JPG	1	0	0
testdatall/test-015.JPG	1	0	0
testdatall/test-017.JPG	1	0	1
testdatall/test-019.JPG	1	0	1
testdatall/test-023.JPG	1	0	1
testdatall/test-027.JPG	0	1	1
testdatall/test-029.JPG	1	0	0
testdatall/test-031.JPG	1	0	3
testdatall/test-033.JPG	1	0	2
testdatall/test-034.JPG	1	0	1
Total	9	1	10

Table 10: False Positives for Haar Training

A training session was also conducted with 238 positives and 110 negatives, with similar training settings as in the first training session (Table 1), but with the feature type as LBP. Table 11 shows the false positives count for the performance test results for this Adaboost training with LBP features for the first 10 test images.

Test Image File Name	Hits	Missed	False Positives
testdatall/test-014.JPG	1	0	12
testdatall/test-015.JPG	0	1	10
testdatall/test-017.JPG	1	0	5
testdatall/test-019.JPG	1	0	7
testdatall/test-023.JPG	0	1	9
testdatall/test-027.JPG	0	1	7
testdatall/test-029.JPG	1	0	9
testdatall/test-031.JPG	1	0	8
testdatall/test-033.JPG	1	0	12
testdatall/test-034.JPG	1	0	10

Test Image File Name	Hits	Missed	False Positives
Total	7	3	89

Table 11: False Positives For LBP Training

Training with 530 positive images and 600 negative images, 48x48 window size with LBP features achieved much better results. The training session had the same parameters as in Table 6, but with LBP as the feature type. Because OpenCV does not come with a performance measure tool for LBP classifiers, it was not possible to plot the ROC curves for comparison. A tool was built based on the Haar performance tool to compare the results as shown in the source code, Appendix I. However, it did not work as expected, since the latest LBP object detection module in OpenCV does not return sensitivity data as in the Haar object detection. However, using the tool to compare the results from the Haar algorithm in Table 6, the LBP results were better. LBP achieved 88% recall and very low false positive rate, as compared to 72.8% recall for the Haar training in Table 6, which had similar sample sizes.

Visualization of Results

The results can be visualized in two ways as mentioned in section 4.2. Firstly, visualization using ROC curves, which plots false positive rate against true positive rate. A second way of visualizing the data is using 1-Precision curves which plot 1-Precision against true positive rate. The graphs were plotted with Gnuplot version 4.6.

The ROC curve in Figure 29 compares the results of the first three training sessions, showing performance of the classifier with the different training sample sizes. The results show that the best classifier was achieved with the 513 positives and 230 negatives. This is because of the high rate of recall and precision (low false positives). This can be shown more clearly in the ROC curve in Figure 30, which has plotted only the two best training sessions from Figure 29, i.e., the one with 513 positives and 230 negatives and the other with 485 positives and 835 negatives. The 485 positives training session has both a lower true positive rate (lower recall), but also a slightly higher false positive rate (less precision).

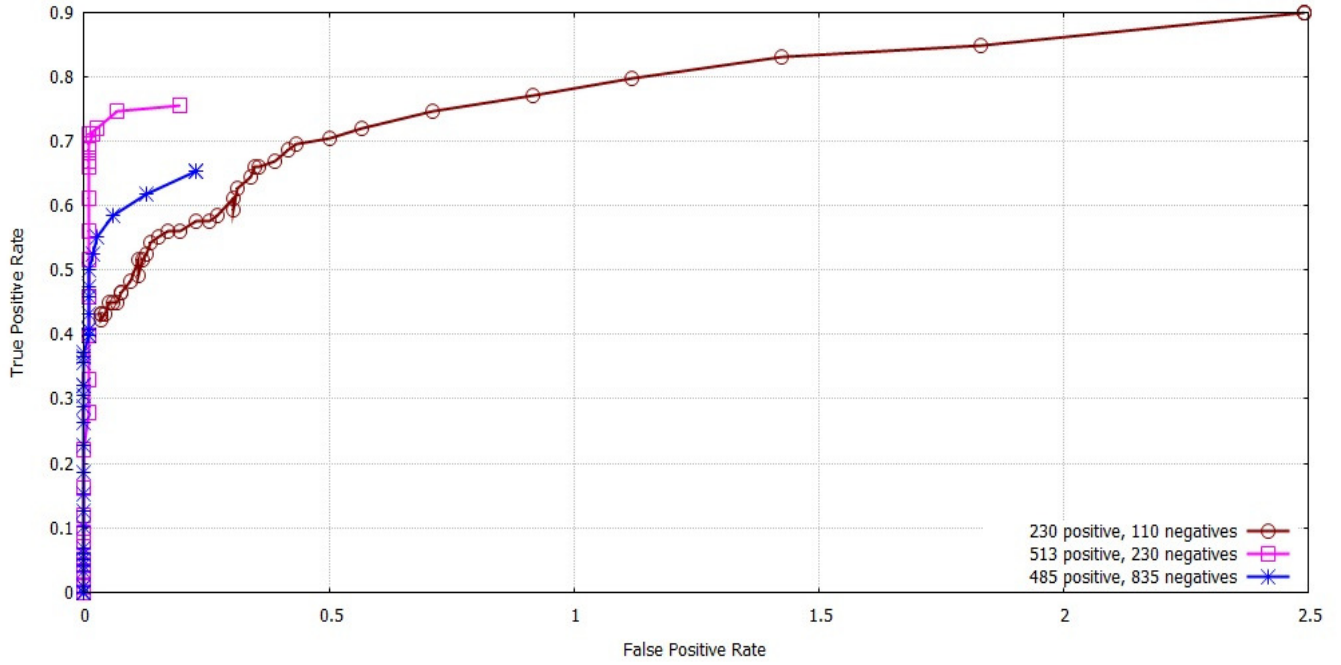


Figure 29: ROC Curve Comparing Performance Based On Training Sample Size.

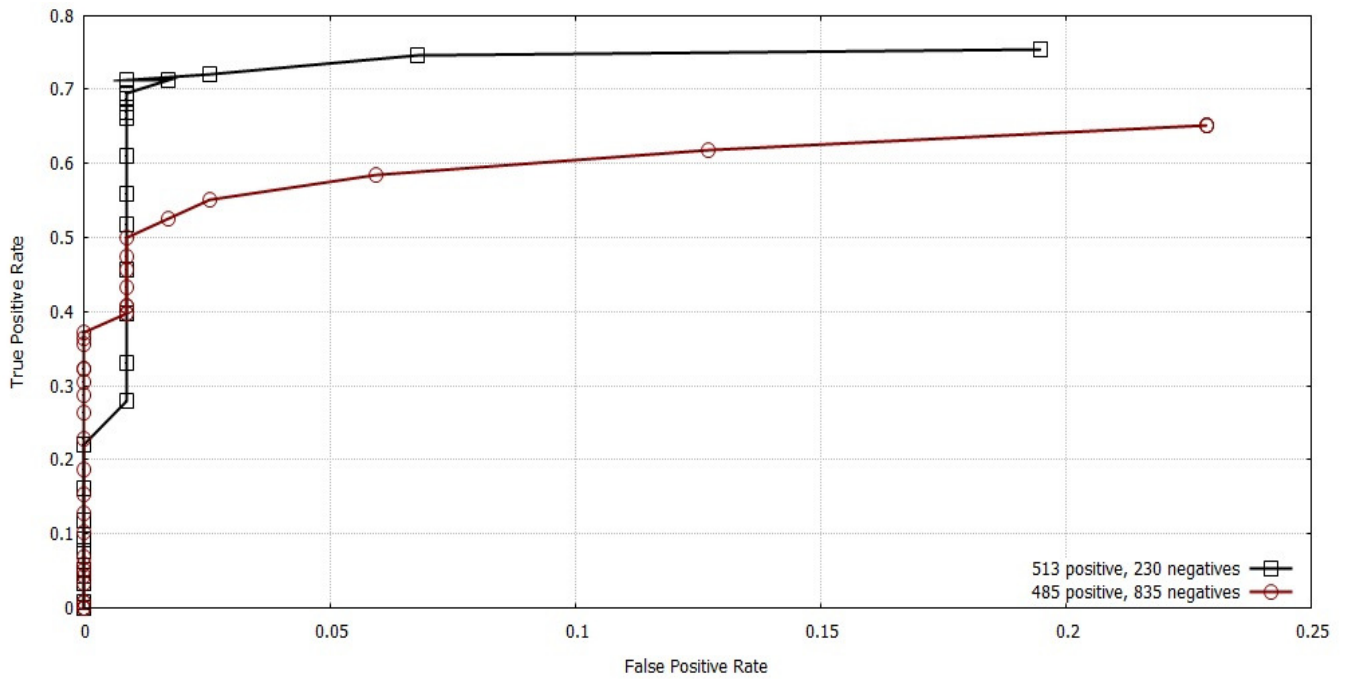


Figure 30: ROC Curve Comparing Performance of Best Two Training Sessions.

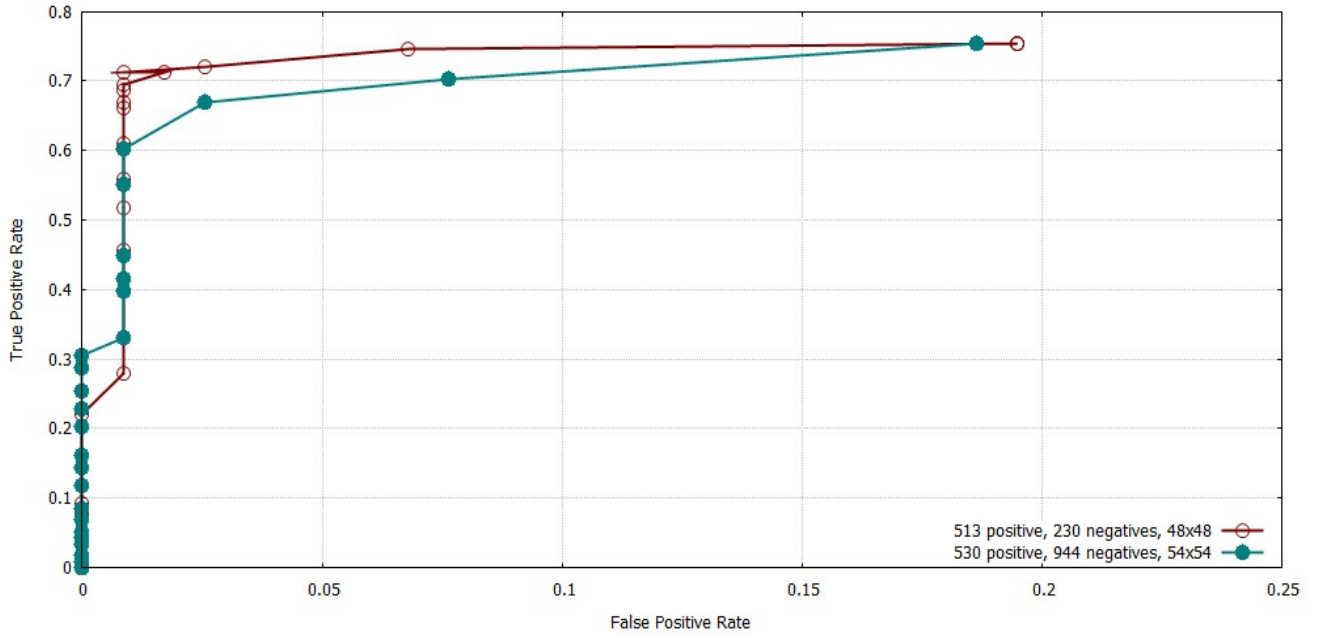


Figure 31: ROC Curve Comparing Performance Based On Scanning Pixel Window Size.

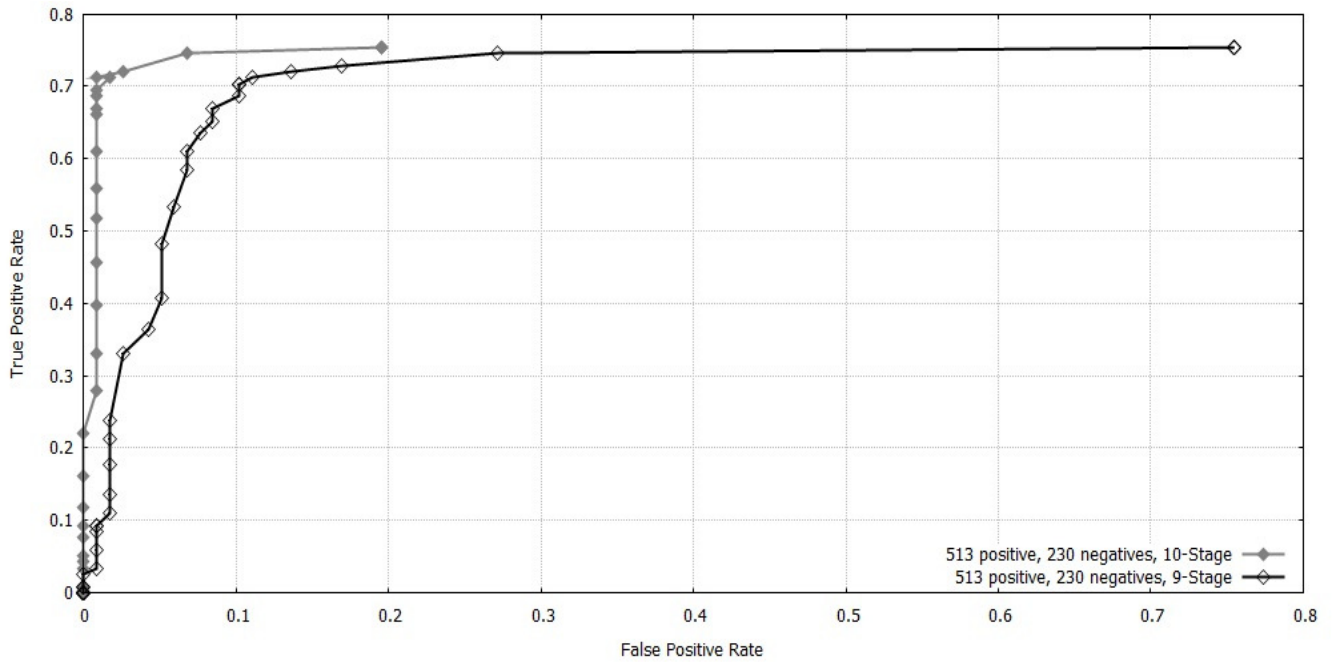


Figure 32: ROC Curve For A 10-Stage Classifier And 9-Stage Classifier.

The ROC curve in Figure 31 on the other hand compares the performance of the best training session with 48x48 pixel window size and the best training session with 54x54 pixel window size. It shows that there is no much difference between the two window sizes and training with either window sizes can achieve good results. However, training with a higher window size is slower than with a smaller window size.

The ROC curve in Figure 32 compares the lower stages and higher stages of object detection. While lower stages may have equal or higher performance rates as higher stages, they have much less precision than higher stages.

Figure 33 shows a 1-Precision Curve for the first 3 results, showing how the 3 trained classifiers compare. This is consistent with the ROC curves in Figure 29 and Figure 30 and shows the best training session with the best performance to be the session with 513 positive and 230 negative examples.

Figure 34 shows a ROC curve comparing the training sessions with 48x48 pixels window size (Table 6) and 24x24 pixels window size (Table 7). Both training sessions were conducted with the same sample sizes and the only difference was in the window size.

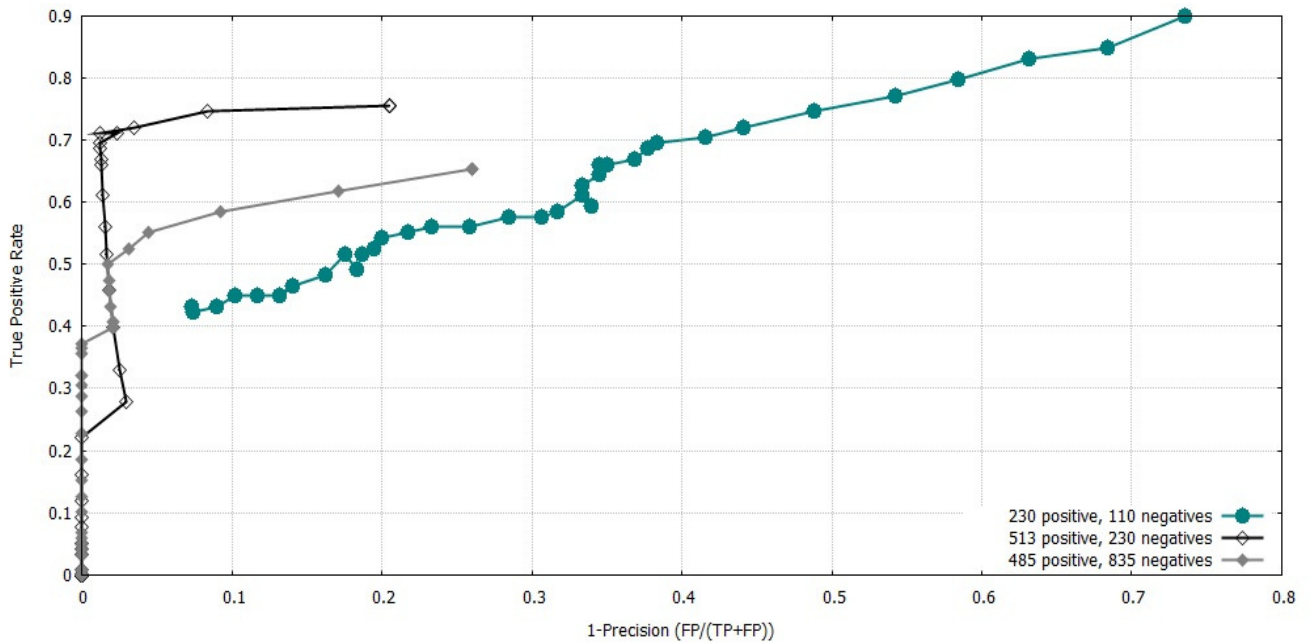


Figure 33: 1-Precision Curve For The First Three Training Sessions.

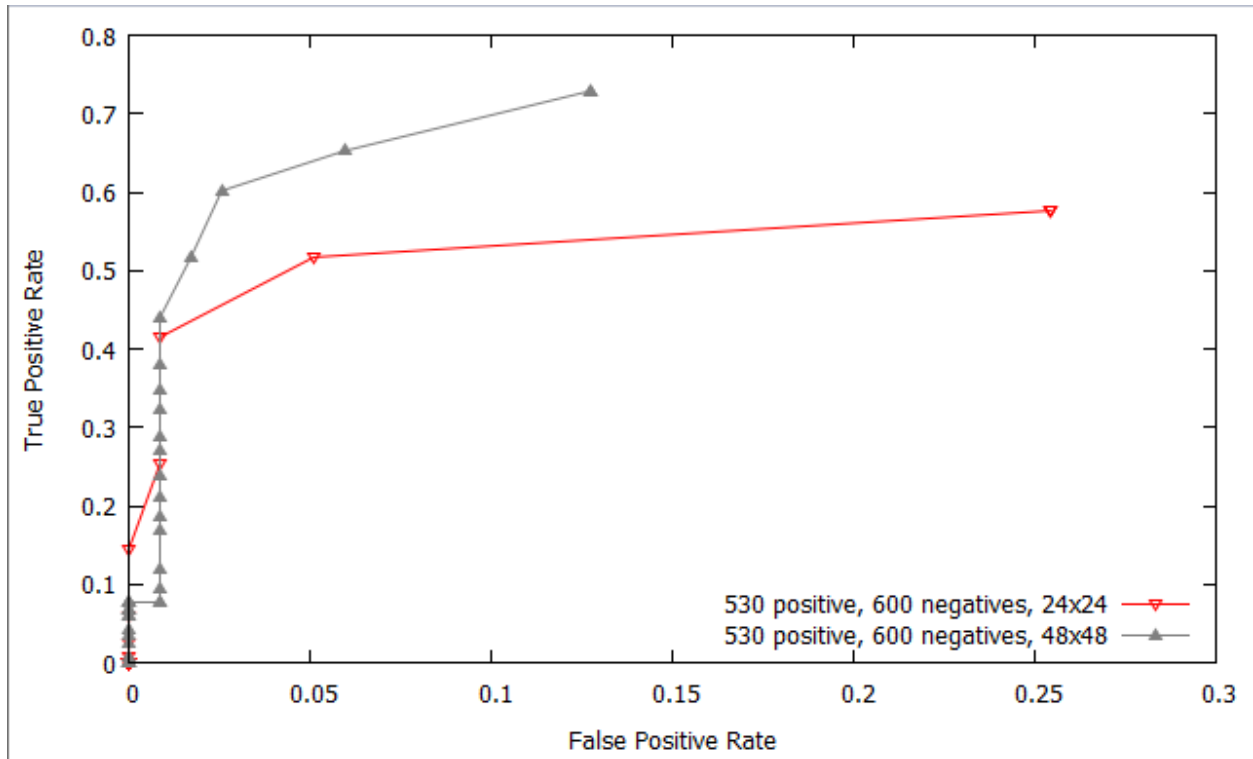


Figure 34: ROC Curve Comparing 24x24 and 48x48 Window Size

Discussion of Results

The following conclusions can be drawn from the presented results:

1. Good performance can be achieved with Adaboost training both when the positives examples are more than negative examples and vice versa. This is shown in Figure 33, where the performance for both cases have been presented. While in most cases training examples are in the range of 5,000 to 10,000 to get more accurate results, with Adaboost training, a lower number of positive and negative examples can still be used to achieve good detection rates. In this study 500 positive examples were used to achieve good results, both with 200 negative examples (half the positive size) and 944 negative examples (almost double the positive examples count). The sixth training session shown in Table 6 also tried to compare performance based on an almost equal number of positive examples and negative examples (530 positive and 600 negative examples). This performance was also relatively good with a true positive rate of 72.8% and a false positive rate of 12.7%. This false positive rate was the lowest achieved in the study (highest precision).
2. A trade-off has to be struck between the false positive rate and true positive rate. For example while the results in Table1 show a performance of 89.8% recall, it has a very high false positive rate of 249.1%. This means that this training session cannot be considered as the best. The best session is where there is relatively high recall and very low false positives as in Table 2, which achieved a recall of 75.4% and false positive rate of 19.4%. This is depicted by the 1-Precision curve in Figure 33, which shows that the curve for the training with 230 positives and 110 negatives does not follow the standard curve of 1-Precision and ROC curves. The other two curves on the other hand follow the pattern of ROC curves and 1-Precision curves. This could be attributed to the low training samples, making the training algorithm less realistic since the sample size used did not closely match the real world.
3. In comparing the performance of the training sessions on different scanning window sizes, there was not much difference in performance between the 48x48 pixels window size and 54x54 pixels window size. As shown in Figure 31, both training sessions resulted in similar curves, and finally the two curves met at the same point. This can be ascertained by the fact that apart from having a larger scanning window size of 54x54 pixels, the training sample size was also higher, making the effect of the window size, if any, negligible. This is important to note since the time required to train with a higher window size is greater than that of a lower size. As long as the window size selected represents the aspect ratio of the object to be detected and does not make the object too stretched, a lower window size can be chosen to improve the training time. Figure 34 shows, however, that halving the window size to 24x24 pixels reduced the recall from 72.8% to 57.6%, and doubled the false positive rate from 12.7% to 25.4%. In terms of

detection speeds, the 24x24 window size was almost 3 times faster (5.8 seconds) as compared to 14.8 seconds for the 48x48 window size. A trade-off can therefore be achieved between the window size and the speed of detection, versus the window size and recall and precision values.

4. Within the same training session, lower stages may have higher true positive rates, but also lower precision as shown in Table 8 and Table 9 compared to Table 2. The ROC curve in Figure 32 also depicts this difference. While the 10-stage classifier achieved a false positive rate of 19.4%, the 9-stage classifier has a false positive rate of 75.4%, and the 8-stage classifier had a false positive rate of 136.4%. This can be attributed to the fact that in Adaboost learning, higher stages are more complex and will reject more false positive regions, while maximizing the recall of the classifier. This is consistent with the description of Adaboost classifiers described by Viola and Jones (2004). Each successive stage in the cascade reduces the false positive rate and decreases detection rates. A trade-off target has to be selected, where there is minimum reduction in positives and maximum decrease in detection. However, an algorithm can have lower number of stages and perform well. The difference is between one stage of the cascade classifier and the next.
5. A comparison of the use of Haar features and LBP features in meter detection can also be presented. From the results shown in Table 10 and Table 11, the number of false detections in LBP for the test data was 89 compared to 10 for Haar features, when training was done with very low sample size. However, with higher sample sizes, LBP performed just as well as Haar training. The results show that for meter detection, both Haar features and LBP features can achieve good recall and false positive rates.
6. Just as with the performance tool, the Adaboost configuration files from the training sessions were also able to effectively detect new meters during the tests on the prototype. In addition, the prototype was able to locate the meter reading and binarize the cropped image containing the meter reading. This was then presented to an open source OCR software, Tesseract, for recognition. As shown in Figure 25, the binarized image has some noise near the meter reading area. Testing with Tesseract OCR engine showed that the noise can be challenging to OCR readers, and therefore requires more accurate OCR software, common with commercial OCR software. However, it could also be possible to build a classifier just for the meter reading numbers, which will then read the numbers without presenting them to OCR software. This can also be more accurate.

CHAPTER 6: CONCLUSION

6.1 Achievements

The main goal of this study was to develop a prototype that can intelligently read meters, as a way of offering an alternative solution to the problem of automated meters. This goal has been achieved in the following ways:

1. The study has identified the machine learning algorithms that one can use for solving the problem of reading meters automatically, without the need for manual reading of meters. The study has compared techniques that have been used in related problems and identified the suitable algorithm, Adaboost learning, for use in the problem of reading of meters.
2. Using Adaboost learning algorithm a classifier was trained on previous examples. In the experiments conducted, the system was able to achieve 75.4% accuracy. This is significant in this first attempt to solve the problem of reading of meters. The study also compared what results were achieved with different parameters of the learning algorithms.
3. The study was also able to come up with a design that can be used to solve the problem of automated reading of water meters and transmission of the readings to a utility provider. By using a system that interacts with the utility provider's existing data, the system is able to ensure that the data submitted from the automatic reading process is accurate. This can help to solve the problem of trust. In addition the study proposes a way of using GPS technology to verify the customer's location.
4. By interacting with existing work on machine learning techniques to solve the problem of reading meters, this study was also able to identify the modern tools that are being used to solve problems of learning in the computer field. One of the major fields in machine learning is in the area of computer vision. By identifying these current technologies, this study has given a pointer to those who are interested in this area of study.
5. The study only considered cases where the meters are upright. There is need to look for ways of detecting meters in any orientation or rotating the meter to an upright position before detection.

6.2 Research Contributions

The major contribution of this study is to solve a problem that affects many people, which, if implemented, can improve service delivery to citizens. By designing a prototype that can automatically read meters and send the readings to a utility provider, the study has shown how common problems can be solved by use of new technologies. The fact that this study uses machine learning techniques is also a major contribution to what can only be seen as the future of computing as such, i.e., the use of machine learning techniques in solving problems that would be difficult to solve otherwise.

6.3 Recommendation / Future Work

This study recommends the following areas for future work:

1. While this study achieved a 75.4% accuracy of the learning algorithm, object detection techniques such as face recognition have achieved much better results (Chen and Youlle, 2004). It is therefore necessary to seek ways of improving the algorithm.
2. OpenCV also offers other machine learning techniques that can be used to solve the problem of object detection. The study concentrated on only one of the algorithms, ie, Adaboost learning. A comparison was also made between different feature types used by Adaboost. These include Haar features and LBP features. Future work can explore other algorithms and how they compare with Adaboost learning.
3. This study proposed a way of transmitting the meter readings to the utility providers using techniques such as GPRS and GPS location settings on a smart phone. However, while the study has shown that this can be done since the platform used supports porting to smart phones, future study will have to actually port the prototype to a smartphone.
4. One challenge with meter readings is the noise around the meter reading region. This makes OCR detection challenging. More accurate OCR software such as commercial OCR software can, however, be used to achieve better performance. Future work can look at how to achieve better results with the actual reading of the binarized image. This may include building a classifier to recognize the meter numbers, instead of using OCR software.
5. The study was also not able to comprehensively compare performance between LBP features and Haar features. This is because the OpenCV LBP object detection module does not return sensitivity data, which is useful for generating the ROC curves data. The study was however able to come up with a simple performance measure for LBP. Future work can expand on this to create a complete performance measure tool for LBP.

6.4 Assumptions and Limitations

While this study has achieved good results in meter reading, this was achieved by taking various assumptions:

1. The utility providers and users are willing to use the technology to take meter readings.
2. Utility providers will incorporate the backend infrastructure to support the technology.
3. Utility providers use standard meters. However, in future, to achieve results with different kinds of meters, the algorithm will have to be trained on those meters.

The study also had the following limitations:

1. Due to time and budget restrictions, the dataset collected was lower than the ideal dataset of thousands of training samples. The use of a larger training sample dataset may have led to a higher recall than what was achieved.
2. The prototype could not be run on an actual device, even though the study shows that this is possible with the tools that have been used.

REFERENCES

- ABDOLLAHI, A., DEGHANI, M., ZAMANZADEH, N., 2007. SMS-based Reconfigurable Automatic Meter Reading System. *International Conference on Control Applications*, October 2007, pp.1103 - 1107.
- AGARWAL, S., AWAN, A., ROTH, D., 2004. Learning to Detect Objects in Images via a Sparse , Part-Based Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 26 (11), 2004, pp.1475-1490.
- ALPAYDIN E. (2010) Introduction to *Machine Learning* (2nd Edition). Newyork: MIT Press.
- BALA, K., BABU S. V., 2012. Remote Wireless Automatic Meter Reading System Based on GSM. *National Conference On Electrical Sciences*, 2012, pp.115–118.
- CABIOC, M.D., GERARDO, B.D. & BYUN, Y., 2011. SMS-Based Automatic Billing System of Household Power Consumption Based on Active Experts Messaging. *Communications in Computer and Information Science*, Volume 266, 2011, pp 229-238.
- CASACUBERTA, F. et al.,2005.The naive Bayes model , generalisations and applications.
- CHEN, X., YUILLE, A.L., 2004. Detecting and Reading Text in Natural Scenes. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004 Computer Vision and Pattern Recognition, Volume 2, 2004, pp 366-373.
- ESCALERA S., XAVIER BARÓ, JORDI VITRIÀ, AND PETIA RADEVA, 2009. Text Detection in Urban Scenes. *Conference of the Catalan Association for Artificial Intelligence*, 2009, pp 33-34.
- EMGU, *The EMGU Project*, v2.4.9., <http://www.emgu.com>, (June 2013).
- FERGUS, R.,FEI-FEI, L., PERONA, P., ZISSERMAN, A., 2010. Learning Object Categories from Internet Image Searches. *IEEE* Volume 98(8). Available from <http://www.robots.ox.ac.uk/~vgg/publications/2006/Fergus06/fergus06.pdf> [Accessed 12 June, 2013] and <http://www.robots.ox.ac.uk/~vgg/data/>.
- FREUND, Y., SCHAPIRE, R.E., 1999. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence* Volume 14(5),1999, pp.771–780.
- GALLE, C., (2010). *Automatic Meter Reading and the Advanced Metering Infrastructure Best Practices : Considerations in Wireless Design Automatic Meter Reading and the Advanced Metering Infrastructure*. [Online] Enfora. Available from <http://www.enfora.com/resource/AMRAMIBestPracticesWhitepaper.pdf>. [Accessed: 10/2/2013]
- GONZALEZ, C. R., WOODS, E. R. (2007) *Digital Image Processing* (3rd Edition). Prentice Hall.

- GUIDA, C., COMANDUCCI, D, COLOMBO, C., 2011. Automatic Bus Line Number Localization and Recognition on Mobile Phones — A Computer Vision Aid for the Visually Impaired. *Proceedings of the 16th international conference on Image analysis and processing*, Volume Part II, 2011, pp.323–332.
- Intel, *Intel Open Source Computer Vision Library*, v2.4.9., <http://sourceforge.net/projects/opencvlibrary> (June 2013).
- JUNG, K., KIM, I., JAIN, A., 2004. Text information extraction in images and video : a survey. *Pattern Recognition*, Volume 37 (5), May 2004, pp. 977-997
- McNABB, J. (2011) Vulnerabilities of wireless water meter networks. [Online]. Available from: http://media.blackhat.com/bh-us-11/McNabb/BH_US_11_McNabb_Wireless_Water_Meter_WP.pdf. [Accessed: April, 2013]
- MITCHELL M. T. (1997) *Machine Learning*. Newyork: McGraw-Hill.
- MURTY, N., DEVI, S., 2011. Bayes Classifier. *Undergraduate Topics in Computer Science*, Volume 0, 2011, pp 86-102.
- PAN, Y., HOU, X., LIU, C., 2008. A Robust System to Detect and Localize Texts in Natural Scene Images. *The Eighth IAPR International Workshop on Document Analysis Systems*, 2008, pp 35-42.
- PRATT, K. W. (2001) *Digital Image Processing* (3rd Edition). Prentice Hall.
- SHAREF, T., ISA, A., HASAN, A., TOORANI A., YADGAR A., (2013). Automated Meter Reading System Based on BASIC Stamp2 Microcontroller. *Asian Journal of Scientific Research*, Volume 6 (1), 2013,pp 88-97.
- SUN, D., ZHENG, S., 2009. Research and Design of Meter Reading System Based on ZigBee Wireless Sensor Network. *WSEAS Transactions on Circuits and Systems*, Volume 8(1), 2009, pp. 31-40
- VIOLA, P., JONES, M., 2004. Robust real-time face detection. *International journal of computer vision*, Volume 57(2), pp.137-154.
- WARFEL, Z. (2009). *Prototyping a Practitioner's Guide*. NewYork. Rosenfield Media.
- Zyl V. J. (2011) Introduction to Integrated Water Meter Management. Water Research Commission.

Appendix I: Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using System.Diagnostics;
using Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;
using Emgu.CV.UI;
using Emgu.CV.GPU;
namespace AutoMeterReader
{
    public partial class MeterReadingDetectForm : Form    {
        Image<Bgr, Byte> imageDetect;
        Image<Bgr, Byte> detectedImage;
        Image<Bgr, Byte> imageDetCopy;
        Image<Gray, byte> binarizedReading;
        Rectangle detectedRect;
        Rectangle meterReadingRect;
        private static MeterReadingDetectForm sForm = null;
        private OCRReader _ocrDetect;
        bool gray_in_use = false;
        public MeterReadingDetectForm()
        {
            InitializeComponent();
            _ocrDetect = new OCRReader("tessdata/");
        }
        public static MeterReadingDetectForm Instance()    {
            if (sForm == null)
            {
                sForm = new MeterReadingDetectForm();
            }
            return sForm;
        }
        private void detectBtn_Click(object sender, EventArgs e)    {
            int meterAcNum = 0;
            decimal meterRed = 0;
            int largestRect = -1;
            cmdReadNumber.Enabled = false;
            if (Int32.TryParse(acNumTextBox.Text, out meterAcNum) == false)
            {
                clsVariables.isTextMsg("Enter your Account Number with the Water
Company");
                acNumTextBox.Focus(); return;
            }
            if (accountExists(meterAcNum) == false)
```

```

        {
            clsVariables.isTextMsg("The Account number entered does not exist.
Kindly confirm that the number is correct");
            acNumTxtBox.Focus(); return;
        }
        OpenFileDialog Openfile = new OpenFileDialog();
        if (Openfile.ShowDialog() == DialogResult.OK)
        {
            //Load the image
            imageDetect = new Image<Bgr, Byte>(Openfile.FileName);
            imageDetCopy = imageDetect.Copy();
            //Display the Image
            imgPictBx.Image = imageDetCopy;
            imgPictBx.Update();

            long detectionTime;

            lblTime.Text = string.Empty;
            List<Rectangle> meters = new List<Rectangle>();
            DetectMeter.Detect(imageDetect, "cascade_test.xml", meters, out
detectionTime);
            //get the largest rectangle.
            if (meters.Count > 0)
            {
                int[] rectArray = new int[meters.Count];
                int curCount = 0;
                foreach (Rectangle meter in meters)
                {
                    rectArray[curCount] = meter.Width * meter.Height;
                    curCount += 1;
                }
                largestRect = rectArray.ToList().IndexOf(rectArray.Max());
            }
            if (largestRect != -1)
            {
                imageDetect.Draw(meters[largestRect], new Bgr(Color.Red), 2);
                detectedRect = meters[largestRect];
                cmdReadNumber.Enabled = true;
            }
            imgPictBx.Image = imageDetect;
            imgPictBx.Update();
            lblTime.Text = detectionTime.ToString() + " ms";
            if (validateMeterReading(meterAcNum, meterRed) == true )
            {
                transmitMeterReading(meterAcNum, meterRed);
            }
        }
    }
    public bool accountExists(int intAcNumber)        {
        int result = 0;
        string sql = "";
        DBUtils oado;
        try
        {
            oado = new DBUtils();

```

```

        sql = "select acct_number from customers " +
            " where acct_number=" + intAcNumber.ToString();
        DataTable myDT = oado.ExecuteTable(sql);
        if (myDT.Rows.Count > 0)
        {
            //get the only row in the table
            DataRow myRow = myDT.Rows[0];
            Int32.TryParse(myRow["acct_number"].ToString(), out result);
        }
    }
    catch (MySqlException) { throw; } //handles connection open & close,
ExecuteNonQuery
    catch (InvalidOperationException) { throw; }
    return result == 0 ? false : true;
}
public bool validateMeterReading(int intAcNumber, decimal meterReading)
{
    decimal dbReading = 0;
    bool result = false;
    string sql = "";
    DBUtils oado;
    try
    {
        oado = new DBUtils();
        sql = "select cur_reading from meters " +
            " where act_number=" + intAcNumber.ToString() + " order by
reading_date desc";
        DataTable myDT = oado.ExecuteTable(sql);
        if (myDT.Rows.Count > 0)
        {
            //get the only row in the table
            DataRow myRow = myDT.Rows[0];
            dbReading = (decimal)myRow["cur_reading"];
        }
        if (meterReading > dbReading)
        {
            result = false;
        }
        else
        {
            result = true;
        }
    }
    catch (MySqlException ex) { MessageBox.Show(ex.Message); ; } //handles
connection open & close, ExecuteNonQuery
    catch (InvalidOperationException ex) { MessageBox.Show(ex.Message); ; }
    return result;
}
public bool transmitMeterReading(int intAcNumber, decimal meterReading)
{
    SmartSQL ssql = new SmartSQL();
    DBUtils oado = null;
    string sql = "";
    bool result = false;
    try

```



```

        {
            oado = new DBUtils();
            ssql.StatementType = SmartSQL.STATEMENT_TYPE.TYPE_INSERT;
            ssql.AddTable("meters");
            ssql.AddFields("ACT_NUMBER", "CUR_READING", "READING_DATE");
            ssql.AddValues(intAcNumber, meterReading,
DateTime.Today.ToString(clsVariables.PREF_DT_FORMAT_DB));
            sql = ssql.SQL;
            ssql.Reset();
            oado.ExecuteNonQuery(sql);
            result = true;
        }
        catch (MySqlException seE)
        {
            //handles connection open & close, ExecuteNonQuery
            clsVariables.isTextMsg(seE.Message, MessageBoxIcon.Error);
        }
        catch (InvalidOperationException iopE)
        {
            //handles connection open
            clsVariables.isTextMsg(iopE.Message, MessageBoxIcon.Error);
        }
        catch (Exception ex) { clsVariables.isTextMsg(ex.Message,
MessageBoxIcon.Error); }
        finally
        {
            oado.Close();
        }
        return result;
    }
    private void cmdReadNumber_Click(object sender, EventArgs e)      {
        imgPictBx.Image = imageDetCopy;
        imgPictBx.Update();
        detectedImage = imageDetCopy.Copy(detectedRect);
        imgPictBx.Image = detectedImage;
        imgPictBx.Update();
        Image<Bgr, byte> template = new Image<Bgr, byte>("tempimagell.jpg");
        Image<Bgr, byte> imageToShow = detectedImage.Copy();
        using (Image<Gray, float> result = detectedImage.MatchTemplate(template,
Emgu.CV.CvEnum.TM_TYPE.CV_TM_CCOEFF_NORMED))
        {
            double[] minValues, maxValues;
            Point[] minLocations, maxLocations;
            result.MinMax(out minValues, out maxValues, out minLocations, out
maxLocations);
            if (maxValues[0] > 0.5)
            {
                Rectangle match = new Rectangle(maxLocations[0], template.Size);
                Rectangle meterRect = new Rectangle(match.X + match.Width + 9,
match.Y+7, match.Width * 7, 35);
                imageToShow.Draw(meterRect, new Bgr(Color.Red), 1);
                meterReadingRect = meterRect;
                imgPictBx.Image = imageToShow;
                imgPictBx.Update();
            }
        }
    }

```

```

    }
}
private void cmdOCRRead_Click(object sender, EventArgs e)
{
    Image<Bgr, byte> tmpImage = imageDetCopy.Clone();
    Image<Bgr, byte> myImage = tmpImage.Copy(detectedRect);
    Image<Bgr, byte> img = myImage.Copy(meterReadingRect);
    ProcessImage(img.Convert<Gray, Byte>());
    imgPictBx.Image = binarizedReading;
}
private void ProcessImage(Image<Gray, byte> image)
{
    Stopwatch watch = Stopwatch.StartNew(); // time the detection process
    List<String> metreadings = new List<String>();
    List<string> words = _ocrDetect.FindMeterReading(image, metreadings, out
binarizedReading);
    watch.Stop(); //stop the timer
    lblTime.Text =
String.Format(watch.Elapsed.TotalMilliseconds.ToString());
    Point startPoint = new Point(10, 10);
    for (int i = 0; i < words.Count; i++)
    {
        Debug.Print(String.Format("Meter Reading: {0}", words[i]));
    }
}
}
}
}

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.GPU;
namespace AutoMeterReader
{
    public static class DetectMeter
    {
        public static void Detect(Image<Bgr, Byte> image, String meterFileName,
List<Rectangle> meters, out long detectionTime)
        {
            Stopwatch watch;
            //Read the HaarCascade objects
            using (CascadeClassifier meter = new CascadeClassifier(meterFileName))
            {
                watch = Stopwatch.StartNew();
                using (Image<Gray, Byte> gray = image.Clone().Convert<Gray, Byte>())
                //Convert it to Grayscale
                {
                    //normalizes brightness and increases contrast of the image
                    try
                    {
                        gray._EqualizeHist();
                    }
                    catch (AccessViolationException ex)
                    {
                        Debug.Print(ex.Message);
                    }
                }
            }
        }
    }
}

```

```

        }
        catch (Exception ex)
        {
            Debug.Print(ex.Message);
        }
        Rectangle[] meterDetected = meter.DetectMultiScale(
            gray,
            1.1,
            3,
            new Size(48, 48),
            Size.Empty);
        meters.AddRange(meterDetected);
    }
    watch.Stop();
}
detectionTime = watch.ElapsedMilliseconds;
}
}
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Drawing;
using Emgu.CV;
using Emgu.CV.OCR;
using Emgu.CV.Structure;
using Emgu.CV.UI;
using Emgu.Util;
namespace AutoMeterReader
{
    public class OCRReader : DisposableObject
    {
        /// The OCR engine
        private Tesseract _ocr;
        public OCRReader(String dataPath)
        {
            //create OCR engine
            _ocr = new Tesseract(dataPath, "eng",
Tesseract.OcrEngineMode.OEM_TESSERACT_ONLY);
            _ocr.SetVariable("tessedit_char_whitelist", "1234567890");
        }
        public List<String> FindMeterReading(Image<Gray, Byte> gray, List<String>
strmeterreadings, out Image<Gray, Byte> binarizedImage)
        {
            Image<Gray, Byte> filteredReadingShow;
            using (Image<Gray, Byte> tmp1 = gray.Copy())
            {
                //Resize. This size of front results in better accuracy from tesseract
                using (Image<Gray, Byte> tmp2 = tmp1.Resize(1000, 800,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC,true))
                {
                    //removes some pixels from the edge
                    int edgePixelSize = 2;
                    tmp2.ROI = new Rectangle(new Point(edgePixelSize, edgePixelSize),
tmp2.Size - new Size(2 * edgePixelSize, 2 * edgePixelSize));
                    Image<Gray, Byte> mrdImg = tmp2.Copy();
                }
            }
        }
    }
}

```

```

        Image<Gray, Byte> filteredReading = FilterMeterReading(mrdImg);
        filteredReadingShow = filteredReading.Copy();
        Tesseract.Character[] words;
        StringBuilder strBuilder = new StringBuilder();

        using (Image<Gray, Byte> tmp = filteredReading.Clone())
        {
            _ocr.Recognize(tmp);
            words = _ocr.GetCharactors();
            for (int i = 0; i < words.Length; i++)
            {
                strBuilder.Append(words[i].Text);
            }
        }
        strmeterreadings.Add(strBuilder.ToString());
    }
    binarizedImage = filteredReadingShow.Clone();
    return strmeterreadings;
}

private static Image<Gray, Byte> FilterMeterReading(Image<Gray, Byte>
mreadImg)
{
    mreadImg._EqualizeHist();
    //smoothed
    Image<Gray, byte> smoothedGrayFrame = mreadImg.PyrDown();
    smoothedGrayFrame = smoothedGrayFrame.PyrUp();
    //canny
    Image<Gray, byte> cannyFrame = null;
    cannyFrame = smoothedGrayFrame.Canny(50,50);
    //smoothing
    mreadImg = smoothedGrayFrame;

    //binarize
    Image<Gray, Byte> thresh = mreadImg.ThresholdBinaryInv(new Gray(50), new
Gray(255));
    thresh._Not();
    thresh._Or(cannyFrame);
    return thresh;
}
protected override void DisposeObject()    {
    _ocr.Dispose();
}
}

// Test LBP Performance

namespace AutoMeterReader
{
    public partial class TestPerfForm : Form
    {
        public TestPerfForm()
    }
}

```

```

    {
        InitializeComponent();
    }
private void lbpTestBtn_Click(object sender, EventArgs e)
{
    //read file and store details in
    int counter = 0;
    string line; double scale_factor = 1.1;
    float maxSizeDiff = 1.5F; float maxPosDiff = 0.3F;
    int minNeighbors = 3; double distance;
    bool found = false; int rocsz = 40;
    int hits=0; int missed=0;
    int falseAlarms = 0; int totalHits =0;
    int totalMissed =0; int totalFalseAlarms =0;
    long totalDetectionTime = 0;
    //118 is the number of test images
    TestDataDetails [] testDataAll = new TestDataDetails[118];
    int[] pos = new int [rocsz]; int[] neg = new int [rocsz];

    // Read the file and display it line by line.
    System.IO.StreamReader file = new
System.IO.StreamReader("testdatall.dat");
    TextWriterTraceListener myWriter = new
TextWriterTraceListener(System.Console.Out);
    Debug.Listeners.Add(myWriter);
    while ((line = file.ReadLine()) != null)
    {
        //split string
        string[] lineDataS = line.Split(' ');
        if (lineDataS.Length == 6)
        {
            testDataAll[counter].filename = lineDataS[0];
            int.TryParse(lineDataS[1], out testDataAll[counter].refcount);
            int.TryParse(lineDataS[2], out testDataAll[counter].x);
            int.TryParse(lineDataS[3], out testDataAll[counter].y);
            double.TryParse(lineDataS[4], out testDataAll[counter].width);
            double.TryParse(lineDataS[5], out testDataAll[counter].height);
            testDataAll[counter].calcx = 0.5F * testDataAll[counter].width +
testDataAll[counter].x;
            testDataAll[counter].calcy = 0.5F * testDataAll[counter].height +
testDataAll[counter].y;
            testDataAll[counter].calcWidth = Math.Sqrt( 0.5F *
(testDataAll[counter].width * testDataAll[counter].width +
testDataAll[counter].height * testDataAll[counter].height));
        }

        counter++;
    }
    file.Close();
    //now we can detect the loaded files
    //Load the image
    Image<Bgr, Byte> imageDetect; long detectionTime;
    Debug.WriteLine
(string.Format("+=====+=====+=====+=====+\n"));
}

```

```

Debug.WriteLine (string.Format("|
|Missed| False|\n"));
Debug.WriteLine
(string.Format("=====+\n"));
for (int i = 0; i < 118; i++)
{
    imageDetect = new Image<Bgr, Byte>(testDataAll[i].filename);
    List<Rectangle> meters = new List<Rectangle>();
    //DetectMeter.Detect2(imageDetect, "cascade_test03.xml", meters,
scale_factor, minNeighbors, out detectionTime);
    DetectMeter.Detect(imageDetect, "cascade_test03.xml", meters, out
detectionTime);
    totalDetectionTime += detectionTime;
    missed = 0; hits = 0; falseAlarms = 0;
    if (meters.Count > 0)
    {
        counter = 0;
        TestDataDetails[] detectedDat = new
TestDataDetails[meters.Count];
        foreach (Rectangle meter in meters)
        {
            detectedDat[counter].x = meter.X;
            detectedDat[counter].y = meter.Y;
            detectedDat[counter].calcx = 0.5F * meter.Width + meter.X;
            detectedDat[counter].calcy = 0.5F * meter.Height + meter.Y;
            detectedDat[counter].calcWidth = Math.Sqrt(0.5F *
(meter.Width * meter.Width + meter.Height * meter.Height));
            detectedDat[counter].neibors = meters.Count; counter++;
        }
        found = false;
        for (int j = 0; j < detectedDat.Length; j++)
        {
            distance = Math.Sqrt(0.5F * ((detectedDat[j].calcx -
testDataAll[i].calcx) * (detectedDat[j].calcx - testDataAll[i].calcx) +
(detectedDat[j].calcy -
testDataAll[i].calcy) * (detectedDat[j].calcy - testDataAll[i].calcy)));
            if ((distance < testDataAll[i].calcWidth * maxPosDiff) &&
(detectedDat[j].calcWidth > testDataAll[i].calcWidth /
maxSizeDiff) && (detectedDat[j].calcWidth < testDataAll[i].calcWidth * maxSizeDiff))
            {
                testDataAll[i].found = 1;
                testDataAll[i].neibors = Math.Max(1,
detectedDat.Length);
                found = true;
            }
            Else
            {
                falseAlarms++;
                neg[Math.Min(detectedDat[j].neibors, rocsz - 1)]++;
            }
        }
        if (testDataAll[i].found == 1)
            hits++; pos[Math.Min(1, rocsz - 1)]++;
        Else
            missed++;
    }
}

```


Appendix II: Training Cascade File

```
<?xml version="1.0"?>
<opencv_storage>
<!-- Automatically converted from trainout\cascade.xml, window size = 48x48 -->
<cascade_test type_id="opencv-haar-classifier">
  <size>
    48 48</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  0 0 45 30 -1.</_>
                <_>
                  15 10 15 10 9.</_></rects>
              <tilted>0</tilted></feature>
              <threshold>4.7469174861907959e-001</threshold>
              <left_val>-9.8198199272155762e-001</left_val>
              <right_val>1.</right_val></_></_>
            <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  0 0 45 33 -1.</_>
                <_>
                  15 11 15 11 9.</_></rects>
              <tilted>0</tilted></feature>
              <threshold>5.8714973926544189e-001</threshold>
              <left_val>-9.8230087757110596e-001</left_val>
              <right_val>1.</right_val></_></_></trees>
            <stage_threshold>1.7699124291539192e-002</stage_threshold>
            <parent>-1</parent>
            <next>-1</next></_>
          <_>
        <!-- stage 1 -->
        <trees>
          <_>
            <!-- tree 0 -->
            <_>
              <!-- root node -->
              <feature>
                <rects>
                  <_>
```



```

        1 0 42 33 -1.</_>
        <_>
        15 11 14 11 9.</_></rects>
        <tilted>0</tilted></feature>
        <threshold>7.9169398546218872e-001</threshold>
        <left_val>-7.3109245300292969e-001</left_val>
        <right_val>9.3203884363174438e-001</right_val></_></_>
    <_>
    <!-- tree 1 -->
    <_>
    <!-- root node -->
    <feature>
        <rects>
            <_>
            44 35 4 9 -1.</_>
            <_>
            44 35 2 9 2.</_></rects>
            <tilted>1</tilted></feature>
            <threshold>5.8609610423445702e-003</threshold>
            <left_val>-7.6618534326553345e-001</left_val>
            <right_val>9.6327316761016846e-001</right_val></_></_></trees>
    <stage_threshold>1.6585347056388855e-001</stage_threshold>
    <parent>0</parent>
    <next>-1</next></_>
</stages></cascade_test>
</opencv_storage>

```

Appendix III Installation Of OpenCV And Other Tools

1. To install OpenCV, download the latest compiled version for the Windows platform from <http://opencv.org/downloads.html> and install on your computer. When you install OpenCV, it comes with libraries already compiled.
2. Download Microsoft Visual Studio Express redistributable from <http://www.microsoft.com>. If you are interested in development, you can also download the Express edition.
3. To code in Microsoft Visual Studio C#, you need to also download the Emgu library and install from http://www.emgu.com/wiki/index.php/Download_And_Installation.
4. You need to decide which Windows platform you will use, whether it is 32-bit or 64-bit. You cannot mix the two platforms, and the 32-bit is an easier platform to work to begin with, before exploring 64-bit.
5. When you install the precompiled version of OpenCV, it comes with all the libraries required to execute OpenCV applications. You can easily get things working by copying the libraries from the bin folder in the OpenCV installation folder to the Windows system folder in C:\Windows\System32. The libraries can also be copied to the application folder.
6. The precompiled OpenCV installation also comes with source files which can be compiled using Visual Studio Express edition. Most of the training tools and examples are not available until compiled. This can easily be done by opening the ALL_BUILD project file and compiling the tool you want.